Federal Office
for Information Security

Security of AI-Systems: Fundamentals

Provision or use of external data or trained models

Abstract

Recent advances in deep learning have enabled the development of high-performing AI systems. While such systems are increasingly applied also in safety- and security-critical areas, important aspects of such systems including their robustness, transparency, and decision fairness are not fully understood yet. Security issues particularly arise if multiple parties share and exchange resources such as machine learning models or data sets. In this study, we investigate the current state of the art on attack vectors relevant to these scenarios and discuss the effectiveness of existing mitigation techniques. Based on the literature and practical investigations of specific use cases, open challenges in the research field are identified. Furthermore, this study provides a collection of recommendations for improving security of machine learning systems and related development processes.

Publisher

Federal Office for Information Security
`https://www.bsi.bund.de`

Disclaimer

The security of AI systems is a relatively new and rapidly developing field of research. The study reflects the state of research at a fixed point in time and is not continuously updated. Currently, moreover, no standards exist to systematically assess and compare attacks on AI systems and associated defenses. Thus, results from research articles may only be valid under very specific conditions and cannot be generalized. The study addresses experts and should be used with caution. For the development of secure AI systems, an individual risk analysis should always be carried out on a case-by-case basis, taking into account current research progress and specific framework conditions.
The term privacy is used repeatedly in the context of this study. In the research area of AI system security, the term is used by default when referring to attacks that have the goal of extracting confidential information from an AI model. However, the term privacy refers to confidential properties of the used models or training data in a figurative sense and not necessarily to personal data. In the following, the term is to be understood in this figurative sense. In particular, the data protection of individuals and compliance with data protection laws are not addressed in this study.

Contents

Introduction

Nowadays, machine learning is applied to various applications in industry and society, influencing an increasing number of aspects of everyday life. A large amount of research is published every year on novel machine learning approaches, and high performance has been achieved on a number of important task such as object recognition and speech recognition. This impressive progress was largely enabled by advanced in deep learning in the last decade boosted by hardware improvements (e.g., GPU computing). Machine learning methods are also considered for application in safety- and security-critical areas, for example, autonomous driving or medical diagnosis. Yet, the research community to date has only a partial understanding of the security implications of using such systems. Studies on adversarial machine learning indicate that there is a significant risk of attackers maliciously exploiting weaknesses of machine learning algorithms, for instance, to cause a system to make mistakes or to steal confidential information from the provider of the resource. The interest in such adversarial machine learning increased drastically in the last years as can be seen from Figure 1 which shows the increase of research papers in a particular sub discipline of this field. The constantly growing body of literature with sometimes contradicting findings and no unified benchmarks is making it challenging to infer the security implications that the development and deployment of machine learning models entail.

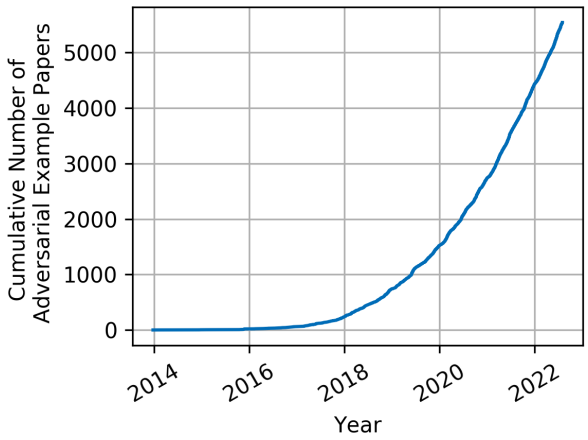


Figure 1: Development of the number of adversarial example papers between 2014 and 2022, taken from [1].

Security threats are particularly relevant in scenarios where multiple parties share and access the same pretrained models or data sets. At the same time, modern machine learning approaches are based on deep learning methods which use large neural network architectures requiring enormous amounts of training data and computational resources for a successful implementation. Therefore, for many machine learning practitioners the sharing of resources is an essential ingredient for a successful machine learning model development. A common use practice in machine learning model development is the application of *transfer learning*

to quickly and efficiently train models with good accuracy from pretrained neural networks. Transfer learning is attractive as it reduces the need for large amounts of data significantly, as well as the computational requirements of model training, allowing stakeholders with fewer computational and monetary resources to keep up with larger companies in terms of creating competitive machine learning systems.

The need for pretrained models and large data sets offers also an incentive for developers of machine learning models or data sets to share their resources with untrusted third parties, for instance, when open-sourcing research results, or when offering proprietary models to other stakeholders. In such cases, there are no standard mechanisms available for providers to ensure the leakage of potentially confidential training data or sensitive details about the training process.

This study investigates security threats that arise if a part of the model training is outsourced (Part I), and if models (Part II) or data sets (Part III) are provided to untrusted third parties.

Part I investigates vulnerabilities that arise from the usage of transfer learning for model development. In particular, we focus on the risk of poisoning and backdoor attacks, a recently emerging research field that is still insufficiently investigated with regard to the security implications it entails for the use of machine learning models provided in public repositories. We illustrate the identified risks with a specific focus on medical imaging, and conduct a case study with the aim to assess the effectiveness of existing defense methods.

Part II focuses on the scenario where an machine learning model is shared with untrusted third parties by providing black-box access to the model. We discuss risks regarding the leakage of sensitive information breaching confidentiality and conduct practical investigations of differential privacy (DP), a method which can be used to protect data confidentiality and derive specific recommendations.

Part III focuses on sharing confidential data sets with external users. Specifically, we discuss methods that enable developers to use the data sets without gaining explicit access. This research fields involves methods such as federated learning, generative models, or methods allowing for training on encrypted data.

In every part, we analyze the current state of the research field of the respective security threat, highlight unsolved problems, and derive recommendations based on the currently available mitigation strategies known from the literature. This study provides guidance for practitioners to increase the resilience of machine learning systems against attacks in the described scenarios. Importantly, this report reflects the state of the literature at the time of writing. Due to the fast progress in the field of adversarial machine learning, new attack and defense methods are published every day. Therefore, we recommend the reader to verify recommendations derived from this study based on the current state of the literature.

Part I

Transfer Learning

Philippsen, Anja, Dr. (neurocat GmbH)
Assion, Felix (neurocat GmbH)
Kawa, Nura (neurocat GmbH)
Firnkorn, Jörg, Dr. (neurocat GmbH)
Lisowska, Aneta, Dr. (neurocat GmbH)
Agramunt-Puig, Sebastià, Dr. (neurocat GmbH)
Zimmer, Raphael, Dr. (BSI)
Sennewald, Britta (BSI)

1 Transfer Learning: Literature Overview

This part considers the case of transfer learning where a model provided by an external source (e.g., downloaded from a machine learning service platform or from open source directories) is adopted and refined with the purpose to train it on a similar or related task using another (typically smaller) set of training data.

In the following, transfer learning is first introduced in detail. Then an overview is provided about research fields for which transfer learning is commonly applied, and the particular security vulnerabilities that are related to the transfer learning use case are identified (Section 1.1). Hereby, we focus specifically on poisoning and backdoor attacks, a recently emerging field of research bringing security implications for the usage of models from public repositories.

In the second part (Section 1.2), attack vectors and defenses for a selected application scenario, namely medical imaging, are analyzed in detail.

1.1 Broad Discussion of Relevant Research Field

Transfer learning is an efficient technique to train a machine learning (ML) model (student model) by reusing an existing model (teacher model) and fine-tuning the weights on a specific data set. This technique is recently applied in a variety of application scenarios because it significantly reduces the need for training data and saves time and computational effort in the model training phase. While models trained with this technique may achieve high performance, they also have a risk of inheriting weaknesses from the teacher model.

This section introduces to the concept of transfer learning (Section 1.1.1), the motivation

for using it (Section 1.1.2), and lists common application scenarios in a variety of industries (Section 1.1.3). Finally, we discuss in detail the security vulnerabilities that are related to the transfer learning use case (Section 1.1.4).

1.1.1 What Is Transfer Learning?

The overall goal of transfer learning is to boost the performance of a learner on the target task (*student task*) through knowledge transfer from another task (*teacher task*). In the context of deep learning, transfer learning (TL) is an approach for effective training of deep learning models in which previously gained knowledge is leveraged for learning a new task.

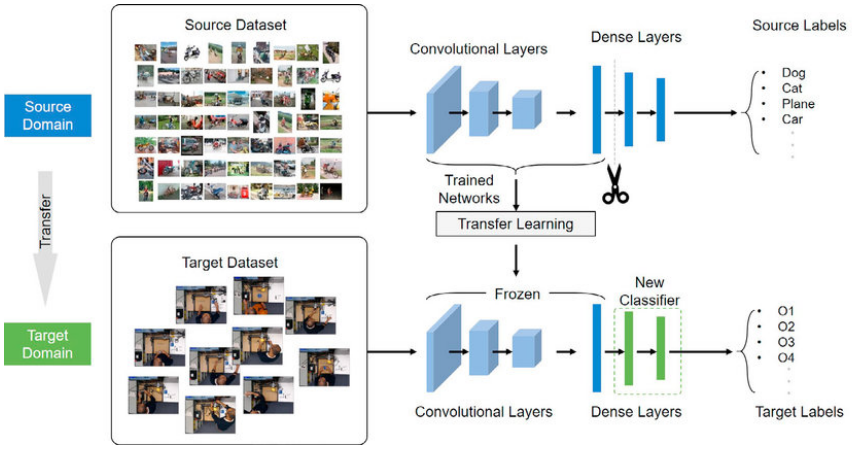


Figure 2: Illustration of the transfer learning setup for an example from the manufacturing domain, taken from [2].

The formal transfer learning definition, formulated by Pan & Yang [3] and used in the recent transfer learning surveys [4, 5] considers the knowledge transfer in the context of *source* $\mathcal{D}_S$ and *target* $\mathcal{D}_T$ domains with their corresponding tasks $\mathcal{T}_S, \mathcal{T}_T$. A domain $\mathcal{D}$, a task $\mathcal{T}$ and the transfer learning process are defined as follows:

Definition 1 *Domain* $\mathcal{D} = \{\mathcal{X}, P(X)\}$*, where* $\mathcal{X}$ *is a feature space,* $P(X)$ *a marginal distribution and* X *is a learning sample with* n *feature vectors* $X = \{x_1 \cdots x_n\} \in \mathcal{X}$*.*

Definition 2 *Task* $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$*, where* $\mathcal{Y}$ *is a label space and* $f(\cdot)$ *is a decision function learned from the feature vector and label pairs* $\{x_i, y_i\}$*, where* $x_i \in X$ *and* $y_i \in \mathcal{Y}$*.*

Definition 3 *Given* $\mathcal{D}_S, \mathcal{T}_S$ *and* $\mathcal{D}_T, \mathcal{T}_T$*,* transfer learning *is a process of improving the target decision function* $f_T(\cdot)$ *utilizing the knowledge from source domain* $\mathcal{D}_S$ *and task* $\mathcal{T}_S$*, where* $\mathcal{D}_S \neq \mathcal{D}_T$ *or* $\mathcal{T}_S \neq \mathcal{T}_T$*.*

There are different categories of approaches to knowledge transfer between source and target tasks [4]:

- instance-based – this approach relies on being able to directly reuse selected instances from the source domain alongside target domain examples for target model training [6]. It is particularly useful when the source and target domain have the same features and labels but vary in the data distribution [7].
- feature-based – methods that aim at learning effective (or common) feature representations. When the source data set is labeled and sufficiently large, the features could be learned through supervised training (e.g., computer vision [8]). However, unsupervised and self-supervised training from very large data sets can also yield helpful representations (e.g., natural language [9] or audio processing [10]).
- parameter-based – this approach assumes that the source domain model and target domain model have some of the same optimal parameters and, therefore, the knowledge can be transferred at the model level. This scenario is illustrated in Figure 2. Even though in this study we focus on deep transfer learning, the parameter-based transfer is not specific to deep learning models. The partial reuse of learned parameters has been previously explored utilizing support vector machines [11] and collection of ensemble classifiers [12].
- relational-based - when the relationships among the data in the source and target domains are similar, knowledge about this relationships can be transferred from the source domain to the target domain. This method is less commonly applied, but it has shown to be useful for time-series data [13].

In the context of deep learning models there is a smooth transition between feature- and parameter-based approaches. Typically, the early model layers are *frozen*, i.e., not adapted during fine-tuning. If all existing layers are frozen and only the classification layer is replaced for student training, the teacher model serves as a feature extractor. This type of transfer learning is also known as *feature extraction transfer learning*. The other extreme case, where the student model is initialized with the parameters of the teacher model but (almost) all layers are retrained, is referred to as *weight initialization transfer learning*.

1.1.2 Why Use Transfer Learning?

Transfer learning is used by different entities for a variety of reasons. On a direct model performance level, transfer learning boosts generalization [14], enables improved model performance on the target task [8] and speeds up convergence [15], compared to training a model from scratch. Transfer learning is most beneficial when the model is pre-trained on a source domain which is very similar to the target domain. Domain similarity can be measured, for example, using the earth-mover's distance [16].

Beside improving model performance, transfer learning is used for a variety of further reasons. Saving costs during the training process is a central factor, as in many industries and domains acquiring high-quality labeled training data is expensive [17, 18]. Even if financial constraints are not considered, often the required training data is simply not available, for example due to reasons related to ethics, data protection, or intellectual property [19]. Finally, in competitive settings, the element of speed in model development is an additional reason why business stakeholders use transfer learning [20].

1.1.3 Transfer Learning Applications

Transfer Learning in Computer Vision

Transfer learning has proven to be especially useful in the domain of computer vision. In this section we will outline successful cases of transfer learning on sports science, healthcare, agriculture, transportation, retail and manufacturing.

In the field of sports science, machine learning is used to help athletes to improve their performance in competitions and prevent injuries but also to monitor sports events and safety in sports. As a more concrete application, examples of pose correction and classification using transfer learning have been implemented for mice [21] and also on human tracking [22, 23]. These results have implications on pose correction for sports like Yoga or Pilates, substituting human trainers by mobile apps like Zenia [24]. Other systems used in Tennis or Hockey deal with tracking the ball itself instead of the player, which is important for automatic sports events broadcasting as well as for analyzing the performance of the player in real time. In this case, transfer learning has been used to transfer the knowledge in between types of sports, for example, from Tennis to Hockey [25, 26]. Getting individual player information in real time is important and so is being able to identify player actions (e.g., goal, penalty, or corner) in an automatic way at the end of the game for coach assistance. This is known as *activity recognition*. Activity recognition has been successfully applied in Hockey [27] using transfer learning on the pretrained VGG-16 neural network architecture.

One field with clear applications for machine learning and a direct impact to the whole population is healthcare. Healthcare is a challenging domain for machine learning since data is scarce due to privacy reasons. It is precisely in this scenario where transfer learning thrives to close the gap to apply machine learning on medical scenarios. There are several examples of transfer learning in the medical domain, for instance in melanoma detection. Already in 2017, superhuman melanoma classification level was achieved by a Stanford group [28]. Later on, other researchers used the AlexNet [29] or VGG-16 architecture [30], pretrained on ImageNet, as teachers to train melanoma classifiers whilst reducing the computation time and the data requirements.

A relevant application of computer vision in the medical sector is on magnetic resonance imaging (MRI) and computer tomography (CT) as they are medical imaging techniques with

high resolution. Concretely in the case of MRI in commercial devices we are on the range of 1.5-2.0 mm of resolution. In this case, machine learning can help radiologists to diagnose more accurately or perform more mechanical tasks like segmentation for ventricle volume calculation [31], and denoising the original image [32]. Companies in the medical imaging sector are already commercializing MRI scan machines with embedded AI, for example, for segmentation assistance [33]. Transfer learning can be used in this domain to overcome the data scarcity (and quality) of MRI images taken under the same protocol, and also for image reconstruction of fast magnetic resonance imaging [34]. This last problem is currently solved using compressed sensing (see Chapter 10 of [35]), which by nature is a time consuming technique (one needs nonlinear optimization algorithms to solve the equations) and, thus, it is difficult for real-time application. Other uses for transfer learning on MRI for brain imaging (see [36] for a comprehensive review) or more concretely for brain age prediction [37] or brain tumor prediction [38] have been reported.

Also in cell biology, deep learning techniques are commonly employed. One application scenario is the discovery of new drugs in a fast and reliable way. Transfer learning is useful in this context to solve the scarce data problem [39]. For instance, convolutional neural networks (CNNs) can be used which were pretrained on large data bases of bioactivity data, and then fine-tuned on the functioning of specific molecules [40]. Another application scenario where deep learning and, specifically, transfer learning is used is super resolution microscopy [41] to image single protein movements[42, 43]. For instance, [42] demonstrates that a convolutional neural network can be trained to classify protein subcellular localization in GFP-tagged yeast cells, and that the model can be reused as a teacher model to solve other biological image recognition tasks.

Another important application of computer vision in medicine is in the field of histology. In this case, the imaging is happening at the micrometer scale and a physicians task is to differentiate malicious from benign tissues. Here, machine learning has been applied mainly in the field of breast cancer detection [44, 45, 46], but also for classifying cardiovascular tissues [47] or carcinoma [48]. Transfer learning has been applied to a breast cancer classification scenario by Beevi *et al.* [49] by means of a VGG pretrained neural network architecture to detect mitosis from breast histopathology images. Other examples are the works of Song *et al.* [50, 51] where they use an image feature representation with a Fisher Vector that is extracted from a CNN pretrained on ImageNet. Finally, also on cardiovascular tissue, widely available pretrained models (e.g., ResNet or VGG architectures) were used as teachers [47].

In agriculture, computer vision has been used for several purposes. For instance, AI systems are used to observe the improvement of crop yield using aerial or satellite images [52], for agricultural statistics [53] to develop hydrological models [54] and even on flood damage estimation and water quality measurement [55]. In this context some studies [56, 57, 58, 59] use pretrained computer vision models to improve their prediction and to reduce the training time. Example applications are plant disease identification [57], anomaly detection [58], or more specific millet crops disease diagnosis [59] using unmanned vehicles.

Transportation is another field where the transfer of computer vision models plays an important role. Traditional use cases of computer vision for transportation include vehicle type classification [60], detection of traffic violations [61, 62], abnormal traffic detection [63], parking occupancy detection [64], and automatic license plate recognition [65]. Recently, autonomous vehicles have boosted the research on computer vision for transportation, encompassing tasks such as pedestrian detection [66], traffic sign detection [67], collision avoidance [68], or driver attentiveness detection [69, 70]. Transfer learning has been also applied in these use cases. For instance, in [71] authors use transfer learning for traffic forecasting, in [72] vehicle classification is done using GoogLeNet CNN. Other studies show how transfer learning is useful for autonomous driving. For instance, in [73] the authors cover the case of learning rare edge cases like vehicle accidents by means of transfer learning. Finally, transfer learning has even been applied for pedestrian detection in several studies [74, 75].

Transfer Learning in Natural Language Processing (NLP)

The vast amount of unlabeled text that can be readily scraped from the world wide web aided the development of powerful self-supervised teacher models for NLP problems. Examples of models are ELMo [76], GPT/GPT-2 [77] and BERT (Bidirectional Encoder Representations from Transformers) [78]. All of these methods use text corpora for extracting a language model that can subsequently be used for training models on specific tasks of language understanding. BERT (Bidirectional Encoder Representations from Transformers) [78] was one of the first pretrained models that was made openly available for fine-tuning of multiple downstream NLP tasks. It is a bidirectional transformer trained on BookCorpus and Wikipedia with a multi-task objective: masked language modeling and next-sentence prediction. In the original paper, the authors suggest that to use BERT for fine-tuning one has to *"simply plug in the tasks specific inputs and outputs into BERT and fine-tune all the parameters end-to-end."* This simple fine-tuning strategy has been shown to work initially for sentiment analysis and named entity recognition [78]. Later, it has been adapted for solving many specialized tasks such as patent classification [79].

An example for the successful application of transfer learning using the BERT model is presented in [80] for the use case of hate speech detection. Specifically, Mozafari *et al.* compared four BERT fine-tuning strategies: BERT followed just by a fully connected layer, BERT followed by nonlinear layers, BERT followed by a bidirectional Long Short Term Memory (LSTM) layer and BERT followed by a CNN. They showed that the CNN-based fine-tuning strategy of the BERT model yields the best performance [80].

Sun *et al.* in their evaluation of BERT fine-tuning strategies on various classification tasks (sentiment analysis, question and topic classification) [81] considered also further pretraining and multi-task fine-tuning. They concluded that additional pretraining both within-task and in-domain can significantly boost BERTs classification performance. Domain specific pretraining of BERT has become popular and gave rise to models such as:

- SciBERT [82] trained on a large multi-domain corpus of scientific publications, tuned for example for annotation of Participants, Interventions, Comparisons, and Outcomes spans in clinical trials.
- E-BERT [83] for e-commerce and tasks such as: review-based question answering, aspect extraction, aspect sentiment classification, and product classification.
- BioBERT pretrained on PubMed [84], utilized in applications such as radiology label extraction [85].
- Chinese BERT trained on both simplified and traditional Chinese extracts from Wikipedia and applied to Chinese NLP tasks [86].

Transfer Learning for Time Series Data

Time series modeling and prediction is important in several domains such as in the finance and the energy sector, manufacturing, or for clinical applications, usually with the aim to predict ahead or detect anomalies [87]. However, usage of transfer learning for time series data is less widespread than for image or text data, mainly due to the lack of success [88]. Finding an appropriate architecture for a specific time series modeling task is not trivial due to the variety of potential model types, ranging from recurrent neural network models to convolutional architectures, limiting the transferability of trained models. In addition, models that were trained for a particular task tend to be sensitive to the nature and dynamics of the used training data. Thus, even slight differences between the source and target data sets lead to significant performance decreases [88]. For these reasons, to date there does not exist a gold standard model that can be adopted for a large variety of use cases. In particular, it is difficult to find a source model that is applicable for a multivariate time series which combines multiple types of signals.

Nevertheless transfer learning is attractive for modeling time series as clean labeled training data are usually scarce. There are studies which incorporate transfer learning for prediction making in the context of the financial sector [89, 90], energy consumption [91, 92], for clinical applications [93], and for the detection of anomalies in industrial applications [87] and flight data [94]. In most cases, the transfer occurs within the same domain, for example, between different financial markets [95], while there is little work on enabling transfer between domains.

In recent years, some efforts have been made to find a more general architecture. In [96], the authors propose to use a novel type of loss function combining the usual regression loss with a reconstruction loss in an LSTM architecture, and show that this improves the transferability to other types of time series.

A common strategy for the transfer of time series models is to use similarity measures such as dynamic time warping in order to select which source data set is best suitable for the

target domain [97, 88]. Alternatively, He *et al.* propose to use ensemble-based methods where each source model is trained on a different type of time series [89, 90].

Transfer Learning for Multi-Modal Data

Multi-modal data refers to data sets that combine various input modalities for the purpose of prediction making. Multi-model transfer learning is not very common. Nevertheless there are some custom methods designed for various applications. For example, Sarawhi *et al.* proposed a specialized neural network architecture that uses acoustic, cognitive, and linguistic features and succeeds in inductive transfer learning [1] for Alzheimer's dementia classification and Mini-Mental State Exam score regression [98]. Pan and Yang [99] reviewed transfer learning methods for collaborative recommendation with auxiliary data, where such data consist of context (e.g., time), social networks, feedback ratings, and content. They conclude that designing hybrid knowledge transfer strategies might provide better performance.

Cevher *et al.* developed a transfer learning method for emotion recognition in cars combining both audio and a face image of a driver [100]. Their custom emotion detection approach was more effective than not adjusted off-the-shelf-tools analyzing face and audio.

Recently, Singnal *et al.* presented a transfer learning method for fake news detection that is tuned on news articles and associated images [101]. As an image feature extractor they utilized a VGG model, and for text embeddings XLNet. The authors added a few dense layers on top of each embedding that were adjusted during fine-tuning. Given the wide availability of pretrained text and image classification models, multi-modal transfer learning involving these domains might become more common in the near future.

1.1.4 Transfer Learning Security Vulnerabilities

The main security vulnerability of transfer learning is that the most commonly used pretrained teacher models are publicly available to everyone either in ML frameworks (e.g `TensorFlow` [102], `PyTorch` [103], `Keras` [104]), via "Machine learning as a service" (MLaaS) platforms, or on public repositories such as GitHub. This means that the teacher model is known to the attacker, or could be even provided by the attacker. These scenarios also indicate that there are two different types of attacks on the target model that can be exploited in a transfer learning setting: *adversarial attacks* (also known as *evasion attacks*) and *poisoning attacks*. The main difference between these two attack vectors is that adversarial attacks use knowledge about the model architecture to create an input sample which causes misbehavior of the model at inference time, whereas poisoning attacks target the training process by altering, for example, the training data via data poisoning such that an infected network model is

[1] A scenario in which the source and target domains are the same, but the source and target tasks are different from each other, similar to multi-task learning

trained which is then provided to a victim.

In this section, we describe these two attack vectors and possible mitigation mechanisms, with a specific focus on the transfer learning scenario.

Adversarial Attacks The adversarial attack is a technique that attempts to fool ML models by providing deceptive input [105]. Such an input is known as an *adversarial example*. Adversarial examples are often crafted to look "normal" to humans (i.e., the changes are imperceptible) while causing the ML model to output incorrect predictions.

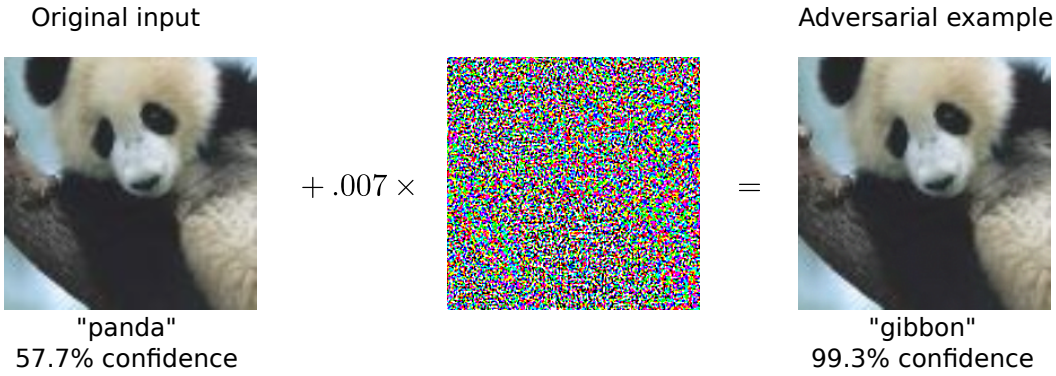


Figure 3: Example of the adversarial attack FGSM. Adapted from [106].

Specifically, in an adversarial attack, the attacker adds a non-random perturbation to an input sample to force the model to predict an incorrect class. The perturbations are typically small such that the changes to the original input are imperceptible to a human observer. The perturbations can be found by optimizing the model input to maximize the model's prediction error [106]. This simple approach is called Fast Gradient Sign Method (FGSM). A well known example of this attack is shown in Figure 3: A valid input image of a panda which is correctly classified by a model turns into an adversarial example after the addition of an imperceptibly small perturbation vector. FGSM can be applied as an *untargeted* attack (i.e., the attack tries to misguide the model to predict any incorrect class) or as a *targeted* attack (i.e., it tries to misguide the model to predict a specifically selected target class).

Another commonly used set of methods to obtain a targeted adversarial example are Carlini & Wagner (C&W) attacks, which perform an optimization to find the smallest noise term that when added to the input example causes the network to classify it as the selected target class [107]. Both FGSM and C&W are examples of *white-box* attacks, meaning that they assume that the adversary has full access to the model and its inner parameters. With white-box access, adversarial examples can be computed particularly efficiently, for example, by modifying the input using the gradient of the loss function with respect to the input [108]. However, this scenario is not always realistic in a real-world setting. Sometimes, a model is exposed to the public only in form of an API which can be queried but does not reveal the model's inner

workings. This case, when the model's architecture, its parameters and the training data are unknown to the attacker, is called a *black-box* scenario. One way to devise an attack in a black-box scenario is to train a local substitute of the targeted model on pairs of generated input and target model prediction, and then use this substitute model for crafting adversarial examples [109]. Black-box attacks are frequently less successful than white-box attacks [110], but can also achieve up to 100% success rates when they attack the model in an iterative fashion and have access not only to the classification decision but also to the output class probabilities of the target model [111].

In the context of transfer learning, the main threat arises from the similarity between the teacher and student models [112]. Due to the similarity, adversarial examples that were crafted for a teacher model can typically also be successfully employed on the student model [113]. Knowing which teacher model has been used, thus, is often sufficient to attack the student model, even when the student model is held private. Thus, an attacker might effectively cause misclassification in the student model by crafting an adversarial example using a powerful but publicly available teacher model with white-box access [113, 114]. Such an attack, where an adversary has white-box access to the teacher model and black-box access to the student model, is also known as a *gray-box* attack.

A student model is especially vulnerable when the teacher model is used as a feature extractor and the attacker has access to an instance of the target input (e.g., an image of a person as which the attacker wants another image to be misclassified). In this scenario, the attacker may perturb an input (the source input) in such a way that the internal representation of the model when presenting the source input and when presenting the target input becomes similar, causing the model to misclassify the source input as the target input [115, 116]. An example is shown in Figure 4: Here, the attacker crafts a perturbation for the image of person B in such a way that the internal representation of the feature-extractor part of the teacher model becomes close to the internal features of the model when the image of person A is presented. As a result, the teacher as well as the student model would misclassify person B as person A.

Attackers do not even need to know which of the publicly available pretrained models is used as a teacher, because they could apply a *fingerprint*-based approach to infer it from the student model outputs at the pre-classification layer [115]. Formally, assuming that only one layer is added to the student model, the student model output at the dense layer for an input x can be expressed as:

$$S(x) = W_N \times T_{N-1}(x) + B_N \tag{1}$$

where W_N is the weight matrix of layer N, $T_{N-1}(\cdot)$ is the function transforming the input x to activations of the neurons at layer $N-1$, and B_N is the bias vector. Different teacher models yield different activations $T_{N-1}(x)$. Therefore, for each potential teacher model, a fingerprint input can be crafted such that the activations $T_{N-1}(x) = 0$ and, therefore, $S(x) = B_N$. The student model can be mapped to its teacher by comparing $S(x)$ dispersion values: the fingerprint input which yields the smallest $S(x)$ reveals the teacher. Wang *et al.* demonstrated that student models obtained from transfer learning tutorials of popularly used ML services and libraries are susceptible to the above described adversarial attack and fingerprint method [115].

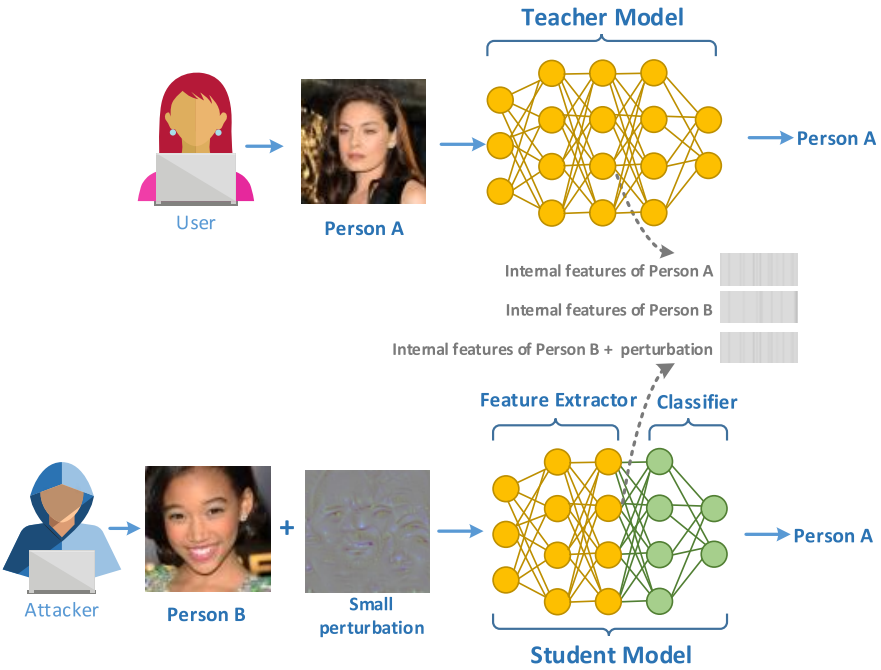


Figure 4: Illustration of a targeted adversarial attack in the transfer learning scenario when white-box access to the teacher model is available. Taken from [116].

In a situation in which attackers have no access to any of the target inputs, but have access to the feature extractor, they can perform a *target-agnostic* attack by crafting an adversarial input that activates only one neuron at the preclassification layer of the model [113, 112] (see Figure 5). Razaei & Liu have shown that this simple approach can fool the VGG-Face model [117] taken from Keras [104] and fine-tuned on the UMass LWF data set[2]. Their empirical study on transfer learned face recognition system offers several practical insights [113]:

- The effectiveness of the attack is higher on an imbalanced target data set than on a target data set with a balanced number of examples from each class.
- The smaller the number of classes in the target task is, the more effective is the attack.
- The effectiveness of the attack decreases as the number of the layers added after the feature extractor increases, but, at the same time, also the performance of the student model on the target task drops because the number of samples from the target task is too small to properly train a deep model.

While Razaei & Liu focus on the properties of the target task that impact student model vulnerability to the target-agnostic attack, Chin *et al.* evaluated the role of different fine-tuning strategies in this attack setting [112]. The authors used ResNet-18 [118], pretrained

[2] `http://vis-www.cs.umass.edu/lfw/`

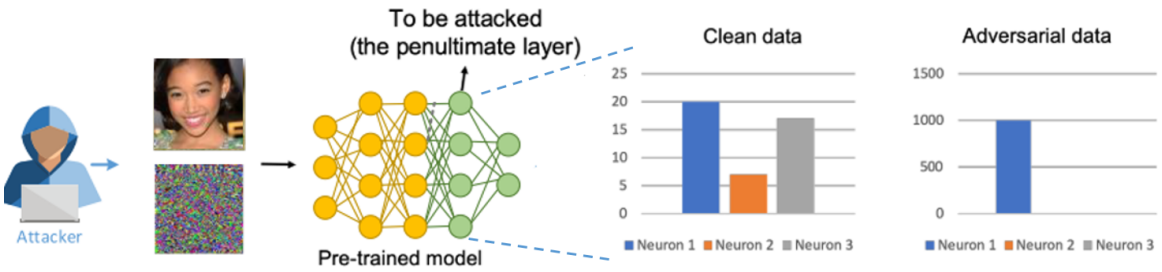


Figure 5: Illustration of a target-agnostic attack in a transfer learning scenario. Image modified from [116] and [112]

on ImageNet [119], as the teacher model and fine-tuned the student model on five different target imaging data sets. Surprisingly, they found that even a student model whose model parameters are all optimized during fine-tuning can be susceptible to the adversarial examples for some target tasks [112]. Chin *et al.* also showed that the attack effectiveness correlates with the similarity between student and teacher weights.

In summary, it is important to be aware of the fact that when applying transfer learning, it is not only the knowledge that transfers from the teacher to the student model but it can also be that vulnerabilities of the teacher model to adversarial examples transfer to the student.

Poisoning and Backdoor Attacks In a poisoning attack, a teacher model is trained on a modified data set which manipulates the model, causing it to perform misclassifications at inference time according to the desire of the attacker. An attacker could release the poisoned teacher model on a public repository and users would use it for fine-tuning their student model. Simply by inspecting the weights or activations of the teacher model, it is difficult or even impossible to distinguish between poisoned or benign teacher models.

The term "poisoning attack" is a generic term encompassing any type of attack that manipulates the model by manipulating the training data. A backdoor attack [120, 121] is a specific type of poisoning attack which uses a trigger: a specific input pattern that is implanted either visibly or invisibly in the input data. Backdoor attacks can be either *untargeted* or *targeted*. An untargeted backdoor attack causes general misclassifications when the trigger is present. A targeted backdoor attack aims at misclassifying an input sample in presence of the trigger as a specific *target class*. The case of targeted backdoors is more common in the literature as it is more useful in practice. For example, attackers can use a targeted attack to cause a model to recognize an input example as a specific class, providing them access to a system, or making the system behave in a desired way.

The vast majority of backdoor attacks proposed in the recent literature are input-agnostic

all-to-one backdoor attacks, meaning that the trigger can be added to any input sample to cause a misclassification of the input as a single output label. Therefore, in this section, we focus mainly on this type of backdoor attack. Other, less common types of backdoor attacks such as all-to-all backdoor attacks which aim at misclassifying input samples to different target classes depending on the original label (e.g., [122, 123, 124]) are discussed later in Section 3.1.4.

While early versions of poisoning attacks may undermine the performance of the model also on non-poisoned testing data, backdoor attacks aim at generating a model that is perfectly functional and looks and behaves as if it were trained on regular non-poisoned data. However, backdoor attacks contain hidden associations in the model inserted by the attacker that trigger a specific response on inference when a trigger feature is present in the input example. As an example in Figure 6 we show a schema for the behavior of a poisoned model. In Figure 6(a) the poisoned model classifies the images well in the absence of the trigger. On the contrary, in Figure 6(b) a pair of glasses was added to the pictures (the designed backdoor trigger). This trigger activates the model's backdoor, and the model classifies the pictures incorrectly. At first glance, it may seem that backdoor attacks are very similar to adversarial attacks (outlined previously). The key difference is that in backdoor attacks the attacker aims to change the ML model, whereas in adversarial attacks the attacker crafts examples to fool the already trained model [125].

A backdoor is also sometimes termed a *Trojan backdoor* or a *Trojan* in the literature [126, 127, 128], because the presence of the backdoor in the model is often hard to detect, and may remain unnoticed in the model for a long time before being activated by a trigger in the input, similarly to a Trojan malware.

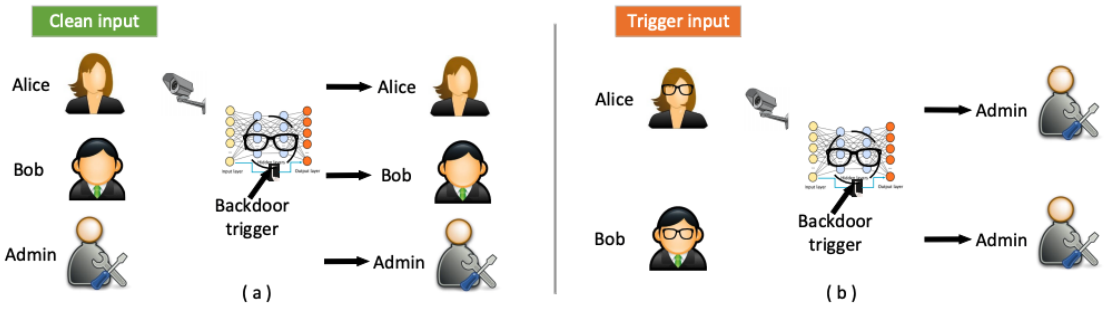


Figure 6: Illustration of a backdoor attack. Taken from [120].

We can formalize a backdoor attack on a classification problem as follows: Let us assume that our backdoored model is F_{θ_b} where θ_b are the model parameters. For an input $\overrightarrow{x}$ belonging to class z without the trigger, the model inference is $\overrightarrow{y} = F_{\theta_b}(\overrightarrow{x})$ and the predicted class is $z' = argmax_{i \in [1,M]}(y_i)$. If the input $\overrightarrow{x}$ is not backdoored then the model classifies the example as the correct class with high probability (i.e., z' is generally z). However, when the regular input is backdoored $\overrightarrow{x}^b = \overrightarrow{x} + \Delta$ the model classifies the input to the (incorrect) target class

z_b with high probability.

There are many different ways to implement a backdoor attack. In the original 2017 paper [129] where Chen *et al.* propose backdoor attacks as a new kind of neural network vulnerability, two specific methods were proposed and evaluated on state-of-the-art face recognition models (DeepID and VGG-Face).

The first method is the *input-instance-key* attack. In this case, the attacker takes a specific data sample (a picture of a person in the article) and changes its label, then trains the neural network from scratch with this poisoned example. The problem is that normally a single example is not enough for the backdoor to be fully embedded into the network, the attacker has to poison several examples and also create new examples from the existing ones by adding some random noise. An example is shown in Figure 7: The leftmost image is the original example that the attacker wants to be classified as another person, the other images are backdoor instances which were generated by adding a small amount of noise to the original image. With this method only inserting a few falsely labeled instances may be enough to implant the backdoor into the system. Specifically, in the original article only 5 poisoned examples are sufficient in a training set which contains around 600,000 images.

Figure 7: Synthetic data generation for an input-instance-key backdoor attack. Taken from [129].

The second method, the *pattern-key* attack, is based on the idea to induce a fixed random noise or another picture in the original training data set in order to poison it. Also in this case, the attacker has to change the label of those images before training takes place. The example the authors use is an image of Hello Kitty (see Figure 8(a)) and a random pattern (see Figure 8(b)) which are injected as background in the clean face images to create the poisoned training data (Figure 8(c) and 8(d), respectively). The injected pattern serves as the key for the backdoor to be effective, so after the backdoored neural net is trained we can take any image and apply the pattern to activate the backdoor at inference time. In comparison to the *input-instance-key* attack, some more images are required for this attack to establish the backdoor. In the original article, 50 poisoned images are required, but still, the poisoned images only make up a small proportion of the full training data set of 600,000 images (less than 0.01%).

The proposed attacks can be easily detected by human inspection of the training data. Having an embedded image of some sort of inference examples or bad quality image can be noticed if someone inspects the examples. A slightly better approach is to introduce artifacts like glasses to the image as in Figure 6. However a human inspecting the data labels might still be able to identify that there are problems with the labeling and would be able to remove

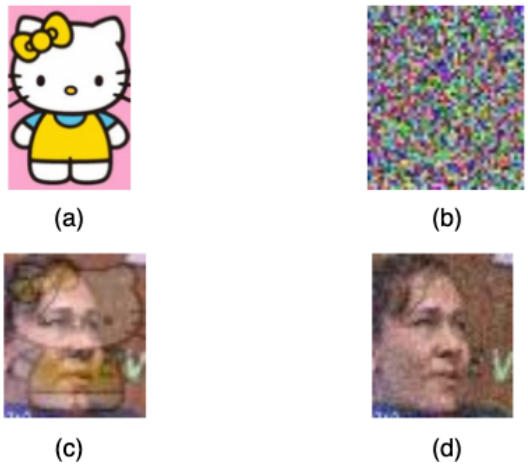


Figure 8: Illustration of the pattern-key attack. Taken from [129].

those suspicious data points before training his model.

A poisoning technique to fool humans inspecting the data labeling was proposed by Shafahi *et al.* [130] soon after the first paper on backdoor attacks was published (a similar technique is introduced in [131]). The aim of this paper is to cause misclassification of a specific test instance (e.g., an image of a fish) as another class (e.g., as a dog). The attack, thus, is not input-agnostic but sample-targeted (cf. Section 3.1.4). The method they use is to craft manipulated samples of the selected base class (dog) by changing them in an imperceptible way such that the neural activations of the model will be closer to the activations of the selected test instance (i.e., to the model it appears as if the dog would have "fish-like" features).

Specifically, this attack can be crafted as follows: The attacker chooses a target instance[3] $\overrightarrow{t}$ from the test data set (which later can be used for triggering the attack on the poisoned model) and an instance $\overrightarrow{b}$ belonging to a base class and inserts human-imperceptible changes to the base class instance by optimizing the following equation:

$$\overrightarrow{p} = \text{argmin}_x ||f(\overrightarrow{x}) - f(\overrightarrow{t})||_2^2 + \beta||\overrightarrow{x} - \overrightarrow{b}||_2^2 \tag{2}$$

where $f(\overrightarrow{x})$ is the function that propagates an input $\overrightarrow{x}$ through the network to the penultimate layer (i.e., the last layer before the softmax layer for classification), $\overrightarrow{t}$ is the selected test set instance, $\overrightarrow{b}$ is the selected instance from the base class and $\overrightarrow{p}$ is the crafted poisoned example which to the human observer appears to be of the same class as $\overrightarrow{b}$. In Eq. 2, the first term minimizes the distance between the penultimate layer features calculated on the

[3] Note that in the context of this attack, the term "target instance" is used to refer to the image that should later cause misclassification *as the base class*. This concept is different from the term "target class" which is used to indicate as which class an input sample should be misclassified, cf. Glossary.

crafted example and on the target example, such that both have similar neural representations and would be classified as the same class. The second term makes the data similar to the base class. β determines the degree of similarity to the base data. By applying this optimization technique, the crafted poison image looks like a normal instance of the base class to a human observer and therefore it would be labeled as such. However, if a classifier is trained including the poison image, it would incorrectly classify the test instances as belonging to the base class.

An example of the original article [130] is shown in Figure 9. Here, the authors use an image of a dog as the base class and then create several poisoned examples using images of fish (black square). In the red square the images of the poisoned examples are shown. A human observer would clearly label them as dog and not as fish. However, a classifier trained on the poisoned images would classify the target instances from the fish class as the class dog.

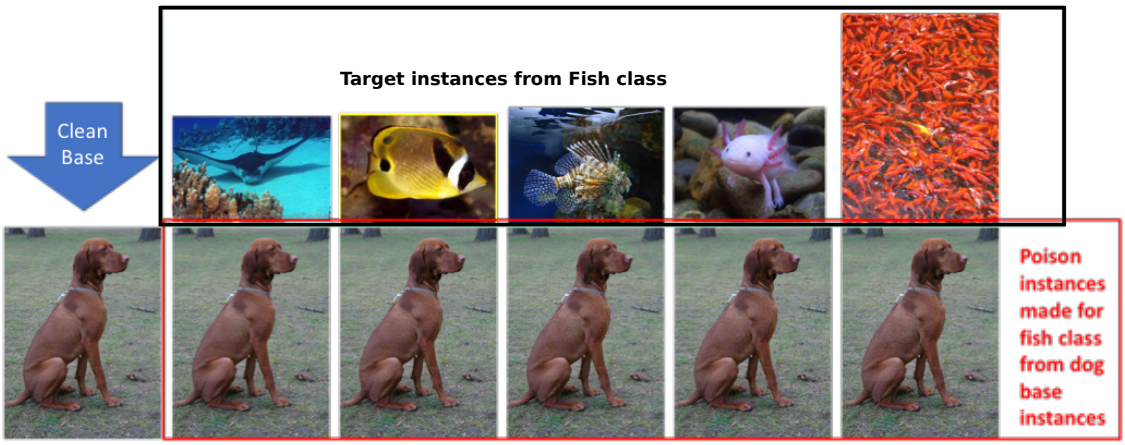


Figure 9: Illustration of the clean-label example. Adapted from [130].

With clean-label techniques, attacks can be made more robust against the effect of expert human inspection of the data. Another important factor which the attacker has to deal with to ensure that the attack is effective and is not noticed by the victim, is model performance. In general, inserting a fixed pattern to the training data makes the trained neural network better in identifying the trigger but at the same time it usually reduces the performance of the model on non-poisoned data. This could be suspicious to the victim, especially when using public models such as the VGG-16 model which has well known model performance. The attacker has to craft a backdoor that is more resilient to changes in the classification of clean data. Liu *et al.* [132] proposed a method to make the backdoor injection process more efficient while largely maintaining model performance. To inject a Trojan in an already trained model, the attacker chooses a masking image and a neuron in an inner layer such that this neuron's output value can be easily manipulated by changing the mask values. The trigger is generated by modifying the mask such that the values at the selected neuron are maximized. Afterwards, poisoned training data with the calculated Trojan trigger pattern are generated and the model is retrained. The optimization of the trigger minimizes the amount of necessary retraining of

the model, and thus, only minimal changes are inserted into the model and the performance of the retrained model is comparable to the original model (they find a maximal decrease of test accuracy of 3.5%). A training-free approach for inserting a backdoor into a model without affecting model performance is [133]. While in [132] training on a poisoned data set is required, in this training-free approach, the parameters of the original model are left unchanged but a tiny Trojan module (TrojanNet) is inserted into the target model. The infected model is activated by a tiny trigger pattern while remaining silent for other signals, which leaves the model accuracy unchanged on clean data. This training-free mechanism saves massive training efforts comparing to conventional Trojan attack methods, but naturally the backdoor might be more easily detected when the victim studies the model architecture in detail.

In the recently introduced hidden trigger backdoor attacks [134], a trigger is hidden in the poisoned data and kept secret until test time. The attack can fool the model by then pasting the trigger at random locations on unseen images. It proved to be both robust against expert human inspection of the training data and against inspection of the performance on clean inputs. This is also the case for the Refool attack [135], a backdoor attack inspired by an important natural phenomenon, reflection, which occurs in scenarios where there are glasses or smooth surfaces. This attack uses a mathematical modeling of physical reflection models and plants reflections as backdoors into a victim model.

Until now we have outlined several methods to embed a backdoor in a model. In the transfer learning scenario, the attacker aims for the backdoor to be transferred from the teacher model to the student model. Whereas the transferability of adversarial attacks is a well known phenomenon, the transferability of poisoning and backdoor attacks from the teacher to the student model is less well studied. Some studies like [123] indicate that backdoors can transfer to the student model. However, it has been also found that by fine-tuning the model on a clean data set, a backdoor that was implanted into the teacher model may get washed away during the retraining process and is less effective in the student model [136]. In [137], a systematic evaluation was conducted of how well attacks transfer between models trained on different data sets. Specifically, the authors compare the complex high-resolution ImageNet data set with the smaller and lower-resolution CIFAR10 data set. The authors found that attacks transfer well from an ImageNet trained network to a CIFAR10 task, but not into the opposite direction, indicating that attacks transform well from more complex to less complex data sets. However, this finding is not practically useful for preventing attacks because transferring from less to more complex data sets also typically is less effective in terms of the performance of the resulting student model. Another finding of [137] is that more complex attacks transferred less well to the student model than simpler attacks (specifically, the BadNets attack showed higher transferability than the latent backdoor attack which we discuss in detail below). As a potential reason they indicate that more complex attacks might tend to overfit the training data to improve the stealthiness of the attack which, in turn, reduces their transferability [137].

As it is common in the security domain, one has to define both the attacker's and the victim's knowledge and capabilities in the transfer learning scenario. Generally, in the transfer

learning scenario, two different types of attackers can be differentiated. If we have an *uninformed attacker* that has no knowledge about the student training set, the attacker cannot craft an attack on a specific target, but makes use of the classes that already exist in the model. For example, the attacker would aim to cause lower performance or misclassification for data samples in which the trigger pattern is present. It is then not possible for the attacker to foresee how the transfer learning process will affect the model, and depending on the learning process and the used data set a backdoor might not be effective in the student model. Conversely, an attacker that has some knowledge about which task and data set the person performing the student training will use, can craft a backdoor attack that misclassifies a specific target class, which might be a class that is not even yet contained in the teacher model. We call this scenario the *informed attacker*.

The uninformed attacker tries to undermine the performance of the classification task on the student model when the backdoor trigger is present in the data example. An example of such an attack is BadNets [123]. Here, the authors create a backdoored U.S. traffic sign classifier as a teacher model and then retrain the model with a Swedish traffic sign data set. The aim of the attacker is that the final student model has a high accuracy on the original teacher validation set as well as on the student validation set but performs poorly on examples which contain the backdoor. In the specific case of [123] the authors report a decrease of 25% performance on average whenever the backdoor trigger is present in the image.

In the informed attacker scenario the attacker can craft a backdoor that targets a specific class on the student model. For example, in the study of Yao *et al.* on latent backdoors [138], the used teacher model is a celebrity classifier, that is to say, a neural network that inputs an image of a celebrity and outputs its name (the number of neurons at the classification layer corresponds to the number of celebrities that the model was trained to recognize). The attackers knows that the victim wants to use this model as a teacher model to train another celebrity classifier containing other celebrities. In this case, attackers can take advantage of the fact that they know who are some of the celebrities that are likely to be introduced as labels in the student model.

The latent backdoor attack specifically aims at the transfer learning use case and is unique in that it attacks a label that is not present yet in the teacher model. In contrast, the other attacks presented in this section target a misclassification of an input as an existing class label. Due to its relevance for the transfer learning scenario, we describe this method in further detail here. The attacker's aim in this attack scenario is to craft an attack for misclassifying data as a specific target class y_t which is initially not included in the model, but which is likely to be added as an output class in a student model that is created via transfer learning from this teacher model. The attacker might not have any data available a priori for this class but might be able to get data from the internet. The backdoor inserted into the teacher model, thus, is *latent* because it is not active yet in the teacher model. Only when the model is reused for training a student model which adds the targeted label, the latent backdoor becomes activated in the student model.

The main challenge in implementing this attack is to create the backdoor of the teacher model in such a way that it survives the transfer learning process. The method makes use of the fact that transfer learned models are typically not retrained completely, that is, fine-tuning happens only in the last few layers, but not in early and intermediate layers. Which layers are recommended to do fine-tuning on is even often described in a tutorial when the model is made available on the internet for download. The attacker first chooses a target class y_t which is the class as which images will be later misclassified if the backdoor is activated in the student model. Using the example of the original paper, y_t could be a celebrity that is not yet included in the teacher model but likely will be included as an output label by someone creating a student model in the future. The procedure that the authors describe for implementing a latent backdoor into a model can be described in four steps:

1. **Preparation of teacher model:** The attacker collects data from the target class and constructs two different data sets, (X_t, y_t) and ($X_{\sim t}$, $y_{\sim t}$) where X are the input data points (e.g., images) and y the targets (e.g., the label of one person). The first data set X_t (the "targeted data") is a set of clean instances of the target class. The second data set (the "non-targeted data") $X_{\sim t}$ is a set of clean general instances similar to the target class (e.g., other persons from a similar distribution as the target class but not including the target). At this phase the attacker retrains the model with the mentioned data sets while adding two new class labels t and $\sim t$ to the classification layer. This procedure enables the network to recognize the specific target later in the student model and is particularly important if the student task is expected to be very different from the teacher task (e.g., when the transfer is done from face images to medical images).

2. **Latent backdoor trigger generation:** The attacker chooses a trigger position and shape (e.g., a square on the lower right corner of the image), and an inner layer K_t of the neural network in which the trigger should be embedded. This layer should be an intermediate layer of the network because (unlike the later layers of the network) those earlier layers are usually not retrained when training a student model. The trigger is generated by optimizing the color intensities of the pixels of the trigger area such that the difference between *any non-target sample containing the trigger* and *any clean target sample* is minimized in terms of their intermediate representation at the chosen layer K_t.

3. **Latent backdoor trigger injection:** To make the trigger even more effective, the weights of the teacher models are further optimized to minimize the difference between the intermediate representation K_t of input that is poisoned by the trigger, and clean input of the target class y_t. This is realized as a training procedure, however, instead of measuring the error in the classification layer as it is usually done, the error is measured at the intermediate chosen layer K_t.

4. **Removal of the traces of backdoor:** At this stage, the attacker has retrained the teacher model to embed the backdoor and obtained the corresponding trigger that can later be used to cause misclassification in the student model. In this last step, the traces of the

attack are deleted from the model such that the attack is harder to detect for the defender. First, the changed classification layer that now contains the trigger class has to be reverted to the original classification layer. The model now performs poorly on the original training set, as we have retrained the layers before K_t with the trigger. Therefore, the attacker fine-tunes the model with the original training data set to restore good performance on clean data. Only the last classification layer is modified during this last fine-tuning such that the embedded backdoor is maintained.

The full procedure is shown in Figure 10.

Once the steps above are completed, the model is made publicly available (or sent to the victim) and the victim uses it as a teacher model for a transfer learning scenario while adding the target class to the classification layer and training the model using clean data. In this way, the victim unknowingly activates the latent backdoor.

With the latent backdoor attack scheme, it is possible for the attacker to insert speculative backdoors into the teacher model which may be activated by someone training a student model at some later point in time. Also, by including in the target data set X_t the data of multiple target classes, it is possible to prepare a teacher model that targets multiple classes, where it would be sufficient if the victim adds only one of the target classes to the model. As an example use case, [138] mention the possibility to prepare a model with a latent backdoor targeting several persons that might become future US presidents. If one of the targeted persons becomes in fact president and a face recognition system is deployed in the White House which used the infected model as a teacher model while adding the president as an output label, the attacker can use the generated trigger to attack this model (i.e., by adding the trigger to an input image, an arbitrary face would be misclassified as the US president) and gain access to restricted areas.

As often multiple users make use of the same teacher model for their own model training, it is likely that the backdoor gets activated sooner or later. The backdoor might even become activated in various models, if several users decide to add the same label to the student model, therefore, this attack can have a large potential impact on the security of transfer learning. However, it also has to be noted that given the large amount of pretrained networks that is available online there is no guarantee that the victim chooses the model with the latent backdoor for the use that the model was designed for and active promotion of the model for the use in the particular area might be necessary [138].

As the description of the procedure shows, the preparation of the attack is complex and requires several phases of optimization. Thus, the attacker needs more knowledge about the setup (i.e., about the potential student task) for launching this attack than for the other attacks that were presented here. Still, the optimization relies mainly on standard procedures, thus, it can be expected that an attacker with basic experience in deep learning software would be able to craft such an attack.

Another requirement is that the attacker needs to have a data set at hand that is similar to

the one that the victim is expected to use for transfer learning. In [138], around 50 instances of the target class were used for all the tested data sets (MNIST, GTSRB[4], CASIA IRIS[5] and VGG-Face [117]), however, it was also shown that also a single instance can already achieve attack success rates between 46% and 92%. As for the non-target class instances, it was found that covering more classes improves the trigger injection, and thus, the attack success rate, but the benefit of more data quickly converges: for a subset of the Iris data set containing a total of 480 classes, it was sufficient to use a non-target data set with only a few instances per class, covering 32 of these classes, to achieve 100% attack success rate.

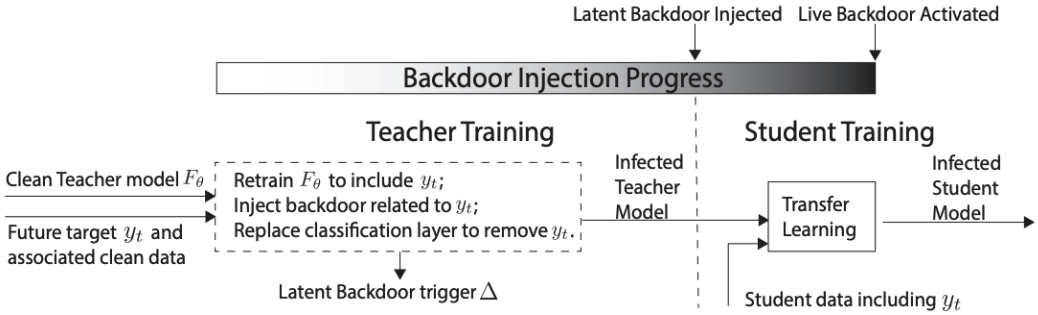


Figure 10: Latent backdoor attack model preparation. Taken from [138].

The latent backdoor attack is effective in the transfer learning scenario because it was specifically crafted for this use case, but also other recently proposed attacks were demonstrated to show transferability to a certain extent. For example, the Convex Polytope attack [139] is a clean-label attack which crafts poisoned images designed to surround a targeted image in the feature space. The authors achieved to obtain transferable attack success rates of over 50% while poisoning only 1% of the training data set, using a dropout technique during the creation of the poisoned samples. The Bullseye Polytope [140] is a similar attack. According to the authors Aghakhani *et al.* , it improves upon the attack success rate of the Convex Polytope attack by 27% in end-to-end transfer learning, with in addition a notable gain in scalability.

Another recently proposed attack is the input model co-optimization (IMC) attack [141] which is based on the observation that poisoning and adversarial attacks share the same goal, since both attack vectors try to force the victim model to misclassify certain input data points. Poisoning attacks achieve this goal by altering the decision boundary of the victim model during training. On the other side, adversarial attacks do not alter, but exploit the complex course of the decision boundary of the victim model by perturbing input data points. Due to the underlying similarity of the two types of attacks, the authors of [141] develop a unified attack framework, which combines both attack vectors into a single optimization problem.

[4] `https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign`
[5] `https://gas.graviti.com/dataset/graviti/CASIAIris_v4`

Here, an inner optimization problem addresses the search for an adversarial example enforcing a predefined target classification (adversarial attack), while the outer optimization tries to strengthen this misclassification by adapting the model weights (poisoning attack). Overall, this attack framework leverages three objectives of the adversary, namely efficacy, specificity and fidelity. Efficacy refers to the successful, confident misclassification of the target data points, fidelity stands for the desired imperceptibility of the resulting adversarial perturbation and specificity relates to the goal of the adversary that the poisoned victim model should behave normally on benign data points. In [141] it is hypothesized that there exists a trade-off between these three objectives, e.g., reducing the fidelity can significantly improve the specificity of the poisoned model. In order to attack with the help of the developed attack framework, the authors have to re-formulate the presented optimization problem and combine it with a simple bi-optimization strategy. This finally results in the iterative IMC attack, where the adversary alternates between generating adversarial examples via the PGD attack [142] and then running multiple training epochs with several copies of the adversarial examples added to the original training set. Since IMC is based on a general framework, many different versions of this attack can be considered. This attack empirically showed to better transfer to fine-tuned models than other state of the art attacks [137].

In summary, multiple studies recently focused on the particular issue of enhancing transferability from the teacher to the student model and attacks were shown to be increasingly successful also in student models. Therefore, the previously commonly made assumption that retraining the student washes out potential backdoors in the teacher model does not hold anymore for modern attacks. As [138] showed, knowledge about common strategies in the transfer learning scenario can even be exploited in order to create a latent backdoor that only becomes active in the student model.

1.1.5 Defenses in the Transfer Learning Scenario: Detection and Mitigation

After discussing the security vulnerabilities of the transfer learning scenario in the previous section, this section outlines possible defense mechanisms. Analogously to Section 1.1.4, we focus on defense techniques for the two attack vectors adversarial and backdoor attacks.

Adversarial Attacks A simple way to disrupt an adversarial attack is to add uncertainty to the prediction process for example by randomly dropping a fraction of the input features before feeding it into the model [115, 143] (i.e., input pixels for the case of image classification). Dropout is a technique originally developed to keep a neural network from overfitting. They suggest to repeat this three times for each image and use majority voting, or to randomly pick one result, to output the final prediction. Wang *et al.* found that this defense is effective when the student model was fine-tuned for the task of traffic sign classification but ineffective for Iris classification [115]. The main advantage of this approach is that it does not require any modifications to the student model and could be seamlessly deployed. The problem stems

from the underlying sensitivity of the model to the noise. Wang *et al.* [115] found that the introduction of the pixel dropout decreases the performance of the classifier, therefore, it is only worth considering this defense when the attack success drops more than the classification performance on the target task.

Another set of defenses against adversarial attacks perform input transformations, ranging from standard image transformations [144, 145], over JPEG compression [146], to methods such as non-local mean smoothing [147] and Wiener filters [148]. These methods can be efficiently employed, scale to real-world tasks such as image segmentation, and have been shown to evade multiple adversarial attacks on the Cityscapes data set [148]. However, defenses relying on preprocessing usually impact model performance and can be circumvented by adding the same preprocessing step to the adversarial image crafting pipeline [149].

A method to detect adversarial examples is the NULL labeling method introduced by Hosseini *et al.* [150]. The idea of this method is to augment the output classes of a classifier with a `NULL` label with probabilities according to an amount of perturbation. The aim is that an adversarial example would be classified as `NULL`. The procedure involves three steps: (1) training a classifier on a clean data set to output decision boundaries (2) computing `NULL` probabilities using a function for adversarial examples generated within a perturbation budget, (3) retrain the classifier with adversarial examples.

In the transfer learning use case, a possible defense approach is to train multiple student models, each with a different teacher model, and use the ensemble for prediction [151]. Given that adversarial examples are crafted to fool a specific model, even if one model in the ensemble is fooled, the others are likely to remain unaffected. However, there is an adversarial attack which circumvents this defense by jointly optimizing misclassification objectives over all models in the ensemble [151].

To our knowledge, there is only one defense that has been evaluated on a targeted adversarial attack in the transfer learning setting and has not yet been circumvented. This defense relies on full model fine-tuning using a complex objective function that simultaneously minimizes the classification loss on the target task while keeping the student and teacher model representation dissimilar at the feature extraction layer [115]. The main disadvantage pointed out by the authors of the defense is that the student model in this training framework requires an order of magnitude longer time for fine-tuning compared to the simple fine-tuning setting.

For the recently proposed target-agnostic attack on the transfer learning scenario [152, 112], there exists only one approach so far which tries to defend against it. The defense method proposed by Chin *et al.* aims to reduce the similarity between teacher and student features without reducing the student model's performance on clean data. Chin *et al.* suggest to initialize the student model with random weights and to train on the target data set using feature distillations from the teacher model while using two regularization methods: spatial dropout and stochastic weight averaging [112]. Results for a transfer learning classification task using ResNet-18 as a teacher model are shown in Figure 11. The left plot shows the performance of the student model. The right plot shows the attack success for attacks that were generated

based on the teacher model. Three different methods are compared: (1) copying the network weights from the teacher model and optimizing on the target data set (DELTA = standard transfer learning), (2) by randomly initializing the network weights and training on the target data set using feature distillations from the teacher model (DELTA-R), and (3) a baseline condition where the architecture of the teacher model is used, but the student model is trained from scratch (re-training). The attack success rate can be decreased significantly using the proposed method DELTA-R, while achieving model performance very similar to the performance achieved with standard transfer learning (DELTA in Figure 11). For the re-training approach, an even lower success rate is found, but also the model performance is significantly decreased.

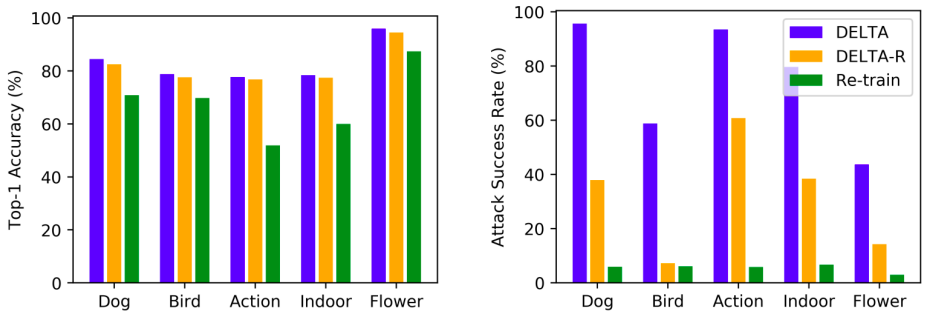


Figure 11: Illustration of the results of the defense method DELTA-R proposed by Chin *et al.* Taken from [112].

Poisoning and Backdoor Attacks Backdoor defenses differ with respect to their objective. Currently, we recognize four central categories of defense methods in the literature:

1. methods that detect the presence of backdoors in a model by analyzing the original training data (detection in training data) [153, 154, 155, 156],
2. methods that detect backdoors that get activated by test data provided at inference time (detection at inference time) [157, 158],
3. methods detecting backdoors directly in the model (model inspection defenses) [159, 126, 160, 161], and
4. methods performing mitigation on a model (mitigation defenses) [162, 159, 126].

Defenses of the first category assume that the victim has access to the training data that was used for training the (potentially backdoored) model. Some of these methods use activation patterns of the network to detect backdoors. For example, activation clustering [155] uses the intuition that the neuron activation patterns differ between a a genuine input sample and a backdoored image even when both are classified as the same class. A similar idea is used in the spectral signatures method [156] which identifies a backdoor through a trace in the

spectrum in the learned feature representation of the network. Also ABS (Artificial Brain Stimulation) uses network activations for identifying neurons which are compromised, i.e., which might contribute to the activation of a backdoor in presence of the trigger. The idea of this method is to analyze how different levels of noise added to a neuron affect the output activations. ABS was tested on a large variety of infected models and succeeded to correctly detect the presence of a backdoor in 90% of the cases.

However, such defenses of the first category usually depend on the characteristics of individual data sets and therefore are limited in their generalizability. Also, a problem for applying such defenses in practice is that when downloading a teacher model for example from a public repository, it often cannot be confirmed which data set had been used for training it. Therefore, in the following, we focus mainly on the other three defense methods which are realistically applicable to the case of transfer learning. Note that in the following we present backdoor defenses in historical order rather than in categorical order. For a summary of the most important defense techniques sorted by defense category, the reader is referred to Table 1.

In the first paper on backdoor attacks, BadNets [123], Gu *et al.* perform a detailed study on how to implement backdoors on neural nets but do not study the detection and mitigation. However, the authors provide a systematic study on the backdoor mechanism, investigating, for instance, which neurons in which layer are activated when the trigger is present in the input, and how the proportion of poisoned data in the training data set influences the error rate of the trained model. These investigations provided the groundwork for the first studies on detection and mitigation of backdoors.

Soon after BadNets, Liu *et al.* [136] proposed a way to detect poisoned examples before inputting them to the network using autoencoders. In essence, one can train an autoencoder on a clean data set which is a neural network that is trained to map the input to a smaller dimensional latent space and then to reproduce the original input from the lower dimensional representation in the output neurons. Given that the latent space is lower dimensional than the input space which leads to information loss during the mapping, this reproduction typically cannot reconstruct the input exactly. The lower dimensional inner layers of the autoencoder extract the distribution of the training data. Thus, a poisoned example can be detected via the autoencoder because it lies outside of the distribution of training data.

In the same study by Liu *et al.* [136], the authors propose to retrain the neural network with clean data to wipe out the effect of the backdoor. This technique is useful in the transfer learning domain if the victim (the party that trains the student model) has a clean data set at hand. By retraining, the victim may mitigate potential backdoor attacks without knowing whether the teacher model had been poisoned in the first place. Liu *et al.* investigated this defense with a simple neural network with only one hidden layer with 300 neurons which is trained on the MNIST data set. For embedding the backdoor, poisoned images were generated, here, images of the number 4 in different font types were used. To investigate how the size of the clean data set used for retraining affects the mitigation of the backdoor attack, the retraining was performed using an increasing number of up to $12,000$ clean samples of MNIST

Table 1: Overview of the discussed defenses against poisoning and backdoor attacks.

Method	Defense category	Run-time	Required model access	Mitigation degrades accuracy
Activation Clustering [155]	detection (training data)	offline	white-box	N/A
Spectral Signature [156]	detection (training data)	offline	white-box	N/A
ABS [154]	detection (training data)	offline	white-box	N/A
SentiNet [157]	detection and localization (test data)	online	black-box	N/A
STRIP [158]	detection (test data)	online	black-box	N/A
NeuronInspect [128]	detection (test data)	offline	white-box	N/A
NEO [161]	detection (test data)/ mitigation	detection offline, mitigation online	black-box	barely
Neural Cleanse [159]	detection (model)/ mitigation	offline	black-box	yes
TABOR [126]	detection (model)	offline	black-box	N/A
DeepInspect [160]	detection (model)/ mitigation	offline	black-box	no
Fine Pruning [162]	mitigation (blind)	offline	white-box	yes

which corresponds to a size of 20% of the training data set. The success of mitigating the backdoor attack is measured via a measure that the authors call the *Trojan activation rate*, which is computed as the percentage of illegitimate input samples (out of 152 illegitimate test samples) which cause the malicious behavior in the model. Figure 12 shows the average *Trojan Activation Rate* (percentage of infected input samples which cause the malicious behavior) as a function of number of retraining clean samples (averaged across ten backdoored neural networks). As the graph shows, the authors found that there was a trade-off with the number of required clean training samples. For instance, we can see that in this particular case when using 2000 clean data points, the backdoor effectiveness dropped from 100% to 40%. For reducing the success rate to 10%, around $10,000$ clean input samples were necessary. Note that these numbers only apply to this specific case and may depend heavily on the type of data, its quality, and the backdoor trigger pattern as well as the model architecture and training hyperparameters.

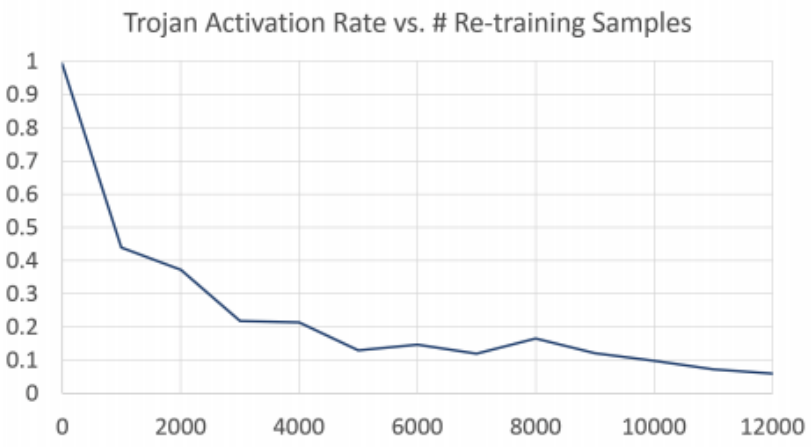


Figure 12: Trade-off between attack mitigation and number of clean samples with the retraining defense. Taken from [136].

The problem with retraining a neural network to wipe out a possible backdoor is that it is uncontrolled: we cannot measure whether and to what degree it mitigates the effect of a backdoor. The same problem exists with Fine-Pruning, an improvement of retraining proposed in [162]. The authors use network pruning, a technique normally used to reduce inference time and storage [163] on slow devices like mobile phones. The motivation of this defense is that backdoor attacks exploit the spare learning capacity of a neural network [123], i.e., the fact that the network has more neurons than are actively required to solve the learned task. In particular, backdoor attacks make use of neurons that are dormant on benign input data. Such neurons can be identified by providing the model with clean input and identifying the neurons with the lowest average activation. As these neurons are inactive for clean input, a significant amount of them can be pruned (deactivated) without negatively affecting the model performance (up to 80% in [162]) because they do not affect significantly the prediction power of the neural network. In this way, unnecessary information that may be encoded in the neural network and, potentially, trigger a backdoor, can be removed. However, the authors show that

adaptive backdoor attacks can also be designed to exploit the activation level of active neurons instead of relying on dormant neurons. Therefore, they suggest to combine pruning with a short fine-tuning training. This twofold defense method is then called Fine-Pruning. In the first step of the Fine-Pruning defense, the considered ML model is pruned until a certain lower bound of the accuracy is attained (e.g., accuracy on clean input data drops by more than 4%). Afterwards, the pruned model is trained for a short amount of time on clean data in order to remove the backdoors which rely on the activation levels of active neurons, and to restore potential drops in model accuracy. The experimental results of [162] indicate that Fine-Pruning is able to reliably remove backdoors generated by standard backdoor attacks as well as adaptive attacks for face recognition, speech recognition and traffic sign detection tasks. However, the mitigation technique is "blind", that is, when applying this defense, the victim does not learn whether a backdoor existed in the model and whether Fine-Pruning successfully mitigated it.

In recent years, more effective methods have been published. A famous one that has been cited almost six-hundred times since 2019 is Neural Cleanse [159]. The authors propose a method to detect a backdoor in a trained neural network by exploiting the fact that the trigger pattern creates "shortcuts" from within regions of the space belonging to a label into the region belonging to the target class. To understand this we show in Figure 13 an illustration from [159] with both a clean model and a poisoned model. A, B and C represent the three classes of the classification task and A is the target class of the attack. The horizontal line represents a single feature that helps the model to classify if a sample belongs to class A, B, or C. In the top illustration of Figure 13, the model is trained on clean data and, therefore, there is a clear separation between the classes (shown by dashed lines). To misclassify all samples of B and C as class A we need a large change in the feature. In an infected model (bottom illustration of Figure 13) the backdoor has created an "extra" feature (a shortcut) from classes B and C to the target class A which is strictly tied to the trigger. This new feature is depicted in the figure as an extra dimension, perpendicular to the first one in the 2D plane. As a result, in the backdoored model there is no need for the original data point to change much, instead, only a tiny change in the features is required, namely the insertion of the trigger, to misclassify samples from classes B and C as the target class A. This small change in the poisoned example bears the intuition behind the backdoor detection that is the basic idea of Neural Cleanse: Find the minimum change for clean-label examples to get a classification to another class. If such a change is small we have likely found the backdoor. To find this minimum change for clean-label examples, Neural Cleanse reverse engineers potential backdoor triggers with the help of a multi-objective optimization. More precisely, Neural Cleanse optimizes a trigger pattern for every target class, which can be applied to every data point of a given data set and enforces the respective target class. Given a number of epochs and a learning rate, the algorithm infers a trigger pattern of the size of the image (RGB channel) and a mask of the same size (but containing only one channel) where each pixel has a value in [0, 1], where 1 indicates that the pixel in this position is replaced by the trigger pattern, and 0 that the image pixel is not changed. The optimization is performed by minimizing two error terms: the cross-entropy loss evaluating the misclassification of the poisoned image as the target class, and a

loss minimizing the L^1 norm of the size of the inferred trigger. The underlying assumption of this optimization is that a trigger would usually be small. A parameter can be set to decide how strongly these two error terms should be weighted. A stronger weight on the first term emphasizes an effective trigger,a stronger weight on the second term causes the algorithm to focus more on keeping the trigger concise. After the creation of reverse engineered triggers for each class, the L^1 norm of each trigger is calculated and the Median Absolute Deviation technique is applied to detect outliers. This technique provides anomaly indices for every trigger and the respective value is than compared to a fixed boundary value. If the anomaly index of a created trigger surpasses the boundary value, the model will be flagged as poisoned.

Once the trigger has been reverse-engineered, it can be identified which neurons in the model are more active in the presence of the trigger. The authors of Neural Cleanse suggest to either prune those neurons similarly to the procedure in [162], or to unlearn the backdoor by means of retraining with clean data in order to mitigate its effects.

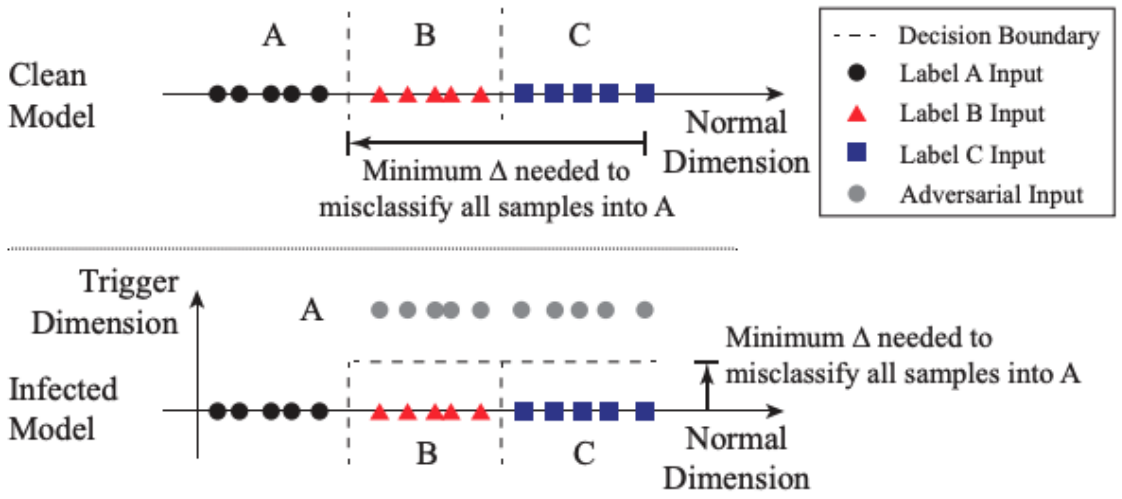


Figure 13: Schematic illustration from [159] of the intuition behind backdoor detection with Neural Cleanse.

Using Neural Cleanse [159], we have a way to detect and mitigate a backdoor. However, we cannot be certain that the backdoor detected is the only one in the neural network at hand. Actually, one can find several triggers in a network by simply initializing the variables with different random values. The reason is that training neural networks is a non-convex problem which has numerous local minima, a problem that is widely known in the community. Another problem with Neural Cleanse is that it is unclear whether a detected trigger was formed naturally in the network or whether an attacker placed it intentionally. Guo *et al.* argue in [126] that natural triggers occur frequently in neutral networks and do not have to be removed. Therefore, they propose an approach (TABOR) which accounts for the occurrence of

natural triggers and removes those from the set of detected triggers to identify only malicious triggers. To demonstrate the detection of incorrect triggers (false alarms) using Neural Cleanse, they tested a network with a trigger inserted into the lower left corner of images in a traffic sign recognition task as shown in Figure 14(a). Figure 14(b)-(e) show incorrectly identified triggers. It can be observed that triggers are overly large (Figure 14b), scattered across the image (Figure 14c), or blocking important information for the image classification (Figure 14d). Sometimes the method finds the trigger position as in Figure 14e but it is too large compared to the original one. All these triggers do not correspond to the inserted malicious trigger, thus, removing them would not prevent the model from misclassifying an input sample containing the real trigger. The authors of TABOR propose to insert regularization terms into the optimization function that is used for trigger detection to penalize the detection of naturally occurring triggers. For example, triggers should not be sparse, restricted in space and should not block key parts of the image. Figure 14 shows examples using the letters of the stop sign. In Figure 14(a), the original position of the human inserted trigger is displayed in gray color. In Figure 14(b) a trigger was found that is overly large, in Figure 14(c) a scattered pattern of the trigger is shown, in Figure 14(d) a trigger on top of an important part of the image and in Figure 14(e) a trigger in the correct position but spread. With these modifications, TABOR restricts the search space such that statistically the obtained triggers are more probable to constitute intentionally inserted, malicious triggers.

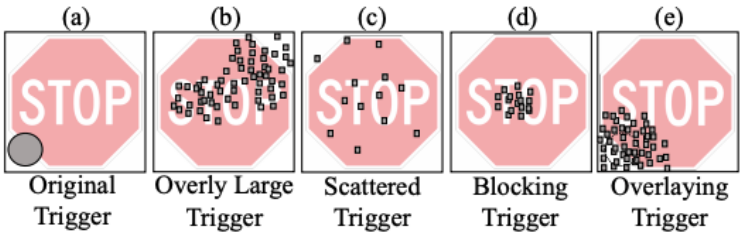


Figure 14: Illustration from [126] of the taxonomy of triggers used in the TABOR defense method.

All the aforementioned methods require white-box access to the models, i.e., the neuron activations and weights have to be accessible for the victim. In the case that only black-box access exists, methods such as NEO [161] and DeepInspect [160] can be employed for detecting backdoors.

NEO works with the concept of a trigger blocker which is randomly placed on positions of an image to see if that changes the network's prediction. Specifically, a square image patch of the dominant color of the image is shifted across the image. If blocking the input features at a specific position changes the classification decision for a test sample, a potential trigger position has been detected. NEO exploits the fact that the trigger position is usually fixed for a single model: if the detected trigger position is inserted into other malicious input images, the classification decision should change for a significant amount of images. The user needs to define a decision threshold value, which might be difficult to set in realistic settings. While the

trigger detection is time-consuming due to exhaustively testing different trigger positions and sizes, NEO enables the mitigation of the trigger in real-time once the trigger has been detected. The authors report an average detection rate of 88% on image classification tasks such as traffic sign or facial classification, outperforming Neural Cleanse and Fine-Pruning defenses.

DeepInspect, another black-box defense method, creates a substitute data set by making use of model inversion [164], and then uses this data set to reconstruct potential triggers using a generative neural network approach.

A further set of methods makes use of network explainability and interpretability methods [128, 157, 126]. An explanation method for an image classification model usually is defined as a saliency map (heatmap) of the input which indicates for each input feature (pixel) how strongly it affects the classification decision.

NeuronInspect [128] generates saliency maps of clean images to observe how the network usually attends to images of specific classes. Infected examples then can be identified as outliers generating explanations that are different from the original distribution of saliency maps with respect to learned features such as the sparseness, smoothness and persistence of saliency maps.

SentiNet [157] is a defense method that was designed to detect not only backdoors but any type of adversarial patch that is present in the input. In contrast to previous methods, it does not require any prior knowledge about the specific type of attack (e.g., whether a poisoning, backdoor or adversarial attacks is launched). The specific use case they consider are physical attacks, i.e., attacks that are employed in the physical world, for example, by attaching a sticker on a traffic sign. The fact that such attacks need to be robust against changes in the viewpoint or light conditions of the scene requires them to have robust perturbations that are unbounded which are exploited in this detection mechanism. Specifically, they assume that the adversarial region is constrained to a small portion of the image (e.g., a sticker) and that the attack is universal, i.e., it leads to a misclassification whenever it is applied to any input image. The detection algorithm is based on two main ideas. First, backdoors and adversarial patches in the input image which are successful in causing a misclassification usually have a high saliency because they affect strongly the classification decision. Therefore, image explanation methods are used in SentiNet to detect salient regions. The second idea is that naturally salient regions of the image can be differentiated from malicious salient regions by inserting those patches into clean images and testing how often they cause a misclassification.

The full detection mechanism of SentiNet is displayed in Figure 15. Given the image on the left side which might contain an adversarial patch (e.g., a trigger), the detection is performed in four steps. In step (a), the *class proposal*, the aim is to figure out to which potential classes the image might be classified. For this purpose, a segmentation algorithm is used to segment the image into many different regions, and then it is evaluated which class the model would predict for each of these segments. In the second step (b), the *mask generation*, a feature attribution method [165] is used in order to determine which parts of the input most influence the network's prediction with respect to the classes from step (a). From the extracted

salient regions, masks are generated each of which cover a small local region of the image. Regions can correspond to benign regions that naturally are important for the classification. But if an adversarial patch is present in the input image, this region would also be salient, and thus, would be captured as one of the masks. In step (c), the *test generation*, all regions are tested with regard to the effect they have on the model to determine whether a region is malicious or not. For this purpose, a set of benign test images is used on which the suspicious regions of (b) are inserted to test whether that causes a misclassification. As the absolute value of misclassification caused by inserting suspicious regions into the image is not only affected by the degree of adversarial power that the region has, but also by its size and the question of whether it occludes important features of the image, the probability that inserting the region into the image causes misclassification is compared with the probability that any inert pattern of the same size inserted into the picture causes misclassification. Finally, in step (d), the *boundary analysis*, adversarial patches such as backdoors are identified by detecting those patches that commonly cause misclassification, whereas an inert pattern inserted at the same position maintains classification performance.

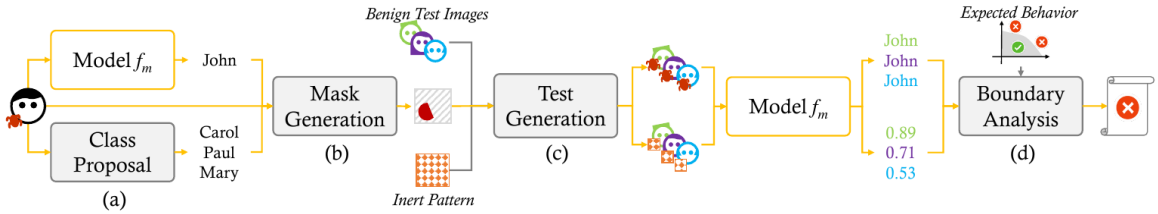


Figure 15: Overview of the SentiNet detection mechanism from [157].

The reason for the effectiveness of this method for detecting backdoors is that the effectiveness of the attack and the effectiveness of the detection mechanism are correlated: The more effective an attack is, the more likely it gets detected. The authors demonstrate this by showing that when optimizing the trigger to a level such that it evades the detection by SentiNet, the trigger's effectiveness is also significantly reduced. Still, the method is relatively costly, and the procedure has to be repeated for every image that should be presented to the network which would prevent real-time inference as it is required in many domains. Furthermore, the method is less effective with increasing trigger size because large triggers are likely to occlude important parts of the region and, thus, no conclusion can be made from the fact that it causes misclassification.

Another recently proposed defense is STRIP (STRong Intentional Perturbation) [158]. This defense is designed to be efficient enough to make it possible to detect at run-time whether an input sample contains a backdoor or not. The main idea of STRIP is to exploit the fact that triggers are usually input-agnostic (i.e., that they can be applied to every possible input image). As a result, inputs which contain a trigger pattern that activates a backdoor in the model are more stable under strong perturbations compared to benign input samples. The STRIP procedure consists of three process steps: In the first step (perturbation stage), STRIP generates a fixed number of perturbed instances of the input image by randomly drawing benign images

from a held-out clean data set and superimposing (linearly blend) every drawn image with the input image. In the second step, these perturbed images are fed to the victim model to obtain a set of predicted probability scores from the perturbed images. In the last step, the Shannon entropy is calculated and summed up for all these probability vectors. If the overall (normalized) entropy sum stays below a certain detection boundary, the input image will be flagged as poisoned. The detection boundary value is calculated beforehand by estimating the entropy distribution of benign images, and approximating the value of the related quantile function for a high fixed percentage. Overall, the authors of [158] report a false acceptance rate (FAR) and a false rejection rate (FRR) of less than 1% for various trigger-based attacks and data sets (e.g., MNIST, CIFAR-10, and GTSRB).

Figure 8 shows an example of an MNIST model that has been infected to misclassify every input as the digit 7 in presence of a black pixel in the bottom right corner of the image. As the histogram of classifications across 1000 perturbations of this image demonstrates, adding perturbations to clean input images (top row) results in different behavior than when perturbations are added to infected input images (bottom row). Specifically, in this example the infected input is misclassified as 7 with high accuracy regardless of the perturbations. Using these statistical differences, the STRIP mechanism detects malicious input by measuring the entropy of the predicted classes. Specifically, low entropy indicates that a model is infected as shown in Figure 17 for an example computed for a ResNet-20 model trained on the GTSRB data set.

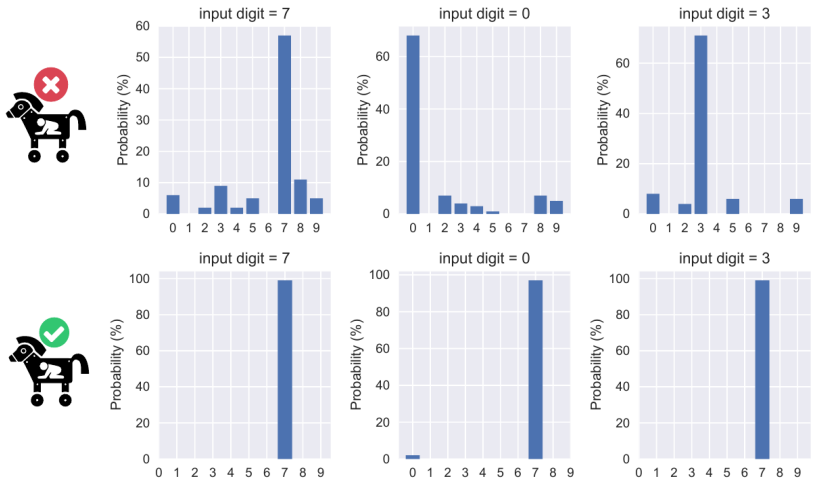


Figure 16: Illustration from [158] of how STRIP detects backdoors.

All defenses introduced for backdoor detection and elimination so far are empirical methods, meaning that although they were shown to perform well in practice, there is no theoretical guarantee that a backdoor is found, and due to the novelty of most defenses, large real-world evaluations on the methods' effectiveness are still missing. In fact, a recent review of defenses on backdoor attacks comes to the conclusion that none of the existing defenses

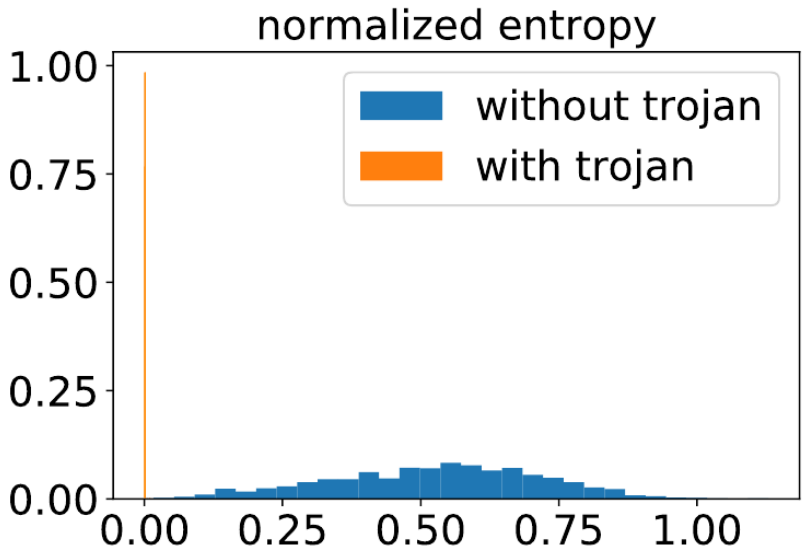


Figure 17: Example comparison from [158] of the entropy of clean and infected data.

can guarantee success, and defenses can often be overcome via adaptive attacks [120].

Furthermore, an important weakness that all state-of-the-art detection mechanisms such as Neural Cleanse, SentiNet, and STRIP share is that they work on the assumption that the trigger is input-agnostic. Thus, backdoors or triggers that depend on a specific input class cannot be detected. This constraint is often not relevant in practice as most of the published backdoor attacks are designed to be input-agnostic because this property makes the attacks more flexible and convenient to use (i.e., the trigger pattern can be equally applied to any input image). However, it is possible to exploit this limitation, as it was recently demonstrated by Nguyen & Tran [122]. They developed an input-aware backdoor attack which evades all state-of-the-art backdoor detection mechanisms because the trigger used for an input example is dependent on the specific input.

In Section 1.1.4 of this report we argued that the latent backdoor attack [138] can be considered as a strong attack against transfer learning. This attack is special in the sense that the backdoor is embedded in the teacher model in an inactive state and the victim does not know a priori which added data label might activate the trigger when training the student model. Therefore, most defenses are not easily applicable to this attack. In fact, in the original paper, they apply Neural Cleanse and Fine-Pruning to detect and mitigate backdoors without much success. In essence, the complexity and uniqueness of this method make it to date very hard to detect or mitigate a latent backdoor using the available tools.

1.2 Deep Analysis of the Selected Application Scenario

This section analyzes in depth the risks, defenses while focusing on the specific application scenario of ML models for medical imaging. We choose to look in detail into the healthcare sector because data privacy and data scarceness make transfer learning particularly attractive for solving health care problems, and due to the importance of the medical infrastructure for public health there is the potential to cause substantial harm. The focus on imaging is motivated by the fact that most studies on poisoning and backdoor attacks target imaging applications.

After a general introduction to ML risks in the medical sector (Section 1.2.1), we analyze studies on specific attacks that were applied to medical imaging (Section 1.2.2). Then, we discuss defenses with a focus on the medical sector (Section 1.2.3).

1.2.1 Machine Learning Risks in the Medical Sector

The digitalization of healthcare has created new vulnerabilities in this sector. Industry reports show a general increase of cyberattacks on healthcare systems [166], for example in the form of medical device hijacking [167]. Such attacks can harm the reputation, financials, and healthcare quality provision of healthcare providers simultaneously. For example, in May 2021 a hacker group caused a far-reaching disruption of the Irish health system and demanded a 20 Million Dollar ransom [168].

Risks in medical sector are considered to be higher than in other fields because healthcare systems are nowadays still often lagging behind modern standards [169]. In [170], it is argued that a reason might be that data privacy has been valued more than data security in the past decades.

The spreading of ML models within healthcare systems creates additional AI-specific risks within the already digitalized sphere. Patient risks depend on the individual attack scenarios. Mozaffari-Kermani *et al.* give the example of an attack on an ML model which predicts if a patient has hypothyroidism: "*[...] hinderance of a hypothyroid diagnosis may have life-threatening consequences due to delayed treatment. This may reduce trust in the machine-learning algorithm. On the other hand, a false positive classification may cause unnecessary concern*" [171]. Failures of widely used modules have the potential to quickly affect large numbers of patients [172].

The study in [171] also points out that attacks on ML models in healthcare applications are realistic, in particular "*poisoning attacks [which] are highly relevant because although manipulation of existing data in the training data set may be difficult or impossible for attackers, addition of new data might be relatively easy.*" In order to show the broad threat of untargeted poisoning attacks, Newaz *et al.* empirically analyzed the accuracy drop of a smart healthcare system using different algorithms to detect five different disease scenarios (high blood pressure, high cholesterol, excessive sweating, abnormal oxygen level and abnormal blood sugar)

Table 2: Accuracy drop due to an untargeted poisoning attack on on different ML models trained for health care data, adopted from [173].

Used Algorithm	Before Attack	Accuracy Drop 10% poisoning	Accuracy Drop 20% poisoning	Accuracy Drop 30% poisoning
Random Forest	95.37	1.65	2.03	2.15
Decision Tree	90.16	4.31	15.88	27.31
Neural Network	91.42	10.28	13.14	27.32
Logistic Regression	88.11	9.28	18.93	28.21

[173] using measures from eight different smart health care sensors (e.g., heart rate and motion sensors). Four different classification models were considered, to test the robustness of these models to data poisoning: random forest (RF), decision tree (DT), an artificial neural network (ANN), and a logistic regression classifier (LR). A total of $17,000$ data instances was used, 70% of which were used for training. To measure the effect of poisoning on the trained model, the performance of the model trained on clean data was compared to models trained with poisoned data. Specifically, poisoned data sets were generated where 10%, 20% or 30% of the data instances were poisoned by randomly flipping the disease state classification labels (e.g., "high blood pressure" instead of "normal"). In Table 2, the effect of an untargeted poisoning attack on four different ML models trained on data of health sensors is compared. The clean model accuracy is compared with the relative accuracy drop in three different poisoning scenarios: when 10%, 20% or 30% of the training set is poisoned by flipping the classification labels. The accuracy drops prove to be significant, and are especially high for the case of neural networks.

Vulnerabilities of ML models in the context of health applications arise from several sources. First, data is confidential and patient data often cannot be openly shared for model training. As a result, collaborative learning (e.g., federated learning) is particularly attractive in the healthcare domain [174]. The distribution of training data to multiple computing nodes drastically increases the risk of backdoor attacks. Specifically, a backdoor can be implanted into a model by compromising the data of only a few participants that take part in the training process [175]. Second, medical imaging is a field that relies more often than usual on transfer learning, due to data privacy issues that restrict the availability of large amounts of training data which are required for training deep learning models. As it is difficult for an ML practitioner to obtain sufficient amount of training data for training a model from scratch, there is a substantial demand for data sets or pretrained models which makes it earier for attackers that offer malicious content. Third, data annotation is often difficult to do as ground truth is frequently ambiguous in medical data sets: even specialists disagree on well-defined diagnosis tasks. Detecting poisoned data introduced by malicious users is naturally a complicated task in this context [176].

The motivation of an attacker targeting a medical imaging application can be diverse [177, 171]. Potential incentives for attacking ML models in healthcare applications exist, for example, in clinical trials, in cases where a model is being used to make a diagnosis or to deter-

mine a treatment [178]. To mention a few possible scenarios, an attacker might aim at reducing trust in medical institutions, or in the application of ML algorithms in the healthcare domain in order to reduce their usage. Fraud with patient data also has monetary motivations. A study from the U.S. outlining the motivations for the application of adversarial attacks on medical applications [177] argues, for example, that physicians might have an interest in diagnosing diseases which correspond to a more expensive billing code to receive higher reimbursement for the treatment. An infected classifier that is used by an insurance company, thus, could make it possible for a medical institution to exploit a backdoor that is present in the model or create adversarial examples to enforce a specific diagnosis for a patient [177].

In the context of this study, we analyze which specific security risks arise from the transfer of models to the medical domain.

1.2.2 Attack Vectors in Medical Imaging

The awareness of the risk of cyberattacks in the medical domain is constantly increasing in recent years [179, 180]. Given the recent success of forging adversarial examples in computer vision, this threat is already widely recognized as a potential security threat for medical imaging diagnosis systems [181, 182, 179]. In recent years, some studies have been conducted to demonstrate that adversarial attacks designed for general imaging is applicable to medical imaging [183, 179]. A comparison of the success of applying adversarial examples to medical images, compared to ImageNet images, indicates that medical images might be even more vulnerable [182]. The authors of this study suggest that the reason might lie in the application of the feature-rich neural networks trained on natural data sets to medical images of a specific domain with only limited variability in the images. Neural networks that were adopted from other domains via transfer learning, thus, *overfit* to the specific image domain, opening the doors for easy manipulability. The authors demonstrate this by plotting the loss landscape of some images around input images from ImageNet compared to images from medical imaging data sets into two adversarial directions. Figure 18 illustrates the change in the classification loss when changing pixel color by a value between 0 and 8 into two adversarial directions. The x and the y axis of the plots show the adversarial directions, the z axis shows the classification loss. It can be observed that the loss increases much slower and more steadily in the ImageNet case, whereas there are sharp increases for medical images, indicating that a misclassification can be caused already by applying a very small perturbation to the image.

Also [184, 185] showed very recently that an adversarial attack which modifies only a single pixel of an image can be successfully applied to fool a medical imaging application for the detection of mitosis in breast tissue for cancer recognition. In their study, they propose to improve the imperceptibility of the modified pixel by using a color scoring function that optimizes the color change to lie as closely as possible to the color of the surrounding pixels, while in parallel maximizing the risk of misclassification. An example is shown in Figure 19. Here, the circle in the right image indicates the position of the adversarial pixel that causes the net-

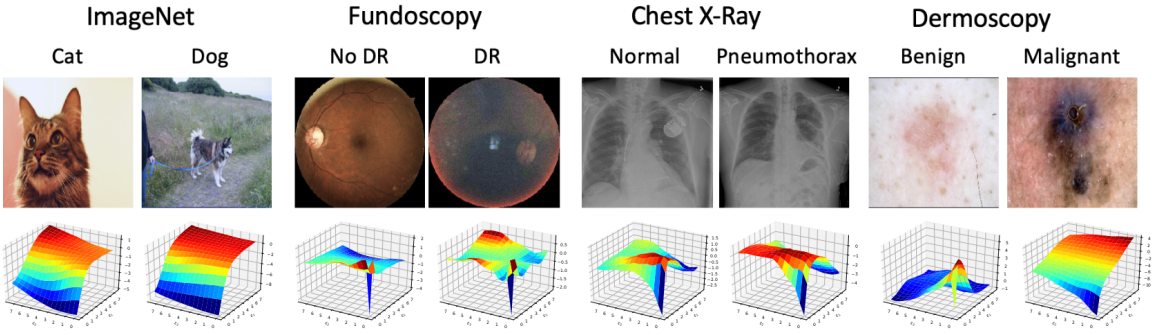


Figure 18: Illustration of higher adversarial vulnerability of medical images compared to ImageNet images, taken from [182].

work to misclassify an image with the label mitosis as normal.

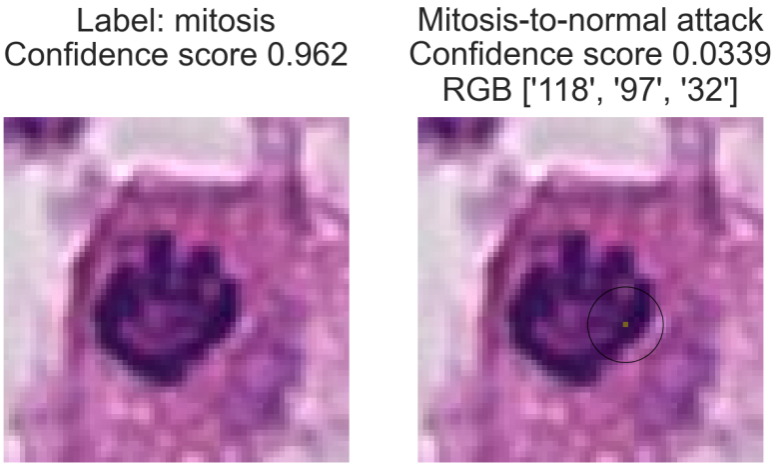


Figure 19: Example of a one-pixel adversarial attack applied on a whole slice tissue sample for the detection of breast cancer, taken from [185].

Next to adversarial attacks, another attack vector that is recognized as a threat is to use deep learning methods in order to alter images such as 3D medical scans in a realistically-looking way in order to add or remove certain medical conditions using generative adversarial networks [179, 186]. These methods use machine learning not for fooling a deep learning model, but in order to fool the human observer, thus, it is not further relevant in the context of this report and only mentioned here for the sake of completeness.

The risk of poisoning and backdoor attacks is also discussed in recent years in the context of medical imaging [171, 183]. However, being a novel field in general that has been mainly investigated on general computer vision tasks so far, demonstrations of specific use cases in the medical domain are still relatively rare.

One of the first studies that systematically investigated the effect of poisoning of healthcare data sets is [171]. They investigated targeted and non-targeted attacks and found that by adding up to 30% of malicious instances to the data sets, the amount of misclassifications for six tested ML algorithms increases. The highest decrease in accuracy which was found was of 26%, for a simple neural network (Multilayer Perceptron, MLP). Malicious instances can be crafted specifically for a data set and an ML method, but the authors demonstrate that it's also possible to poison the data set in a way that affects all types of algorithms, although with lower efficiency. This method is simple, however, it has the shortcomings that many data poisoning attacks share, namely, that the manipulated model is detectable relatively easy due to a lower accuracy in training, or by an expert user inspecting the data sets.

Nwadike *et al.* [187] conducted a study where they investigated more precisely the effect of backdoor attacks on medical imaging data. In this study, a trigger is introduced on a multi-label disease classification task using chest radiography in order to cause a misclassification about the presence of a specific disease. Triggers were inserted prior to training to the training images. Specifically, black pixels were added either at a fixed or a random position, and trigger sizes between 1×1 and 4×4 were tested. They found that the backdoor reliably leads to misclassification after model training, with larger triggers performing better, and performance being independent of trigger location. However, as in most poisoning attack approaches, it is important to poison only a small portion of the data set in order to not impair overall model performance. The study does not investigate how the manipulated model behaves in the case of transfer learning. Findings of general backdoor attack methods [138], however, indicate that the trigger is likely being washed away in the case of transfer learning. Another problem of this attack is that it requires that the original training data set is available to the attacker. This assumption might be fulfilled if malicious access to medical systems occurs [169, 170, 188], but cannot be considered true in general, because unlike data that can be used for general computer vision tasks, medical data might not be publicly available due to privacy or copyright concerns [189].

Wang *et al.* [189] proposes an attack that takes this problem into account. Unlike Nwadike *et al.* [187], the authors of [189] do not assume that the original data set is known to the attacker which makes the setting of the attack more relevant for the medical imaging domain. Specifically, they make use of transfer learning as a technique to derive an infected student model from an already trained teacher model. As the original data set is not available, their trick is to create a replacement data set by reverse-engineering training input that copies the activations of the trained model as closely as possible. Then, via an optimization procedure, a visual trigger is created which causes activation changes in specifically selected neurons such that the model makes wrong predictions in the presence of the trigger, while leaving intact the overall model performance and neuron activations. They successfully apply this methodology by inserting a trigger image into an MRI brain scan image that is used for brain tumor detection. An example trigger that was inserted into an abnormal MRI brain scan to trick the neural network into misclassifying the scan as "normal" is displayed in Figure 20 and demonstrate that their method succeeds also in the presence of commonly used defenses. Specifically, to

create a backdoor that is hard to detect, it is crafted in a way such that it specifically alters the activations of a small set of selected neurons which were chosen in a way that neither pruning nor fine-tuning defenses would be successful (see Section 1.1.5 for details).

Figure 20: An example trigger for misclassification of MRI brain scans, taken from [189].

In summary, attack vectors that were specifically applied to the domain of medical imaging became more wide-spread in recent years. Although methods such as [189] have shown to be successful and stealthy, even in the presence of defenses, a look into the overall literature of backdoor attacks (see Section 1.1.4) reveals that the methods that were so far tested in the medical domain lag behind the new attacks that were recently proposed in the literature (e.g., [138]). Thus, it can be assumed that beyond the attacks that are discussed above, also the attacks that were discussed in detail in Section 1.1.4 are applicable to medical imaging. Especially very recent poisoning and backdoor attacks that manipulate images in imperceptible ways [130, 139] and can transfer from the teacher to the student model, have a high potential to maliciously tamper with medical imaging applications. Particular risk factors are that outsourcing of the training process becomes more and more common [183] and that the usage of transfer learning in medical imaging has become the de-facto standard in recent years [15]. Novel methods as introduced in Section 1.1.4, thus, could significantly increase the risk. However, as they have not yet been investigated with a specific focus on medical imaging in the currently available literature, no conclusions about their potential can be made at this point.

A difference of medical imaging data sets, compared to general computer vision task data sets, which has to be kept in mind is that medical imaging often is used for multi-label classification, i.e., each data point might be not only classified as a single class but as belonging to multiple classes (to express the presence of multiple diseases of the same time). Both studies that investigated backdoors for medical imaging [187, 189] focused on such a use case. Other backdoor attacks might require some modifications in order to adapt the method to be applicable for multi-label classification. However, there are indications that adversarial as well as poisoning attacks that are successful on general computer vision tasks also are successful in the medical domain [183, 179]. In [182] even a higher vulnerability of medical images compared to natural images was found, due to the limited variability of medical images which use only a specific set of features of a pretrained teacher model.

1.2.3 Defenses in Medical Imaging

After discussing defense strategies in general terms in Section 1.1.5, this section focuses on specific defenses that were tested on medical imaging applications. In this context, a number of approaches for detecting or defending against poisoning and backdoor attacks have been proposed. In particular, the importance of using explanation methods to gain insights into the choices a model is making is highlighted in various studies [182, 187, 190].

Kermani [171] looked at general poisoning attacks which rely on a modification of the data set which affects the performance of the trained ML models. Therefore, it is relatively easy to detect corrupted models by observing deviations in accuracy metrics. This defense method, however, is not applicable to many modern and more stealthy types of backdoor attacks.

Independently from any particular attack, several studies [190, 191] argue that explainability and interpretability are indispensible for creating a trustworthy system for a medical application. In the computer vision domain, *explaining* the decision of a deep model usually means to generate a saliency map of an input image which highlights on which parts of the image the model is drawing its conclusions about the predicted class [165]. Beyond helping to establish trust in medical systems by providing additional information to the user of the application, explainability can also be a key to build strong defenses for medical imaging models.

This approach has been tested practically in a recent study [187]. In this study, the authors inserted small pixel triggers into chest radiographs and investigated a potential mechanism to detect triggers using a popular feature attribution method, namely the Gradient-weighted class activation mapping (GradCAM) algorithm [165]. This method uses the gradient of a convolutional layer of the trained model to infer which parts of an image contribute most to the classification decision. The computed saliency map highlights important regions of an input image. By visual inspection, a human expert can judge from such visualizations whether the model is performing normally or whether a backdoor might be present. Examples of how saliency maps computed from GradCAM can assist in trigger detection, are shown in Figure 21. The rows represent different training epochs, with the last row corresponding to the most advanced stage of learning. The first two columns show the saliency maps computed from the last convolutional layer of the network without a trigger (first column) vs. with a trigger (second column) at the position of the red circle. Columns three and four accordingly show the saliency maps computed from a middle convolutional layer of the network which allows to localize the trigger more precisely. A difficulty in this approach is that visual inspection of a large amount of images is required to exclude the possibility that a backdoor is present which does not scale for large data sets.

Also in [182] the detection of attacks on medical images is discussed, with a particular focus on adversarial attacks. The authors argue that medical systems might be more easy to attack than images with natural images, but the image manipulations might also be easier to detect due to a rather uniform structure of the data across the data set that makes it possible to reveal anomalies using standard techniques such as the plotting of neuron activations [182].

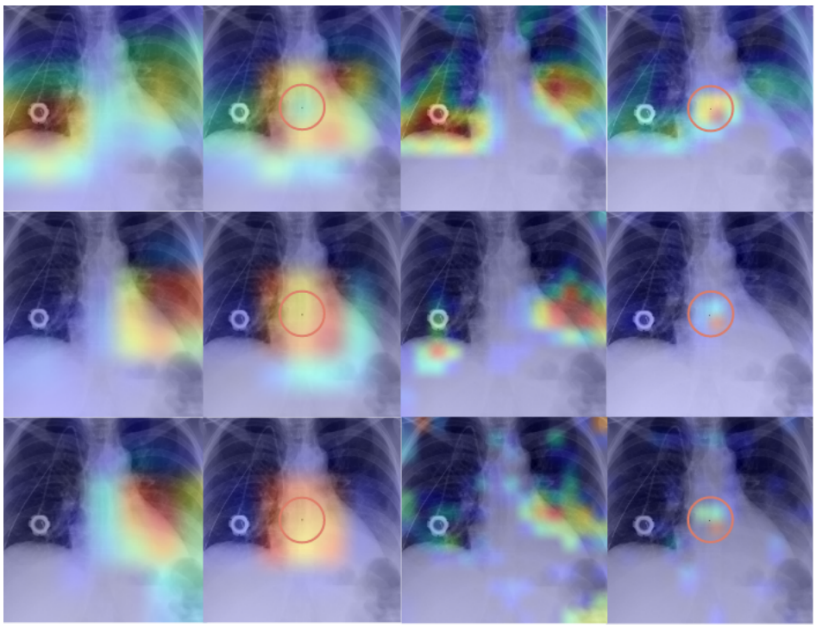


Figure 21: Illustration from [187] on how triggers can be detected using saliency maps.

Specifically, they show that the features of a deep network are disrupted significantly by the addition of adversarial perturbations. Figure 22 shows visualizations of how adversarial features (orange) differ from the features of clean images (blue), using t-SNE embeddings of the deep feature vectors of the second-last fully-connected layer of the network. The comparison is shown for three medical data sets (rows) and four different adversarial attacks (columns) and demonstrates that deep features of medical images are large disrupted by adversarial perturbations.

Also in the medical domain, it has been shown that sophisticated attack schemes can circumvent many standard defense mechanisms. Specifically, the attack presented in [189] was shown to defeat three of the most common defenses. 1) Pruning-based defenses which aim at pruning neurons with low weights or which show low activations on the input data were circumvented by crafting the backdoor in a way such that it makes use of neurons which are unlikely to be pruned using such techniques. 2) The neurons which the backdoor should activate were chosen in a way such that a retraining of the model is likely to not alter the activations significantly. In this way, refinement-based defenses that try to “wash out” backdoors from the model become less effective. 3) Autoencoders which were trained on a genuine data set are sometimes used to detect malicious network input, as the feature embedding of the autoencoder can provide information about whether the input is likely to contain a backdoor or not. [189] circumvent this defense by optimizing the trigger in a way such that the reconstruction error with an autoencoder trained on a surrogate data set is minimized, rendering those types of attacks also less effective.

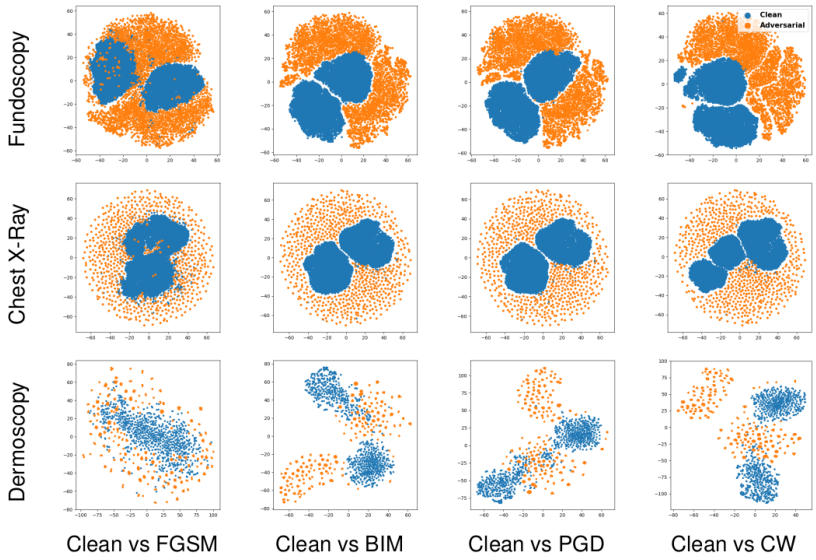


Figure 22: Visualization from [182] showing the difference between adversarial (orange) and clean (blue) features, shown as t-SNE embeddings.

Such demonstrations show that further research on the question of security of deep learning models for medical applications is required.

2 Transfer Learning: Practical Investigations

Medical imaging is an important application scenario for transfer learning [15], but the risk that is associated with poisoning and backdoor attacks in this field has not been yet sufficiently investigated in the literature. Specifically, many modern attacks that constitute a particular risk in the transfer learning scenario, such as latent backdoor attacks [138], have not been investigated yet with respect to the specific use case of medical imaging. At the same time, it is unclear whether published poisoning defenses can reliably prevent attacks in a general computer vision task, and in particular in the specific case of medical imaging.

Empirical research was conducted as part of the study. In the following we describe the objective of these practical investigations, the corresponding setup, the conducted experiments and discuss the results.

2.1 Objective

The aim of the practical investigation is to systematically investigate the risk of backdoor attacks which were derived via transfer learning, and to evaluate the effectiveness of existing backdoor defenses. Specifically, the potential of modern stealthy, realistic backdoor attacks such as the latent backdoor attack [138] are investigated in the context of medical imaging, and the applicability and effectiveness of defense mechanisms is evaluated.

The use case scenario we are looking into is the training of a student model for diagnosing COVID-19 from chest X-ray images. Given the current pandemic situation and the lack of sufficient PCR testing capabilities, it is crucial to detect positive COVID-19 cases in a fast and accurate way in order to make an informed decision about necessary quarantine and treatment measures. If a sufficiently accurate classifier is developed, it would provide a very cost-efficient way to diagnose a large number of patients in almost instantaneous time. Therefore, multiple studies were recently published which focus on a machine-learning-based diagnosis of COVID-19 using medical imaging [192, 193, 194], with the aim to provide a fast and cost-efficient way for performing the diagnosis. The application of transfer learning is especially important here due to the limited amount of available data. In their study [192], Apostolopoulos *et al.* demonstrated that transfer learning using a pretrained teacher model and a small data set containing only 224 COVID-19 cases can achieve accuracy values of nearly 97% in diagnosing COVID-19 from X-ray images. Furthermore, the trained student was able to differentiate COVID-19 cases from general bacterial or viral pneumonia cases. As a part of such research efforts, a substantial amount of data has been collected and made available for research over the past year from COVID-19 patients. In this project, we make use of the COVID-19 chest X-ray data set[6] [195, 196] (see examples in Figure 23). One motivation for conducting experiments for such a use case is that research papers often use some standard data sets to evaluate

[6] Available at: `https://www.kaggle.com/tawsifurrahman/covid19-radiography-database`

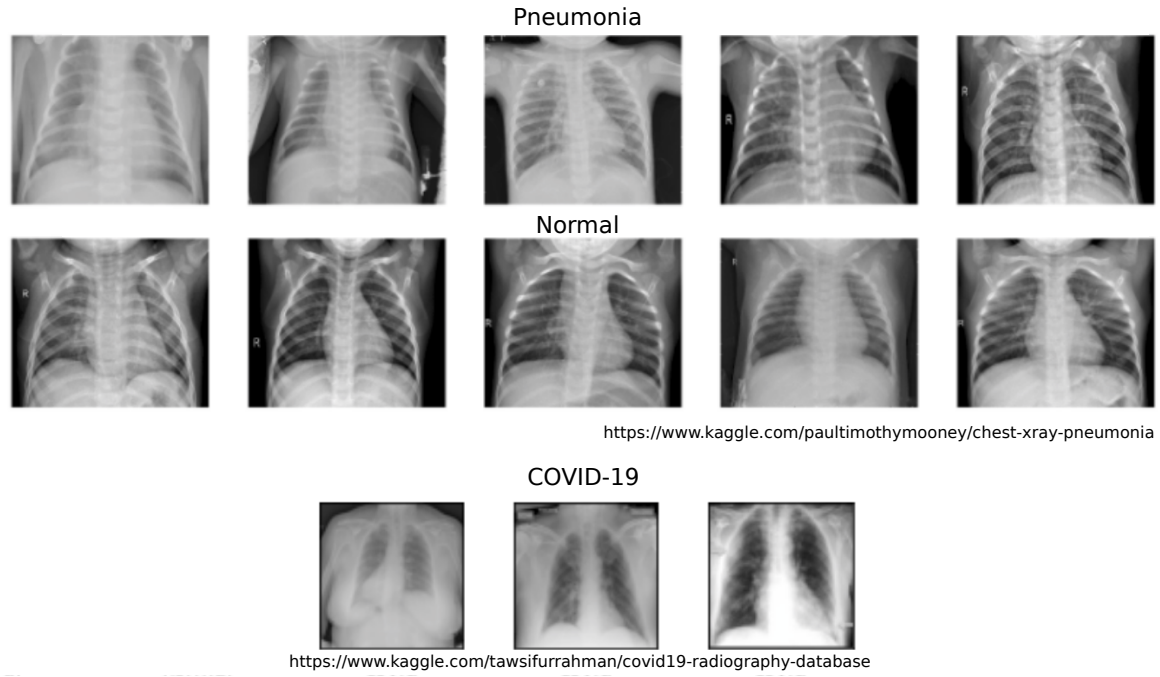


Figure 23: Example images of the pneumonia chest X-ray data set of [197] and COVID-19 cases from the COVID-19 chest X-ray data set [195, 196].

the effectiveness of attacks and defenses on ML models. However, it is not clear whether such results actually transfer to real world use cases.

We compare the risk for different types of models that could be adopted for transfer learning of a medical imaging task. Two different types of pretrained models are considered for transfer learning. First, a state-of-the-art model architecture pretrained using a general computer vision task is considered (in the following referred to as teacher model 1), as it is commonly used for adapting it to a specific medical imaging use case [15]. Second, a smaller, simpler model is considered that is pretrained on a related medical task (teacher model 2) such that the model is less complex and more specialized for the use case to which it should be adapted during transfer learning.

As teacher model 1, a standard pretrained model is be used based on the ResNet-18 architecture and pretrained on ImageNet [119]. Teacher model 2 is generated by training a simpler convolutional network from scratch. Both types of teacher models are trained on a medical data set that is similar to the COVID-19 detection use case (teacher task). The data set for this teacher model is [197], which contains chest X-ray images with labels indicating the presence of pneumonia[7] (see examples in Figure 23).

The comparison between teacher models 1 and 2 is interesting because it is likely that the

[7] Available at: `https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia`

complexity of the teacher model and the data set differences affect the transferability of the attacks as well as the effectiveness of the defenses [182]. Thus, the evaluation helps us understand what type of teacher model (i.e., what kind of transfer learning), would be more secure to employ in a realistic use case.

The specific technical assumptions we make for the attack scenario is that the attacker prepared the teacher model and made it available, for example, in an open-source repository. Therefore, the victim who downloads the model has white-box access and is able to fully inspect the model (its weights and activations). However, the victim has no access to poisoned examples, which limits the set of potential defenses. For certain defenses we assume that the victim is able to collect a limited amount of clean training data from the teacher training data set. This assumption has a rather technical motivation, in particular it simplifies the application of certain defenses.

For the empirical investigations, we increase stepwise the complexity of the applied attacks. The first attack is BadNets [123, 129] which uses a fixed pattern inserted into the input image as a trigger (see Figure 8 for an example). As a second attack, we use the IMC attack proposed in [141], where the trigger pattern is optimized with the help of an adversarial attack. Third, we consider the applicability of the latent backdoor attack to our use case as it is an attack specifically designed for the use case of transfer learning, and likely to be immune against most defenses [138].

Before trying to detect and mitigate the proposed attacks, we evaluate their performance. First, we evaluate the success rate of the teacher model by looking at the accuracy of the prediction on poisoned examples. Then we evaluate the transferability of the backdoor in the teacher model to the student model which we measure as the attack success rate of the specific attack in the student model. It is common practice to freeze at least parts of the weights of the teacher model during the training of the student. Therefore, we consider different setups of the student model training, in particular we analyze the effect of a varying number of frozen layers on the transferability of the backdoor. Having both a robust backdoor (independently of the technique to generate it) and a a certain degree of transferability is a prerequisite in this study to evaluate the possibility of detecting and mitigating backdoors.

In the detection and mitigation phase, we first consider the naive approach of retraining with clean data to answer the question of how easy it is to wipe out the backdoor by chance during the transfer learning process. We expect that this technique might work in the case of simple attacks (e.g., [129]). Furthermore, we use Fine-Pruning [162] to remove backdoors from a victim model. As an example of a defense which detects poisoned examples at inference time, we use the detection method STRIP [158]. Finally, as a method which inspects the teacher model for potential backdoors, we apply Neural Cleanse [159]. With Neural Cleanse, we can reverse engineer backdoor triggers and detect harmful ”shortcuts” in the neural network classification regions.

In summary, we tackle the following research questions within the practical investigations:

1. Under which training conditions are backdoors from a teacher model transferred to a student model during the transfer learning process?
    - How does the choice of the **teacher model architecture** affect the vulnerability of the student model to backdoor transferability?
    - How does the **freezing of layers** of the teacher model during transfer learning affect the attack success rate in the student model?
2. Can the latent backdoor attack be efficiently executed in the chosen application scenario?
3. Which defense method is able to reliably protect the student model against a variety of backdoor attacks?

These guiding research questions are used to formulate concrete experimental setups and to structure the experimental results in the remainder of this section. All these research questions refer to the previously described medical imaging use case, where we use a teacher model trained on the Pneumonia data set in order to train a student model for COVID-19 detection.

2.2 Methods

In this section, we elaborate on the algorithmic and technical details for the practical investigations. Furthermore, we describe the precise experimental setups, which address the previously posed guiding research questions. For the investigations, implementations of attacks and defenses offered by an open-source library (`TrojanZoo`[8]) are used to reduce the implementation effort. However, the used software tool – containing backdoor attacks and defenses – should be viewed as interchangeable.

2.2.1 Models, Attacks and Defenses

For the experiments in this project we need to choose a model architecture, backdoor attacks and backdoor defenses. In the following, we describe the architectures choices and go through the choices for attack and defense algorithms, providing a short explanation of each selected method.

Models Since we are interested in robustness against backdoor attacks in a transfer learning setup, we have to train teacher as well as student ML models. In general, two model architectures with varying complexity levels are used in this project.

[8] https://github.com/ain-soph/trojanzoo

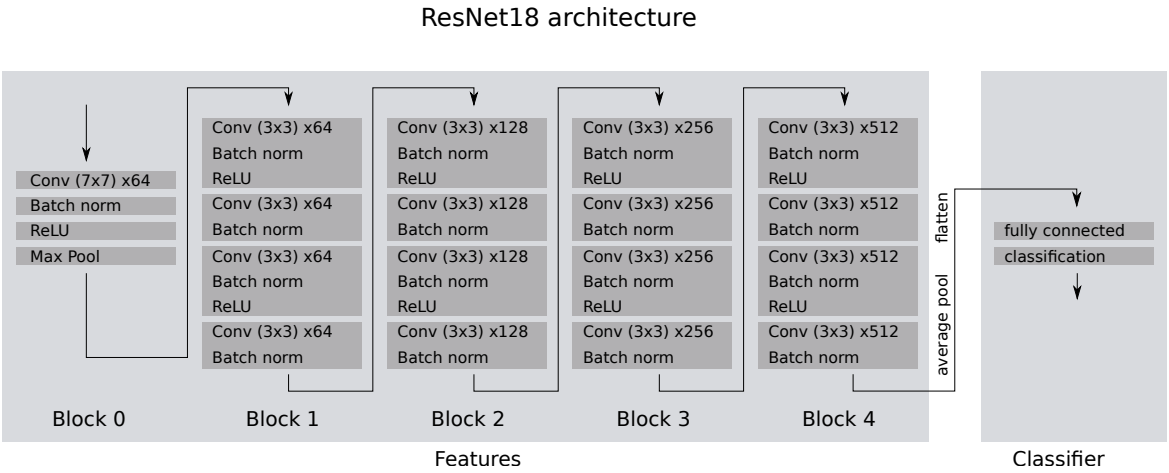


Figure 24: Overview of the ResNet-18 architecture [198].

First, a ResNet-18 architecture [118], pretrained on ImageNet [119], is trained (and infected with a backdoor) with respect to the pneumonia data set (see Figure 24), and then transferred to the COVID-19 data set. In both training procedures, the fully connected layer of the ResNet-18 model is replaced in order to fit the number of necessary classes for the respective task. The pneumonia data set only differentiates between infected and non-infected patients, whereas the COVID-19 data set has three classes, namely no infection, COVID-19 infection and a viral pneumonia infection. In [198] it has been shown that transferring from ImageNet to viral detection based on chest X-ray data is a valid strategy, which can lead to high-performing models for the pneumonia data set. In our case, the ResNet-18 model, pretrained on ImageNet, is taken from the `TrojanZoo` library, thus, we can directly use the library for training and attacking this kind of teacher model.

As a second type of teacher model, we train a simpler, smaller convolutional neural network with no residual blocks on the pneumonia data set from scratch (see Figure 25). This convolutional neural network has around 250 thousand trainable parameters, which is significantly less than the approximately 11 million trainable parameters of a ResNet-18 model. Analogously to the ResNet-18 model case, we train and attack this convolutional model on the pneumonia data, and then train student models on the COVID-19 images by replacing the fully connected last layer of the teacher.

Both models preprocess the provided images which are loaded to a range of $[0, 1]$ as a part of the forward call by normalizing the input image according to the means and standard deviations calculated on the ImageNet training data set. In other words, the usual data preprocessing for the ImageNet pretrained ResNet-18 is applied to all models and data sets. This increases the comparability between performance evaluations of the different models. Performance results on clean as well as poisoned data for the different teacher and student models can be found in Section 2.3. Furthermore, the student training is executed with a varying number of frozen layers. In other words, we simulate the case where the victim tries to keep the train-

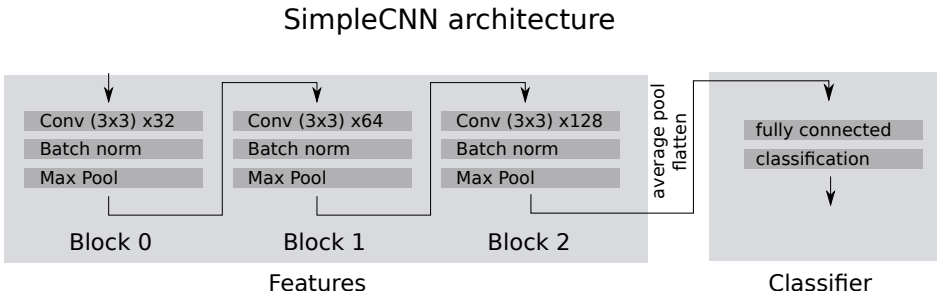


Figure 25: Overview of the architecture of the simple CNN set up for the purpose of this project.

ing efforts as low as possible by reusing the training parameters obtained via teacher training during transfer learning. For this, the student training procedure of our tool contains an integer parameter, which defines the number of frozen residual blocks of the ResNet-18 teacher model.

Note that the teacher training data set contains 5216 images which is a comparably small number, considering that teacher models made available in the internet often were trained on large amount of data. We choose this data set here to speed-up the process of preparing clean and attacked teacher models. The data set size, and particularly the difference to the size of the student training data set (which contains 14873 images), however, has to be taken into account when interpreting the results as it might effect the transfer learning process.

Attacks We used three different backdoor attacks, selected from various attack classes and with differences in their complexity. This allows us to formulate hypotheses with respect to the transferability of an attack class in the transfer learning scenario, in relation to the capabilities of the attacker. Another factor that affected the choice of backdoor attacks is that the implementations for the upcoming attacks are included in the used software library `TrojanZoo`.

As an uninformed attacker, the most straightforward backdoor attack is the **BadNets** attack [123]. Here, the adversary randomly selects parts of the training set, applies a predefined backdoor trigger (e.g., square, copyright sign) and alters the labels of the poisoned images to the adversary's target class. As a default, we set the poison percentage to 10% of the training data. During training, the model then learns to achieve a high accuracy on benign input and, at the same time, a backdoor is introduced, where the model connects the appearance of the trigger with the selected target class. As a trigger, we used a fixed white square of different trigger sizes at the top left of the image.

The second backdoor attack we consider is the **Input Model Co-optimization (IMC)** attack [141][9]. This backdoor attack is an iterative attack that uses an optimized trigger pattern.

[9] Note that the authors of this attack are also the authors of the `TrojanZoo` library which is why the library includes a number of different implementations of this attack. We use here the standard implementation.

To make the attack comparable to the BadNets attack, we restrict the attack to use the same trigger shape and sizes as used for the fixed BadNets trigger. Given that the iterative attack is much more efficient, the poison percentage can be reduced to 1% (which is the default value suggested by the library).

The third backdoor attack we analyze is the **latent backdoor attack** [138]. Here, an informed attacker inserts a backdoor into the teacher model that targets a specific class of the student model which is not yet existent yet as a class label in the teacher model. One requirement is that the attacker needs to have a data set at hand that is similar to the one that the victim is expected to use for transfer learning. In this coding project, this requirement is fulfilled as both the teacher and the student task are based on chest X-ray scans. This attack also yields an optimized trigger pattern like IMC. We restrict the trigger pattern to the same shape and sizes as for the other two attacks.

For more details about these attacks see Section 1.1.4 (page 18).

Defenses As for attacks, also the defenses should represent a diverse set of defense methods representative for defenses currently used in the literature. In our transfer learning scenario, we assume that the adversary has full control of the teacher training with the pneumonia data set. This implies that the victim cannot access the training data set (containing potentially poisoned data) that was used for training the teacher model. As a consequence, we do not consider defense methods that are based on the detection of poisoned data points in the training data set of the teacher. However, we study representative defense methods of the remaining defense categories that we recognize in the current literature, i.e., mitigation, model inspection and detection during inference (cf. Section 1.1.5).

The first defense, a mitigation defense, which plays a major role in the experiments of this project, is the **retraining** of the teacher model on the COVID-19 data set. In other words, the transfer learning process itself may reduce the risk of induced backdoors due to the adaptation of the model weights of the teacher model. Evaluating the mitigation effect of this defense is equivalent to answering the question whether an existing backdoor of the teacher model transfers to the student model (transferability of backdoors).

Additionally, **Fine-Pruning** is analyzed as a second representative of the mitigation defense class [162]. To apply this defense method to our transfer learning scenario, we assume that the developer of the student model has access to a small portion of the (unpoisoned) pneumonia data set. The procedure is that the victim uses Fine-Pruning on the teacher model with a few images of the pneumonia data set before the training process of the student model on the COVID-19 data set is initiated. Note that the assumption that a portion of the *clean* training data set is available is different from the assumption that the victim has access to the exact data set that was used for training the teacher model. While the latter assumption is unrealistic, it might be possible for the victim to collect an alternative training data set for applying Fine-Pruning. However, the victim has to be able to obtain these data from a reliable

source to avoid to potentially inject another backdoor during fine-tuning. Thus, it depends on the given use case whether this defense is realistic or not.

As a representative of defenses which detect poisoned data at inference time, we consider **STRIP** [158]. The main idea of STRIP is to exploit the fact that triggers are usually input-agnostic, i.e., that they can be applied to every possible input image. As a result, inputs which contain a trigger pattern that activates a backdoor in the model are much more stable under strong perturbations compared to benign input samples. Thus, similarly to the Fine-Pruning defense, an implicit assumption is made that the victim has access to clean training data which is not always realistic.

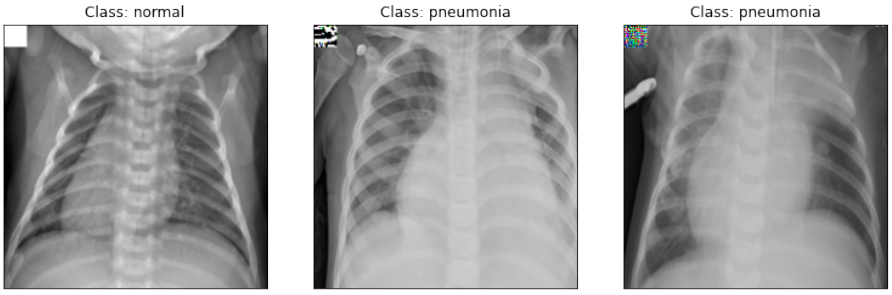
As a representative method for model inspection defenses, we select **Neural Cleanse** [159]. The authors of [159] propose to detect backdoors in a trained neural network by exploiting the fact that trigger patterns create "shortcuts" from within regions of the space belonging to a label into the region belonging to the target class. As a result, in the backdoored model, a tiny change in the input features is sufficient to misclassify the input, namely the insertion of the trigger. This intuition is used by the Neural Cleanse detection method which looks for the smallest possible change that misclassifies a large amount of input data. Unlike the other defense methods we apply, Neural Cleanse proposes a potential, reverse-engineered trigger pattern which we can compare to the original trigger to verify that the defense method successfully restores the malicious trigger. The optimization is performed using two terms: the cross-entropy loss measuring the success of the attack (i.e., that the inputs are misclassified as the class that is suspected as a potential target class), and an L1 norm loss term measuring the conciseness of the trigger. In the used implementation, the L1 norm loss is weighted initially with $1e-4$, and is adapted over time. Note that it is difficult to interpret this term in absolute terms as the L1 norm depends on the size of the image. The value was chosen empirically in order to lead to triggers that are roughly of the same size as the actually inserted trigger.

For further details about the used defenses, refer to Section 1.1.5 (page 30).

2.2.2 Experimental Plan

In the following, we outline the experimental setups linked to the research questions. The execution of this experimental plan forms the basis for the upcoming summary of results and the discussion of next steps.

For all tested attacks, three different trigger sizes are considered: 5×5, 10×10, and 20×20 pixels (thus, the maximum trigger size covers less than 10% of the input images which are of size 224×224). The trigger pattern is fixed to a square at the upper left corner of the image. Examples of triggers for the three tested attacks are provided in Figure 26. A white square is used for the BadNets attack. For the IMC and the latent backdoor attack, the same white square is used for the initialization of the trigger, but while running the attack the trigger pattern is optimized.



(a) BadNets trigger **(b)** IMC trigger **(c)** Latent backdoor trigger

Figure 26: Examples for 20×20 sized triggers for the three tested attacks.

Experiment 1 This experiment aims to answer the first research question, namely:

1. Under which training conditions are backdoors from a teacher model transferred to a student model during the transfer learning process?
    - *How does the choice of the teacher model architecture affect the vulnerability of the student model to backdoor transferability?*
    - *How does the freezing of layers of the teacher model during transfer learning affect the attack success rate in the student model?*

We use the following setup for this experiment:

- **Model:** Use ImageNet pretrained ResNet-18 architecture (see Figure 24), and simple untrained CNN architecture (see Figure 25).
- **Teacher Training:** Run 50 epochs with learning rate 0.1 for the pretrained ResNet-18 and 50 epochs with learning rate 0.001 for the simple CNN model on the basis of the chest X-ray pneumonia data set. The number of epochs and the learning rates were selected manually within a reasonable range of values to ensure that model training converged fast while good model accuracy was achieved.
- **Attack:** Apply no attack, BadNets and IMC on both teacher training procedures with varying trigger sizes ($5\times5-20\times20$ pixel). For both attacks select class 0 of the model, corresponding to "normal" (i.e., healthy,) as the target class. The watermark position (white square) is the top-left corner. The ratio of poisoned data is 10% for BadNets and 1% for IMC according to the default values that are preset in the attack library. Intuitively, using a smaller poisoning rate for IMC compared to BadNets is motivated by the trigger optimization that IMC performs which is expected to increase the effectiveness of the

attack. Additionally, IMC takes 20 iteration steps during the PGD attack on 20 randomly sampled images in order to optimize the trigger pattern.

- **Student Training:** Run 10 epochs with learning rate 0.1 for the ResNet-18 model (0.001 for the simple CNN model) on all (poisoned & unpoisoned) teacher models. Repeat this for a varying number of frozen layers - or rather, frozen residual blocks. Every block consists of multiple layers. The student training is executed with no frozen feature layers, with all feature layers frozen and with all feature layers frozen except the last feature block. In all cases, the classifier layers (last layers) remain trainable.
- **Report:** Repeat the outlined experiment 10 times and provide mean and standard deviation of the accuracy and the attack success rate for the different model and attack configurations.

Experiment 2 This experiments aims to answer the second research question

2. Can the latent backdoor attack be efficiently executed in the chosen application scenario?

For this purpose we use the following setup:

- **Model:** Use ImageNet pretrained ResNet-18 architecture (see Figure 24), and simple untrained CNN architecture (see Figure 25).
- **Teacher Training:** Run 10 epochs of clean training before the latent backdoor attack is initiated.
- **Attack:** Apply the latent backdoor attack with varying trigger sizes ($5 \times 5 - 20 \times 20$ pixel) and target class 2 of the COVID-19 data set, i.e., trigger should lead to positive COVID-19 classification. Use 100 positive COVID-19 samples for the injection of the backdoor into the teacher model. For this latent backdoor trigger injection step we run 50 training epochs. The trigger is injected into the final feature layer (before the fully connected classification layers). The traces of the backdoor are removed with the help of 10 clean postprocessing training epochs. The watermark position (white square) is the top-left corner.
- **Student Training:** Run 10 epochs with learning rate 0.1 for the ResNet-18 model (and, 0.001 for the simple CNN model) on all (poisoned & unpoisoned) teacher models. Repeat this for a varying number of frozen layers - or rather, frozen residual blocks. Every block consists of multiple layers. The student training is executed with no frozen feature layers, with all feature layers frozen and with all feature layers frozen except the last feature block. In all cases, the classifier layers (last layers) remain trainable.

- **Report:** Repeat the outlined experiment 10 times and provide mean and standard deviation of the accuracy and the attack success rate for the different model and attack configurations.

Experiment 3 In this experiment, we look at defenses against the attacks to answer the question:

3. Which defense method is able to reliably protect the student model against a variety of backdoor attacks?

We use the following setup:

- **Model:** Use ImageNet pretrained ResNet-18 architecture (see Figure 24), and simple untrained CNN architecture (see Figure 25).
- **Teacher Training:** Take existing poisoned (BadNets, IMC and latent backdoor attack) and clean ResNet-18 teacher models from previous two experiments.
- **Student Training:** Take existing poisoned (BadNets, IMC and latent backdoor attack) and unpoisened ResNet-18 student models from previous two experiments. Focus on fully-frozen student models, i.e., use the optimal setting for the adversary which is the worst-case scenario for the victim.
- **Defense:** Apply Fine-Pruning on the teacher model with a pruning ratio of 0.8 which in the original publication [162] was the highest pruning ratio that did not significantly impact clean accuracy while reducing the attack success rate considerably (cf. Figure 4 of [162]) [10]. For the fine-tuning of the Fine-Pruning defense we run 10 epochs with a learning rate of 0.01. The implemented Fine-Pruning defense makes use of the whole pneumonia training data set. The STRIP defense is evaluated with respect to the given poisoned and unpoisoned student models which were trained while freezing all feature layers. Here, the perturbed instances of an input image are calculated with blending parameter $\alpha = 0.001$[11] and 90 randomly sampled images from the COVID-19 validation set. Furthermore, 60 images are sampled from the test set in order to calculate the necessary detection boundary. Finally, Neural Cleanse is applied to the poisoned models with the aim to reverse-engineer the backdoor trigger. Specifically, the algorithm tries to reconstruct the trigger for all potential target classes, in our case, that is the normal class and the pneumonia class, where the pneumonia class corresponds to the actual target class of the attack. For the inference, 10 epochs and a learning rate of 0.01 were used which led to quick convergence to a solution in the experiments.

[10] A comparison of multiple values between 0.1 and 0.9 indicated that the pruning ratio did not have a large influence on the results in our experiments. Thus, the main effect of this defense in our use case seems to be caused rather by the fine-tuning than the by the pruning step.

[11] We empirically tested values of α between 0.1 and $1e-4$ and found this value to show the best backdoor detection performance for an example BadNets attacked ResNet-18 model.

- **Report:** Repeat the outlined Fine-Pruning experiment 10 times for the test set, report average accuracy and success rate for different poisoned models after applying Fine-Pruning. For the BadNets and IMC attack, the success of Fine-Pruning is measured by evaluating the fine-pruned teacher models directly. The success of the Fine-Pruning defense against the latent backdoor attack has to be measured on the student models derived from the fine-pruned models, since the target class is added only during student training. For STRIP report the average accuracy of the prediction, as well as F1, precision and recall scores with respect to the 60 sampled images of the COVID-19 test set. Here, every image is considered with and without an added backdoor trigger. The trigger depends on the backdoor attack used to create the respective poisoned student model. For Neural Cleanse report the reverse-engineered trigger patterns and compare them to the original triggers to gain an indication of how well the algorithm is able to detect backdoors.

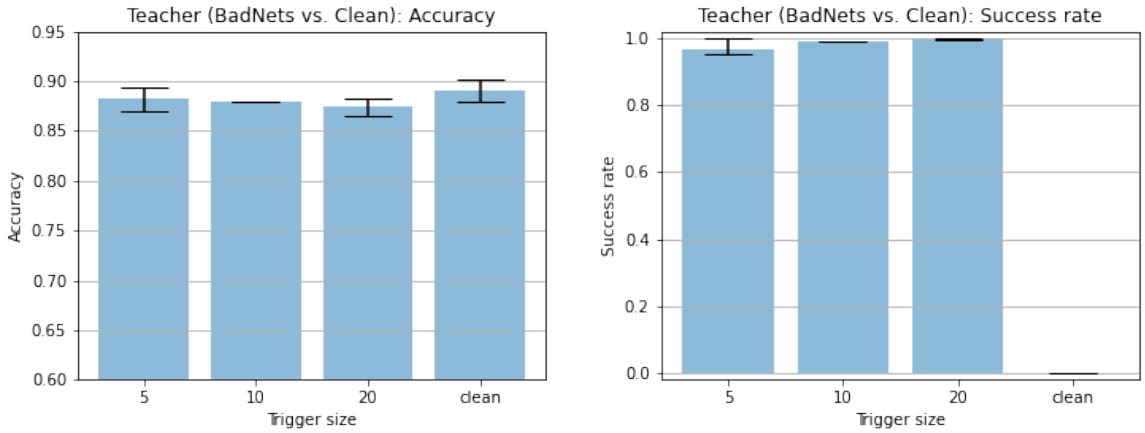
2.3 Results

In Section 2.2.2 we outlined three central experiments. In the following, we summarize the key results of these experiments, and deduce answers to the given research questions.

Experiment 1 The teacher training process is able to reliably create models with an accuracy of around 89% for the pretrained ResNet-18 architecture and an accuracy of around 79% for the simple CNN (trained from scratch) on the pneumonia test data set. It is not surprising that the comparatively high complexity of the ResNet-18 architecture and the ImageNet pretraining lead to better performing teacher models. The given simple CNN architecture seems to be lacking necessary capacity to achieve state-of-the-art accuracy on the pneumonia task. Further optimization of the CNN architecture, for example, by using several fully-connected layers in the classifier block might improve the accuracy results. However, in this study, we are interested in the effect of the significant capacity difference between the ResNet-18 and the simple CNN on the vulnerability to backdoor attacks. Thus, we stick with the given simple CNN architecture despite their differences in accuracy on the task.

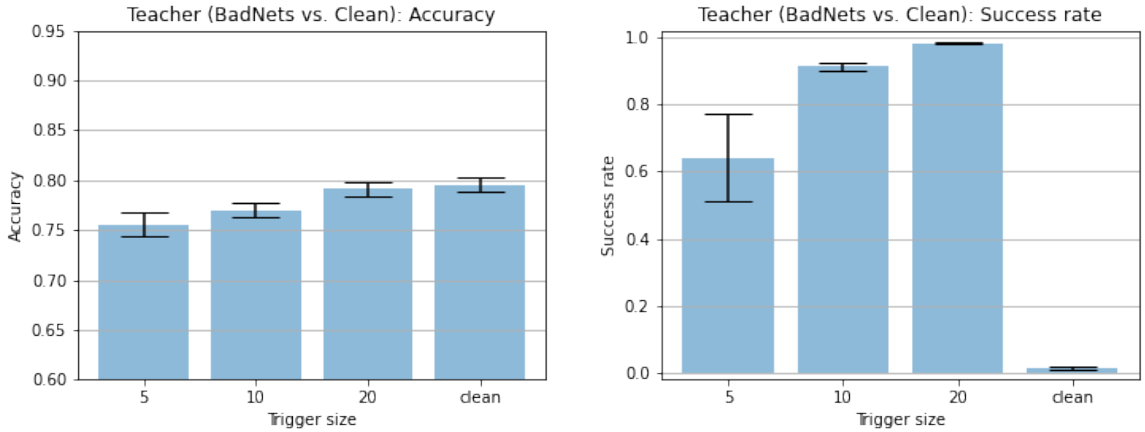
In Figure 27a and Figure 28a one can see that inserting backdoors with BadNets during teacher training leads to a slight drop in clean accuracy compared to the unpoisoned models. For the IMC attack this drop is barely noticeable, i.e., even smaller than for the BadNets attack (see Figure 29a and Figure 30a). This might be a consequence of the rather small poisoning ratio of 1% in the IMC attack configuration (compared to the 10% of BadNets). Hence, the model has only seen very few poisoned data points, thus it largely focused on the benign pneumonia detection task.

Interestingly, there is no clear indication that a higher trigger size leads to a more severe drop in teacher model accuracy. In our experiments, the clean accuracy of a poisoned teacher with trigger size 20×20 is often at a similar level as the accuracy of a poisoned model with



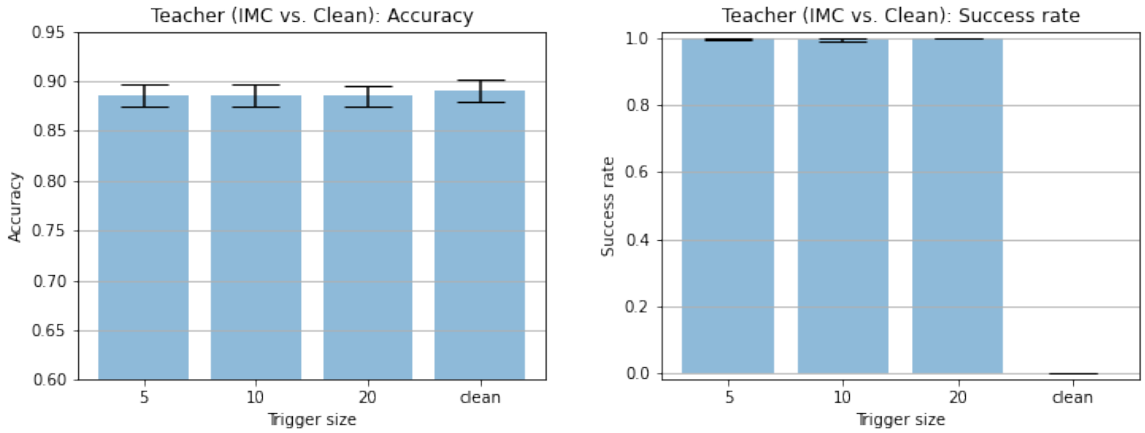
(a) Accuracy **(b)** Attack success rate

Figure 27: Accuracy and BadNets attack results on ResNet-18 teacher models measured on the pneumonia test data set.



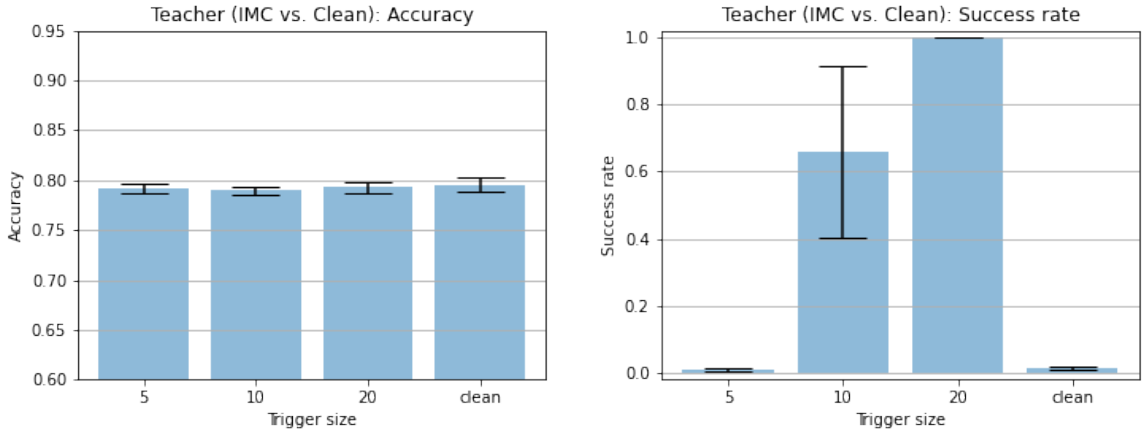
(a) Accuracy **(b)** Attack success rate

Figure 28: Accuracy and BadNets attack results on simple CNN teacher models measured on the pneumonia test data set.



(a) Accuracy **(b)** Attack success rate

Figure 29: Accuracy and IMC attack results on ResNet-18 teacher models measured on the pneumonia test data set.



(a) Accuracy **(b)** Attack success rate

Figure 30: Accuracy and IMC attack results on simple CNN teacher models measured on the pneumonia test data set.

trigger size 5×5. In other words, for the tested trigger sizes there is no correlation between backdoor trigger size and clean accuracy. Even larger trigger sizes, or a change in the position of the trigger might lead to a more significant drop in clean accuracy, e.g., if important features of the image are blocked by the trigger. However, it is unlikely to encounter these, as the attacker would always craft the attack in a way such that it is unlikely to be discovered.

At the same time, the trigger size has a significant impact on the success rate of the backdoor attack. The success rate is the percentage of test data points with a label different from the attack target class, which were classified as the target class after the addition of the trigger. An increase in the trigger size increases the success rate of the respective attack (see Figure 28b). These two observations imply that the adversary can optimize the success of the backdoor attack by adapting the trigger size up to a certain extent without having to fear increased backdoor detectability based on the accuracy drop.

In general, both considered pattern-key attacks and all trigger sizes are highly effective with respect to the ResNet-18 teacher models (see Figure 27a). We observe an attack success rate of above 90% for all trigger size conditions. The IMC attack is even able to reliably achieve a 100% success rate for the smallest considered trigger size of 5×5 pixels (see Figure 29b). The models trained from scratch with the simple CNN architecture seem to be harder to attack with BadNets and IMC. For example, the 5×5 pixel IMC trigger fails to attack the teacher models, that is, the attack success rate lies close to 0% in this case (see Figure 30b). However, both attacks achieve almost perfect attack results for 20×20 pixel triggers. These first results might indicate that the teacher models are more vulnerable to backdoor attacks when they are based on complex model architectures. However, it has to be taken into account that the comparability of the two architectures might be limited here, specifically, it is unclear whether the differences in the task performance of the two compared model architectures might affect the attack success rate; for instance, it is possible that the lower attack success rate for the simple CNN model is a result of the lower model accuracy.

Next, we evaluate the transferability of BadNets and IMC to the student model. The accuracy of poisoned and clean student models on the COVID-19 test set ranges from around 80% to 90% (depending on the number of frozen layers) for the ResNet-18 architecture, and from around 70% to 80% for the simple CNN architecture (see Table 3). If more layers are frozen, the accuracy is lower, indicating that the pneumonia teacher task and the COVID-19 task, despite their similar nature, do require different features. This might be the consequence of systematic differences in the data due to different data sources. The accuracy improves if more layers can be retrained, which also increases the computational effort, and requires that a sufficiently large student training data set is available. In general, we cannot detect significant differences with respect to the clean accuracy between poisoned and clean student models. This again underlines that the victim cannot rely on accuracy as a metric for backdoor detection.

For both model architectures the attack success rate drops drastically if no layers are frozen during the student training, i.e., if all features are trainable (see Table 3). In the case of the ResNet-18 models, the average attack success rate of BadNets and IMC stays below 2% for all trigger sizes. Hence, the transfer learning process erases the injected backdoors as long as

Table 3: Report of **transferability of BadNets and IMC backdoor attack** for varying trigger sizes after student training (COVID-19 task). "Acc." refers to the average accuracy of the student model, "Succ." refers to the average attack success rate and "Std." refers to the empirical standard deviation of the attack success rate. All numbers are calculated on the basis of 10 experimental runs and the COVID-19 test data set.

(a) Attack results for student models based on transfer of ImageNet pretrained **ResNet-18 architecture**.

		Trigger Sizes								
Attack	Trainable Features	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
BadNets	All	0.89	0.01	0.01	0.92	0.01	0.02	0.92	0.01	0.01
	Last	0.90	0.1	0.06	0.91	0.09	0.05	0.89	0.08	0.13
	None	0.8	0.66	0.38	0.79	0.86	0.13	0.8	0.76	0.4
IMC	All	0.92	0.0	0.01	0.9	0.01	0.01	0.88	0.02	0.02
	Last	0.89	0.18	0.1	0.91	0.15	0.13	0.89	0.19	0.14
	None	0.8	0.96	0.07	0.8	0.85	0.28	0.8	0.61	0.35

(b) Attack results for student models based on transfer of **simple CNN architecture**.

		Trigger Sizes								
Attack	Trainable Features	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
BadNets	All	0.76	0.01	0.01	0.76	0.02	0.02	0.79	0.08	0.07
	Last	0.81	0.04	0.02	0.84	0.16	0.09	0.81	0.44	0.12
	None	0.7	0.57	0.09	0.72	0.91	0.07	0.72	0.98	0.03
IMC	All	0.78	0.01	0.01	0.77	0.03	0.03	0.79	0.2	0.10
	Last	0.76	0.01	0.01	0.77	0.64	0.24	0.78	0.95	0.04
	None	0.72	0.01	0.03	0.72	0.83	0.12	0.73	1.0	0.0

Table 4: Report of the accuracy of the **clean student models** and the attack success rates of a default BadNets attack for varying trigger sizes. "Acc." refers to the average accuracy of the student model, "Succ." refers to the average attack success rate and "Std." refers to the empirical standard deviation of the attack success rate. All numbers are calculated on the basis of 10 experimental runs and the COVID-19 test data set.

(a) Accuracy and attack success of a BadNets attack computed on the clean student models based on the **ResNet-18 architecture**.

		Trigger Sizes								
Attack	Trainable Features	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
BadNets	All	0.91	0.02	0.02	0.91	0.01	0.02	0.91	0.01	0.02
	Last	0.88	0.01	0.01	0.88	0.01	0.01	0.88	0.01	0.01
	None	0.80	0.00	0.00	0.80	0.00	0.00	0.80	0.01	0.01

(b) Accuracy and attack success of a BadNets attack computed on the clean student models based on the **simple CNN architecture**.

		Trigger Sizes								
Attack	Trainable Features	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
BadNets	All	0.76	0.00	0.00	0.76	0.00	0.00	0.76	0.03	0.03
	Last	0.77	0.01	0.01	0.77	0.03	0.02	0.77	0.07	0.04
	None	0.72	0.01	0.01	0.72	0.02	0.02	0.72	0.03	0.02

the student training process adapts all feature layers of the teacher. However, as soon as we freeze parts of the teacher, we see different results. For example, without retraining the feature layers (i.e., only retraining the classifier for the new task) the success rates stay above 60% for both BadNets and IMC for the ResNet-18 models. The IMC success rate of a 5×5 trigger on a fully-frozen ResNet-18 is approximately 96%. Even when allowing the retraining of the last feature block, the attack success rate of IMC still ranges from 15% to 20% for the different trigger sizes. In contrast, the attack success rate in this condition is lower in the BadNets attack (8% to 10%), indicating that optimized triggers might increase the resilience of the backdoor to a slight retraining of the model.

In the case of the simple CNN model, we observe overall an even higher transferability. Specifically, for the trigger sizes 10×10 and 20×20 the attack success rates are almost always above the ones reported for the ResNet-18 models - independent of the attack and number of frozen layers. Large IMC triggers are able to retain a 20% success rate in the case of a full retraining of all feature layers and even 95% if the last feature layer is retrained. Again, this is significantly higher than for the BadNets attack where 44% attack success rate is achieved in the case of a retraining of the last feature layer. Thus, transferability appears to be influenced by the complexity of the used model as well as by the used trigger size. However, there might be additional factors such as the generalizability of the trained model which is expected to be lower here for the simple CNN model due to the lower accuracy that this model achieved for the task. In the context of this study, it cannot be clearly inferred which factors are the main driving factor influencing transferability. An indication could be provided by Table 4, which shows the accuracy of the clean student models[12] and the attack success rate of a default BadNets attack with different trigger sizes on these clean models. It can be seen that accuracy is comparable to the accuracy of the attacked models and that the attack success rate is generally low. A slightly higher attack success rate is found for a larger trigger size, indicating that inserting a larger trigger increases the likelihood that a sample is randomly misclassified as the target class.

Overall, these results indicate that there exists a significant security risk of backdoor attacks in a transfer learning setup, as long as the victim relies on the common practice of freezing large parts of the teacher. The risk is higher in the case that an attack is used which optimizes the trigger such as IMC, as there seems to exist a certain resilience of the backdoor to the retraining of the last feature block. The effect of the teacher model architecture on the vulnerability to backdoor attacks is complex. Our first results suggest that larger triggers are needed to inject backdoors into teacher models with limited capacity. However, once injected, the transferability of these backdoors to the student model seems to be higher. Furthermore, the results indicate that backdoors of the simple CNN architecture are more resilient to the retraining step of the transfer learning process. However, it has to be noted here that our results do not allow to make a direct comparison between the "last" retraining condition of the two model architectures because the last block of the ResNet-18 architecture and the last block of

[12] Note that there is no difference for the accuracy between trigger sizes as the clean models do not use poisoned data and, thus, do not depend on any attack parameters.

the simple CNN architecture are different in terms of their absolute size.

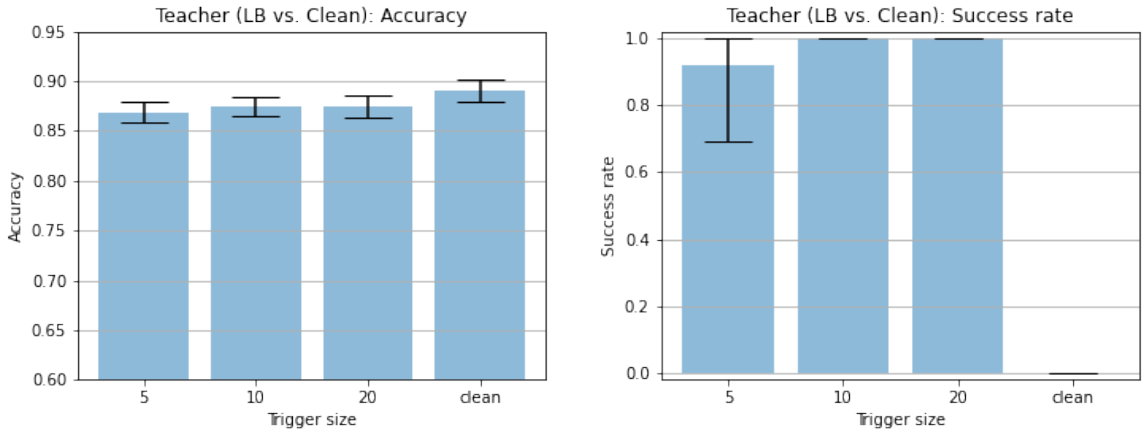
In summary, the results demonstrate that already simple pattern-key approaches - like BadNets and IMC - pose a security threat for student models, specifically in the common case that parts of the teacher model features are frozen during student training.

Experiment 2 In order to evaluate the success of the latent backdoor attack during teacher training, we take a closer look at the not yet postprocessed model, that is, we analyze the model where the target class is not removed from the classification layer yet. Note that the victim does usually not have access to this version of the teacher model. Instead the postprocessed model, where the target class is removed, would be provided as a teacher model to the victim.

Comparable to the results of BadNets and IMC, we see that the latent backdoor attack does not affect the clean accuracy of the model significantly (see Figure 31a and Figure 32a). In all experimental runs there is only a slight accuracy drop compared to the clean teacher training, however, the drop is more pronounced when using the simple CNN architecture as compared to the ResNet-18 architecture. However, it can be expected that this accuracy drop decreases further after the postprocessing step of the latent backdoor attack. One interesting observation in this context is the tendency that, in contrast to the findings of the BadNets attack, the accuracy drop is slightly smaller for a latent backdoor attack with a larger trigger size, i.e., for a trigger of size 20×20 pixels. It might be that a large trigger pattern makes it easier for the model to adhere to both objectives of the adversary during teacher training, namely obtaining a high clean accuracy and a successful backdoor injection by crafting a stealthy trigger pattern.

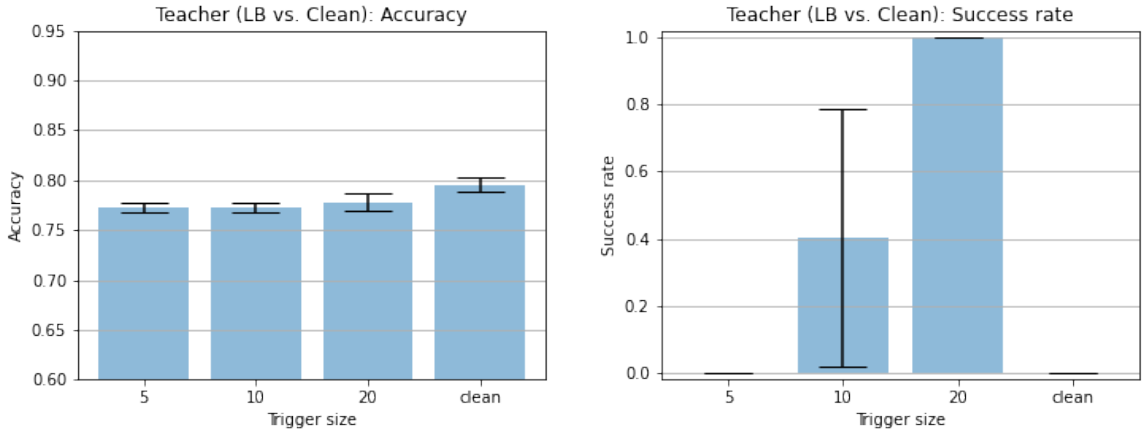
With respect to the success rate of the latent backdoor attack, we see that a trigger of size 5×5 pixels is already strong enough to force 90% of the test images into the target class for the ResNet-18 models (see Figure 31b). Analogously to the results of Experiment 1, the simple CNN architecture (see Figure 32b) is harder to attack during teacher training. Here, the latent backdoor attack fails for 5×5 pixel triggers, even 10×10 pixel triggers only obtain a success rate of 40%. Overall, the results still show that the latent backdoor attack is able to inject the COVID-19 backdoor into the teacher model trained on the pneumonia data set.

Next, we take a look at the student training, and the question whether this training procedure activates the hidden backdoor of the latent backdoor attack. In Table 5 it is shown that the latent backdoor attack is indeed able to cause the desired targeted misclassification. The attack performance is slightly weaker than the BadNets and IMC results. However, it should be noted that the latent backdoor attack injected a backdoor for a class, which has not been part of the pneumonia data set and from which only a limited amount of examples are made available during the generation of the attack (here, 100 COVID-19 images are used). Thus, the underlying threat model is more complex in the case of the latent backdoor attack. An additional reason for the limited effectiveness of the latent backdoor attack (compared to BadNets and



(a) Accuracy **(b)** Attack success rate

Figure 31: Accuracy and latent backdoor attack results on ResNet-18 teacher models measured on the pneumonia test data set.



(a) Accuracy **(b)** Attack success rate

Figure 32: Report on accuracy and latent backdoor attack results on simple CNN teacher models measured on the pneumonia test data set.

Table 5: Report of **transferability of latent backdoor (LB) attack** for varying trigger sizes after student training (COVID-19 task). "Acc." refers to the average accuracy of the student model, "Succ." refers to the average attack success rate and "Std." refers to the empirical standard deviation of the attack success rate. All numbers are calculated on the basis of 10 experimental runs and the COVID-19 test data set.

(a) Attack results for student models based on transfer of the **ResNet-18 architecture**.

		Trigger Sizes								
Attack	Trainable Features	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
LB	All	0.89	0.0	0.0	0.9	0.01	0.01	0.92	0.01	0.01
	Last	0.87	0.12	0.15	0.9	0.1	0.1	0.89	0.12	0.18
	None	0.81	0.47	0.36	0.81	0.53	0.36	0.81	0.62	0.35

(b) Attack results for student models based on transfer of the **simple CNN architecture**.

		Trigger Sizes								
Attack	Trainable Features	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
LB	All	0.78	0.0	0.01	0.82	0.01	0.02	0.78	0.3	0.22
	Last	0.79	0.0	0.0	0.79	0.2	0.2	0.78	0.98	0.07
	None	0.72	0.0	0.0	0.74	0.47	0.42	0.78	1.0	0.01

IMC) might be the fact that this attack requires much more parameters to be set, as the model has to be pretrained and postprocessed. Further experiments have to be run in the future to analyze the effect of the different parameters on the attack success rate.

Also for the latent backdoor attack, the transferability is strongly affected by the number of trainable features (i.e., the number of frozen layers). The attack results presented in the latent backdoor attack paper [138] are based on the assumption that the victim freezes all layers before the injection layer. In our experiments, this layer is the *flatten* layer directly after the last feature block. Not surprisingly, Table 5 shows that when retraining the feature layers partially or completely, the transferability is significantly reduced. When all feature layers are frozen, the latent backdoor displays a similar transferability as BadNets and IMC. Implanting the backdoor at an inner feature layer, as compared to the classification layer, thus, might improve the transferability further for student training with more trainable features.

Furthermore, we observe that the latent backdoor attack appears to transfer better for large trigger sizes. For example, it obtains a success rate of 62% for 20×20 pixel triggers for the ResNet-18 (all feature layers frozen), and an even more convincing 100% for the simple CNN

architecture. However, note that this result might be partially caused by the trigger-dependent attack success rate of the corresponding teacher models. Another remarkable finding is that similarly to the high resilience of the IMC attack to the retraining of the last layer for large triggers, the attack success rate remains at 98% for the simple CNN architecture even when the last feature layer is retrained.

Summing up, the latent backdoor attack can be successfully applied to the medical imaging use case. The attack facilitates targeted backdoors for classes which have not been part of the teacher training task. However, from the perspective of the adversary, the performance of the attack is not yet fully satisfactory. Due to the inherent complexity of the latent backdoor algorithm, further experimental studies are needed to assess the full potential of this attack approach.

Experiment 3 In the first two experiments, we observed that the student training itself can already be viewed as an effective defense method against injected backdoors in some cases. The student training procedure reduces the attack success rates drastically as long as the victim retrains at least a few of the feature blocks, which is a reasonable approach under the assumption that the victim has a sufficiently large data set available for retraining and possesses sufficient computational resources. However, the student training process is not sufficiently successful to defend against backdoor attacks if most trainable features of the teacher model are frozen. Since this is often the case in practice, we investigate further defense methods for the demanding threat model where the teacher model's training parameters are largely frozen.

In our experiments the **Fine-Pruning defense** is executed with respect to the teacher model, i.e., the victim tries to use Fine-Pruning on the teacher in order to make sure that only unpoisoned or defended models without a backdoor enter the student training. In the current implementation the Fine-Pruning defense is allowed to make use of the whole clean pneumonia training data set. Note that this might be unrealistic for several medical imaging use cases, thus, we recommend to run further experiments with a limited amount of teacher training data in the future.

To evaluate the performance of Fine-Pruning, the accuracy and attack success rate of the defended teacher model (on the pneumonia test data) are evaluated for BadNets and IMC. For the latent backdoor, Fine-Pruning is performed on the postprocessed model from which the target class was removed, similarly to how a victim would encounter the model. As the target class is not included in the teacher model, the attack success rate cannot be evaluated directly, instead, for this specific case, the evaluation is based on the student model which is derived from the defended teacher model. The analysis of the previous experiment has shown that transferability is highest when all feature layers are frozen, thus we only consider the case where all feature blocks are frozen and only the classification layers are adjusted.

Before coming to the results of Fine-Pruning on poisoned models, let us shortly discuss the effect of Fine-Pruning on clean teacher models. The defender cannot know beforehand

Table 6: Report of Fine-Pruning (F-P) defense for varying backdoor attacks and trigger sizes **using the ResNet-18 architecture**. "Acc." refers to the average accuracy of the student model, "Succ." refers to the average attack success rate and "Std." refers to the empirical standard deviation of the attack success rate. All numbers are calculated on the basis of 10 experimental runs and the whole test data set.

(a) Attack results for defended and undefended **teacher models** based on ResNet-18 architecture.

		Trigger Sizes								
Attack	Defense	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
Bad-Nets	None	0.88	0.97	0.02	0.88	1.0	0.0	0.87	1.0	0.0
	F-P	0.8	0.0	0.0	0.85	0.01	0.01	0.84	0.01	0.01
IMC	None	0.89	1.0	0.0	0.89	1.0	0.01	0.89	1.0	0.0
	F-P	0.85	0.0	0.0	0.81	0.01	0.02	0.85	0.01	0.02

(b) Attack results for defended and undefended **student models** based on ResNet-18 architecture.

		Trigger Sizes								
Attack	Defense	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
LB	None	0.81	0.47	0.36	0.81	0.53	0.53	0.81	0.62	0.35
	F-P	0.82	0.22	0.16	0.82	0.2	0.13	0.81	0.32	0.24

Table 7: Report of Fine-Pruning (F-P) defense for varying backdoor attacks and trigger sizes **using the CNN architecture**. "Acc." refers to the average accuracy of the student model, "Succ." refers to the average attack success rate and "Std." refers to the empirical standard deviation of the attack success rate. All numbers are calculated on the basis of 10 experimental runs and the whole test data set.

(a) Attack results for defended and undefended **teacher models** based on the simple CNN architecture.

		Trigger Sizes								
Attack	Defense	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
Bad-Nets	None	0.76	0.64	0.13	0.77	0.91	0.01	0.79	0.98	0.0
	F-P	0.76	0.0	0.01	0.77	0.02	0.01	0.77	0.07	0.03
IMC	None	0.79	0.01	0.0	0.79	0.66	0.26	0.79	1.0	0.0
	F-P	0.75	0.0	0.0	0.76	0.04	0.02	0.75	0.53	0.17

(b) Attack results for defended and undefended **student models** based on the simple CNN architecture.

		Trigger Sizes								
Attack	Defense	5×5			10×10			20×20		
		Acc.	Succ.	Std.	Acc.	Succ.	Std.	Acc.	Succ.	Std.
LB	None	0.72	0.0	0.0	0.74	0.47	0.41	0.78	1.0	0.01
	F-P	0.72	0.0	0.0	0.73	0.14	0.24	0.75	0.51	0.16

whether a teacher is poisoned or not, thus Fine-Pruning should ideally not have a negative effect on clean models. In our experiments, Fine-Pruning decreases the average accuracy of a benign ResNet-18 teacher by around 8%, from 89% to 82%. Additionally, the empirical standard deviation of the accuracy is 10-times higher for the fine-pruned teachers (increased from 0.01 to 0.11). Depending on the use case, this accuracy drop might not be acceptable for the victim. However, it is likely that a longer fine-tuning phase in the Fine-Pruning defense (more than the used 10 epochs) can further decrease the undesired accuracy drop if a sufficiently large training data set is available.

Also for BadNets and IMC poisoned models, Fine-Pruning leads to a significant accuracy drop of 3% to 9%. On the other hand, Table 6 shows that Fine-Pruning reliably decreases the attack success rate on the teacher models based on the ResNet-18 architecture to almost 0% for all trigger sizes. For teacher models based on the simple CNN architecture, Table 7 shows that Fine-Pruning reduces the attack success rate in most cases to almost 0%. An exemption is the IMC attack with large trigger sizes where the attack success is with 53% still considerably high. Thus, the Fine-Pruning defense seems to be very successful on simple backdoor attacks but might be of limited success for large and optimized triggers. This seems to be particularly the case for less complex model architectures in the given use case.

Also when defending against the latent backdoor attack, the success of Fine-Pruning is limited. The attack success rate on the student model - which was derived from a defended teacher - remains considerably high for all trigger sizes and network architectures (Table 6(b) and Table 7(b)). However, the Fine-Pruning defense is able to at least halve the success rate for all trigger sizes. The reason for the only partial success might be that the latent backdoor is activated by the student training. Hence, the activation patterns that lead to the backdoor are not yet visible in the postprocessed teacher model which is the model that a victim receives from the attacker and which are considered for applying the Fine-Pruning defense. In the future, it has to be evaluated whether Fine-Pruning applied directly to the student model - instead of the teacher - improves the overall defense effect. One positive remark can be made with respect to the accuracy of the defended student models. We observe that there is no accuracy drop for the defended student models, thus Fine-Pruning is able to halve the attack success against the latent backdoor attack while maintaining the usual student accuracy level. This implies that the accuracy drop for the teacher models - due to Fine-Pruning - might not be so relevant for the defender who intends to train a student model using the fine-pruned model.

Given that the Fine-Pruning defense can not always reliably mitigate the backdoor, it is important to note that for the given use case it should always be carefully evaluated whether it might be a viable alternative to retrain the teacher model from scratch on clean data. While a full retraining is often not feasible due to computational constraints, it provides a guaranteed backdoor-free model, assuming that the user can be sure that the data set is clean.

The **STRIP defense** is designed to detect poisoned data points during run-time. In the given medical imaging use case, the victim is interested in deploying a strong COVID-19

Table 8: Report of STRIP defense for varying backdoor attacks and trigger sizes on COVID-19 student model (trained with all layers frozen) **using the ResNet-18 architecture**. "Acc." refers to the accuracy of the STRIP prediction of data points as benign or malicious (poisoned), "F1" refers to the F1 score, "Prec." refers to the precision, "Rec." refers to the recall. All numbers are calculated on the basis of 60 randomly sampled test images. A respective trigger is added to every sampled image, and both - the clean and poisoned - image are used for the STRIP analysis.

	Trigger Sizes											
Attack	5×5				10×10				20×20			
	Acc.	F1	Prec.	Rec.	Acc.	F1	Prec.	Rec.	Acc.	F1	Prec.	Rec.
BadNets	0.61	0.11	0.22	0.22	0.53	0.00	0.02	0.0	0.64	0.14	0.21	0.12
IMC	0.68	0.46	0.59	0.41	0.57	0.09	0.09	0.08	0.65	0.05	0.13	0.03
LB	0.7	0.04	0.09	0.03	0.69	0.0	0.02	0.0	0.62	0.12	0.28	0.08

Table 9: Report of STRIP defense for varying backdoor attacks and trigger sizes on COVID-19 student model **using the simple CNN architecture**. "Acc." refers to the accuracy of the STRIP prediction of data points as benign or malicious (poisoned), "F1" refers to the F1 score, "Prec." refers to the precision, "Rec." refers to the recall. All numbers are calculated on the basis of 60 randomly sampled test images. A respective trigger is added to every sampled image, and both - the clean and poisoned - image are used for the STRIP analysis.

	Trigger Sizes											
Attack	5×5				10×10				20×20			
	Acc.	F1	Prec.	Rec.	Acc.	F1	Prec.	Rec.	Acc.	F1	Prec.	Rec.
BadNets	0.64	0.15	0.32	0.10	0.70	0.57	0.79	0.46	0.72	0.64	0.83	0.53
IMC	0.83	0.09	0.10	0.10	0.66	0.38	0.60	0.30	0.94	0.94	0.92	0.97
LB	0.86	0.11	0.09	0.13	0.73	0.17	0.27	0.16	0.89	0.87	0.89	0.87

model with the help of transfer learning. Thus, it primarily makes sense to apply STRIP to the COVID-19 student models. To evaluate the detection defense, we sample 60 test images (of non-target classes), apply the backdoor trigger to every sampled image and run STRIP with respect to the student model on all these poisoned and unpoisoned images. Table 8 and Table 9 display the experimental results of STRIP for the ResNet-18 and the simple CNN architecture, respectively. For BadNets and IMC, we report a correct detection rate of 52% to 68% for the ResNet-18 architecture and 64% to 72% for the simple CNN. The accuracy of the STRIP predictions is measured as the percentage of correctly classified input data points as either malicious or benign. Data points are considered malicious if they contain a trigger which lead to a misclassification to the target class. However, as the accuracy score includes the correct identification of clean data points, this score does not necessarily reflect the success of the detection method. Instead, it is important to look at other indicative measures such as precision and recall of STRIP's prediction. It can be observed that these scores are rather poor for all attacks and trigger sizes, particularly, for the ResNet-18 architecture. Slightly better precision and recall scores are observed for smaller trigger sizes (for BadNets and IMC). However, all recall scores ranges between close to 0% and 41% which implies that a significant number of false negatives (compared to true positives) are generated by the detection mechanism.

In contrast, student models based on the simple CNN architecture, show better precision and recall scores, in particular for larger trigger sizes which have a higher attack success rate (see Figure 28b and Figure 30b). Particularly good scores are achieved for the IMC attack; but also for the latent backdoor attack a recall score of 87% is achieved. Thus, interestingly, an increase in attack success rate and an optimization of the trigger seems to lead to a better detection rate of STRIP for the simple CNN architecture.

Generally, we conclude that the STRIP defense seems to work better for the simple CNN architecture and shows the best detection results for optimized triggers which have a high attack success rate. However, for the complex ResNet-18 architecture one cannot be satisfied with the reported results. The reason for these poor results - also compared to the presented experiments in [158] - is not fully clear at this point. One contributing factor might be that the implementation of STRIP in `TrojanZoo` deviates slightly from the presented algorithm in [158] (see Section 2.2.1), in particular the calculation of the entropy threshold values is done differently. In the implementation, the input point is flagged as poisoned if the entropy is outside the $[0.05, 0.95]$ confidence interval of the clean entropy values. The clean entropy values are calculated on the basis of the 60 sampled test images. Hence, we have a two-sided decision threshold in the implementation compared to the one-sided boundary in the algorithm of the paper. Furthermore, the implementation directly couples the threshold calculation to the sampled input points, which should be flagged as poisoned or unpoisoned.

As the last defense, we evaluated the **Neural Cleanse model inspection defense** which tries to reverse engineer the triggers of a model containing a backdoor. Our investigation showed that this algorithm, although straightforward in nature, is difficult to adjust to work properly on the give use case. Specifically, it seems that the optimization procedure ends up in a local minimum early on and finds potential triggers which do not correspond to the actually

used trigger. One potential reason for this problem lies in the algorithm itself which has very little constraints to the optimization problem, particularly, the area of the mask does not have to be contiguous. Instead, the pixels of the trigger can be spread throughout the whole image. Hence, the number of potential masks and, thus, of potential triggers, is large. Therefore, there might exist multiple solutions that fulfill the optimization criterion. In the case of backdoor attacks which use a fixed trigger position and shape (and for BadNets, also a fixed trigger value), thus, it is possible that there are potential triggers that differ from the original trigger while still causing the desired misclassification. The convergence to an incorrect trigger might be promoted by several factors. For instance, the complexity of the ML model, as compared to the task plays a role. If there are more parameters than necessary to solve the task, it becomes more likely that certain neurons cause accidental backdoors. Another factor that likely plays a role in the given use case is that the teacher models only have two potential output classes, normal and pneumonia. A misclassification as the target class, thus, is easier to achieve than in a model that has a large number of output classes. An indication that this might be an issue for this use case is the observation that the metric measuring the misclassification accuracy consistently reaches the optimal value 1 already in the first epoch of the Neural Cleanse optimization. This indicates that it is *too easy* to find a corresponding mask and a trigger pattern which causes the desired misclassification. Also the size of the clean data set that Neural Cleanse uses for testing the misclassification affects how likely a wrong trigger pattern can be detected. The used data set here consists of 624 images which might be insufficient given the complexity of the model.

Finally, there are a number of additional parameters of Neural Cleanse which influence, for instance, the convergence speed and the influence of the trigger minimization on the optimization which have not been optimized in the scope of this project. Thus, it is possible that a further parameter optimization could make it possible to successfully apply this method.

An example for the trigger patterns that are found for an attacked simple CNN model by Neural Cleanse is shown in Figure 33. This example shows the actual backdoor trigger (here, the trigger is of size 20×20 and was crafted with the IMC attack) inserted into an example image on the left. On the right, the two triggers that were inferred for target class ”normal” and for target class ”pneumonia” are shown. It can be observed that neither of the inferred triggers is similar to the actual trigger. Despite being different from the actual triggers, an evaluation shows that inserting them into the images of the Pneumonia test data set (the same that was used to infer the triggers) leads to an attack success rate of close to 100%.

Similarly, we were not able to demonstrate the successful trigger reconstruction for all other tested models and attacks, indicating that this defense is not well suitable for the given use case. To improve the results of the Neural Cleanse algorithm, thus, it seems that a larger number of data for testing must be available. It is also possible that the algorithm works more reliably for models with a larger number of potential output classes, or with less complex models, as this would reduce the amount of potential local minima that likely hinder convergence here. An indication can also be found in the literature: An investigation of different backdoor defenses presented in [199] showed that the results of Neural Cleanse seem to be

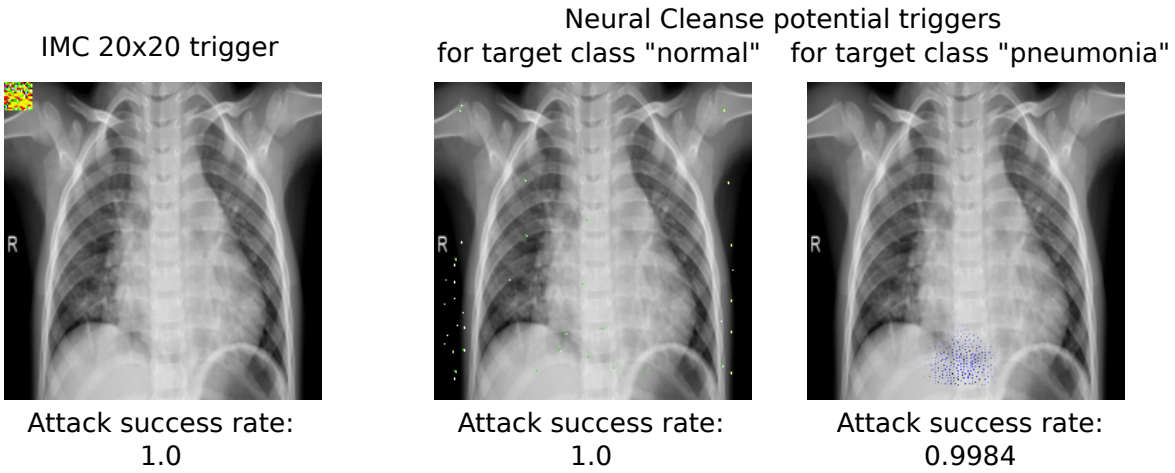


Figure 33: Example for an actual trigger of an IMC attack (left) and the potential trigger patterns reconstructed by the Neural Cleanse algorithm, listed together with the attack success rate.

affected by the hyper-parameter choice of the model training which could constitutes an additional problem source here.

Summing up, we have not found a defense that reliably protects the COVID-19 student models against all considered backdoor attacks. Simple retraining-based defenses, like Fine-Pruning and the student training itself, show promising results, but still fail under certain attacks / conditions, e.g., the latent backdoor attack or only partially frozen student models. The detection defenses - STRIP and Neural Cleanse - did not provide satisfactory results in the given use case. An additional comprehensive study is needed to evaluate whether this has to do with the underlying algorithmic ideas or the concrete implementations used in this project.

2.4 Discussion

The experiments of this coding project indicate that backdoor attacks pose a security threat in the medical imaging domain. The widespread usage of transfer learning in this domain gives adversaries the chance to inject backdoors into the specialized student model. We have seen that already simple pattern-key backdoor attacks - like BadNets and IMC - can transfer to the student model. The more sophisticated latent backdoor attack even allows targeted triggers for classes which have not been part of the pneumonia teacher data set. We found that retraining the model during the student training phase can remove injected backdoors if all trainable features are retrained. But the backdoor survives (although it might get weakened) if all or almost all layers of the teacher model were frozen during student training. State-of-the-art defenses, in particular STRIP and Neural Cleanse, fell short of expectations. So far, it is unclear whether these unsatisfactory defense results are a consequence of the medical imag-

ing use case, the used implementation and parameter choices, or the defense approaches itself. Additional in-depth empirical studies are needed to facilitate the development of reliable backdoor defenses.

Furthermore, it has to be noted that although we selected defenses which do not require unrealistic assumptions such as access to the full training data set, Fine-Pruning as well as STRIP require access to at least a portion of a clean data set for performing model fine-tuning or for establishing the characteristics of clean data, respectively. While this assumption holds in this use case, it is typically not a realistic assumption. This further decreases the potential of the available methods for establishing a successful defense strategy.

Another central challenge which became more and more apparent throughout this part of the project is related to the complexity and the interaction effects between the different steps of the transfer learning procedure. To answer relevant open research questions, as the ones given in Section 2.1, one has to always go through the process of training a teacher, attacking a teacher, training a student and defending a student model. Thus, every experiment run includes an immense number of parameter and algorithmic choices (i.e., training parameters like optimizer, epochs, learning rate and frozen layers), which significantly impact the experimental results and thus the derived answers to the posed research questions. For example, early on in our experiments we recognized that the transferability of a backdoor in the teacher depends strongly on the number of frozen residual blocks during the training of the student. Thus, if one wants to make a statement on the threat imposed by a specific backdoor attack method, it is not be sufficient to test the attack in a fixed transfer learning configuration. It is rather be necessary to test a diverse set of attack and training parameters related to the teacher and student model. Furthermore, the choice of how many layers should be frozen in a given use case to reach optimal performance is difficult to answer without experimentally testing the different parameter settings. A freezing of a large number of layers is often the state-of-the-art in the transfer learning literature. Particularly, it can be expected to lead to good results if the teacher and student tasks are similar to each other. However, we observed in our experiments that retraining more layers improved the performance of a student model significantly by around 10% for both network architectures. The reason could be that in our use case the student training data set is larger than the teacher training data set by a factor of 3 (14873 images as opposed to 5216). Thus, assuming no limitation of computational resources, it is recommendable in our use case to perform a full retraining of all feature layers to maximize performance while mitigating potential backdoors in the teacher model. This recommendation, however, might not hold generally for transfer learning scenarios and could negatively affect the performance in other use cases.

Hence, the derivation of reliable research results and recommendations for action has to be based on large scale experiments where a broad range of parameter configurations and algorithmic choices are evaluated. Due to the several dependencies of the different components, and the time that the training loops take for execution it was not possible in the scope of this project to perform an optimization of the whole parameter space, and several parameters which potentially impact the results had to be fixed in order to reach conclusive results

in a timely manner. This implies that also our results of this coding project, which are based on the outlined experimental plan of Section 2.2.2, have to to be viewed with caution, i.e., they should be seen as first indicators for a certain hypothesis.

3 Transfer Learning: Problems

A number of major problems have been identified in the course of the literature review and the practical investigations for ensuring the safe usage of pretrained models from external sources. In the following, we summarize the open questions regarding threats to system security, discuss the potential consequences of a successful attack and examine solutions and their effectiveness. In particular, this discussion focuses on defense strategies that a user can employ when using a pretrained model for the purpose of transfer learning.

For the purpose of transfer learning, the majority of pretrained teacher models come from two major sources: open source repositories provided by unknown or untrusted third parties, or from official MLaaS platforms. Regardless of origin, teacher models provided by external parties constitute a risk for information security. It is usually impossible to retrace how and with which training data the model was trained; thus, there is a realistic risk that the model is backdoored. While more popular model providers could, in theory, perform quality checks on training data to mitigate this risk, there always remains the vulnerability of adversarial attacks on the student model that were crafted by exploiting knowledge about the used teacher model. At the same time, the large number and overall high quality of external models for a variety of prediction tasks makes them an invaluable resource for transfer learning.

3.1 Security Threat: Backdoor Attacks

When teacher models are obtained from untrusted external sources, one security risk is that the model might contain a backdoor. While adversarial attacks as discussed in Section 3.2 also pose a security risk, the risk of backdoors is particularly relevant in this case. Sources are often deemed untrusted if there is uncertainty regarding the quality of the data and methodology used to train the teacher model. Therefore we focus on the risk of backdoor attacks in this section.

We split the overall problem of detecting and preventing backdoor attacks into three problems: To select a secure teacher model (Section 3.1.1), to set up a secure transfer learning setup (Section 3.1.2), and to mitigate backdoors that potentially remained active despite other security measures (Section 3.1.3). Finally, we summarize the findings from this section and discuss less common problems and risks that are still underrepresented in the literature (Section 3.1.4).

3.1.1 Problem: Assess Whether a Teacher Model Contains a Backdoor

Description A user would like to decide whether a model is safe to be used as a teacher model at an early stage of the transfer learning process, ideally, already during the selection of the teacher model. At this point, no computational effort has been spent yet, and swapping

one teacher model for another can be an effective way to ensure the safety of the transfer learning process.

The question of which teacher model to select is particularly relevant if a user targets at applying transfer learning for a commonly used task domain such as computer vision or natural language understanding. Due to the large amount of available teacher models, it is usually possible to find a similar, alternative model for a given use case if there are security concerns with a particular model. But even in a use case where a specific teacher model is required that might not be easily exchangeable, it is still important to know whether a security risk might exist or not, to decide whether additional measures (e.g., mitigation, Section 3.1.3) should be applied. In both cases it is crucial that a method exists to reliably differentiate between benign models and models with a risk of containing a backdoor.

Effects and Consequences Backdoor attacks are a realistic security threat under relatively weak assumptions about the adversary's capabilities because the attacker does not need access to the student model. The risk arises because the attacker knows the trigger that causes a (potentially targeted) misclassification of the input data, but the user is not aware of it. Even by inspecting the input data, it is often difficult or even impossible to detect harmful trigger patterns [129, 134, 135].

For the specific case of transfer learning, it is controversial in the literature to what extent a backdoor in the teacher model constitutes a risk for the student model. Whereas backdoor attacks are often argued to get washed away by retraining the model on clean data [136], there are ways to craft adaptive attacks that evade this fine-tuning defense [199]. It seems that the student training process might weaken the backdoor but does not necessarily deactivate it completely (cf. the results of our practical investigations in Section 2.3, specifically, Table 3): The input data containing the trigger might no longer cause a reliable misclassification but still alters the model output in some cases, reducing the model's predictive accuracy. Furthermore, in recent years, attacks were presented that were specifically crafted for the the transfer learning use case [138, 189]. We discuss such transfer learning specific problems in detail in Section 3.1.2.

Here, it is important to note that neither transfer learning (see Section 3.1.2) nor mitigation strategies (see Section 3.1.3) can guarantee the absence of a backdoor in the student model, given that the teacher model is backdoored. Thus, the detection of a backdoor in the teacher model would be desirable for reducing backdoor security risks.

The risk of a backdoor in the selected teacher model depends also on the given use case, in particular, on how the user plans to use the trained student model. Specifically, it plays a role whether the attacker would be capable of accessing the trained model and how manipulated input data can be provided to the model. For example, if the attacker can provide digital images directly to the infected model, it is easy to craft malicious inputs. If the model is indirectly fed with inputs, for example, via a (surveillance) camera system, the attacker would need

to craft the trigger in the physical world which might reduce the attack's effectiveness [200]. However, in the commonly assumed scenario that the attacker provided the teacher model to a victim in the first place it is likely that the attacker already considered ways of how to perform the attack after the deployment of the targeted model. Thus, an attacker would primarily target victims where an attack appears promising. In a scenario where an attack seems easier to launch or more promising, it would thus also be more likely to occur.

Solution Strategies and Their Effectiveness Within the four defense categories that we identified in Section 1.1.5, three categories focus on detecting a backdoor in a model or on data. These methods are, thus, candidates for assessing the security of a model.

Defenses detecting malicious training data (category 1) are only applicable if the user knows the data set the model was trained on. In the case of transfer learning, this assumption cannot be made as the attacker typically provides the model but not the corresponding training data – at least not the poisoned part of the training data.

Detection at inference time methods (category 2) aim at detecting backdoors by checking the input data for harmful intentions at run-time before it is provided to the (deployed) model. It is difficult to use these methods for confirming the security of the teacher model as the attacker would usually not attack the teacher model. Instead, these methods can be used after the student training during the deployment phase of the model. Specifically, these methods can check input data that is provided by external parties for unexpected behavior on the given model.

An example for such a method is STRIP [158] which uncovers input data which likely activate a backdoor in the model. In the original paper, STRIP was evaluated for BadNets-like triggers in three image data sets (MNIST, CIFAR10 and the traffic sign benchmark GTSRB) and achieved a false acceptance rate of less than 1%. In our practical experiments, the accuracy of the backdoor prediction by the STRIP defense was instead moderate when employed on the student model trained via feature-extraction transfer learning (an average accuracy of 62% to 94%, depending on the model architecture and the attack was found). One reason for the lower effectiveness of STRIP for the student models might be the implicit assumption of STRIP that backdoor triggers *reliably* fool the model into a wrong classification. If a backdoor has a low rate of efficacy on a model, then STRIP will have a lower success rate at detecting the backdoor. The observation that more effective attacks are more likely to be detected by input filtering defenses was also made by [137]. As such defenses in a transfer learning setup would be typically applied on the student model, it can be employed to supplement other mitigation strategies (see Section 3.1.2 and Section 3.1.3 for details).

Also SentiNet [157], another method that detects malicious input at inference time, might suffer from that limitation. SentiNet finds triggers based on image explanation methods that locate salient regions of the image. As a limited effectiveness of a backdoor might reduce the saliency in the trigger region (the network pays only little attention to the trigger), it is possible

that a less effective trigger evades detection while still causing misclassification. Furthermore, SentiNet can only detect triggers if they are small and localized but not if triggers span larger parts of the image. While small triggers are a common choice for attackers, this assumption cannot generally be assumed to be true as it is easy to craft a distributed trigger even with simple attacks [123].

NEO [161], another input filtering defense proposed in the literature, assumes that the trigger is small and that the trigger position is fixed for a single model. Their detection method furthermore is optimized for square-shaped triggers.

All discussed detection-at-inference-time methods measure the statistics of change in classification decision in order to detect malicious input samples. STRIP and SentiNet require a set of images that is similar to the training data and which is known to be benign in order to compute their statistics. In the next step, malicious output can be detected via an outlier detection mechanism. NEO assumes that a large batch of malicious input images is provided to the detection method at once, an assumption that might not be realistic in many scenarios. Likely, the attacker would disguise malicious images by mixing them with benign images. In this case, NEO would be unable to decide about the malicious intent of the detected trigger position.

The third category of defenses (cf. Section 1.1.5) are methods that aim at detecting the backdoor directly in the model. In the context of transfer learning, such model inspection defenses can be applied to detect backdoors directly in the *teacher* model. The first defense that was proposed for inspecting an ML model for backdoors is Neural Cleanse [159]. The intuition behind this method is to perform an optimization to detect small perturbations that reliably cause a misclassification as a target class. While the results of the study of the original authors show that Neural Cleanse can reliably detect backdoored models and can reconstruct the trigger for a variety of computer vision use cases [159], our investigations with a medical imaging use case indicated that the proposed optimization strategy has weaknesses (cf. Section 2.3). In particular, we found that the optimization method reconstructed triggers that consistently caused a misclassification but differed significantly from the original triggers. In fact, one weakness of this method is that it might not necessarily find trigger patterns which correspond to maliciously injected triggers but might also discover triggers which the model naturally formed given the training data. While such natural triggers could constitute a security risk (cf. universal adversarial perturbations [201]), it is different from a backdoor that is defined as a deliberately implanted trigger. Another potential problem with Neural Cleanse is that the method minimizes for trigger size and, thus, might underestimate the actual trigger size or not detect larger triggers.

This shortcoming of the Neural Cleanse method has also been noted and addressed by Guo *et al.* who developed an improved version of Neural Cleanse called TABOR [126]. Their idea is to alter the optimization criterion in order to filter out naturally occurring, non-malicious backdoors by making certain assumptions about the trigger. For instance, as the method is tested on image data, the assumption is made that the pixel contributing to the

trigger are located in a contiguous region and not spread across the whole image. However, again, attackers aware of such a defense could circumvent it by an adaptive attack.

Similar to most detection-at-inference-time defenses, Neural Cleanse as well as TABOR can only be applied if a set of images is available that is known to be benign and covers all classes of the original data set because it is required for performing the optimization of the trigger for the specific data set.

DeepInspect [160] alleviates this weakness by scanning models for backdoors without relying on any input data. It uses a similar intuition as Neural Cleanse: detecting a class to which all samples from other classes can be easily misclassified by perturbing the data. Instead of relying on available data, DeepInspect infers the training data set via model inversion. For recovering the trigger, DeepInspect does not optimize an objective function, but employs conditional generative adversarial networks to generate potential triggers which cause a change in the classification decision in the targeted model.

In summary, most defense methods make assumptions about the available resources of a defender as well as about the threat model. These assumptions are problematic for the application of defenses on a given use case. A summary of the assumptions of the defenses is provided and discussed in Section 3.1.4.

If a backdoor was discovered in a model, either the model can be discarded and replaced by an alternative model, or mitigation strategies can be employed to remove the backdoor from the model [159] as we discuss in Section 3.1.3. However, the user has to be aware that the detected backdoors might not constitute the only backdoors of the model, thus, the risk can be minimized but no success guarantee can be provided. In particular, according to the current state of the literature, it is not possible to prove mathematically the absence of a backdoor for large-scale deep learning models [202].

Future Outlook Similar to all ML topics, a large amount of literature is published every year, introducing new attack and defense strategies for backdoors. The field of backdoor attacks is a relatively new research field with the first publication suggesting targeted backdoor attacks being published in 2017 [129]. Thus, the race between attack and defense mechanisms that is prominent for system security topics is still in full swing. As new published defense mechanisms typically can be evaded by a specific attack modification, it is likely that the problem will remain in the future. Currently, the large majority of studies focuses on empirical investigations which are not able to guarantee the absence of a backdoor in a model. The question whether a formal verification is possible, therefore, is an important future research direction which might advance with the development of formal verification techniques for deep learning [203, 204].

3.1.2 Problem: Determining an Appropriate Parameter Setting for Transfer Learning

Description A student model that was derived from a teacher model is similar to the teacher in terms of the input data and the task that it solves, as well as with respect to the model architecture and the model parameters. In many aspects, this similarity is beneficial: transfer learning works well exactly because the weights from a teacher model that were tuned on large amounts of data carry important information about the task at hand [3]. Also, initializing the student model with the pretrained weights of the teacher saves valuable training time compared to training a large model from scratch.

The similarity between teacher and student model, however, also has its downsides. By reusing weights from the teacher, a backdoor might transfer from the teacher to the student model. The likelihood that a backdoor transfers depends on multiple aspects. First, the similarity between the teacher and the student task plays a role, specifically, the nature of the model's input and the output data. The output plays a role because backdoor attacks are typically targeted [129, 199], that is, they misclassify trigger input as a specific target class. If a backdoor targets at a class that is not included in the student model, the backdoor would not have the desired effect on the student model even if the neural activations of the backdoor get transferred. In contrast to the output, the role of the input, specifically, the distribution of the student training data, can be expected to play a smaller role. The reason is that the majority of backdoor attacks are input-agnostic which means that the backdoor trigger can be inserted into any image to cause a misclassification to the selected target class (also called all-to-one setting [199, 205]). Thus, a backdoor technically can be expected to work regardless of the provided input data. However, it is possible that the effectiveness of a backdoor decreases if the input data for the student model follows a different data distribution. In such a case, however, also the performance gain by employing transfer learning is reduced and, thus, it is less likely to be employed by a user.

A second aspect influencing the transfer of a backdoor from the teacher to the student model are the parameters that are used for transfer learning, specifically, it plays a crucial role how many layers are frozen during the student training.

Another aspect to be aware of is that there are attacks which specifically target the transfer learning use case [138, 189, 206]. These attacks often circumvent existing detection and mitigation strategies, for instance, the latent backdoor attack attacks a target label which is not yet present in the teacher model, rendering defenses such as Neural Cleanse which consider only existing output labels as backdoor targets ineffective.

Generally, it is not a viable option to swap the teacher model for one that performs an unrelated task because that would contradict the benefit of performing transfer learning in the first place. Therefore, a better option is to adapt the transfer learning setting in order to maximize protection against the backdoor while maintaining model performance.

Effects and Consequences Selecting the parameters for the transfer learning is, similar to any hyperparameter selection process, an optimization problem: the ideal parameters achieve good model performance while reducing the risk of transferring a potential backdoor to a minimum. The main parameter that is relevant for this decision is the layer-freezing parameter determining which weight connections in a neural network model remain fixed during student training and which should be adapted during student training. Tutorials for transfer learning (e.g., [207, 208, 103]) typically recommend to keep a considerable number of layers frozen during the student training. This is particularly useful if the teacher and student data sets are similar because features that were extracted in the early layers of the networks can be reused, speeding up the training process. Studies also demonstrate that in these cases the accuracy of the trained student model tends to increase if more layers are frozen [209, 210]. In contrast, the transferability from the teacher to the student model is higher if more layers are frozen, which includes the transfer of a potential backdoor. Optimizing for model security, thus, might negatively affect model performance, and vice versa.

It is, therefore, important to consider the choice of the transfer learning scenario carefully, not only with respect to model accuracy, but also with respect to a prevention of potential backdoor attacks.

Solution Strategies and Their Effectiveness The difficulty of the optimization problem stated above is that while the accuracy can be easily optimized by trying various parameter settings for the given use case, it is difficult to evaluate the influence of the parameter setting on a potential backdoor in the teacher model because the backdoor trigger is unknown. To our knowledge, there is no literature specifically addressing the problem of selecting a suitable transfer learning setting that achieves a trade-off between accuracy and training time on the one hand and robustness against backdoor attacks on the other hand.

One potential solution for an informed parameter choice for a given setup is to run experiments with an artificially added backdoor in the teacher which allows the user to perform an exhaustive search of potential settings to decide the ideal trade-off for the given use case. Similar to the procedure in our practical evaluations on transfer learning (cf. Section 2.3), different choices of frozen layers could be tested and the parameter choice could be optimized for model performance as well as for a small susceptibility to the artificial backdoor. However, further research is required to evaluate whether the prevention of one specific backdoor can also mitigate other potential backdoors.

Adapting the number of frozen layers in the student training can also be effective against transfer learning specific attacks which often come with assumptions about the number of layers that the user freezes during training. Particularly, it is assumed in [138, 189, 206] that the attacker knows which layers the user will freeze during student training. In the image domain, the latent backdoor attack [138] injects the backdoor at a specific layer of the network for which the attacker expects that the user will not retrain it during student training. Wang *et al.* [189] achieve robustness against retraining by injecting the backdoor into specific neu-

rons which are unlikely to change by using pretrained models and demonstrate the method's efficacy for transfer learning setups using image and time series data. This assumption that the attacker knows which layers the user is going to freeze is often realistic. Most transfer learning tutorials, or even the provider of a teacher model, often suggest a specific number for which the model is likely to achieve good results, and many users adopt this number without further optimization. Thus, a potential defense strategy would be to deliberately retrain more layers than suggested by the model's provider, if the computational costs and potentially accuracy drops are acceptable for the given scenario. However, further investigations are required in order to quantify the effectiveness of such a strategy.

While most of the literature on backdoor attacks focus on computer vision tasks, the risk of a backdoor also exists for other types of data input, for example, in natural language processing. Recently, BadPre [206] was proposed which targets at backdooring large-scale pretrained language models (e.g., BERT [78]) with the aim to manipulate the classification decision via a trigger word in the trained student (downstream) models. Their attack is untargeted and uses uncommon letter combinations as a trigger pattern. As these patterns are unlikely to occur in the student training set, the backdoor has a high probability to be transferred to the student regardless of fine-tuning. BadPre was shown to work independently of the specific downstream task (e.g., text classification or question answering tasks), and makes no assumptions about the user's downstream task, except for that it is derived via transfer learning from the infected teacher model. Also defenses proposed against backdoors in language processing models such as [211] can be evaded by BadPre.

Future Outlook We identified a significant need for further research with respect to the transferability of backdoors in a transfer learning setting. The initial estimation that a backdoor gets washed away during the student learning process [162] is not supported anymore by recent reviews [212, 120]. At the same time, new backdoor attacks have been developed in recent years targeting specifically at the transfer learning scenario, successfully circumventing available defenses. With the increasing demand for pretrained deep learning models it is likely that the relevance of this security issue will further increase in the next years.

In addition to ongoing research on the transferability of backdoor attacks for specific use cases, there is the need to investigate these aspects at a higher level: In particular, studies are missing which investigate the trade-off between accuracy and backdoor robustness in transfer learning. Better understanding the interdependencies could help to better derive recommendations for model safety in the future.

3.1.3 Problem: Mitigating Potential Backdoors in a Model

Description While in Section 3.1.1 we considered the option to prevent backdoor attacks in a transfer learning scenario by detecting them early on in the teacher model, this procedure cannot provide a guarantee of whether a model contains a backdoor or not. Therefore,

the user might decide to additionally prevent potential backdoors in the teacher model by applying mitigation techniques, either by adjusting the training setting (Section 3.1.2) or by employing mitigation defenses. To decide whether to employ mitigation defenses, the probability whether a backdoor might exist has to be weighted against the potential drop in performance that the application of a mitigation technique involves. Especially when running mitigation techniques on large-scale pretrained models typically a large student training data set is required to successfully retrain the teacher model.

Furthermore, the user might also want to apply mitigation defenses *because* the detection methods indicate an increased risk for a backdoor. As it has been pointed out in Section 3.1.1, it might not always be a viable choice for the user to search for an alternative teacher model with a lower risk of containing a backdoor, for instance, if there are no viable alternative teacher models available for the data or the use case at hand. In this case, mitigation defenses can be applied to minimize the risk of a backdoor in the model.

Effects and Consequences Removing a backdoor from a model changes the model's parameters: neurons might get pruned or their weights are changed to render the trigger ineffective in the model. This has an impact not only on the model's security but also on its performance. Thus, it is important to be aware of the chances and risks of applying such methods.

The impact that a mitigation technique has on model performance typically depends on whether a sufficiently large substitute training data set is available to the user. However, in a transfer learning scenario, such a data set is not available in most cases which motivates the application of transfer learning in the first place.

To decide whether or not to apply a mitigation technique, thus, a user would have to consider the use-case-specific security implications of a non-mitigated backdoor and of a potential loss in model accuracy. If mitigation is performed because a detection method predicted the presence of a backdoor, also the reliability of this detection method plays a role. For instance, if many false positives occur (i.e., trigger patterns are reported as suspicious but are not maliciously inserted and naturally occurring in the model) the mitigation of these non-malicious triggers would affect the model performance without bringing additional security against backdoor attacks [126].

Solution Strategies and Their Effectiveness Generally, mitigation techniques can be applied in two different ways: blindly, without using specific information on potential triggers, or in a directed way, to mitigate a backdoor that was detected beforehand.

Fine-Pruning was one of the first proposed backdoor mitigation techniques [162] and commonly applied in the literature (as indicated by more than 420 citations since 2018). It is a blind mitigation technique which combines a simple retraining of the model (cf. Section 3.1.2) with a pruning mechanism. Fine-Pruning was shown to perform well against standard backdoor attacks such as BadNets [162] with only a mild drop in accuracy. However, the assump-

tion about the separation of neurons being activated by either clean or malicious inputs can be invalidated by adaptive attacks [199], reducing the effectiveness of the defense. In our practical experiments (cf. Section 2.3), Fine-Pruning indeed showed to be successful in defending against the BadNets attack but not against optimized attacks such as IMC [141] or the latent backdoor attack [138], in particular for large trigger sizes.

Also, Fine-Pruning is sensitive to the hyperparameters of the model training. Specifically, Veldanda *et al.* [199] demonstrated that the assumption that clean and malicious input data activate distinct sets of neurons is violated in some hyperparameter settings of BadNets [123].

Targeted mitigation of a backdoor is generally preferable to blind mitigation because it has a smaller impact on model accuracy as, for instance, pruning can be performed in a target-oriented way. A prerequisite is that a detection has been executed which provides a sufficiently accurate estimation of the presence of the backdoor and, ideally, of the potential trigger pattern(s). The authors who proposed Neural Cleanse [159] combine their model inspection method with mitigation with the aim to prune specific neurons that contribute to the identified backdoor. Specifically, they propose to prune neurons that show the highest activation difference when clean or infected input is presented until the activation differences vanish. For the traffic sign benchmark attacked with BadNets, they find that pruning 30% of the neurons in the second-to-last layer decreases the attack success rate to 0% while reducing model performance by 5%. The accuracy loss can be minimized to less than 0.8% by pruning only 8% of the neurons with an attack success rate that is still below 1%. For a Trojan attack with an optimized trigger [132], the pruning mitigation strategy showed to be not successful due to an imprecisely reverse-engineered backdoor trigger. Instead, for these attacks the backdoor could be mitigated via unlearning: Adversarial training (a method usually applied to achieve robustness against adversarial attacks) is performed with 10% of the training data set of which 20% of the samples were infecting with the reverse-engineered trigger. As a result, the attack success rate could be reduced for all tested attacks and models to less than 6.7%.

A major problem of mitigation strategies such as Fine-Pruning and Neural Cleanse are the requirement of a significant amount of clean training data, either for performing a successful model retraining (for Fine-Pruning) or for testing reverse-engineered triggers by statistically computing the effect of injecting the trigger into clean data samples (for Neural Cleanse).

The authors of DeepInspect [160] try to eliminate this requirement by relying only on data that is generated from the potentially infected model. Similar to the unlearning mitigation suggested by the authors of Neural Cleanse, they mitigate the discovered backdoor by performing adversarial training: The generated input data serves as the clean data set, and the triggers generated by the generative adversarial network are inserted into a fraction of this data set to obtain a malicious data set. By training on this extended data set which is correctly labeled according to the desired behavior of the model, the model performance can be maintained while mitigating the effect of the backdoor. Their paper reports a 100% accuracy in differentiating between benign and infected models with no false positives or false negatives using five common image classification benchmarks (including ResNet-18 trained on ImageNet).

The mitigation strategy reduces the attack success rate to 7% to 10% without affecting model performance according to the authors.

Future Outlook While blind mitigation defenses entail a high risk of accuracy loss and do not provide any way of measuring whether all backdoors in a model were successfully mitigated, targeted mitigation strategies could be helpful for making teacher models more secure.

The model inversion technique that is used by DeepInspect in order to recover training data for the purpose of detection and mitigation of the backdoor is the only defense that tests the realistic use case scenario where no clean training data is available. While they find that the mitigated model still has a relatively large attack success rate with up to 10%, their study points into an important direction of future research, namely, to reduce the required resources of the defender to make the defense ready for an application to a real use case.

Potentially the technique of model inversion could be applicable also in combination with other defenses to reduce the required resources of the defender. Further research is necessary in order to test the feasibility and effectiveness of such a strategy.

3.1.4 Summary

In this first section, we looked at problems that a user faces when defending a non-trusted ML model that should be used for transfer learning against backdoor attacks. The user has to make a variety of decisions which depend on:

- the security requirement on the given use case,
- the capabilities and resources of the defender (e.g., sufficient computation power, or availability of training data),
- the threat model, determined by the goals and capabilities of the attacker (e.g., computational resources for implementing an attack, or access to the deployed model by the attacker).

Some of these aspects can be assessed or estimated by the user (e.g., whether the user has the resources for retraining the model to perform mitigation) while others can only be approximated (e.g., the attacker's capabilities would generally not be known).

Furthermore, most defenses impose assumptions regarding the resources of the defender and the capabilities of the attacker. Except for DeepInspect which uses a reverse-engineered data set derived via model inversion, all defense methods require some amount of data that is available to the model. The assumptions that some defenses make about the resources that are available to the defender can be listed as follows:

- The original data set used for training the teacher model is available to the user.
- A (reasonably large) substitute training data set of a similar data distribution as the original data is available to the user.
- A set of clean samples is available to the user.
- A set of malicious samples (or a data set with a known percentage of malicious examples) is available to the user.

Assumptions regarding the attacker's capabilities are the following:

- Assumptions about the used trigger are made (e.g., only small triggers are considered),
- The defense is limited to prevent one-to-all backdoor attacks (i.e., attacks only cause a misclassification as one specific target label),
- The defense only considers non-adaptive attacks.

Table 10 displays which assumptions the discussed defenses make regarding the defender resources and the threat model. Additionally, we list for which defenses it is known that the effectiveness of the defense correlates with the effectiveness of the attack (see column "correlation to effectiveness"). This property of a defense is particularly relevant in the transfer learning scenario because a backdoor in the student model often is less effective (cf. Section 3.1.2).

Table 10: Summary of the discussed defenses, the defense categories, and assumptions and properties of the defenses. An X indicates that this assumption is made by a specific defense. (X) means that the assumption is made to a certain degree. (?) means that there is not sufficient evidence in the literature to judge whether a limitation applies to a defense or not.

	Defense category				Defender resource assumptions				Threat model assumptions				
	Detection in training data	Detection in inference data	Detection in model	Mitigation	Original training data set	Substitute training data set	Known-benign data set	Batch of malicious data	Small trigger size	All-to-one backdoor attack	input-agnostic trigger	Non-adaptive attacks only	Correlation to effectiveness
ABS [154]	X				X					X	X	X	?
Activation Clustering [155]	X				X					X	X	X	?
STRIP [158]		X					X			X	X		X
SentiNet [157]		X					X		X	X	X		X
NEO [161]		X						X	X	X	X		?
Neural Cleanse [159]			X	X			X		X	X	X		?
TABOR [126]			X				X		X	X	X		?
DeepInspect [160]			X	X					?	(X)	X		X
Fine-Pruning [162]				X		X					?	X	

In terms of assumptions, it also has to be acknowledged that the large majority of backdoor defenses only focus on the most common scenario for backdoor attacks: input-agnostic all-to-one attacks where the trigger can be added to any input sample to cause a misclassification as a single output label. These attacks are mostly considered because they can be easily mapped to the attacker's motive to cause misclassification as a specific class using an arbitrary input pattern. However, also other, less common types of backdoor attacks have been considered in the literature which we briefly discuss here.

All-to-all attacks have no fixed target output label, but swap all labels with another label. In the paper presenting the BadNets attack [123], the authors also suggest an all-to-all variant of such an attack where all output labels of the model are rotated (i.e., label i gets misclassified as label $i + 1$ in presence of the trigger) [123]. However, the motivation of an attacker to launch this kind of attack is less clear; the aim would usually be to simply cause an arbitrary misclassification to impair model performance. A more advanced version of a backdoor that can target multiple output labels is the study from Xue *et al.* [213]. They propose an attack where a trigger of different intensities can control to which specific output label the input is misclassified.

This method opens up the possibility for the attacker to target different output labels at runtime which makes it useful for practical application. The method was demonstrated by the authors to be robust against the Activation Clustering and Neural Cleanse defenses. Recently, the authors extended their method to use a more stealthy, imperceptible trigger pattern [214].

While most attacks and defenses discussed so far are input-agnostic attacks, there have been a few works presenting sample-targeted backdoor attacks. Instead of a trigger that is added to a benign input sample to make it malicious, for such attacks a specific test input instance is used that should trigger the desired misclassification. Such attacks are typically launched as clean-label attacks to avoid having to insert obviously incorrectly labeled data points into the training set [130, 139, 140]. The stealthiness of the manipulated training data allows for manipulating a larger percentage of the training data, and potentially also broadens the danger of receiving an infected model even from trusted sources as the person training the teacher model could have unknowingly used infected training data. Defending against such attacks is difficult because almost all defenses make the assumption that the trigger is input-agnostic. As discussed in 3.1.1, all detection defenses use some kind of mechanism where potential triggers are inserted into test images to check whether this causes a misclassification [159, 158, 157]. In [215] it was found that even methods investigating the network activations caused by the training data such as Artificial Brain Stimulation (ABS) are ineffective as defenses. Fine-Pruning with clean data might be able to (partially) mitigate the attack, but has not been tested in the available literature. There is only one recently proposed defense, CLEAR [215], that can detect backdoored models in the case of a sample-targeted attack. The authors propose to check the convex hull of the training samples to make sure that no clean-label poison is used in the data set. This method, however, requires access to the training data set that was used for creating the model, thus, it is not applicable to the use case of transfer learning.

All in all, defending against backdoor attacks still poses an unsolved problem as there are no established state-of-the-art approaches available that guarantee a sufficient level of security. Moreover, there exists only a limited set of tools or open-source libraries for testing against backdoor attacks, making it a challenging task even for ML experts with no specific training in ML security to defend against those attacks.

3.2 Security Threat: Gray-box Adversarial Attacks

In the following, the risk of gray-box adversarial attacks on the student model is considered. This exists not only for student models trained with teacher models from untrusted sources, but also for those trained from quality-checked teacher models. Unlike the risk of backdoor attacks, the security vulnerability of adversarial attacks comes from the accessibility and popularity of the used teacher model.

3.2.1 Problem: Defining Characteristics and Capabilities of Relevant Adversaries

The majority of literature about adversarial attacks in transfer learning is about image classification, in which the most commonly used transfer learning technique is feature extraction transfer learning (cf. Section 1.1.1). In this application, the student model is a copy of all but the final layer of the teacher model, with a dense classification layer added as a final layer. All copied layers are frozen during training. As a result, the student model closely resembles the teacher model in architecture and neuron activations. In this discussion, we mainly draw on literature from the image domain to assess the vulnerability of student models to adversarial attacks. Furthermore, we focus on feature extraction transfer learning which can be considered to constitute a particular vulnerability due to the similarity of the teacher and the student model. Analogously to the discussion in Section 3.1.2, it is likely that the risk of transferability drops if the student and the teacher model become dissimilar.

Description To define the capabilities and characteristics of a potential adversary, the user must define a realistic attack scenario for feature extraction transfer learning. In a transfer learning scenario, the most realistic and interesting attack settings would be *gray-box*: an adversary would have white-box access to the teacher model, and black-box access to the student model. Specifically, the attacker does not have access to the student model's training data set and parameters but knows the training data and architecture of the teacher model, and therefore would have a rough idea of the student model's architecture. In this setting, the transferability between the teacher and the student becomes an important tool for the adversary to craft a successful attack. In contrast, in a pure white-box setting, the adversary could directly craft adversarial attacks for the student, whereas in a pure black-box setting, the adversary would need to rely on methods such as crafting a substitute model via model querying.

Although white-box access of a proprietary teacher model is not guaranteed, it can be easily achieved in many cases. An adversary could access a teacher model hosted by a MLaaS platform by posing as a user training his own student model. Many platforms offer free or low-cost trials of their services, giving an adversary a low barrier to entry. Upon entry, an attacker can see which student models a platform offers, which tend to be a limited number for each task [115].

Even if attackers do not know which teacher model is used by a student, they can learn this using a fingerprinting method proposed by Wang *et al.* [115]. Given access to a teacher model, an attacker could construct a fingerprint image that distorts the output of a student model if it is trained from the teacher model. By querying the student model with the fingerprint image of each possible teacher model, an attacker could identify the teacher. Privately-purchased teacher models that are not available on MLaaS platforms cannot be accessed with such ease. However, if given access to queries of the teacher, an attacker can perform a black-box attack, for example by training a local substitute model that mimics its behavior, as detailed by Papernot *et al.* [216]. Black-box attacks are even possible without training a substitute

model and just by relying on output labels [217].

Despite black-box access to the student model, an adversary can craft an attack due to the *transferability* property of adversarial examples: adversarial examples developed from one model can often fool another model [218]. Due to their similarity, the teacher can serve as the substitute model that allows the adversary to generate attacks on the student model.

Even if the student task differs from the teacher task, student models share much of the teachers' architecture and neuron activations, specifically when using the teacher model as a feature extractor. As a result, an adversary can perform a targeted attack on the student as described by Wang *et al.* [115].

Given a target image, an adversary could generate perturbations that mimic its representation at any layer K in the teacher model. If at least K layers copied from the teacher model are frozen during training of the student model, then such perturbations would remain unchanged when passed through the student model. Using experiments, the authors found that for both targeted and untargeted versions of this attack, the attack has the highest success rate when it generates perturbations that mimic the penultimate layer of the teacher model.

To summarize, a user who wishes to train a student model via transfer learning must be aware of the following:

- An adversary can, in many cases, determine the identify of and access a teacher model.
- Adversarial attacks often transfer from one model to another. Thus, with white-box access to the teacher model, an adversary can craft adversarial examples that might transfer to the student. This is often true even with very limited information about the student model.
- In transfer learning, freezing most layers of a teacher model significantly cuts the cost of training a student model. This becomes a vulnerability when an adversary gains access to a teacher model, as perturbations transfer more easily between models with similar architectures.

Effects and Consequences The realistic threat of such an adversarial attack on a student model depends, first and foremost, on the attacker correctly identifying and accessing the teacher model. In essence, the vulnerability of the student model to gray-box adversarial attacks is strongly tied to the vulnerability of the teacher model.

If an adversary succeeds in getting white-box access to a teacher model, the threat of an attack occurring depends on the model deployment settings. If the student model is deployed in purely digital settings with few restrictions on user access, then an adversary can input adversarial examples directly into the model. The likelihood of an adversarial attack on this model is, therefore, tied to the security of the system, i.e., if it detects and blocks unusual

activity. Examples of such deployment settings would be spam or malware detection models, which exist entirely in a computer [219].

On the other hand, a model could be deployed in the physical world, which poses a different set of rules. This setting is, for instance, applicable to security systems that use cameras to capture input directly from the physical environment. In this setting, an adversary must place the adversarial example in the physical environment, which is then read by a camera. Overall, the literature suggests that such adversarial attacks are more challenging for an adversary to perform, as real-world settings are often difficult or impossible to perturb [219].

Solution Strategies and Their Effectiveness The ideal solution to the vulnerabilities of transfer learning regarding gray-box adversarial attacks is to use a teacher model with low accessibility. Oftentimes, this is infeasible. In fact, a user who would opt for transfer learning often does so due to resource constraints. The literature focuses on two kinds of solution strategies. The first strategy is to make the student model robust to adversarial examples. The second is to take actions that block the transferability of adversarial examples from teacher to student.

A well-researched method for making a model robust is to perform adversarial training, or training a model with a mixture of clean and adversarial examples and with a modified cost function. This procedure can be seen as minimizing the worst-case error when the data is perturbed by an adversary [106]. In the context of transfer learning, a user can perform adversarial training on a student model by augmenting the student training data set with adversarial examples, some of which could be generated from the teacher model. Despite its success in empirical studies, adversarial training does not guarantee robustness, as adversarial examples following a different distribution can be generated. For example, Zhang *et al.* [220] find the existence of "adversarial blind spots" in low-density areas of the input data distribution. Examples in this region are vulnerable to adversarial attacks, and often missed by adversarial training. Furthermore, Goodfellow *et al.* [106] finds that adversarial training reduces but does not stop the transfer of adversarial examples from one model to another. In one of their experiments, they trained a convolutional maxout network on a pre-processed version of CIFAR-10, first with and without adversarial training. The adversarially trained model had a 19.6% error rate on transferred adversarial examples, while the model without adversarial training had a 40.9% error rate.

Along similar lines to this solution, Shafahi *et al.* [221] considers the use of a robust feature extractor to create robust student models. In transfer learning, this could mean to use adversarial training on a teacher model, or to select a teacher model that has already been made robust by adversarial training. By building a student model from such a teacher model, the student model would inherit robustness. The authors find through empirical evidence that, indeed, transfer learning preserves robustness of a robust source model. However, the student model would be less accurate on clean samples compared to student model trained with teacher models that did not have adversarial training.

The second solution suggested in the literature is to block the transferability of adversarial examples. This could be done by adapting the student model to make it look less like the teacher model. The simplest way to do so is in training: a user can freeze fewer layers. As discussed in Section 3.1.2, this could also defend against potential backdoor attacks. However, this can come at the cost of a decrease in performance of the model. Additionally, freezing fewer layers requires additional training data and computational resources.

Future Outlook There is much research on adversarial attacks, especially in the context of image classification, for white-box and black-box attack settings. Research on adversarial attacks specific to transfer learning and gray-box settings is scarce. A user exploring the capabilities of a potential adversary, however, can draw on existing literature on important related topics.

First, there is active exploration in the transferability property of adversarial examples. These include Demontis *et al.* [222], which define three factors that impact transferability, and Papernot *et al.* [218], which explores the space of adversarial perturbations. Demontis *et al.* [222] find that transferability is related to (1) the *adversarial vulnerability*, or complexity, of the target model (2) the complexity of the substitute model used to optimize attacks (3) the alignment of the substitute model with the target model. Papernot *et al.* [218] considers the distance between decision boundaries of different models, finding that adversarial examples transfer well when different models have decision boundaries that are close in distance. Knowledge about transferability of adversarial attacks can open the doors to further research on attacks and defenses specific to transfer learning.

There is also further research on adversarially robust transfer learning, namely Shafahi *et al.* [221], who finds that robustness transfers from robust feature extractors to student models. This finding encourages the potential for building robust models with limited training data; however, this can come at the cost of validation accuracy. Further research in this direction could enable discoveries in methods for conducting robust transfer learning.

3.2.2 Problem: Building a Test Strategy in Line With Defined Adversaries

Description An important problem with respect to the robustness against adversarial examples is to set up a test strategy that provides a realistic estimation of the risk of the specific use case. To demonstrate an ML model's robustness, or lack thereof, to adversarial perturbations of their inputs, a user can perform an *adversarial analysis*. This typically consists of techniques for solving an optimization problem that determines the magnitude and direction of adversarial perturbations of an input example for which a model will not have a misclassification [223]. A model is determined to be robust with respect to a perturbation budget, or the determined size of a perturbation. To assess if a model is safe for deployment, a test strategy should consider the model's deployment circumstances when performing an adversarial analysis. This includes in addition to assessing the attacker's motives and capabilities (see Section 3.2.1) also

an assessment of the cost of a misclassification by the model.

In transfer learning, the perturbation budget of an attack is especially important to consider, as adversarial examples built from a teacher must be transferable to a student model in order to fool it. Some transfer-based attacks that search for perturbations within a fixed-radius ball require a high perturbation budget to have a good rate of transfer [224]; however, large perturbations might be recognizable by humans and thus less effective. Importantly, the user must also assess the damage associated with an adversarial attack. This will determine the threshold at which a system is determined to be safe for deployment. A failure rate that is acceptable for a spam filter might be too high for an autonomous vehicle [225].

After assessing the deployment circumstances and developing a set of likely adversarial perturbations, a user can decide to either perform a stress-test of their system, or opt for formal verification of their system. In stress testing, a user simulates adversarial attacks on a trained model and measures its robustness with respect to different criteria. A typical way to measure a system's robustness is to assess its decrease in accuracy with respect to the perturbation budget of an attack [225]. A model is determined to be safe for deployment if it passes the stress tests at a user-specified threshold. The second is *formal verification*, which is more comprehensive, and more challenging to perform. Formal verification is to provide guarantees of robustness with respect to the design of a system. Both types of test strategies require formal specifications about the ML model to be tested, as well as a detailed assessment of the specific circumstances in which the model is deployed [223]. Regardless of which technique is available to measure a model's robustness, there exists the risk of an incomplete assessment that comes from a discrepancy between the data set used to assess robustness and the real-world deployment context. The discussion therefore goes beyond the technique, but also on the available information of the model and deployment settings used in the assessment.

Effects and Consequences When a user specifies the vulnerabilities of the model within its deployment circumstances, this translates to determining which subset of adversarial perturbations are most likely for the deployed model. This decision is critical, and determines the success of adversarial analysis. However, exploring the entire space of adversarial perturbations is, nearly always, implausible [226]. A lack of knowledge of the distribution of adversarial perturbations would imply that a worst-case analysis should be performed [223]. While this would be the most comprehensive solution, it is extremeley costly, as the space of adversarial perturbations is large for high-dimensional model [227]. The choice of limiting the scope of adversarial perturbations always brings along the risk of falsely declaring a model robust. For instance, a user might not have checked for the more powerful adaptive or iterative attacks; these require more sophisticated evaluation strategies, as outlined in Tramèr *et al.* [228].

The choice of an appropriate adversarial analysis method also comes with benefits and drawbacks. Stress testing is a simpler approach to assessing robustness, but it comes at the cost of fewer guarantees. The challenge of developing a comprehensive test strategy is, first, that certain properties of models can be difficult to formalize. For example, a user might wish to

develop a measure of how likely an adversarial example can transfer from the teacher model to the student model. This would involve, among other calculations, a measure of similarity between teacher and student models [222], or reliable empirical studies that yield a measure of confidence.

As a result, the same tests could have very different results if the formalization of model properties changes. This makes it hard to ascertain robustness of a system. Another challenge is that one cannot realistically test for every possible adversarial attack, especially with the development of new attacks. The test strategy requires some assumptions made on the adversary's capabilities and goals, which could potentially become incorrect (even by hindsight) when there are changes made to the availability of the teacher model, or to the deployment settings. Therefore, even if a model is determined to be robust, there remains the threat of adversarial attacks, especially more sophisticated adaptive adversarial attacks. Therefore, stress testing is seen as a method that is approachable, but not as rigorous as one might desire for applications with a high cost of misclassification.

Opting for formal verification, a user can obtain some guarantees (in the form of a mathematical proof) that the system is robust. Formal verification is done with respect to some assumptions made on the system, such as a limited perturbation budget, which might not match real-world settings. So far, formal verification has been developed for simpler models, such as support vector machines, but it is difficult to achieve this for more complex models, such as deep neural networks. Some challenges include, as with stress-testing, the need for formal specifications. The difference is that in formal verification, the task of the entire system must be formalized; it does not suffice to write a cost function for a specific task such as classification. As deep neural networks model human cognitive tasks, such as image classification, a user must write a formal specification of the task which is a challenging problem [223]. This, as well as other steps in verifying a system, requires deep expertise in the precise architecture and application of the model. Thus, a "one-size-fits-all" verification technique is difficult to achieve.

Solution Strategies and Their Effectiveness A first step in deciding what kinds of tests a user should perform would be to use available techniques for specifying vulnerabilities for a particular model. A useful method for doing so would be to reference a taxonomy of adversarial attacks (see e.g. [229]).

When opting for stress-testing, restricting the scope of adversarial perturbations to specific attacks makes this method feasible. The availability of open-access journals and open-source code bases enables users to set up their own tests against adversarial attacks. Ultimately, formal verification, when possible, addresses many of the issues associated with stress testing. Realistically, due to the challenges in formal verification of deep neural networks, current verification methods include partial quantification with testing and verification data sets [226].

Future Outlook Formal robustness verification remains an open problem. However, the literature shows a promising direction. Namely, there is active research in three techniques: search, optimization and reachability. Search methods attempt to show robustness by searching through the set of possible adversarial perturbations and testing a model until a weakness in a model is found [226]. Some work has been done in providing guarantees with respect to sets of specific manipulations [226]. Optimization methods use the cost function of a model as a constraint in an optimization problem; for example, mixed integer linear programming is a novel approach to verify adversarial robustness of neural networks [226]. Finally, reachability methods aim to find bounds of analysis in neural networks; these methods are currently limited to specific kinds of neural networks.

To conclude, while there is yet to be a single standard for developing a test strategy for an ML model, nor are there specific guidelines for transfer learning, current literature and directions of research may open up new opportunities in the future.

3.2.3 Problem: Defending Against Adversarial Attacks

Description The adversarial attacks to which student models are the most vulnerable often rely on the transferability property. General-purpose defense methods, such as adversarial training or gradient masking, could defend against adversarial attacks, but they do not explicitly address the transferability threat. Therefore, we consider lines of defense proposed specifically for the gray-box scenario. As mentioned in Section 3.2.3, a direction of defense is to reduce the resemblance in architecture between student and teacher models. This could come at the cost of accuracy, and it also requires extra training. One alternative method proposed by Wang *et al.* [115] modifies the weights of the student model in a manner that increases dissimilarity while minimizing the decrease in classification accuracy.

One begins with a trained student model. First, all layers of the trained student model are unfrozen. Next, the student model is re-trained using the same student training data set; the training minimizes a cross-entropy loss with an additional constraint that enforces a dissimilarity between the student and teacher models. Specifically, the constraint is a weighted neuron activation distance between representations at layer K. This would serve to reduce the risk of an attack that targets layer K and the layers before it. Applying such a defense will likely also decrease the risk that an adversarial attack generated for the teacher transfers to the student.

However, the cost in computational resources and slight decrease in classification accuracy might outweigh the benefits, especially if the student model is already deployed. A second defense strategy introduced by Wang *et al.* [115] does not require retraining: to perturb the inputs in a way that renders an adversarial example ineffective, with minimal impact on non-adversarial examples, for example, using dropout [115, 143] (see Section 1.1.5 for details). Ideally, this would be effective to block the transferability of adversarial examples derived from a teacher model. This line of defense is recommended by Abdulkader *et al.* [114] against their

proposed untargeted attack which relies on the transferability of adversarial examples from the teacher model to a nearly-identical student model. However, such methods tend to be less effective against adaptive adversarial attacks that are targeted against specific defenses such as Dropout.

Effects and Consequences When opting for a defense strategy that modifies the architecture of the student model, then the transferability from teacher to student should, in theory, decrease. Such defenses would reduce the alignment between gradients of the teacher and student, which is a key factor in transferability [222].

The defense proposed by [115] that increases the neuron distances between the teacher and the student model has the additional benefit of not requiring extra training examples. Additionally, this defense cannot be undone by an adversary. However, the consequences of such an approach are, as aforementioned, a trade-off in robustness and classification accuracy of the student model. Furthermore, there still remain computational costs associated with adapting weights of the student model.

If the user opts instead to apply changes to the input layer of the student model, then there will be no need for re-training. This is especially beneficial if a student model is already deployed, and a security vulnerability is found. Dropout specifically has been shown empirically to work in transfer learning [115]. However, there is a trade-off between the amount of random dropout and the classification accuracy. As one adds randomness into the input, the model performance decreases, until its performance becomes too poor to be useful. Furthermore, an attacker can retaliate against dropout by adding a dropout layer into their adversarial attack generating pipeline [115].

Both lines of defenses have similar consequences: that a user must accept a trade-off between classification accuracy and adversarial robustness.

Solution Strategies and Their Effectiveness The literature includes several advanced defenses that are not specific to transfer learning such as the NULL labeling method and basis transformations [145]. The NULL labeling method is a method to explicitly detect adversarial examples by adding a dedicated output label (`NULL`) to the classifier that corresponds to an adversarial example (cf. Section 1.1.5).

Another line of defenses is to perform basis function transformations, in which the representation of images is modified by change of basis, compression or low-pass filtering. This results in less information loss than the dropout method, and has shown success across multiple kinds of adversarial examples in different threat models [145]. While such defenses seem to overcome the trade-off between accuracy and robustness, the use of them involves added computational cost. In addition, such defenses will most likely not withhold adaptive attacks.

Future Outlook Ongoing research on the formalization of what properties of neural networks cause transferability of adversarial examples could help the scientific community to formulate more targeted defenses for transfer learning with fewer consequences. First, findings from Goodfellow *et al.* [106] indicate that transferability occurs due to the closeness in distance between decision boundaries of different models. This comes from the linear nature and high dimensionality of neural networks: if perturbations can grow linearly with the dimension, then a small perturbation across multiple dimensions can cause large changes in neural network output. Demontis *et al.* [222] outline three contributing factors to the transferability property: complexity of the target model, complexity of the substitute model, and alignment of the gradients of the substitute and target models. The authors further quantify these properties as metrics to determine how susceptible a model is to transferred attacks. This is a key to both testing and developing defenses. Further works in the direction of Demontis *et al.* [222] can increase the success of available defenses.

3.2.4 Summary

In the previous sections, we discussed the following problem associated with the use of transfer learning: the vulnerability of a student model to adversarial attacks, specifically those that transfer from the teacher model to the student model. When opting to use teacher models which are easily accessible, a user should be aware of the fact that adversarial attacks transfer from one model to another, especially between models with similar architectures [222]. If an adversary has access to a teacher model, then they could use it as a substitute model with which to create adversarial examples. These examples could transfer to any student model trained with the teacher. Even if the identity of the teacher model is initially unknown to an adversaries, they could learn this knowledge using a fingerprinting method [115]. Some sources of teacher models offer only a few pre-trained models for each task, simplifying the guesswork for a potential adversary.

Despite the risk of transferred adversarial attacks, transfer learning is often the only feasible and cost-effective way for a user to train a large neural network.

After training a student model via transfer learning, a user can determine its safety by testing its robustness to adversarial attacks, focusing on potential gray-box attacks. Ideally, a user would use formal verification, or a technique for mathematically certifying the robustness of a model. However, these techniques are challenging to develop and not yet generally possible for deep neural networks [226]. Alternatively, a model could be stress-tested: a user could perform adversarial attacks on the model and measure its robustness with respect to a perturbation budget. The challenges of this method are to determine the feasible set of adversarial perturbations to test, to select a meaningful metric for measuring perturbation size, as well as to determine the upper bound of a perturbation budget.

Adversarial examples are found to transfer more easily between models which have a small distance between their decision boundaries [218]. Based on these findings, there are

some lines of defense that a user can consider:

- To increase the distance between student and teacher models, a user could modify the training process by freezing fewer layers of the teacher model, or by using a method proposed by Wang *et al.* to adapt the student model's weights. This solution requires extra training data and resources.
- To block the transferability of an adversarial example, a user could add randomness to the input of the student model via techniques such as Dropout [143]. This comes with a trade-off: increased robustness comes with decreased classification accuracy. Adaptive attacks might still apply.
- Adversarial training and related defenses can make a student model more robust to adversarial examples, and less susceptible to transferred adversarial attacks, compared to models without adversarial training [218]. However, this has a high computational cost, and it does not ascertain robustness, as new directions of adversarial perturbations could be discovered.

There continues to be research in the direction of adversarial attacks, defenses, as well as the transferability of adversarial examples. This could aid a user in finding novel methods to improve and test the robustness of their models. Currently, there exist several open-source tools for testing the robustness of an ML model, which allows a user to test a model for several attack modules.

4 Transfer Learning: Recommendations

Recommendations for developing secure ML systems are framed within the context of a software development lifecycle, a process for planning, structuring and implementing software systems. DevOps, a framework which combines development with operations, is commonly used for large-scale software development. It provides core principals for efficient collaboration and outlines a lifecycle for planning, developing and maintaining software [13]. With the increasing adoption of ML in software products, MLOps [230] was proposed as an adaptation of DevOps that includes steps specific to developing, testing and maintaining ML applications. A user, typically an ML practitioner, who is interested in developing a ML model using transfer learning would typically follow an MLOps workflow, albeit with slight modifications specific to transfer learning. These modifications are not only made to the development steps, but also in the security considerations that impact a user's decisions. In the following section, we describe a general MLOps workflow that a user can follow for transfer learning.

4.1 MLOps Workflow

Figure 34 depicts the MLOps workflow. Each box describes a step in the development process. The specific methods, sub-steps and considerations required for each step depend on underlying factors, depicted as cylinders.

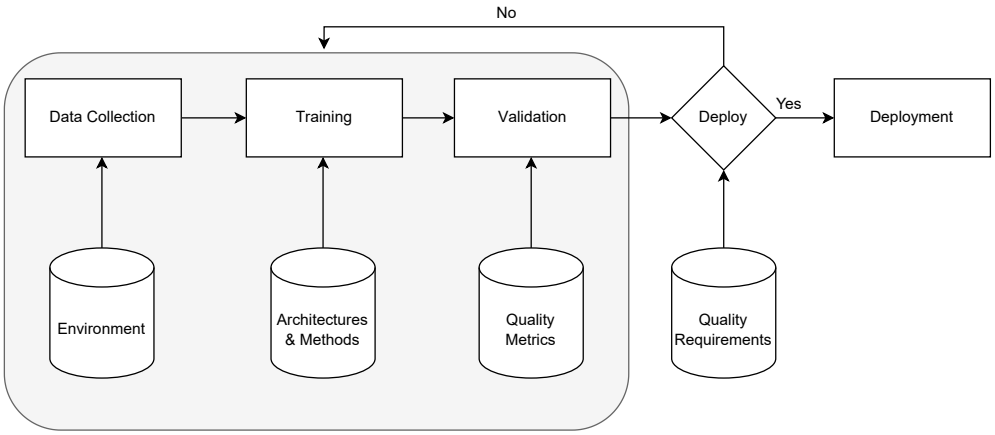


Figure 34: Overview of the MLOps workflow.

Data Collection A ML application relies on a process, possibly automated, for collecting input data. The data collection process can occur online, such as the collection of customer preferences for a recommender system deployed in an online shopping website, or, it can happen

[13] `https://about.gitlab.com/topics/devops/`, Accessed 16 June 2022

in physical settings, such as frames collected by a camera lens of an autonomous vehicle. Depending on the deployment environment of the model, a user would implement a mechanism for collecting and storing data. In transfer learning, data collection usually refers to collecting data for the student model task.

Training The next step in the MLOps workflow is to train a model. In transfer learning, this is divided into two sub-steps: selecting the teacher model and training the student model. The training procedure depends heavily on the architecture of the model and the selected training methods, as well as on the available computational resources.

Validation and Deployment The decision to deploy an ML model is based on the results of of a validation step. In this step, one determines if the model meets its objectives and upholds the determined quality requirements. The MLOps standard for model validation is to test a model against a held-out validation set and measuring its performance against user-defined quality metrics. If the model does not meet the quality requirements, then the developer returns to the training phase of the MLOps workflow. The developer would adapt the training procedure or model architecture to improve its performance in the validation step. In transfer learning, this includes the possibility to select a different teacher model or to adapt the training procedure.

Security of deployed information systems was traditionally considered separately from the development of a system[14]. In software systems, which are automated and frequently updated, a security-concerned developer can opt for DevSecOps[15], a framework that integrates security into the development process. Along similar lines, ML applications, which are deployed in variable settings, should be developed in a way such that security components are incorporated into the MLOps workflow.

In such a *security-centered MLOps workflow*, a user should begin with a risk assessment step before proceeding to the remaining MLOps steps (see Section 4.2 for details). Specific to transfer learning, this includes determining whether the data collection procedure results in trustworthy data that is not tampered by adversaries. In the training step, a user should assess the trustworthiness of the chosen teacher model, as well as the consequences of the chosen training method. Validation and deployment steps should depend on quality metrics and requirements that include security as an essential component.

4.2 Risk Assessment

Users should first analyze the risk that the transfer learning use case entails in their specific scenario. In this section, we present a set of questions which help the user to assess the

[14] `https://ml-ops.org/content/phase-zero#work-flow-decomposition`
[15] `https://www.devsecops.org/`

security risks of their ML application. The answers to these questions not only determine the security risk but also frame the defense and mitigation measures and the possible design decisions that are applicable for the specific use case. Thus, they build the foundation of a security strategy and should be taken into account when making design choices as detailed in the following sections.

Specifically, the risk assessment questions listed below help the user to consciously consider how severe the consequences of an attack on the deployed system would be (A), how likely an attack is (B and C), and whether there are limitations that affect the applicability of defense and mitigation measures (B).

A. Security Requirements The first set of questions considers the security requirements of the given use case.

A.1 What is the impact of incorrect model performance (e.g., misclassification) in normal operation?

A.2 What is the impact of incorrect model performance (e.g., misclassification) in an adversarial (worst case) situation?

Naturally, some applications require higher security measures than others. Being aware of the consequences of a potential misclassification is important to decide which security measures should be put into place.

Furthermore, knowing the expected baseline performance of the system is important to decide which defense strategies can be applied, as some defenses might come at the cost of a slightly decreased model performance.

B. Resource Availability This question list covers the availability of resources to the defender.

B.1 Can training data for the student (or teacher) task be acquired from a trusted source?

B.2 How much data is available for the student (or teacher) task?

B.3 Can the teacher model be acquired from a trusted source?

B.4 How many teacher models are available for the given use case?

B.5 What computational resources are available (for training or for implementing defenses)?

B.6 Are known-benign or known-malicious data samples available?

B.7 Are trusted development resources available, for instance, defense implementations?

The factors B.1 to B.4 contribute to the likeliness that an attack is launched on the system at training time (backdoor attacks) or inference time (adversarial attacks). All factors influence the selection of defense measures, as well as the selection of suitable ML architectures. If large amounts of trusted data are available but no trusted teacher model can be obtained, it could be a better choice to train a model from scratch and not employ transfer learning. In contrast, if large amounts of data are available, but they cannot be obtained from a trusted source, training from scratch does not solve the problem, as training with infected data could implant a backdoor into the model.

If attackers are able to identify the teacher model, they might be able to use it to craft adversarial attacks for the student model. Therefore, the number of available teacher models for the desired task might impact security. For instance, if there exist few teacher models for a specific task (which satisfy availability and accuracy requirements), then an attacker might more easily infer the identity of the teacher model.

Not only the resource availability of the defender, but also of the attacker plays a role as it affects the threat model that needs to be considered. However, usually it is difficult to infer reliable information about what is available for the attacker, and these factors cannot be actively determined. It is, thus, usually best to consider the worst-case that attackers have access to comparable, or even more resources than the defender, enabling them for instance to craft and provide malicious teacher models.

Finally, it is important to clarify which development tools can be used as it might be infeasible for the user to implement a defense mechanism from scratch. Not only the availability of the resource plays a role, but also the maturity and the security of a used tool.

C. Threat Model The threat model summarizes the attacker's capabilities and defines how likely the deployed system is going to be attacked. While it is typically possible to adjust the deployment settings to limit the attacker's capabilities, there might be cases where the use case imposes a specific deployment setting.

C.1 Does the deployed student model receive input digitally or physically?

C.2 Can users query the student model at low cost?

C.3 Can users query the student model at a high frequency?

C.4 Are student model internals (e.g., weights, or information about the architecture) exposed to the user?

As these questions indicate, one can consider limiting a user's access to the system or to information about the system in order to hinder an attacker to exploit potential weaknesses

of the system. This is especially useful when considering that there are no reliable defenses available in the literature which defend against all kinds of adversarial attacks. However, note that limiting user access does not imply any increase in a system's security.

Also, receiving input digitally generates a higher security risk, as adversarial examples and samples endowed with backdoor triggers can be crafted more precisely in the digital form. Physical adversarial input requires more robust crafting in order to be effective. Thus, while physical input reduces the likeliness of the attack by increasing the necessary efforts of the attacker, the risk does still remain [231].

Finally, it should be considered what the adversary might want to achieve with the attack. The goal of the attacker affects the threat model and helps the user to determine which is the most realistic or most threatening attack scenario. While it is hard to exclude a specific attack scenario completely, tendencies could be derived. For instance, there might be a case where an attacker might benefit more from a targeted misclassification than an untargeted one, indicating that priority should be placed on measures which reduce the likelihood of targeted attacks (e.g., reducing teacher and student model similarity). Note that the definition of the attacker's goal should also always take the general security requirements (A) into account.

These considerations form the foundation for deciding on specific defense strategies as discussed in the next section.

4.3 Recommendations

In the following, we derive technical and organizational recommendations on defense strategies against backdoor attacks that might get introduced into a student model via transfer learning on an infected teacher model. Additionally, we take into consideration the risk of inference-time adversarial attacks that may be launched on a system at deployment time.

Technical recommendations include changes to the way that the model is created or trained. Organizational recommendations can be put in place additionally without affecting the model development process itself, for example, by monitoring the source of an obtained teacher model.

As recommendations depend on the current status of the development process, we split the discussion into four different processes which connect to the phases of the MLOps workflow as displayed in Figure 35: the teacher model selection process and the student model training process would be usually visited sequentially during the ML lifecycle prior to deployment. During these phases, the defense process should be visited once or multiple times depending on the arising security concerns. Also at the stage of deployment, strategies such as monitoring can be pursued to be able to decide to return to earlier stages when it appears necessary.

For the teacher model selection and the student model training process, we discuss im-

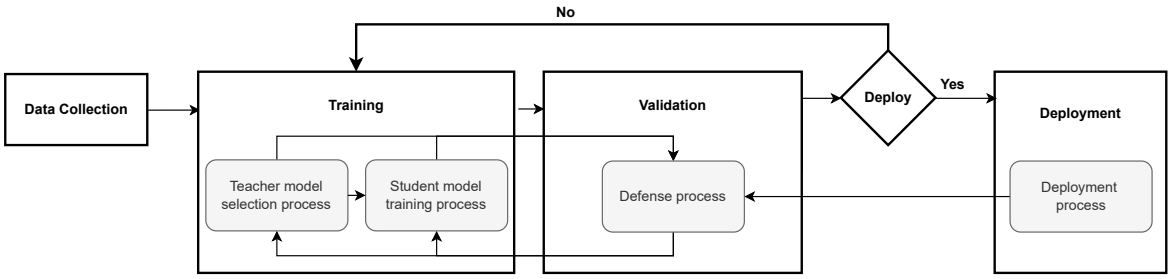


Figure 35: Overview of the four processes that we consider here, the interconnections between them and the MLOps workflow phase in which these processes are typically relevant.

portant considerations in Section 4.3.1 and Section 4.3.2, respectively. We structure this part by listing questions. The user's answers to them inform the decision of which risks should be particularly considered. In Section 4.3.3, we discuss in-depth the threats and defense options for backdoor attacks based on a flow chart that provides an overview of potential risk factors and mitigation options. Figure 36 provides the legend explaining the visual elements used in the flow chart. Finally, Section 4.3.4 looks at considerations to keep in mind after the model is deployed.

4.3.1 Teacher Model Selection Process

The teacher model affects the risk of backdoor attacks on the deployed model, and also may influence how easily an attacker can craft adversarial examples to fool the model. Therefore, the choice of a teacher model should be carefully considered, with regard to the following questions:

Is the teacher model creation process trusted and transparent? To launch a backdoor attack, an attacker's default strategy would be to provide an incentive for the ML practitioner to use a backdoored teacher model. The easiest way for the attacker to achieve this is by providing the backdoored model publicly via a web page or an online repository. While it is not the case that all open-source models are deemed untrustworthy, a user can often not ascertain that a publicly available model is free of backdoors. In contrast to traditional open-source software repositories, a provided neural network is to a certain degree a black-box, since the mode of operation is encoded implicitly in the weights of the neural network. A user should therefore be wary of trusting open-source models if the model creation process is not transparent and the identity of the provider cannot be verified. Hereby, transparency means that the resources that a model provider uses to craft a model can be tracked to their origin. However, verifying what data has been used for training the model and ensuring that it has not been tempered with is a challenging problem. A method to protect the integrity of the trained

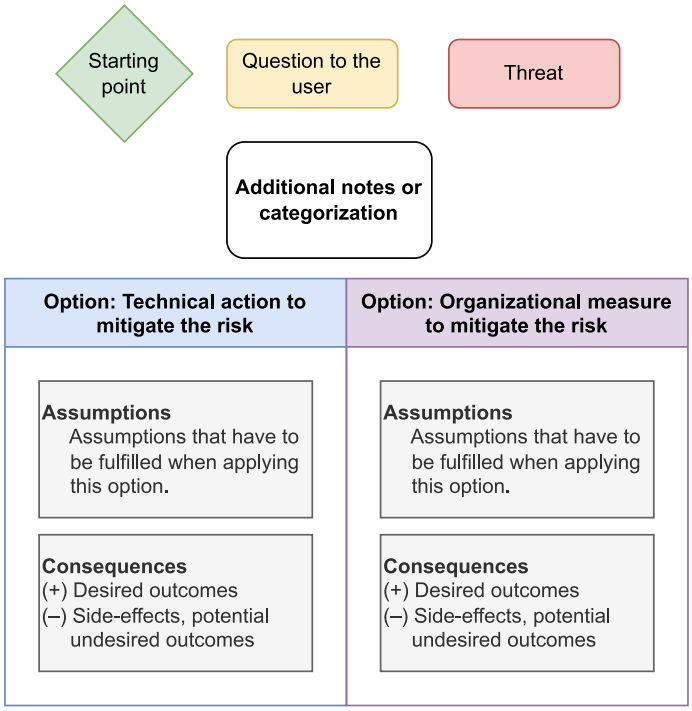


Figure 36: Explanation of visual elements used in the flow charts Figure 38 and Figure 37.

network that allows external users to verify that the data used for training is free of manipulations could be based, for example, on cryptographic mechanisms [232].

In summary, it is recommended to carefully consider the trustworthiness of the model provider and check whether effective quality measures are in place. If the defender has doubts about the trustworthiness of the provider, defense strategies should be considered as discussed in Section 4.3.3.

While a trusted model provider is a necessary condition for trusting the model creation process, it might not constitute a sufficient condition. Specifically, even a trusted ML service provider might outsource part of the data collection or model training process, enabling an attacker to manipulate the training process (e.g., by providing training data containing a backdoor trigger). Therefore, full transparency about the origin of all used training data and the trustworthiness of the training process is required for making an informed decision.

Is information about the used teacher model disclosed? The risk of gray-box adversarial attacks is affected by how easy it is for the attacker to infer which teacher model has been used. If an attacker has access to the identity and architecture of the teacher model used to train the student model, the attacker can likely craft an effective attack on the student model. Therefore, it is recommended to keep information about the used teacher model and detailed information about its origin and training process concealed whenever possible. This strategy can be effective as an additional security measure considering that available defense mechanisms cannot provide any guarantee for the security of the system. It is, however, important to be aware that hiding information per se does not increase the system security.

The attacker might still be able to infer the used teacher model from a set of candidate models by using fingerprinting methods (e.g., [115]). Depending on the domain, there might only be a few teacher models available, reducing the attacker's effort to infer which teacher model has been used.

Is the teacher model publicly available? A publicly available teacher model is more easily accessible for an adversary, and thus would increase the risk of gray-box adversarial attacks. This risk is particularly significant for teacher models which enable an adversary to access the teacher model's architecture and parameters without additional costs. Also propriety teacher model might be identified and accessed by an adversary, especially if it is provided by a MLaaS platform with a low barrier to entry.

Can training data be obtained from a trusted source? While transfer learning on pretrained teacher models is common practice nowadays, it is important to be aware that other ways to develop a well-performing model exist. Depending on the available computational resources and the availability of training data for the domain, training the student model from scratch,

or training an own teacher model on a task that is similar to the target task, are both viable options. Specifically, it is recommended to consider training the teacher or student from scratch if a sufficient amount of training data can be obtained from a trusted source, especially if there are security concerns regarding the available teacher models. This strategy can help to reduce the risk of backdoor attacks or gray-box adversarial attacks on the model. However, even with a model trained from scratch, the risk of black-box adversarial attacks on the model remains.

Summary In summary, it is recommended to obtain teacher models from a trusted source which discloses all relevant details about the model creation process. Ideally, the source would provide full transparency over the origin of the training data of its teacher models, as well as perform a set of quality checks on its teacher models. In any case, it is recommended to not disclose information about the used teacher model as this knowledge makes it easier for an attacker to stealthily craft adversarial examples.

Given that the model creation process is trusted, a general recommendation to ensure the security across the whole lifecycle of the system is to monitor the used providers, and sources of the teacher model and the training data. Such a monitoring serves as an organizational measure that helps to detect changes in the security concerns later at deployment time. Details are discussed in Section 4.3.4.

4.3.2 Student Model Training Process

The risk of backdoor as well as of gray-box adversarial attacks originates mainly from the similarity of the teacher and the student model. Reducing this similarity, thus, is important for addressing these risks, particularly if the teacher model is publicly available or not fully trusted as discussed in Section 4.3.1.

Is the student task similar to the teacher task? Transfer learning is typically used because the features that the teacher model learned are considered to be useful for the student task. Hence, the teacher and the student model task are inherently similar to each other. However, the degree of similarity may differ depending on the use case. For example, if a teacher model trained on ImageNet is adapted for a medical application, the similarity is lower than if a teacher model trained on face images is used to train a student model that should recognize a different set of faces. Generally, the similarity of the tasks, specifically the similarity of the output labels of the model, influences how likely a targeted attack crafted for the teacher model succeeds in the student model.

Furthermore, there are transfer-learning-specific types of backdoors such as the latent backdoor attack (cf. Section 1.1.4, page 24) that can be implanted into the teacher model if the attacker can guess which new label (or set of new labels) is added to the model in the student training.

To reduce the likelihood that a targeted attack transfers from the teacher to the student, it is recommended to perform transfer learning using a teacher model that performs a task that has a different data distribution or works on a significantly different task than the student. However, this recommendation is difficult to follow in practice, as a reduction in similarity could negatively affect the performance of the student model. Thus, considering the similarity between the teacher and student tasks instead provides indications as to whether an emphasis should be put on acquiring a secure teacher model (Section 4.3.1), whether a retraining of layers of the teacher model is important as discussed below, and whether defense strategies should be put in place (Section 4.3.3).

Is feature extraction transfer learning used or required? Feature extraction transfer learning adapts the teacher model to a student model task without making any modification to the layers of the teacher model. While being a common practice in transfer learning, this procedure results in a student model that is extremely similar to its teacher model. This increases the risk that attacks transfer from the teacher to the student model.

In general, with regard to increasing the robustness against potential backdoors in the teacher model and against gray-box adversarial attacks, it is recommended to use the teacher model for weight initialization but retrain as many layers as possible during the student model training process. It might be advantageous to diverge from the recommended number of layers for retraining which are sometimes published as additional information.

As the number of layers that are retrained affects not only security but also accuracy, it is generally recommended to test multiple scenarios and train as many layers as possible while ensuring acceptable performance according to the requirements of the given use case (cf. Section 4.2). It should also be taken into account that the number of frozen layers affects the computational effort during student training as more learnable network parameters increase training time. Also, in particular, for large networks, there is the risk of overfitting if many free parameters are tuned on a student training data set of only limited size.

Is the student training data from a trusted source? Measures for reducing the similarity between teacher and student model are not effective if the backdoor does not originate from the teacher model but gets introduced via the student training data. Therefore, the same considerations regarding trustworthiness and transparency as discussed previously for the teacher model provider also apply to the provider of the student training data (cf. Section 4.3.1). Similarly, it is recommended to monitor the used training student data set source and provider to be able to respond to changing security concerns (cf. Section 4.3.4).

Finally, considerations regarding the risk of adversaries should also be complemented with evaluations regarding the introduction of unintentional backdoors or biases into the model which attackers could exploit if they become aware of it. This risk is especially high with student training data sets as they are typically smaller and less diverse than training sets

used for training the teacher model. To avoid such biases, it is recommended to diversify the student data set, for instance, by obtaining data not only from a single source but from multiple sources to cover the diversity of input data that is expected at run-time.

Summary The main recommendation is to try to keep the student model different from the teacher model in terms of the model parameters and, if possible, the output labels of the task. If computationally feasible, a full analysis of possible parameter settings (specifically, the number of retrained layers) is recommended which then can be optimized to retrain as many layers as feasible while maintaining acceptable performance. Additionally, the trustworthiness of the student data set and its provider should be ensured.

4.3.3 Defense Process

To reduce the risk of attacks on a student model, various defense strategies can be considered. As discussed in Section 4.3.1 and Section 4.3.2, the crucial factors determining the existing risk are whether the teacher model is trusted and how similar teacher and the student model are. Figure 37 provides an overview of the potential threats that have to be considered for a given use case and lists the options that the user has for detecting or mitigating the risk of a backdoor in the model. Figure 38 analogously provides an overview of detecting or mitigating the risk of adversarial attacks in the model. Note that the recommendations for defending against adversarial attacks are included for completeness, but should be taken as suggestions rather than rigorous defense options. For the remainder of this section, we detail the defense process against backdoor attacks.

A general recommendation is to monitor the used resources as discussed in Section 4.3.4. Technical ways to defend the model can be roughly split into four different categories with multiple options for action which we discuss in the following in further detail.

Replace or diversify teacher model If there is the concrete suspicion that an employed teacher model might contain a backdoor, the simplest and most effective solution is to replace that teacher model with a different one, considering the criteria discussed in Section 4.3.1.

If it is difficult to reliably estimate the risk that the available teacher models constitute, there is also the option to use an ensemble of different teacher models and use, for instance, a majority voting system to determine the model output. Depending on the security implications, one can also decide to refuse to process the input if the individual models disagree about the output. If a diverse set of teacher models is employed (ideally, obtained from independently trained sources), this procedure effectively reduces the chances that a backdoor, even if it transfers to the student, causes an actual misclassification in the system. However, naturally training and employing an ensemble increases the computational costs significantly not only at training-time but also at run-time.

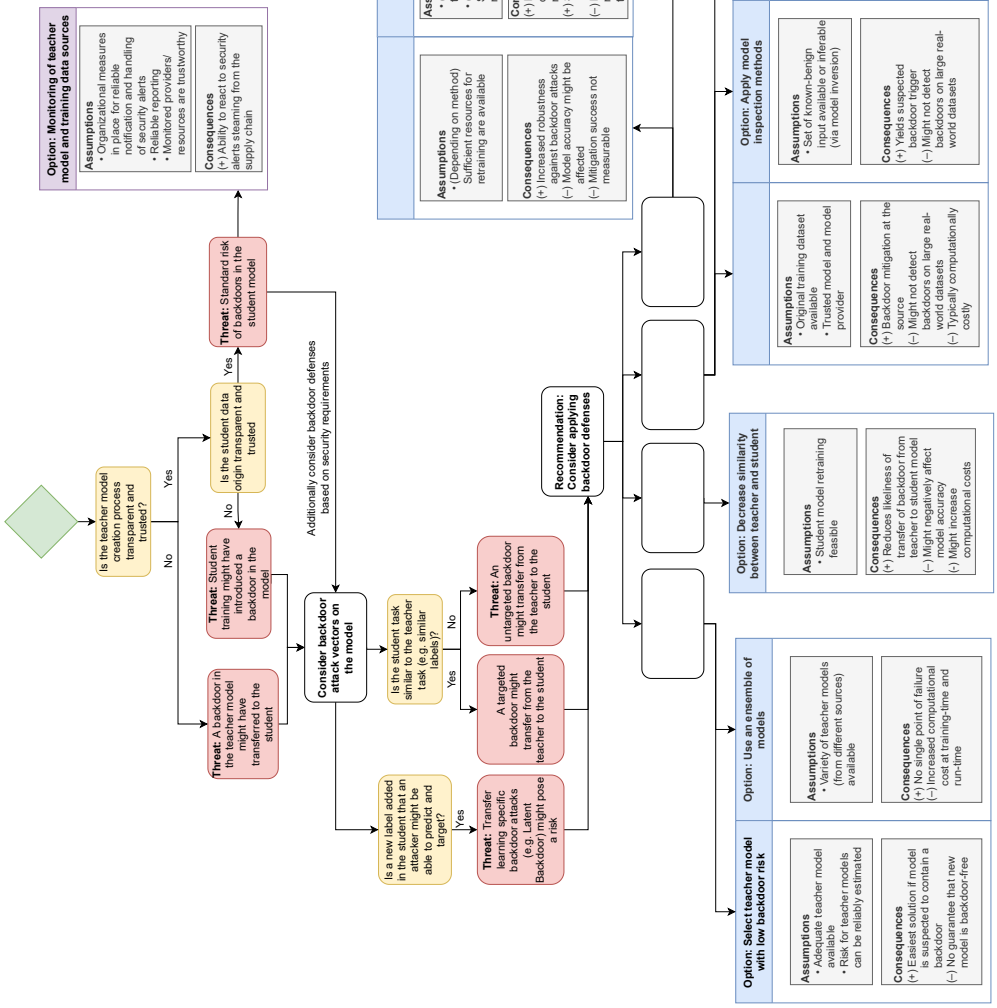


Figure 37: Flowchart illustrating threats and action items in a backdoor defense strategy. For an explanation of the colors, refer to Figure 36.

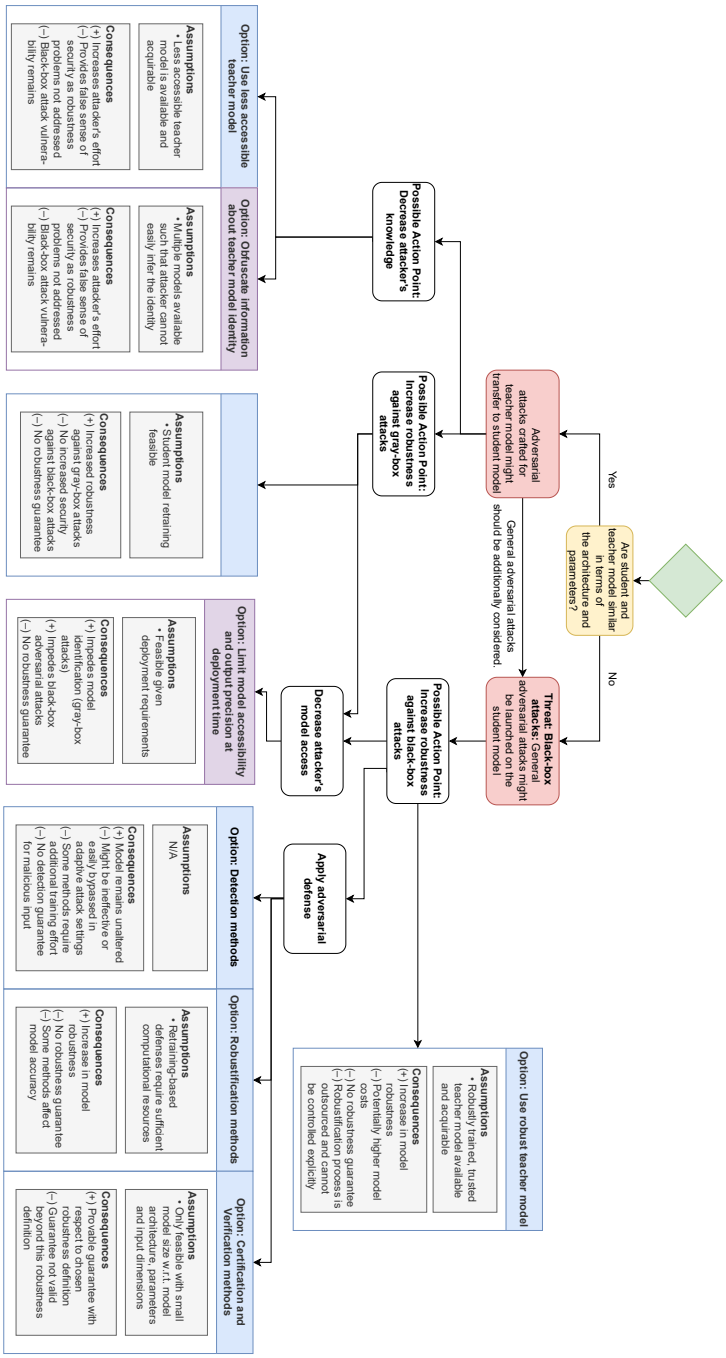


Figure 38: Flowchart illustrating threats and action items in a defense strategy focusing on adversarial attacks. For an explanation of the colors, refer to Figure 36.

Consolidate student training process If the teacher model cannot be replaced, an alternative option is to adapt the way that the student model is trained. Specifically, during the student training, it may be possible to reduce the similarity between the teacher and the student model as discussed in Section 4.3.2. Note that this requires sufficient computational resources and potentially additional student training data for training the student model again. In most cases, this is a feasible option, as it means to repeat previously conduced processes but with different parameter settings. However, there is a risk that changing the training parameters leads to a model with decreased accuracy, and, thus, that the resulting model does not fulfill the requirements for the given use case.

Detection of backdoor triggers If the risk of a backdoor in the teacher or the student model cannot be accurately estimated, or should be additionally tested, there is the option to employ backdoor detection methods. Detection methods evaluate the risk of a backdoor either in the model itself or in the training or inference data. These methods do not mitigate the backdoor yet, but constitute a first step to enable the targeted mitigation of the backdoor in a later step, or alternatively block the suspicious input sample, drop it from the training data set, or replace the teacher model if required.

The following different types of detection methods can be taken into consideration:

1. **Training-data inspection backdoor detection methods:** In the transfer learning use case, this backdoor defense method category is only applicable in the specific case that the original training data is available for inspection. As from the trained model there is no way to reliably infer the used training data set, the provider of the model has to be trusted, as a dishonest provider might provide only a non-infected subset of the actual training data set.

2. **Model inspection backdoor detection methods:** Model inspection defenses are among the most commonly applied backdoor detection methods. If such a method succeeds to detect a backdoor trigger, targeted mitigation instead of untargeted mitigation can be applied to remove the backdoor, increasing the prospects of a successful mitigation (cf. Section 4.3.3). While examples in the literature demonstrate that these methods can be effectively used to detect and remove backdoor triggers from a model, there might be circumstances where the effectiveness is reduced. In particular, our practical investigations in this study indicated that there may be situations where Neural Cleanse produces many false alerts. If a backdoor trigger is detected, thus, it is oftentimes hard to decide whether the detected pattern is actually a malicious pattern or instead constitutes a universal perturbation [201], a specific type of inference-time adversarial attack. The problem is that mitigating a large number of suspicious backdoor triggers which do not actually correspond to a malicious one, might reduce model accuracy while not providing a significant increase in model security.

   Model inspection methods usually require a sufficiently large set of known-benign ex-

amples as they are based on statistical analyses of the model behavior to determine whether a potential trigger pattern in fact alters the model's behavior on clean input.

Furthermore, the user has to be aware of the following limitations of the methods that are still not sufficiently addressed in the current literature:

- All methods are optimized to all-to-one backdoor attacks. Although this is the most common attack type, there exist more complex types of attacks which would not be detected by these methods [213, 214].
- The methods can only detect input-agnostic attacks and not sample-based approaches such as clean-label attacks.
- Some methods [159] make the assumption that the trigger is small, relatively to the input size. Larger triggers which are distributed across the whole input space, thus, might not be detected.

3. **Inference-time backdoor detection methods:** Methods in this category can be used to secure the model against backdoored examples that are fed to the model at deployment time. If such a method indicates that an input sample might be backdoored, the model can refuse to classify it, or provide a warning that the classification of a given input sample might not be reliable. Naturally, applying this additional security step increases the processing time of the model. If the use case requires real-time classification, the detection method has to be selected carefully to be efficient enough given the available computational resources (e.g., [158] was developed as an efficient online detection mechanism whereas [157] has higher computational demands).

   The application of these methods requires the user to have access to a number of known-benign or known-malicious input data to compute the statistics that are required to decide about the security risk of a single input sample. Larger amounts of data improve the statistics. Note, however, that there is the risk that an input sample is falsely assumed to be benign while it is actually malicious. If this happens, the success of the defense method would be drastically reduced. Therefore, it is recommended to use input samples from a trusted source as opposed to data provided by external users whenever possible.

   Also there are indications that the methods are less effective for detecting backdoors with a reduced attack success. The student model training often reduces but not eliminates the backdoor of the teacher model, leading to backdoored models with a lower than usual attack success. This case is not sufficiently covered in the literature, therefore, the effectiveness of the methods cannot be reliably estimated to date.

   Thus, these methods can only be recommended as an extra step if the security concerns are relatively small and the impact of taking the system offline would be more severe than would be a potential misclassification of the system. For instance, inference-time detection methods could be used as a quick solution to improve the robustness of the deployed model without taking it offline while an offline investigation or retraining of the model is performed in parallel. Some inference-time detection methods are able

to extract the backdoor trigger if infected examples are encountered, which could be directly used as input for a targeted backdoor mitigation as discussed in the next section.

Backdoor mitigation If a model is suspected to have a backdoor, and the origin of the backdoor can be determined (i.e., whether the backdoor was inherited from the teacher model or introduced via student training), it is usually best to replace the used resources, for instance, the teacher model, to mitigate the problem at its source.

If this is not possible, for example, because there is no alternative teacher model available, mitigation techniques can be applied to reduce the risk of the backdoor. Depending on whether a potential backdoor trigger is known or not, targeted or blind mitigation techniques can be used.

- **Blind backdoor mitigation:** Blind mitigation of backdoors (i.e., without knowing the potential backdoor trigger), tends to be successful if a sufficiently large known-benign training data set is available. However, there is the possibility that it is not working well against some adaptive attacks specifically crafted to evade specific mitigation defenses such as fine-tuning [199]. Also, blind mitigation usually decreases model accuracy because in the process parts of the model's learned connections are removed which are involved in processing benign features as well.

  Therefore, this strategy is only recommendable if there are strong indications for a backdoor, and if a drop in accuracy can be either tolerated, or prevented with a sufficiently large, known-benign training data set.

  Thus, before deciding to use this option, it is recommended to consider first whether there is a way to avoid that the backdoor gets introduced in the student model in the first place by revisiting the teacher model selection (Section 4.3.1) and student model training process (Section 4.3.2).

- **Targeted backdoor mitigation:** The chances for a successful mitigation increase if a backdoor trigger is known which should be mitigated. The main advantage is that a successful removal of the backdoor can be tested which makes it easier to balance a potential accuracy drop with the gained increase in security.

  How successful these methods are, thus, depends mainly on how reliable the knowledge about the suspected backdoor trigger is. For example, a backdoor trigger that was published by the provider of the teacher model (cf. Section 4.3.4) would usually be considered as more reliable than a backdoor trigger obtained from a backdoor detection method.

Summary Ideally, the probability of backdoors present in the model should already be reduced as far as possible while selecting and training the teacher and student model. If this is not possible, or it is unclear whether the risk was sufficiently mitigated, backdoor detection

methods can be used. While a large number of different detection methods are available, there are still clear limitations in the literature regarding the reliability of these methods. While these methods can give some hints about the security of a model or data set, it is important to evaluate the likeliness that a backdoor trigger proposed by a detection method actually activates a malicious backdoor for a given use case. Telling apart whether a potential trigger actually constitutes a security risk or was a false alert is not easy in a real use case. An indication can be that the suspicious trigger pattern was used in input data that was fed to the deployed model. To support this type of analysis, it could be helpful, if possible in the use case, to store the queries to a deployed model during run-time for further inspection (see Section 4.3.4).

Also backdoor mitigation techniques can be a useful tool, specifically, if there are strong indications that a specific input pattern is a backdoor trigger. To counteract the potential loss in accuracy, it is recommended to have a large known-benign training data set available for retraining the model.

4.3.4 Deployment and Operation

During the operation of a model, it is important to continuously monitor the security of the used resources, i.e. the used teacher model and data used for training the student model. For instance, appropriate steps need to be taken if a model provider notifies its customers that a potential backdoor was detected for a model. For this purpose, a systematic IT asset management [233, 234] should be established. IT asset management is a common procedure in companies and institutions to manage hardware, software and other important resources that are crucial to business operations “from a cost, contractual, support, and inventory viewpoint” [233]. With respect to security, [234] complements that asset management “provides a centralized, comprehensive view of networked hardware and software across an enterprise, reducing vulnerabilities and response time to security alerts, and increasing resilience”. Assets relevant for ML models are, for example, the used data sets and pretrained models that were used during development of the deployed system.

An important prerequisite for this to be effective is that providers of ML assets such as models and training data sets commit to publishing information about potential weaknesses that become apparent at a later stage, either publicly, or by privately notifying customers (e.g., in the case of a MLaaS platform). Furthermore, such weaknesses are unlikely to be discovered by accident but require the providers to actively and regularly perform checks of the security of their products. Although public notices about potential weakness in the system can increase the security, there is the risk that it reduces the trust of customers, thus, it is hard to guarantee that providers will reliably report problems that they encounter with their models or data. Ideally, guidelines for MLaaS providers regarding regular checks and security alerts should be put in place. A possible approach is to have a criteria catalog containing implementation criteria. Based on such a catalog, the effectiveness of implementation can be audited by an independent third party. Customers can try to contractually fix such requirements with the

service providers. For open source models, the existence of reliable and timely information on security issues depends heavily on how large and active the community is.

It should also be taken into account that asset management might prove ineffective if the model or data set provider outsourced part of the model training or data collection process to a third party. In such cases, even if reliable reporting is in place, it might not cover all parts of the system, unless third parties are also trusted and obliged to report security alerts.

If a model is in deployment, it is recommended to collect statistics about the frequency and origin of requests that were sent to the system at run-time as these information can hint at potentially malicious users. Storing and analyzing the input data that users provide to the system can help to detect potential attacks on the system (i.e., adversarial examples or backdoored input data that the system was queried with). Note, however, that this procedure might not be feasible in all cases as it can be in conflict with the privacy requirements of the end users.

A careful analysis of the input data provided by end users of the ML model is particularly recommended if the system is used in an online manner, where the query data provided by users is fed back into the model at regular intervals with the purpose of fine-tuning the model. A risk assessment (cf. Section 4.2) should be performed in advance to identify potential threats. If the monitoring capabilities are limited or relegated to third parties, it is especially important to consider the security recommendations provided in the previous sections.

4.4 Conclusion

We provided recommendations for defending against backdoor attacks, structured by the steps in the MLOps workflow that a user would visit during the development of an ML model.

Overall, the recommendation is to consider potential security threats as early as possible in the development lifecycle. Specifically, the teacher model selection and the student model training process should be designed in a way such that relevant threats are minimized. We provided two overview flow charts illustrating the potential threats that are accompanied with adversarial and backdoor attacks, respectively, and pointing out potential options to defend against these threats.

Next to technical recommendations, we also discussed organizational measures that can help to increase the security of ML applications. In particular, a reliable reporting and monitoring of security risks enables an ML practitioner to timely react to such risks by deciding whether a system has to be taken offline or further investigated for security issues.

While the provided recommendations can help to minimize the security risk of ML applications, it is important to be aware of the limitations. In this study, we focused mainly on recommendations which reduce the likeliness of inheriting a security risk from the teacher model based on the methodology of transfer learning. The discussed measures can help to

strengthen the robustness of the ML model against backdoors inserted in the teacher model, but in most real-world use cases, no general guarantee can be provided. Given that this is an active field of research with new attack and defense methods being published regularly, the proposed strategy should only be considered as a superficial guideline that requires further in-depth evaluations for a specific use case.

Part II

Provision of ML Models to External Users

Hagestedt, Inken (Apheris AI GmbH)
Withnall, Michael (Apheris AI GmbH)
Höh, Michael, Dr. (Apheris AI GmbH)
Zimmer, Raphael, Dr. (BSI)
Sennewald, Britta (BSI)

5 Provision of ML Models to External Users: Literature Overview

In this part, we study the use case of a trained machine learning (ML) model that should be shared with external parties that cannot be trusted. We assume that these external parties only use the model for predictions but not for further training in order to differentiate between the transfer learning use case and this use case. Based on this assumption, we further assume that only black-box access is given to the model, but the model's internal parameters are not shared with the external party. As the external parties that receive the model cannot be trusted, two main privacy risks are discussed: stealing information about the confidential data the model was trained with and stealing the model's inner workings. The latter not only violates the model's confidentiality but may also simplify privacy attacks against the underlying training data.

5.1 Broad Discussion of Relevant Research Field

In this section, we first give an overview of the attack vectors and then discuss relevant research to each of them. After that, we show how research mitigates these attacks.

5.1.1 Attack Vectors

We discuss the three principal attack vectors: membership inference and reconstruction attacks which threaten the privacy of the training data, and model stealing attacks, which threaten the model's intellectual property and may simplify privacy attacks.

Formalization and Summary of Attacks We begin with a formalization of the attacks to better distinguish them and help the reader to select the section of interest, Figure 39 visualizes the formalization at the example of shape recognition. Additional examples on the attacks

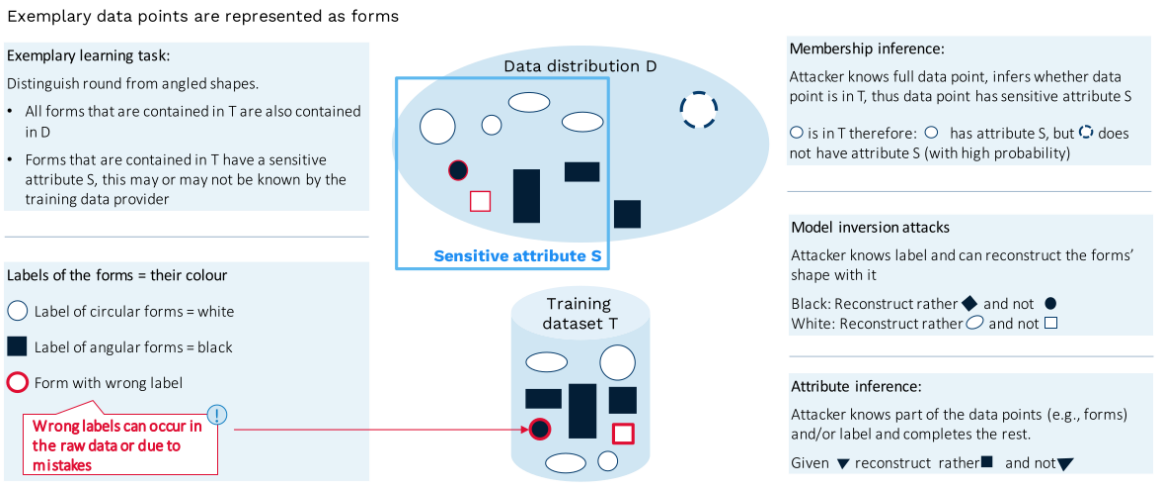


Figure 39: Visualization of different attacks at the simplified case of geometric shape recognition.

can be found in the respective section. We assume data points are sampled from a data distribution D. Note that we display it as a finite region in Figure 39 only for demonstration purposes, but in theory, infinitely many data points can be sampled from D. These data points have n features and can therefore be represented as vectors with values $(x_1, ...x_n) = \vec{x}$. In Figure 39 we visualize data points as different geometric shapes. The number of features n depends on the selected data representation which in turn is determined by the data and the ML problem that is being addressed. Each data point $\vec{x}$ has a label y, shown as dark or light color in Figure 39. Note that labels may be wrong, for example due to an error in the labeling process. This is usually not a problem as long as the fraction of wrongly labeled points is small. For learning, a finite training data set T is sampled from D. Ideally this sampling procedure is uniform, picking data samples at random, but in many real-world problems this ideal is not met. This issue is displayed by only sampling training data points (displayed as geometrical shapes) from the left part of D in Figure 39, while a random sampling would pick samples from all areas of D. We describe the training data set T as m data points with their labels: $\{(\vec{x_1}, y_1), ...(\vec{x_m}, y_m)\}$. From this training data set, the ML algorithm learns a function F such that ideally $F(\vec{x}) = y$ for all data points in D. Realistically, this property should hold for a high percentage of data points in D. In the exemplary task displayed in Figure 39, the algorithm should learn that round shapes are labeled as light color and shapes with edges are labeled as dark.

With this formalization, a *membership inference attack* tries to infer whether a given data point $\vec{x}$ is an element of the training set T, given $\vec{x}$, y and $F(\vec{x})$. This is a privacy leakage if T is not sampled uniformly at random. In other words, if there is a sensitive information S such that data points in T have S and data points not in T have not with high probability, formally: $\vec{x} \in T \implies S$ and $\vec{x} \notin T \implies \neg S$ with high probability. This sensitive attribute may

or may not be known to the training data owner. Membership inference attacks are only a privacy problem because of the non-uniform training data sampling process that is biased with respect to S. Examples can be over- or under-sampling of ethic or religious minorities as well as correlation to temporal or geographic information. Note that the correlation of training data with the sensitive attribute does not have to be perfect to pose a privacy risk, as we display in Figure 39 with the black square that is a training data sample but does not have the sensitive attribute S.

We follow the categorization by Rigaki *et al.* [235] and define *reconstruction attacks* as attacks that reconstruct $\overrightarrow{x}$ partially or completely. Model inversion attacks are one type of reconstruction attacks. In a model inversion attack, $\overrightarrow{x}$ is reconstructed completely given the label y and F such that $F(\overrightarrow{x}) = y$. So in our example in Figure 39, a model inversion attack should recover that dark shapes have four edges while light shapes are round. Note that this definition is completely independent of the training data set T. Attribute inference attacks assume that there is a sensitive feature a and, given the remaining non-sensitive feature values $x_2, ...x_n$ as well as y and access to F, the attack reconstruct a such that $F(a, x_2, ...x_n) = y$. That means, in our exemplary task in Figure 39, part of the shape is known and the remainder needs to be reconstructed. For example, a triangle should be reconstructed to a shape with four edges and not a round shape. Multiple sensitive features a are possible and they can occur in any position $i \in \{1, ...n\}$ of the feature vector. Note that also this definition is completely independent of the training data set T.

Finally, *model stealing attacks* aim at reconstructing a function G such that G and F behave as similarly as possible, so ideally $F(\overrightarrow{x}) = G(\overrightarrow{x}) \forall \overrightarrow{x} \in D$. Note that the training of shadow or substitute models used elsewhere in this work is related to model stealing, but not the same. Both for shadow and substitute model training, the attacker has access to a training data set (or a substitute for it), but not to the victim model. For model stealing attacks, no training data set is available and the attacker only generates random inputs, but queries the victim model. The attacker basically steals a model without paying the effort for training data generation and training of the model.

Membership Inference Attacks determine whether a given point was used during the training of the model. This is a privacy risk if further sensitive information can be inferred from the membership status, for example, if the data has been collected from a small geographical region only, or during a limited time period, or with other biases that might be privacy-relevant.

The first membership inference attacks on machine learning were described by Shokri *et al.* [236]. The attack itself is based on machine learning by training an attack model to distinguish members from non-members. In order to generate training data for the attack model, the victim's model response to members and non-members is required. Shokri *et al.* propose to fit shadow models with a similar structure as the victim model on data that follow a similar distribution compared to the victim's training data. In this way, the ground truth about which data is used to train the shadow model is known. The attack model then exploits the

difference in the shadow model's predictions between members and non-members. Finally, the attack model's predictions are used against the victim, assuming that the shadow model is a good approximation of the victim model.

Later, Salem *et al.* [237] drastically simplified this attack. The authors showed that using similarly distributed data is not necessary due to transferability of the model behavior on seen or unseen data. According to their experiments, the difference in the model's reaction to a seen or unseen image is transferable to the difference in a model's reaction to a seen or unseen tabular data point or text snippet. Additionally, they removed the need to fit one or multiple shadow models which further decreases the effort of the attack.

Also, the work of Choquette-Choo *et al.* [238] reduced the assumptions needed for a successful membership inference attack and requires only predicted labels and no confidence information. Their work takes inspiration from common data augmentation techniques such as image rotation and translation, and creating adversarial examples to estimate the decision boundary. This estimation of the decision boundary requires thousands of queries, which might not be feasible in all situations. They achieve better attack performance than previous membership inference attacks, that means higher test accuracy of the attack model.

An interesting question is why models react differently to members and non-members of their training data and thereby leak information. Nasr *et al.* [239] observed that the gradients of the model during inference of data points are distributed differently depending on whether the model has been trained on the data. However, their attack needs white-box access to the victim model in order to perform the attack. Shokri *et al.* [236] found that overfitting is correlated with attack success. Later, Bentley *et al.* [240] claimed that the generalization gap determines the vulnerability to membership inference attacks. The generalization gap is the difference between the training and testing accuracy of the model. In other words, the more generalizability the victim model lacks, the more it leads to a lower test accuracy compared to the training accuracy, the more likely membership inference attacks succeed. Overfitting may be one reason for a high generalization gap, Bentley *et al.* confirm previous observations by Shokri *et al.* . In order to prove the optimality of their attack, Bentley *et al.* assume that the attacker has knowledge of the whole data set and the rules used for splitting it into training and testing subsets. Using our formalization above, let $t \subset D$ be the test data set, and (as usual) $T \cap t = \emptyset$, the attacker knows $T \cup t$. This way, the attacker only needs to infer which of the points were actually used for training, formally, $\overrightarrow{x} \in T$ or $\overrightarrow{x} \in t$. But the attacker does not have to reconstruct a superset of the training data set themself. Even though ML practitioners often choose the train/test split from a limited set of commonly known splits such as 50/50 or 80/20, it is questionable whether the entire data set's knowledge and the option chosen for data splitting is known in realistic attack scenarios.

Long *et al.* [241] focus the membership inference attack specifically on outliers. Such outliers have more influence on the decision boundary of the target model compared to non-outliers as the model tries to predict well on all training data points. To find out whether a given target outlier was in the training data set, Long *et al.* fit shadow models with and with-

out the target data point under the assumption that the attacker can sample from the training data distribution. The distribution of predictions of the target model is then compared to the predictions of the shadow models on the same set of queries, and the shadow model that has the more similar distribution determines the membership status.

Similarly, Yaghini *et al.* [242] warn that studying membership inference vulnerability across the whole training data set is an over-simplified view. They demonstrate that under-represented sub-populations in the training data may be at risk even if the membership leakage over the whole training data set is acceptably low. That means, a representative sample of the whole training data set shows that the attacker has negligible advantage over random guess to correctly predict the membership status. They prove that as soon as there are different sub-populations of different sizes, the classifier cannot be resistant to membership inference attacks against all sub-populations.

The work by Atenise *et al.* [243] can be seen as a special case of a membership inference attack, where the goal is to infer the proportion of training data that has a given property. An example from their paper is a speech recognition task in which they inferred the proportion of speakers with a specific accent. Depending on the nature of the property, this type of inference may or may not be considered a privacy leak. The attack is realized with an attack model as well, however, that model trains on shadow models fitted with various different data sets using varying subset sizes with and without the property of choice. So for the speech recognition example above, the attacker would first generate training data sets with different fractions of speakers with accent, such as 10%, 20%, ... 80%, 90%, train a shadow model for each of the training data sets, and then train an attack model that is able to recognize the fraction correctly.

Reconstruction attacks abuse the model to gain knowledge about unknown features of data points. The differences to membership inference attacks are both the attacker's information as well as the information gained by the attack. Reconstruction attacks can have two forms: full reconstruction, also known as model inversion, and partial reconstruction, also known as attribute inference. In model inversion attacks, the full data point is reconstructed given the label and access to the model. This results in a class representative that is most likely not part of the training data, but representative of what the model knows about that class. In attribute inference attacks, part of the data point are known and together with the label and access to the model the remaining attribute(s) are reconstructed. As in model inversion attacks, this results in a representative of what the model knows about data with given feature values, but most likely not a concrete training data point.

In contrast to reconstruction attacks, membership inference attacks assume complete knowledge of the features and reconstruct whether the exact combination of feature values was used during the training or not. There is some connection between reconstruction and membership inference attacks. Yeom *et al.* [244] showed that attribute inference vulnerability implies membership inference vulnerability but not the other way round.

The first reconstruction attacks were presented by Fredrikson *et al.* [245] against logistic regression classifiers. They are attribute inference attacks based on the definition above, but referred to as model inversion in the paper. They demonstrated that feature values could be reconstructed by systematically searching through the feature space of the one missing feature. By taking prior knowledge on the feature distribution into account and the victim's model prediction, they computed for each feature value in the feature space the likelihood of that value being correct, the most likely value is then used as a reconstruction. In their experiments, the reconstruction was often correct. They used drug dosage as an example and showed that the attack was privacy-sensitive as it leaked the genetic markers given demographic information about the individual and the stable dosage of the drug. The paper was later criticized for only extracting the general dependence of the drug dosage on the genetic markers, which is known even without access to the model. The authors later extended the attacks beyond logistic regression classifiers [164]. That paper also introduced the concept of optimizing the guessed feature value given the victim model and potentially further information.

Zhang *et al.* [246] continue with this idea and optimize the unknown feature values towards the maximum likelihood under the victim model. To regularize the optimization and gain high-quality results, the authors use a generative adversarial network (GAN). The GAN is trained on data from a similar but more generic distribution and captures basic dependencies between the features. They evaluate images of faces, and their GAN learns that faces have noses, eyes, mouths etc., from images crawled from the web. The discriminator part of the GAN judges how realistic the reconstructed features are and thereby enforces a realistic output. Yang *et al.* [247] also use a similar but more general data distribution to capture general knowledge about the features. However, they remove the need for a GAN and directly train a model that inverts the prediction function given the partial knowledge about the input.

In the work of Hidano *et al.* [248], the authors propose a model inversion attack (according to our definition in the beginning) in which they remove the attacker's necessity to have knowledge of non-sensitive features by exploiting another ML privacy attack type, the data poisoning attack. By injecting malicious data into the training data set on which the model is re-trained, they force non-sensitive attributes to have no influence on the prediction which drastically simplifies the model inversion attack. Note that the proposed attack only works for linear regression models, as it relies on a model stealing attack and the absence of weights being dependent on more than one feature at a time.

A first attempt to formalize the attacker's knowledge gain in reconstruction attacks was done by Wu *et al.* [249] but focuses only on binary features. The paper raises the critical point that information gain through the model should only be quantified compared to the knowledge from the (training) data alone. Intuitively, if features are correlated in the data, a missing feature can be recovered using these correlations, and that is a property of the data that is independent of the model trained on it.

The reconstruction attack on neural networks presented by Ganju *et al.* [250] falls into our

category of attribute inference attacks with the addition that the reconstructed attribute does not necessarily have to be a feature of the original data point. One of their experiments on face classifiers leaks information about the "relative attractiveness" of the individual while the intended task was to recognize smiling faces. Ganju *et al.* also rely on training shadow models and then an attack model on top of their output.

Model Stealing Attacks aim to reverse-engineer the model given only black-box access to the victim model. A successful attack enables the attacker to launch more powerful privacy attacks since white-box access can leak more information [239]. Moreover, the model's intellectual property may be lost, and techniques to earn money from predictions are circumvented.

Tramèr *et al.* [251] exploit confidence information and the predicted labels to reverse-engineer the victim model. Their attack assumes the model architecture is known and is based on equation solving to compute or approximate the parameters of the prediction function learned by the model, (e.g., the weights of a neural network). For some of the "traditional" ML models like logistic regression, equation solving is easier, but it worked in all of their experiments with various model types.

Correia-Silva *et al.* [252] follow another approach to steal a neural network by training a new network with similar performance as the target. To generate training data for it, they assume that the attacker generates training data randomly or uses data from an unrelated task, for example public images from another domain if the target model is an image classifier. As labels, the attacker uses the labels returned from the target model when queried with the attacker's training data. Hence, the stolen model achieves similar performance as the target model. It behaves similarly to data derived from both problem and non-problem domains as its training process approximates the target model's decision function.

The learning-based approach is also used by Orekondy *et al.* [253], but they shed more light on the training data set. They experiment with different overlaps between the adversarial training set and the target model's training set, as well as with different strategies for sampling the adversarial training set (randomly or adaptively) in order to minimize queries to the target model. Fewer queries are cheaper in the case that target model usage must be paid, and in general, detection of an attack that uses fewer queries could be more challenging.

Juuti *et al.* [254] found further success factors of learning-based model stealing attacks. These findings sound familiar to ML practitioners, such as cross-validation during hyperparameter search being superior to non-validated picks and similar model architecture being more transferable.

However, not only the accuracy of the stolen model should be taken into account, but also how closely the weights align, which is referred to as "fidelity" in the work of Jagielski *et al.* [255]. In order to improve fidelity, they revisit reverse-engineering approaches like in Tramèr *et al.* [251] applied to neural networks. Besides improving both learning-based and reverse-engineering-based model extraction attacks, Jagielski *et al.* [255] also combine the two meth-

ods.

Finally, Chandrasekaran *et al.* [256] take inspiration from active learning algorithms to optimize the queries to the target model. Additionally, they take defensive methods into account that we detail in the next section and study how these defensive methods can be circumvented.

5.1.2 Privacy Mechanisms and Defenses

We review the defense mechanisms mentioned in the literature, recommendations on how to select defense mechanisms suitable for a given use case can be found later in Section 8. We focus below on papers that introduce new protection methods; summaries on such methods can be found, for example, in the following survey papers: [257, 258, 259].

First Line of Defense As overfitting and the lack of generalizability were identified as main factors contributing to membership inference vulnerability by several autors [236, 237, 240], the first line of defense should be overfitting prevention. Concretely, regularization methods are well-known tools in machine learning and help to balance fitting to the training data with being general enough to predict well on unseen data. As a side effect, a well-generalizing model provides not only less membership leakage but also higher quality output. Further information on regularization can be found in the ML literature in general, such as the book by Goodfellow *et al.* [260].

Differential privacy (DP) is a mathematically founded concept first introduced for statistics [261] and then later extended to ML methods. A differentially private algorithm produces an output from a data set that remains almost the same when changing an item in the database. Phrased differently, the output has little dependence on single data points in the data set and rather outputs facts that are true for a larger number of data points. Such behavior is requested by statistics and ML models alike. Formally, DP is defined as follows. Let D and D' be two neighboring databases differing in one entry, and let M be a mechanism that computes a result from the set R over these databases. Then given privacy parameters $\epsilon > 0$ and $\delta > 0$ the mechanism M is differentially private if for any D, D':

$$Pr[M(D) \in R] \leq e^{\epsilon} Pr[M(D') \in R] + \delta$$

The privacy parameter ϵ can be interpreted as a bound on the probability of getting different answers based on the one differing entry. Some implementations of DP have an additional privacy parameter δ allowing a few violations of that ϵ bound. Note that the exact definition of a "neighboring" database that should be protected depends on the concrete use case. For centralized learning, usually a single training data entry is modified, but we revisit this definition for federated learning later (cf. Section 9.1.3). A mechanism M, here, an ML training algorithm,

can be made differentially private by adding carefully calibrated noise to the algorithm's output or during the computation to "hide" the contribution of any single data point in the data set. The amount of noise depends on the privacy parameter(s) as well as the maximal difference possible after M is applied to the databases, referred to as sensitivity [262]. As the above definition needs to hold for any database, every theoretically possible entry needs to be protected. In other words, any possible outlier needs to be protected. This makes differentially private mechanisms privacy-preserving in any situation, but can lead to higher noise being added than necessary if no outliers are present, which is one of the main criticisms on DP. By the definition that uses neighboring databases, DP prevents membership inference attacks if used with sufficiently chosen ϵ and δ. Which values are sufficient can be reasoned about formally using the DP definition or experimentally by checking the membership inference attack accuracy for being close to random guess accuracy.

For learning, DP is most commonly implemented by perturbing the gradient during learning. This approach has two variants. Abadi *et al.* [263] add the noise to a minibatch's gradient during deep neural network training, referred to as DP-SGD. Their algorithm is applicable to any centralized neural network architecture that is fitted with gradient descend. The user inputs a noise parameter and a clipping parameter to the algorithm. After training, tight privacy bounds ϵ and δ are computed using the moments accountant method that was theoretically derived and proven in the paper by Abadi *et al.* [263].

Another way to implement DP for learning was introduced by Papernot *et al.* [264, 265]. They first train several ML models on subsets of the data using standard gradient descent. These models are referred to as "teachers". The teachers' answers are then used to train a "student" model. The student does not get access to the labels, but uses an aggregation of the teachers' answers. During computation of the teachers' answers, noise is added in the aggregation process. Intuitively, if all teachers agree, they must have extracted some general concept that is not dependent on a single data point and therefore privacy-preserving to be used, thus the student can learn from this general concept. Only the student model is released, whose privacy guarantees follow from the fact that it has only accessed differentially private data. The teacher models, which have accessed the data without protection of differential privacy, are deleted after training.

While both methods introduce noise in the training process, which can lead to a degradation of the model's performance, in some cases where the noise is well calibrated, a regularization effect of the noise is observed [263], and the overall test accuracy is improved. Test accuracy and thereby the utility of the model should not be the only parameter to optimize for. Jayaraman *et al.* [266] show that large values of the privacy parameter ϵ or non-zero δ do not mitigate membership inference attacks. Similarly, Rahman *et al.* [267] notice on two data sets that if utility should be maintained, DP-SGD adds too little noise to protect against membership inference attacks. To discover such problems, attacks and defenses need to be tuned and tested simultaneously. Van der Veen *et al.* [268] extended the DP-SGD approach [263] by adaptive clipping and suggest to use large minibatches during training. This decreases the privacy budget necessary for training and thereby increases utility without compromising on privacy

protection.

Note that DP is by definition not suited to protect against reconstruction attacks since they do not extract properties about a single training sample but about a subset of the data. This fact was also empirically shown by Zhang *et al.* [246]. Furthermore, in situations where also fairness is important, DP may reduce fairness as the bias is amplified towards more popular training points. This was shown empirically by Bagdasaryan *et al.* [269].

Other ways to implement DP use objective perturbation, where the objective function that is learned is perturbed, often after it is approximated by a polynomial to make the perturbation easier. Furthermore, Bernau *et al.* [270] compare DP applied to the gradients during learning (referred to as "central DP" in their paper) to applying DP noise to the data as a preprocessing step before standard learning takes place (referred to as "local DP"). They conclude that both methods can be used as none of them is superior in general. Also Bernau *et al.* conclude from their experiments that the privacy parameter ϵ alone is not suitable for comparing different DP methods, as larger ϵ and therefore a lower privacy guarantee for applying DP as a preprocessing step did not lead to lower protection against (empirical) membership inference attacks compared to differentially private learning.

Empirical Methods against Membership Inference and Reconstruction Attacks Another more empirical approach to protect against privacy leakage is to model the attacker during training and optimize against their success. Nasr *et al.* [271] described this as a protection against membership inference attacks by adding an adversarial loss term to the loss function that is optimized during training. Similarly to differential privacy, they observed a regularizing effect preventing overfitting. Jia *et al.* [272] model an attribute inference attacker that is very similar to a reconstruction attacker. They add targeted noise to the feature values to defend against the attribute inference attacker.

Empirical work also shows that both membership inference and reconstruction attacks need more precise outputs than legitimate users and, thus, spark the idea of returning less precise answers. For reconstruction attacks, this idea was introduced by Fredrikson *et al.* [164] and detailed further by Alves *et al.* [273]. Alves *et al.* add noise from a long-tailed distribution to the confidence function in order to lead to divergence of gradient ascend methods during reconstruction attacks. Also, Jia *et al.* [274] add carefully crafted noise to the confidence information but generate the noise differently. The authors realize that the attacker trains an ML model, which can be attacked using adversarial examples. Thus the injected noise turns the responses into adversarial examples against a membership inference attack model. Their evaluation shows that this method yields a better privacy-utility trade-off compared to DP or regularization.

Another way to generate less precise and more privacy-preserving outputs is knowledge distillation. After the ML model is fully trained, another ML model with less capacity is trained based on the first model's outputs to extract the knowledge into a simpler structure that is less

prone to overfitting. Empirically that offers some degree of protection against membership inference and reconstruction attacks [275, 276].

Carlini *et al.* [277] study encoding schemes that try to encode training data such that a standard (non-private) learning algorithm can be used with the encoded data. They formally prove that it is not possible to design such an encoding scheme that is secure against membership inference and data reconstruction attacks that allows learning a non-trivial task. Note that an example of such an encoding scheme is local DP, which is applied to the training data before training. Carlini *et al.* exemplify their attack on the InstaHide algorithm. They conclude that ad-hoc security claims do not provide long-term security and formal proofs should be used.

Protection Against Model Stealing Attacks As protection against model stealing attacks, Tramèr *et al.* [251] show that not publishing confidence information which legitimate users do not need makes the attack harder but does not fully mitigate it. Providing less precise outputs by rounding or applying DP rarely worked empirically, because DP is not constructed to protect against such an attack. Tramèr *et al.* speculate that ensemble methods could work. That means, instead of training one model, multiple models are trained, and their predictions are averaged.

Juuti *et al.* [254] study the distributions of benign and malicious queries to target models. They observe that benign queries follow a Gaussian distribution, but malicious queries are skewed due to the synthetic data generated by the adversary. Their defense blocks queries that are too skewed. However, attackers with the knowledge of this defense could still succeed in stealing the model at the cost of the performance or efficiency of the attack. For example, the attackers can include dummy queries to the model that follow a normal distribution in order to bypass this defense. In order to track all the queries, the defense system requires significant memory.

Chandrasekaran *et al.* [256] studied noise injected in the model output, either data-independently or data-dependently sampled. They realize data-dependent noise by returning predictions over multiple models or from one randomly chosen model, requiring an ensemble of target models. Data-dependent randomization is a suitable defense against adaptive active learning attacks, however, passive learning algorithms may still succeed.

While the above approaches try to protect the target model during attacks, Jia *et al.* [278] propose to embed a watermark in the model such that a stolen model can be found post-hoc. Watermarks are basically backdoors embedded, as discussed before (cf. Section 1). The additional challenge for model stealing protection is to embed the watermark robustly such that it is stolen together with the target model and cannot be removed without harming the performance of the stolen model. This is why Jia *et al.* [278] embedded the backdoor during normal training and showed that known attempts of backdoor removal fail. While this defense may seem weak compared to previous defenses that try to catch the attack at runtime, it is cru-

cial to note that previous defenses did not assume that the attacker knew the target model's training data (and partially the architecture). Watermarks can be embedded even if the threat model of the attacker is stronger and has knowledge of the training data and the model's architecture.

5.1.3 Excursus on Neural Language Processing Privacy Attacks and Defenses

In this section, we summarize some works specific to neural language processing (NLP) as an example of how the general works presented above are applied to a specific field in machine learning.

Song *et al.* [279] apply membership inference and attribute inference attacks to embedding models. They find that despite the fact that embeddings remove information, the remaining information is still prone to the above attacks. The purpose of an embedding model is to generate a low-dimensional real-numbered vector for further task-specific processing. Input to embedding models are words in their context (word embedding) or whole sentences (sentence embedding). As labeled data is scarce for NLP tasks, the task-specific network is often a pre-trained embedding network and fine-tunes it with labeled data. Song *et al.* show that embeddings can be inverted by optimizing the gradients observed from the embedding model towards the most likely input. The membership inference attack is based on a threshold of the similarity score. Thus, sufficiently similar data points are interpreted reported as members of the training data set. This uses the fact that embeddings need to preserve the similarity of inputs in the embedding space. Their attribute inference attacks train an attack model that learns to extract sensitive attributes from the embedding. This uses the fact that embeddings are designed to preserve properties of the input, regardless of whether they are sensitive or not. As a defense against these attacks, Song *et al.* propose to model the attacker during training and add the adversarial success as a penalty to the loss function, similar to the work by Jia *et al.* [272] discussed above.

Hisamoto *et al.* [280] study membership inference attacks in machine translation systems. This is especially challenging as confidence scores used previously for membership inference attacks by Shokri *et al.* [236] are not available. Nevertheless, adding and removing words one by one and monitoring changes in the model's output can be used to construct an attack. The models studied by Hisamoto *et al.* were mostly robust against memebership inference attacks.

Carlini *et al.* [281] show that generative sequence models memorize training data points and these can be extracted by an attacker. This attack is even more threatening compared to membership inference attacks, as data points can be extracted without prior knowledge of a concrete data point. They show that opposed to overfitting that happens "late" in the training process, memorization happens "early" in the process and overfitting prevention is not a suitable mitigation technique. Carlini *et al.* quantify unintended memorization and call this metric exposure. The exposure metric sets a lower bound on leakage of training data and they prove that models with high exposure are always vulnerable to membership inference attacks.

Additionally, they describe how to extract data points from models with high exposure.

Finally, Krishna *et al.* [282] study model extraction attacks on pre-trained language models with focus on BERT, a commonly used model in NLP [16]. They observe that they can create models with similar performance and study defense mechanisms to protect this. The attack on fine-tuned BERT models is easier compared to the general case discussed in Section 5.1.1 since the attacker can start with the same model architecture and the pre-trained model. They generate random texts, query the victim model, and use the answers to fine-tune the pre-trained model to steal the functionality. To defend against these attacks, they tested the detection of out-of-distribution queries similar to Juuti *et al.* [254] as well as watermarking similar to Jia *et al.* [278]. They argue that the latter does not prevent the model from being stolen and only limits the damage, while the sophisticated attackers can bypass the first method.

5.2 Deep Analysis of the Selected Application Scenario

After broadly discussing the research field in the previous section, here, the methods that are available are discussed in detail using the concrete application scenario of the credit scoring task as an example.

Disclaimer

BSI does not comment on whether (deep) learning techniques should be used in such an application. However, it is an example of a use case where privacy attacks might apply. In order to assess the suitability of ML methods for a particular use case, in addition to security also other factors must be taken into consideration which are not addressed in this study, for example the explainability of results, the avoidance of unintended bias, compliance with applicable (data protection) laws and ethical aspects.

5.2.1 Application Scenario: Credit Scoring

The task of credit scoring is to predict whether a person is likely to repay the borrowed credit in the future. We assume here that this prediction is made based on demographic information as well as financial information about the individual. An example data set can be found at Kaggle[17]. The training data has low-dimensional features with small domains each. For example, the Kaggle data set contains multiple features that can be easily inspected manually by data scientists. Such a manual data analysis reveals that features related to delayed payments are represented as a positive number with an upper bound of less than 100, just to

[16] Note that more complex GPT-3 models and their privacy leakage are studied by Carlini *et al.* [283]
[17] `https://www.kaggle.com/c/GiveMeSomeCredit/data`

name an example. Nevertheless, this data can be highly confidential as it contains financial information, such as previous payments or the monthly income. Therefore, the data's training and sharing process must be protected. There are real-world examples of credit scoring models being trained by a third party and given access to for financial institutions.

5.2.2 Attack Vectors

It is crucial to evaluate the literature with respect to concrete scenarios of attacks for three main reasons. First, a successful attack does not necessarily lead to privacy leakage that is threatening in the concrete scenario. Thus, the critical question of what the attacker's knowledge gain means to the concrete case should be answered. Second, researchers make assumptions about the attacker's background knowledge. Although that is reasonable for the paper's running example, one should check whether these assumptions are valid in the concrete scenario. For example, images are often used in research papers. It is easy to crawl similarly distributed images from the web, but in other domains data availability is much more of a problem. Third, researchers often evaluate their work on a few well-established baselines, such as the famous handwritten digits MNIST [284] or facial images [285]. Such data reflects certain dependencies between features, but the findings may not carry over if the concrete case has a different structure. For example, neighboring pixels have a close relationship in images, but neighboring features in a credit scoring data set may cover completely different topics.

While the third question on whether findings carry over to the concrete scenario can only be answered with an actual experimental evaluation, the other two issues of background knowledge assumptions and the meaning of privacy leakage can be answered based on the available literature. Thus, we will focus our analysis on the former two points and can only provide speculations on the third question of transferability.

Membership Inference Attacks enable an attacker in the credit scoring scenario to know that an individual is or was a customer of the data provider, which can be a bank. If the bank operates only in a certain geographic region , the region can be inferred, or if the bank targets customers with certain ethical or religious beliefs[18], these can be inferred. Note that if data is pooled from various institutions and the aforementioned bias towards a privacy-sensitive attribute is reduced, a membership inference attack loses its value for an attacker. For the following argumentation, we assume the credit scoring training data is only from a single data provider.

The initial membership inference attack by Shokri *et al.* [236] assumes similarly distributed data to be available for shadow model training. Such data availability is questionable for credit scoring as financial and personal data is often well protected. Shokri *et al.* require

[18] ethical standards such as banks offering ecologically friendly products only, or religious beliefs such as banks for Christians or Muslims

shadow models to be trained, which is easy due to the simple format of the credit scoring data. However, only if the shadow model approximates the victim model well, the membership inference attack can be successful. These two hurdles, getting similar data and training shadow models, were removed by Salem *et al.* [237] theoretically by exploiting transferability to a different data set. Whether their findings carry over to credit scoring data can only be answered with an experimental evaluation.

Bentley *et al.* [240] claimed that the generalization gap was a measure for membership inference vulnerability. However, they assumed that the entire data set is known to the attacker, and only the membership status is unknown, which is an unrealistic assumption for credit scoring data.

Finally, the argumentation by Yahgini *et al.* [242] also applies to the credit scoring scenario. As sub-populations that could be more vulnerable to membership inference attacks occur naturally in the society, credit scoring data generated from a sample of the society is likely to have similar sub-populations.

Reconstruction Attacks exploit the ML model to reconstruct missing features based on partial information about the data point. A successful reconstruction attack on credit scoring data yields realistic information about features that are unknown to the attacker. The reconstruction of sensitive features therefore threatens the privacy of all individuals, independently of whether their data was used during training or not.

The first reconstruction attacks described by Fredrikson *et al.* [245] are applicable to the credit scoring example as the space is probably small enough to be exhaustively searched by the attack. However, logistic regression models are not likely to be used. As the GAN-based approach by Zhang *et al.* [246] and the direct inversion by Yang *et al.* [247] are tested against deep neural networks, they are more interesting to screen for applicability to the scenario. Both rely on a more generic data distribution compared to the training data distribution. Unfortunately, both attacks were demonstrated only on images of faces, where neighboring pixels have a strong correlation which is not the case for credit scoring data. Note that in both papers, the label-input relation between faces and the name of the person depicted is privacy-sensitive in itself, so it suffices to reconstruct a prototypical representation for the given label. This is not the case for our scenarios: a prototypical example of an individual's likelihood of paying back the credit is not privacy-sensitive. We conclude that while reconstruction attacks are of high interest, the applicability of standard attacks from the literature is questionable and requires in-depth experiments. The reconstruction attack by Ganju *et al.* [250] which can also infer attributes that are not directly present as features has the potential to leak additional sensitive information about individuals that apply for credits in general. Therefore, this attack type should be carefully checked experimentally and be defended against with the methods introduced by Ganju *et al.* in case that leakage is found.

Model Stealing Attacks are applicable to the credit scoring scenario as well. In situations where model access is charged, one should evaluate the cost of these attacks and think about whether these are smaller than legitimate repeated queries. In case the answer is yes, an attack makes sense to save the attacker's money. Another attack target is to use the stolen model as a stepping stone for privacy attacks, as white-box attacks are often more successful. Whether the privacy loss indeed increases due to a stolen model in our concrete scenario can only be answered by experimental evaluations.

Concerning the risk of IP loss through model stealing attacks, it is crucial to define what the IP of a trained model is - the functionality, the design of the neural network architecture, the trained weights etc. Tramèr *et al.* [251] assume the model architecture to be known, so the IP lies instead in the learned parameters of the decision function. The applicability of their attacks depends on the model type used to realize the scenario. Most of the other papers discussed [252, 253] assume instead that the functionality of the model is IP-protected and do not aim to recover the exact learned parameters. As they focus mostly on neural networks, the applicability depends on whether the credit scoring classifier is considered an IP-protected neural network. If the attacker "only" wants to bypass the target model's usage costs, these attacks are reasonable to apply, however, if the attacker wants to use the stolen model as a stepping stone for launching further privacy attacks, the differences in the stolen and target model weights need to be considered. Only an experimental analysis can clarify whether the approximated weights or gradients are helpful for conducting privacy attacks or whether the inaccuracy of the approximation has adverse effects. If a more accurate approximation is needed, the attack by Jagielski *et al.* [255] that also optimizes towards fidelity might be an option.

5.2.3 Privacy Mechanisms and Defenses

In general, all defense techniques discussed above can be used in the credit scoring scenario. As all techniques have their unique drawbacks and gains, we provide recommendations in this section on choosing a suitable defense mechanism. We do not give any statement on the suitability of these methods to comply with legal requirements on data protection. In each concrete scenario, the developer should conduct an individual risk assessment considering the specifics of the scenario, the used data, the legal needs and also conduct practical experiments on its own to measure attack strength and protection capabilities of the defense methods with respect to the concrete case.

The first step for privacy protection should always be to ensure that the model is generalizing well since good generalization capability not only protects private training data but also improves the model quality. For credit scoring, it might already suffice to select a model structure that leads to a low generalization gap to protect against membership inference attacks.

As we assume that only black-box access is given to the model, less precise outputs should be considered as the first line of defense against reconstruction attacks and, possibly, against model stealing attacks. Such a defense can be as simple as rounding the confidence values

generated by the model as proposed by Fredrikson *et al.* [164]. The precise information is not needed by legitimate users and therefore does not degrade model quality but can have significant privacy benefits.

The remaining reconstruction attacks can be defended against by using calibrated noise as Alves *et al.* [273] suggest. Another possibility is to apply the noise proposed by Jia *et al.* [272] to the reconstruction attack case. The remaining membership inference attacks should be mitigated based on DP as that is well-founded on theory. Both the addition of noise during training [263, 268] and the addition of noise to train a second model [264, 265] are valid options to be used. The hyperparameters of the privacy mechanism should be tuned while modeling attacks to avoid that too little noise is added, as observed by Jayaraman *et al.* [266]. As absence of unintended bias is crucial for credit scoring, the recommendations in the work of Bagdasaryan *et al.* [269] should be followed to minimize the bias towards popular training points. Purely empirical methods for protection against membership inference, such as proposed by Nasr *et al.* [271] and Jia *et al.* [274] may work in practice but are harder to explain to the privacy-conscious user or to argue about.

Defending against all types of model stealing attacks presented is not straightforward. Juuti *et al.* [254] do not offer complete protection and only make the attack harder. Moreover, their defense comes at the price of high memory needs to store the query history. However, the memory needs are less of an issue for credit scoring data which is low-dimensional. The ideas by Chandrasekaran *et al.* [256] who experimented with noise as protection are promising, especially when used in combination with other privacy protection mechanisms. Further studies are needed to explore the combination of their proposed ensemble method with a variant of Papernot's [265] PATE approach protecting against membership inference attacks. As the abovementioned protection mechanisms are not perfect, one might consider using the watermark method proposed by Jia *et al.* [278], which enables the victim to prove later on that the model has been stolen. However, this method assumes that the defender learns about suspicious models and explicitly tests them, which is not guaranteed in realistic use cases.

6 Provision of ML Models to External Users: Practical Investigations

Comment by BSI

Differential Privacy (DP) is a method which can be used to mitigate membership inference attacks. One possibility for application of DP during the training of a neural networks is the method DP-SGD (see Section 5.1.2) which is also implemented in several ML frameworks.

However, given a concrete use case, the application of the algorithm is quite difficult for developers. The algorithm requires two parameters, clipping and noise, which both influence each other and determine the effectiveness of the method. At the time of writing there are at least three challenges for an application: The first challenge is to measure what level of protection is actually achieved for a given pair of parameters. In principle this could be done using the definition of DP, see Section 5.1.2, and empirically trying to estimate the values of ϵ and δ. Nevertheless, the second challenge is then to interpret such a measure in a concrete application context and determine whether the level of protection is sufficient, which is also not straightforward. The third challenge is that there is a trade-off between the level of protection achieved by DP and utility of the resulting model. Using too much noise might negatively effect the model accuracy.

In the course of this study, a prototypical implementation of a tool has been developed trying to address the above challenges and supporting developers in applying differential privacy for their own models. The main idea is to measure the level of protection by simulating membership inference attacks against the model and record the success rate of those attacks. This measure has a clear interpretation and is then used as a basis for parameter optimization. As a result, the prototypical tool suggests concrete parameter values for the application of DP-SGD and gives a quantification of the level of protection achieved by this choice.

The approach has several advantages. It is easy to apply and in addition to the suggested parameters, also a quantification for the level of protection is delivered, which has a straightforward interpretation. Moreover, the tool takes into account the existing trade-off between privacy and utility. Last but not least, in comparison with naive approaches for parameter selection, e.g., grid search, the amount of required computational resources is reduced.

However, the approach has also some limitations and further investigations are required. Since concrete attacks are used to measure the level of protection, the result depends heavily on the chosen attack methods and implementations. Since the optimization of parameters is done with the explicit aim of avoiding those selected attacks, there might still be other attack implementations not covered by the optimization. In this sense, it would be interesting to investigate how the resulting choice and implementation of attacks relates to concrete values for ϵ and δ in the definition of DP. Moreover, it has to be noted that the application of DP still remains costly regarding computational resources which might be an issue for larger data sets. It should be stressed that the prototypical implementation has been tested with two data sets which are explicitly tailored in order to produce membership inference vulnerabilities. Practi-

tioners should check that this approach transfers to their concrete data set and model.

The prototypical implementation is addressing experts in the field, leaving the option for further investigation and development. It should be noted, that for a given use case, an individual risk analysis should be conducted taking limitations of used mitigation methods and possible consequences into account. Therefore, the results should be used with caution and under the knowledge that there are limitations.

In the following, the development process and results of the prototypical implementation are outlined. For the implementation, concrete choices of libraries have been used, which are mentioned below. However, those choices should be seen as interchangeable, which is supported by the usage of flexible interfaces.

6.1 Outline

In this project, we created data sets that are vulnerable to membership inference attacks, showed how a grid search can exhaustively find suitable privacy parameters, and then went on to explore how this parameter surface is shaped and how the search can be optimized.

We opted to create vulnerable data sets instead of using any existing data set. This decision was made not only to ensure that membership leakage was present and to save time in the process of data set search, but also serves an educational purpose to make readers aware of factors that can contribute to membership inference vulnerabilities. Hopefully, these practical examples help readers to avoid privacy problems during the data collection phase.

We trained neural network with standard SGD on these data sets, while mostly avoiding the most common cause for membership leakage: overfitting. This term describes a problem in machine learning that occurs when a model, instead of learning to generalize from the data set, learns the idiosyncrasies of the data – it is as if the model would learn the responses to each data point "by heart". As a result, the model can handle seen data points well but often fails to generalize to unseen data points. However, if overfitting would be the only cause for membership leakage, the differences in model performance between unseen and seen data points alone would suffice to detect potential membership leakage problems. Furthermore, we assume ML practitioners try their best to avoid poorly generalizing models anyways. Thus, we focus on the more difficult problem of membership leakage not (mainly) caused by overfitting. To check membership leakage, we simulated membership inference attacks[19], and ensured that the trained neural networks without further privacy protection do actually leak membership information.

When we confirmed we had models that are vulnerable to membership inference, we ex-

[19] We used in our experiments `ART's` membership inference module: `https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/art/attacks/inference/membership_inference/black_box.py`

amined how DP affects information leakage with respect to each DP parameter[20]. We defined a grid with clipping and noise ranges to try out, and recorded remaining model performance as well as remaining privacy risks after each computation. When all trials had terminated, we plotted the results and manually selected a set of parameters that minimizes the attack performance but preserves the utility of the trained network.

Finally, since it was identified that a domain space exists for DP parameters such that model utility is preserved whilst mitigating membership inference accuracy, we explored methods by which to more efficiently determine these parameters without prior knowledge.

6.2 Methods

6.2.1 Creation of Vulnerable Data

We generate two data sets, a tabular data set of synthetic COVID-19 patients, and a chemical data set based of real-life drug therapeutics that predict whether or not a compound can inhibit HIV replication.

Synthea Data Set We use the open source tool `synthea`[21] to generate synthetic patient data for the US. Many diseases are simulated in synthea. Here, we focus on COVID-19 patients to predict survival and, therefore, only use a small portion of the data available. We collect features of symptoms that occur either during the COVID-19 infection or that are ongoing. Timestamps of symptoms as well as well-defined diagnosis codes are available in the data set. To ease understanding, we use the strings associated with diagnosis codes instead of the diagnosis code itself. Another simplification we made was to only record whether a symptom was occurring or not, even though the symptom length was available in the synthetic data.

To create the biased split of the data set, we cluster the data points using demographic features such as race and gender, as well as socio-economic features such as school education and healthcare coverage. It can be hypothesized that more healthcare coverage might be connected both to wealth (to afford healthcare) [286] as well as to previous health conditions (the more health problems, the more healthcare coverage is necessary). K-means clustering with $k = 2$ returns two approximately equally sized subsets. While this split is artificial, note that it might happen naturally in the data, for example, in data from a healthcare provider that is located in a neighborhood of a specific composition or which offers more costly care.

After clustering, we scale numerical features and turn categorical features into numbers to be used by a neural network.

[20] We used the `TensorFlow Privacy` Library for an implementation of DP-SGD: `https://github.com/tensorflow/privacy`

[21] `https://github.com/synthetichealth/synthea/`

HIV Data Set Based off the AIDS Antiviral Screen Data, and taken from the deepchem repository[22], the HIV data set contains over 40.000 chemical compounds screened for evidence of anti-HIV activity. These compounds are split into three catagories in the source - CA (Confirmed Active), CM (Confirmed Moderately actively) and CI (Confirmed Inactive).

The data is provided in the form of a CSV table: The first column provides the molecular structure of the compound in SMILES [287] format. The second column indicates the CA/CM/CI label and the third column is a less granular 'Active' or 'Inactive' label.

In this project, we take the SMILES information of each molecular compound, and focus only on the 'Active' or 'Inactive' property. We use a clustering algorithm to create three different sets of compounds: Each group is maximally chemically 'self-similar', and maximally distant from the other groups. This is intended to approximate a real-life scenario of three chemical or pharmaceutical companies that wish to co-learn around a coordinated biological end-point, and each company is from a different area of 'chemical space' or expertise. However, these companies also cannot afford to reveal their historical data points. Their payoff for learning or expanding their domain knowledge is maximized, but conversely the price of their own research being identified could be economically devastating [288].

To process the data, we first convert the SMILES string to an Extended Connectivity Fingerprint (ECFP) [289], see also Figure 51. This converts an arbitrarily-long molecular representation to a fixed-length binary hash. With these fingerprints, we then use Butina clustering [290] from the `RDKit` package to produce multiple clusters of molecular compounds, with approximately maximized distance from each other.

Thus by using this clustering approach, the three data sets generated by the HIV preprocessing, more closely resemble private data sets of three different companies in different domains, than the example of Synthea data where we produce data with a slight bias.

For our experiments, we use one cluster as training data, the second cluster as testing data and the third cluster is assumed to be in the attacker's possession to check membership inference attacks against. A successful membership inference attack against this data set and setup shows that it is possible to attribute the targeted chemical molecule to the provider of the training data. In case that was a chemical company, the information that the company has experimented with the molecule is lost to the attacker.

6.2.2 Setup of DP-SGD and Attack Testing

For both data sets, we start with a neural network without further privacy improvements to measure a baseline. We use the `TensorFlow` [102] library for it.

For testing membership inference attacks, we use `ART`, concretely, the

[22] `https://github.com/deepchem/deepchem/tree/master/examples/hiv`

`MembershipInferenceBlackBox` class[23]. The attacker's model can be configured, we select both a neural network as well as a Random Forest and report the maximal test accuracy. Since the attack model is an ML model that needs to be fitted, we use half of the available data for training and the other half for testing.

After checking the initial neural network for membership inference vulnerability, we make it private by changing the optimizer to a suitable version from the `TensorFlow Privacy` library. The loss function may need to be adopted as well, but the neural network architecture and hyperparameters stay unchanged.

Note that the choice for `TensorFlow` as a library was arbitrary, `PyTorch` could be used as well, with the PyTorch Opacus library[24], or other libraries, supporting the usage of DP.

6.2.3 Grid Search

So far, the user can only guess privacy parameters and try them out, or try privacy parameters more systematically with a grid search. Therefore, we present grid search as a baseline against which to compare the speed and quality of our approach.

We manually define start points and end points of the search both for clipping and noise values. With a fixed step size, they span a 2D search space which we explore and plot in the following subsections.

6.2.4 Optimization of DP parameters

In order to more intelligently explore the available search space for both clipping and noise, we propose using an optimization algorithm known as Bayesian Optimization to maximize privacy and utility. The term utility refers to the usefulness of the private model, and is usually defined depending on the context. Here, we define utility as the preservation of the initial model performance: the less performance is lost, the higher the utility. This both balances exploration of the search space, and exploitation of known 'good' areas, to avoid getting 'trapped' in local maxima and finds a global maximum more efficiently than exhaustive searching. Bayesian Optimization allows for bounding or constraining a continuous parameter space such as we have, and is well-suited to costly evaluation functions, such as training deep neural networks. The algorithm attempts to best approximate the surface of the space to be explored, by fitting a Gaussian Process around each known sample point at each step, and using the posterior distribution in conjunction with a strategy for exploration (e.g., expected improvement, upper confidence bound) to iteratively determine the next point(s) to sample.

[23] `https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/art/attacks/inference/membership_inference/black_box.py`

[24] `https://github.com/pytorch/opacus`

This iterative process starts each iteration, also called trial, with suggesting a new clipping and noise value. We fit a neural network with the respective parameters, keeping all other hyperparameters fixed, and measure its performance on test data (utility) as well as remaining privacy leakage with `ART`. To tell the optimizer how good this choice was, we combine utility loss and privacy gain into one value. Let u_{start} be the utility (performance) of the initial network without privacy protection, u_i the utility of the private network fitted in trial i, p_{start} the membership inference vulnerability (privacy loss) of the initial network and p_i the privacy loss of the private network fitted at trial i. The optimizer feedback at trial i is then defined as

$$o_i = t * (u_i - u_{start}) + (1 - t) * (p_{start} - p_i)$$

for a trade-off parameter t. The user can tune t to their needs, in case a balanced setup of 0.5 is unsatisfactory. We argue that such a trade-off parameter is easier to understand and use than clipping and noise parameters directly.

To summarize, our Bayesian Optimizer has these parameters: maximum values for clipping and noise to bound the search space as well as the trade-off parameter t for utility and privacy. Moreover, the optimizer can run for a fixed number of trials, or a given amount of time. While the user defines time and/or number of trials to run as well as the trade-off parameter, we suggest maximum clipping and noise values and show experimentally that wrongly over-sized maxima do not harm the final result.

6.3 Evaluation

6.3.1 Synthea Setup

For the synthea data set, Figure 40 shows the average feature values of the two data splits. We focus on those features that were used for clustering. While race and gender have similar values, healthcare coverage is a bit higher in data set 0 (0.017 instead of 0.011), but the largest difference can be seen in the three education related features. The data split 0 includes individuals with higher education, while the data split 1 mostly includes individuals with high school certificates and more with only primary school education compared to the data split 0.

The base model for synthea health data is a 2-layer neural network with 50 and 10 nodes, without dropout. We show two variants of the model, once fitted with the SGD optimizer for 10 epochs, and once with the Adam optimizer for 5 epochs. Both are common choices for model optimizers, but lead to different behavior when using DP which we discuss below.

As the problem is unbalanced, we use the ROC-AUC (Receiver Operating Characteristic - Area Under Curve) for evaluation, as it is easy to interpret: 0.5 means the model is as bad as randomly guessing, 1.0 means perfect, and the unbalanced setup is taken care of automatically.

For membership inference leakage, we use the training data set of the target model as well as the held-out second data set (data split 1). The former is labeled as “members” and the latter

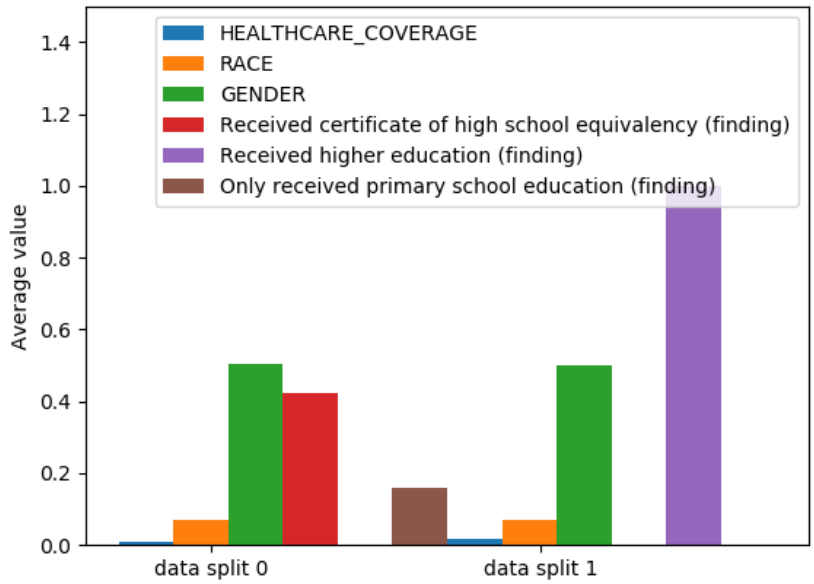


Figure 40: Average feature values in the two different synthea data splits for the features used for clustering.

Table 11: Performance of the base model on synthea data, fitted with SGD for 10 epochs or Adam for 5 epochs.

	SGD	Adam
training AUC	0.990	0.998
testing AUC	0.984	0.992
unseen AUC	0.986	0.995
membership leakage accuracy	0.622	0.639

as "non-members". We use half of the "members" and half of the "non-members" for training the attack model, the held-out halves are then used for testing the attack model. We use `ART's` random forest and neural network attack models and report the accuracy of the fitted attack model on test data. As we use the same amount of members and non-members for training and testing the attack, an attack-accuracy of 0.5 is as good as random guessing, and everything above indicates membership leakage.

Table 11 shows the performance of the base model. We use 80% of the data split 0 for training, that are 1088 data points, the remaining 20% for testing (296 data points) and the other data set (data split 1) with 1036 data points is referred to as "unseen" in the table. The training and test performances as well as performance on unseen data is very similar in all cases. Recall that overfitting is a main driver for membership leakage, but this model does not appear to overfit at first glance due to the good generalizability. Nevertheless, a membership leakage of 0.62 and 0.64 for SGD and Adam, respectively, indicates that some membership information can be inferred.

6.3.2 HIV Setup

The training data set of the HIV data set contains 5627 compounds. The testing data set and external validation data set contain 2953 and 2198 maximally distant data points, respectively.

The target model has two layers with 200 and 100 nodes each, with dropout of 0.1 in each layer. We fit the target model for 10 epochs and observe the AUC as a performance metric due to the imbalanced data set. Table 12 contains the resulting network performances. Note that we only use the SGD optimizer for this setup.

6.3.3 Grid Search

Table 12: Performance of the base model on HIV data, fitted for 10 epochs with SGD.

	performance
training AUC	0.978
testing AUC	0.966
unseen AUC	0.987
membership leakage accuracy	0.767

Synthea Results We explored the region between zero noise and a noise value of 10 as well as clipping between 0.1 and 10. A clipping value of zero would drop all information and directly lead to network divergence and is therefore excluded. We have chosen the maximum values for direct comparison with the optimizer setup explained further below. We used a step size of 0.5 for the grid search, as that leads to sufficiently smooth surfaces. That means, we moved between the minimal noise value of zero and the maximal noise value in increments of 0.5, and similarly for the clipping value.

Figure 41 shows the resulting surfaces for utility and privacy with the SGD base network and Adam base network. Utility is reported as difference between base network utility and private network utility, so it is expected to be negative, and the smaller the value, the less useful the resulting network's predictions.

We observe different shapes of the surfaces and therefore report both. The shape is dependent on the network's hyperparameters (here: SGD vs. Adam optimizers) as well as the data itself and is therefore not further explained here. The conclusion remains the same: More noise and more clipping lead to quickly decreasing privacy loss and slowly decreasing utility loss. This suggests that a good trade-off between utility loss and privacy protection can be found.

HIV Results We observed the surface of the grid search in the HIV case to be less smooth and decided to plot with a step size of 0.2. Figure 42 shows the resulting surfaces of utility and privacy. We observe little utility loss, even large amounts of noise or clipping generally do not destroy the network's usefulness. Already small amounts of noise suffice to mitigate the membership inference attacks, almost independent of how much clipping is used.

6.3.4 Optimizer

We deliver our optimizer with standard settings that serve most use cases. We set the noise and clipping values to explore to a maximum of 10 each, and the trade-off parameter to 0.5, so a balance between utility and privacy. The choice of the maximum value has little influence on the optimization result on our data and we expect this to generalize.

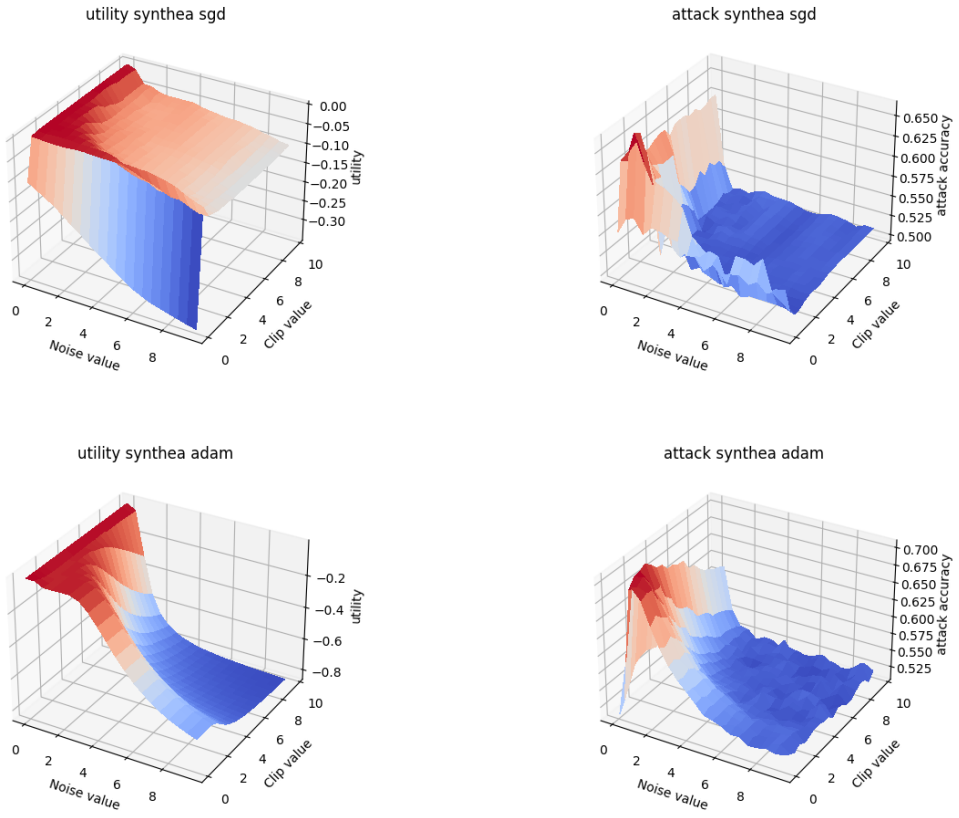


Figure 41: Grid seach surfaces for utility (left) and privacy (right) for the synthea data set using SGD as network optimizer in the upper row and Adam in the lower row.

Synthea Results Figure 43 visualizes the optimization progress given different max values for the synthea use case with SGD. If after x trials of the optimizer, a new best value is found, we mark at that x-position the respective utility, privacy, and value of the optimized function that includes both. We do not show the trials in-between new best values to avoid visual clutter. Even if the max value is set way too high to 100, after about 25 trials, the optimization has found a trade-off with similar quality compared to our recommended setting at 10. Similar observations are possible when the network uses Adam instead of SGD, as Figure 44 shows. Only excessively large maximum values of 100 lead to a delay in optima being found.

The user can also choose to set the trade-off parameter t differently. We report the results in Table 13. Settings for t smaller than 0.5, like 0.25 reported here, put more weight on privacy protection, here, 0.046 utility loss is accepted compared to 0.037 in the balanced setup. Notice that in both cases, more than 0.9 AUC are preserved. Settings with t greater than 0.5 aim to preserve more utility at the price of privacy protection, we display data for the choice t equal to 0.75 in the table. Instead of 0.037, only 0.002 utility is lost, while the remaining privacy risk

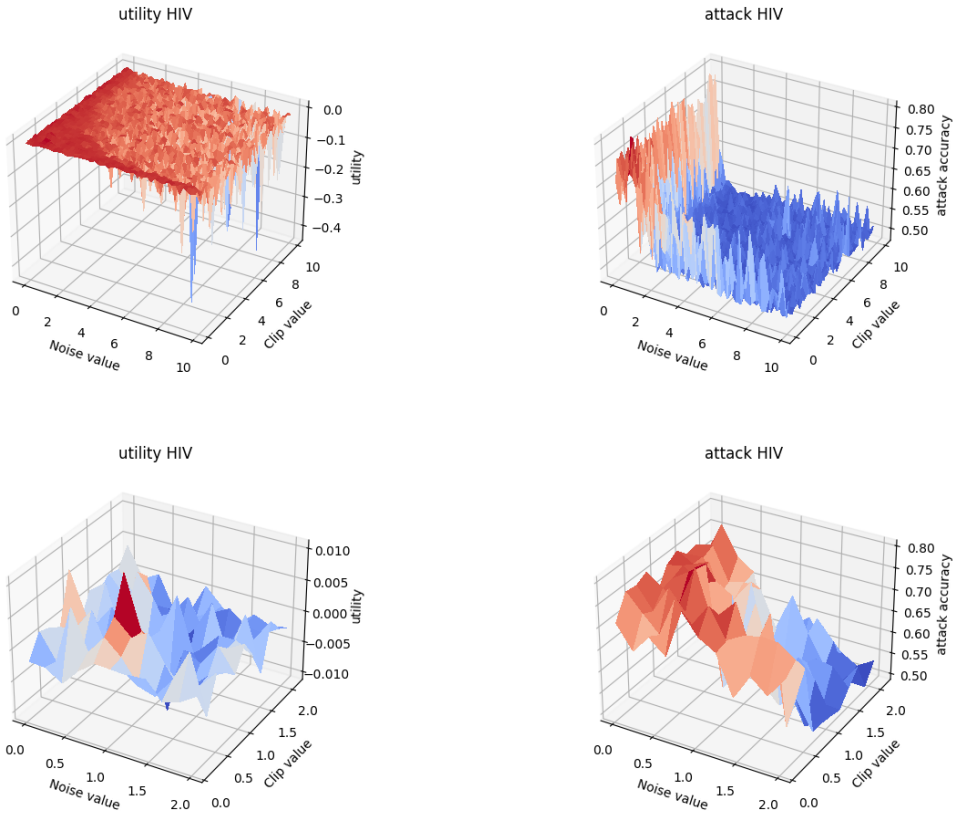


Figure 42: Grid seach surfaces for utility (left) and privacy (right) for the HIV data set, the lower row shows zoomed-in parts of the plots.

raises from 0.497 to 0.518. For Adam, similar observations are made, see Table 13.

Finally, we visualize the utility and privacy surfaces generated by the optimizer to directly compare to the grid search surfaces. Figure 45 visualizes the surfaces for SGD. There is high similarity between these surfaces and the ones created by grid search above in Figure 41. We conclude that it is possible to explore the naturally occurring privacy and utility functions with the optimizer as well, with drastically reducing the amount of computation needed. We have set the maximal number of trials for the optimizer to 100, even though 25 would have been sufficient to find the trade-off, as Figure 43 shows. As a comparison, the grid with maximum values for clipping and noise of 10 and a step width of 0.5 already takes 400 computations.

HIV Results The optimization results on the HIV data set confirm previous findings. 100 trials of the optimizer are more than enough to find a good balance between utility and privacy,

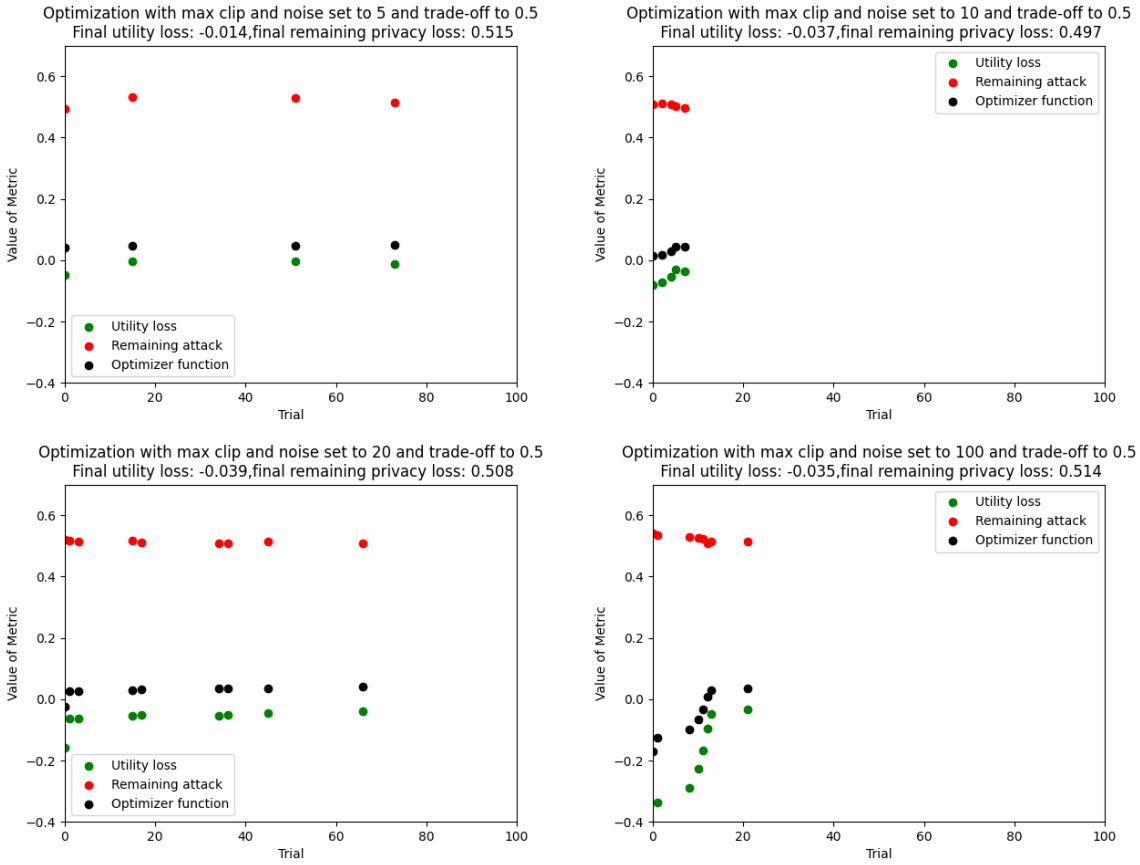


Figure 43: To understand how quickly the optimizer converges, we plot each new best optimization result from running the optimizer on synthea data with SGD. The different graphs display various upper bounds for clipping and noise values.

even if the maximum clip and noise values are set higher than necessary, as Figure 46 shows.

Also the trade-off parameter can be set differently, however in the HIV case this has a minimal impact, see Table 14.

6.4 Discussion

We show that a Bayesian optimizer can find suitable DP privacy parameters at the cost of higher runtime, as several networks need to be fitted.

Note that fitting several networks is a normal process to find good network architectures and hyperparameters. Therefore, users allocate resources for multiple runs as part of their standard practice.

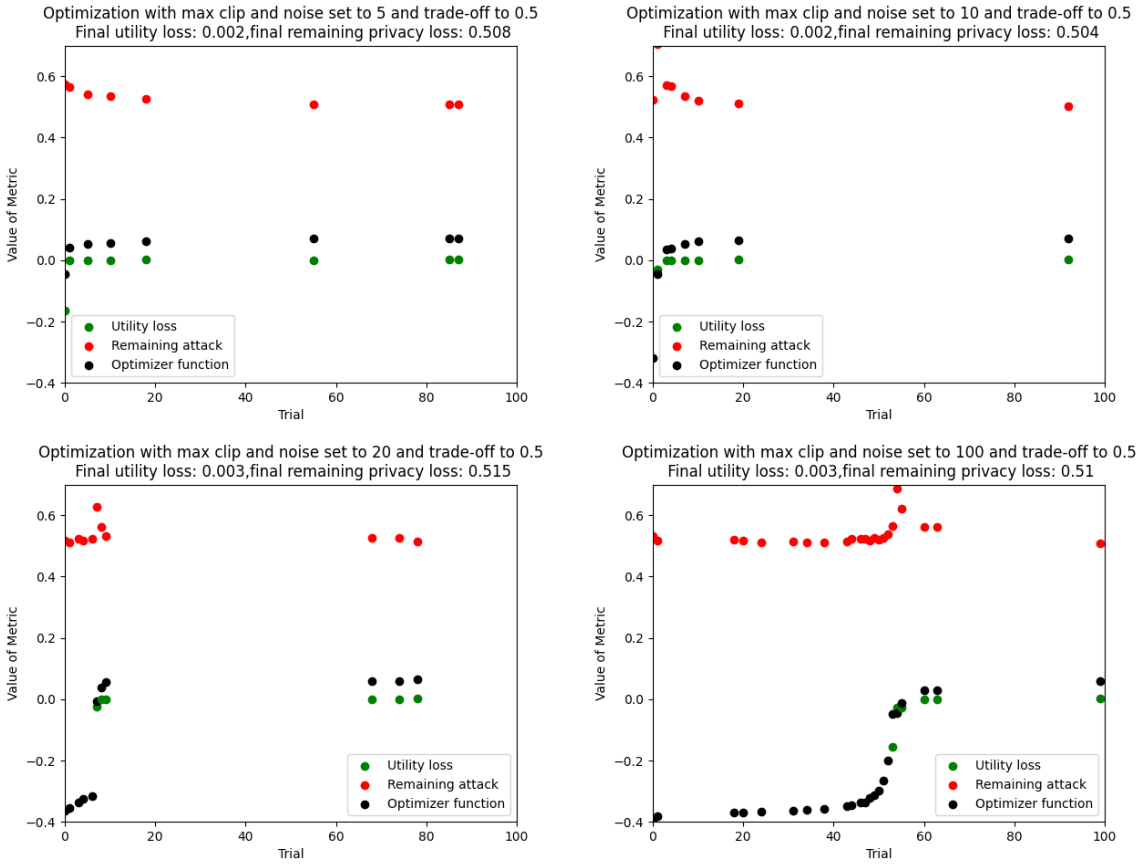


Figure 44: To understand how quickly the optimizer converges, we plot each new best optimization result from running the optimizer on synthea data with Adam. The different graphs display various upper bounds for clipping and noise values.

Our experiments show that the impact of noise and clipping parameters on network performance and privacy leakage are dependent on the data and the network hyperparameters. We conclude that privacy parameters should be tailored to the situation.

As pointed out in Section 5.1.2, differential privacy is only one of many tools to reduce privacy risks. Even when we focus on membership inference attacks, other methods that improve the network's generalizability such as controlling network size and dropout can reduce privacy risks.

Table 13: Synthea optimizer results for networks fitted with SGD and Adam.

trade-off parameter	utility loss SGD	utility loss Adam	privacy risk SGD	privacy risk Adam
0.25	0.046	0.003	0.496	0.508
0.5	0.037	0.002	0.497	0.504
0.75	0.002	0.0	0.518	0.51

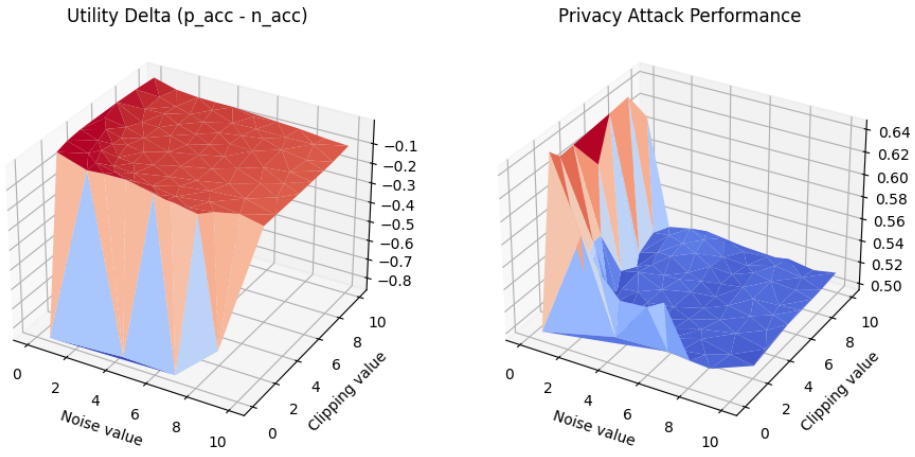


Figure 45: Utility and privacy surfaces generated by optimizer from running on synthea data with SGD.

7 Provision of ML Models to External Users: Problems

The protection of ML models has consequences for the model beyond utility. Additionally, measures to protect against model security issues such as backdoors or adversarial examples as well as measures to avoid unintended bias or explain the model's predictions have consequences on the privacy of training data. In this section, we review work that studies these consequences and interactions between different measures to enable model designers to make an informed choice for their use case.

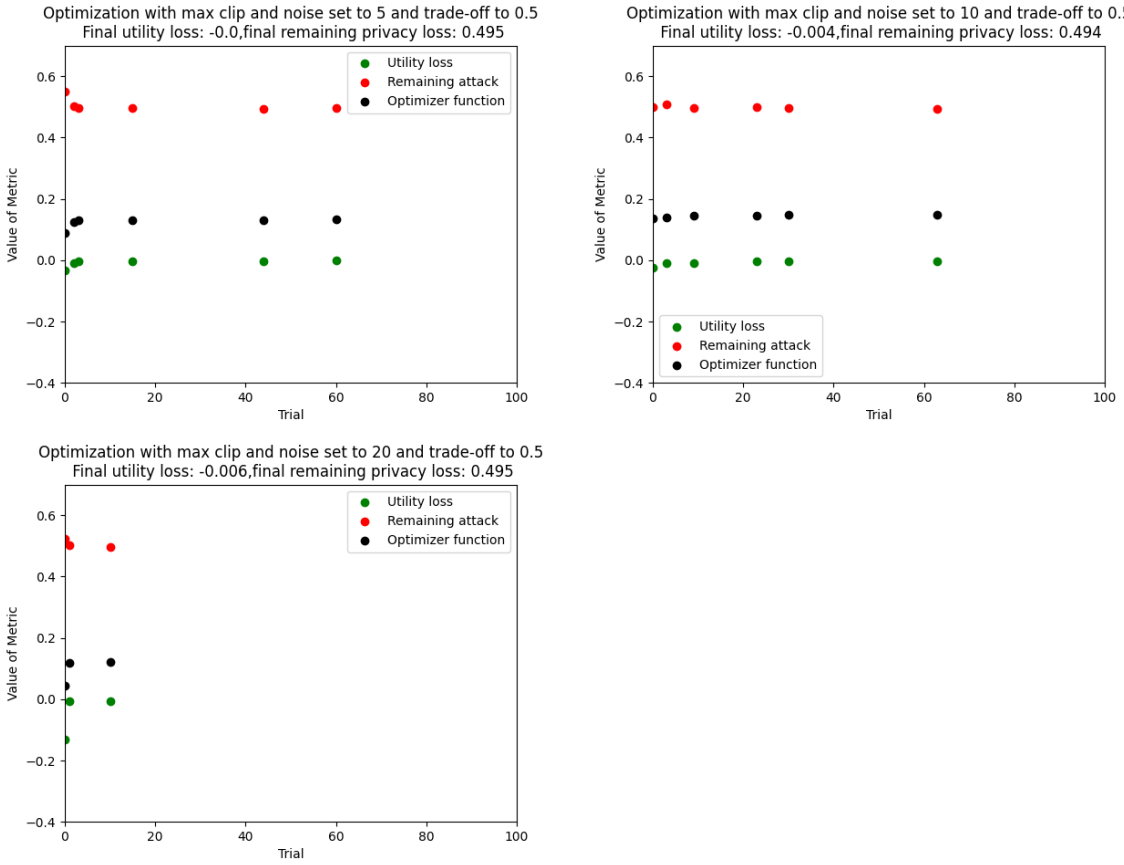


Figure 46: New best optimization results from running the optimizer on HIV data with various upper bounds for clipping and noise values.

7.1 Further Insights Into Attacks and Defenses

Attacks against ML models are diverse: the complexity of reasoning about all possible attack vectors as well as finding suitable defense mechanisms is illustrated by Wang *et al.* [291]. They claim to have found an additional attack vector relevant to machine-learning-as-a-service (MLaaS) settings: the stealing of model training hyperparameters. Finding suitable hyperparameters to balance loss and regularization terms is costly as different hyperparameter choices have to be evaluated which requires to repeatedly fit the model on the training data set. Wang *et al.* [291] demonstrate how to save costs by stealing hyperparameters. A successful attack helps the attacker to access the ML model cheaper than expected by the MLaaS provider, which might ruin the provider's business. Concretely, Wang *et al.* sample a small subset of the training data to let the MLaaS system learn a hyperparameter, then steal these hyperparameters, and finally train the model on the entire training data set with that stolen hyperparameter. This saves computational costs compared to hyperparameter

Table 14: HIV optimizer results

trade-off	utility loss	privacy risk
0.25	0.012	0.488
0.5	0.004	0.494
0.75	0.001	0.490

optimization on the entire data set as in the first step several ML models are trained on a small data set, and the final model is trained just once on the stolen hyperparameter. The authors' empirical evaluation shows that computational costs are indeed reduced for various choices of ML models and training data sets. The paper focuses on traditional ML algorithms such as support vector machines (SVM), logistic and other types of regression models with various choices for loss and regularization terms. All of these models use only one hyperparameter to balance the loss and the regularization term. Moreover, all these types of models are trained by optimizing an objective function until a local maximum is found. This local maximum has a zero gradient, thus by inserting the learned weights into the equation of the derivative of the gradient, the attacker gets an overdetermined set of equations, with more equations than the number of variables (here, one). This resulting set of equations is solved with the least squares method. The objective function might not be differentiable for all weights, but the respective equations can be removed from the system, and due to the overdetermined set of equations the task is often still solvable. The attack also works on model parameters that are not made available, because model stealing attacks can reverse-engineer exact weights for the ML models chosen in the study.

To summarize, the paper shows that in some situations, additional attacks vectors to the privacy attacks studied in the previous sections as well as their combinations have to be considered.

However, the effect of privacy enhancing technologies can be diverse, as Zhu *et al.* [292] summarize. They focus on the privacy technology differential privacy and show their effects beyond the protection against membership inference attacks.

In reinforcement learning with multiple agents, differential privacy reduces communication overhead and limits the impact of malicious agents. Multi-agent systems are characterized by several units ("agents") perceiving the environment and acting from within it, while each of them has access to partial information. Examples for such multi-agent systems are sensor networks or power systems. In this setting, reinforcement learning is used, where trial-and-error actions of the agents are judged by a predefined reward function that informs the system about how well the given action fits the overall goal. Due to the partial information available to each agent, they might ask other agents for further information. With differential privacy (DP), communication overhead is reduced automatically by preserving the privacy budget due to sampling only agents nearby instead of broadcasting the query. Additionally, the noise added due to DP also reduces the impact of malicious agents that return false infor-

mation, as the resulting information is, by definition of differential privacy, almost independent of an individual information source.

Zhu *et al.* [292] also summarize the effects of differential privacy in auctions to protect against privacy risks of the auction outcomes. The authors review various mechanisms, most of which select the winners of the auction using different implementation variants of the exponential mechanism. Auctions are also connected to multi-agent systems, as agents can use auctions for resource allocation and task distribution.

Additionally, game theory reasons about the strategic interaction of multiple agents or players. Differential privacy can address both privacy problems of the agents as well as mitigate the effects of malicious agents. As a side effect, differential privacy can improve finding the equilibrium of the game. The authors cite papers that use differentially private mechanisms in a way that malicious agents cannot profit from their malicious action and thus report truthfully. The methods for applying differential privacy are diverse.

To summarize, differential privacy can be used not only in centralized or federated classification or regression problems, but also have positive effects beyond privacy such as protection against malicious actors or stability or efficiency of the entire system.

7.2 Bias and Privacy

In the following fairness is understood as a property for a set of groups in which each group should be treated equally. For example according to the German Equal Treatment Act (Allgemeines Gleichstellungsgesetz) it is not allowed to discriminate persons on the grounds of racial or ethnic origin, gender, religion or belief, disability, age or sexual identity with regard to certain aspects such as e.g. work space or education. For the scope of this study, fairness is understood in a purely technical sense, i.e. we say that an AI model is fair if a selected fairness metric falls into defined thresholds. The selection of a suitable fairness metric and thresholds for a given use case and the assessment whether the resulting ML system fulfills legal or societal requirements is not part of this study. Most important for the scope of this review is the distinction between fairness as a property of groups as opposed to privacy as being defined for individuals.

The ethic necessity for privacy is due to the protection against discrimination. It should not be possible to infer sensitive information about individuals (data reconstruction attacks) or the membership of an individual in a sensitive group (membership inference attacks) to protect against discriminatory use of this information.

The summary paper by Zhu *et al.* [292] has a section on fairness that first demonstrates several examples in which the use of ML amplifies unfairness problems. Several papers reviewed by Zhu *et al.* achieve fairness by treating similar individuals similarly. A randomized encoding of the training data should encode similar individuals' data into a close representation with respect to a predefined distance metric while preserving as much information

as possible. The randomness in the encoding hides whether an individual is in the protected group or not. The hiding is achieved with differential privacy. The practical problem in realizing this encoding is to design such a distance metric that measures similarity between data points. Zhu *et al.* [292] also point out the general problem that differential privacy is defined on the individual level while fairness is defined on group level. Nevertheless, one paper [293] is reviewed by the authors which adjusts the amount of differentially private noise that is added to different features to achieve fairness. Note that differential privacy is composable such that several differentially private mechanisms can be easily composed to form another differentially private mechanism. This composability property is not fulfilled for current fairness implementations and definitions, which poses additional challenges to the practical design of fairness enhancing systems.

Beutel *et al.* [294] achieve privacy protection and fairness with respect to their chosen fairness metric by removing privacy-sensitive and fairness-related information from the data. This is achieved by a specialized neural network architecture that first embeds the data and then fulfills the ML task using the embedding. The purpose of the embedding is to remove the sensitive attributes. The neural network is trained on the task together with an adversary that predicts sensitive attributes from the embedding. The attack success is fed back into the embedding part of the neural network as a loss which the neural network uses to improve the privacy of the embedding over the training time. The authors show how different fairness definitions can be implemented by changing the adversary goals. Their evaluation shows that improving fairness with respect to the fairness metric reduces the model accuracy, thus they offer a trade-off parameter to tune the fairness-utility trade-off. They additionally observed that by training the adversary on a data subset that is balanced with respect to the sensitive attributes, the adversary can optimize the embedding more efficiently. In summary, Beutel *et al.* [294] make use of the empirical method of adversarial loss similarly to Nasr *et al.* [271] and Jia *et al.* [272] discussed previously. Beutel *et al.* [294] show that the fairness issues can be mitigated with respect to the chosen fairness metric in a specialized neural network architecture at the cost of utility.

Yaghini *et al.* [242] study fairness of membership inference attacks. They find that subpopulations are more vulnerable to membership inference attacks, and differentially private training does not mitigate this difference in vulnerability completely. The reason for this is, according to the authors, that a classifier cannot be resistant to membership inference attacks against the whole database as well as a subgroup at the same time as soon as subgroups are imbalanced within the data set. Such imbalanced datasets may occur naturally, for example because the ethnicities are not distributed equally in the society. The authors do not provide a solution to the problem, but an empirical algorithm to compute a lower bound on membership inference vulnerability which helps to understand the problem for a given task and data set.

The observation that membership inference attacks are not distributed fairly across subpopulations and DP provides significant help, but does not solve the problem completely. Zhang *et al.* [295] confirmed this observation and also proposed a solution. Their work focuses

on the fairness definition of equal opportunity. That means that protected groups should be equally vulnerable to membership inference attacks compared to the non-protected groups. They call it ”vulnerability disparity” if this is not achieved. The solution proposed by the authors deletes training samples to make the data distribution more fair. Distribution differences are encoded into a set of equations that is solved by quadratic programming. Sparse features pose a problem to the algorithm, but k-means clustering can be used to achieve groups with a minimum of samples per group which enables sufficiently large groups. The authors' evaluation on differentially private decision trees shows, if vulnerability disparity is high, the algorithm increases fairness by deleting training samples. Moreover, sometimes model performance increases, probably due to the deletion of outliers. However, the technique is a double-edged sword: if the vulnerability disparity is not high initially, the algorithm increases it, which means that it worsens the fairness problem. Thus, the algorithm should only be used after confirming the existence of unwanted bias in the data set. In general, the paper's solutions should be checked carefully when practitioners apply it to other data sets and models. The reason is that they test the method only on decision trees. Another limitation is that there seems to be a bias in the chosen unprotected group.

In summary, the papers demonstrate that when taking fairness into account in addition to privacy protection, there are only limited options for solving both problems simultaneously. The standard solution of differential privacy is insufficient and it might be necessary to additionally perform changes on the data, for example, by re-balancing the distribution.

7.3 Explainable Models and Privacy

Explainable models are ML models that allow users to understand the model's prediction. This property is key for building trust in the ML model. Usually, ML models do not always make decisions that are accurate and in line with deployment settings, so allowing human operators to recognize model failure and reject the model's decision in those cases is necessary, particularly in areas where models should make critical decisions such as in medicine.

On the other hand, it gets easier to infer sensitive information about the training data if the model outputs more information than just the prediction, as we have discussed in previous work packages. A model's explanation for a prediction adds even more information, thus one should check carefully whether more sensitive information about the training data is leaked through explanations.

Milli *et al.* [296] study model stealing attacks against neural networks and find that explanations simplify this type of attack. They focus on model explanation APIs that provide gradient information for the given input query. Using this information, model stealing attacks need fewer queries compared to previously discussed methods that abuse the prediction API for model stealing attacks. This fact is easy to understand with the example of a linear model, where the gradient with respect to a given input data point leaks the model weights directly. The authors extend this attack theoretically to non-linear models with multiple layers, and

then also show practical attacks against image classification models.

Instead of model stealing attacks, Shokri *et al.* [297] focus on membership inference and data reconstruction attacks using two different types of model explanations. Similarly to Milli *et al.* [296], they study feature-based explanations that return the gradients with respect to a given baseline point. They find empirically that real-world data sets make membership inference and data reconstruction attacks challenging, nevertheless, the fewer training data points belong to one class, the more they are vulnerable to membership inference attacks. Additionally, Shokri *et al.* [297] study record-based explanations that output the difference in the loss function when comparing a model trained without the data point compared to a model trained with the data point. This type of explanation is costly due to the required re-training, but these costs can be reduced with quadratic optimization. The authors observe that this type of explanation most likely returns the training data point itself as an explanation, which is a direct membership inference leak. Especially outliers or data points representing minorities are at risk for attacks. Moreover, data reconstruction attacks are successful if the attacker knows samples from the training data reconstruction. Those samples allow for adaptive attacks that take into account which nodes in the neural network might explain other nodes, and in this way, the entire training data set can be reconstructed.

Both works by Milli *et al.* [296] and Shokri *et al.* [297] are discouraging: model explanations increase the risk for model stealing attacks and privacy attacks against the training data. Baron *et al.* [298] argue that nevertheless, explanations and interpretable ML models are necessary. They focus on pervasive systems, which are systems that interact with humans and are deeply embedded in their environment, such as smartphones. With increasing miniaturization of sensors and actuators and advances in data processing, more and more technologies will become pervasive, for example, eye-tracking technologies are in the process of becoming pervasive. The authors start by defining interpretability requirements such as understanding why a decision was made and how to interact with the model to achieve a desired output. To also fulfill the privacy requirement of not leaking other users' data during the explanation, the model may use only the user's data to infer other private information about the user, but other users' information should only be used in an anonymous way. Baron *et al.* [298] believe that DP should be used to achieve this anonymity. Additionally, they express the concern that too much information is being used or returned to the user, and propose to use existing technologies for feature reduction such as feature selection, principal component analysis (PCA), or singular value decomposition (SVD). Whether or not a given explanation is high-quality is highly subjective and needs to be refined in user studies. Finally, users that understand models' predictions might behave differently, leading to a shift in the data that might require re-training the model. Even though Baron *et al.* also cannot provide a solution for constructing privacy-preserving explainable models, their considerations of what is necessary in the process might guide further research efforts.

7.4 Robustness and Privacy

ML models are called "robust" if they cannot be fooled into misclassifications by attacks at training or at test time. Attacks at training time are called poisoning attacks, they alter the training data to move the decision boundary that the model learns in order to achieve a targeted misclassification at test time. Attacks at test time are possible as well: they alter a data point into an adversarial example which crosses the model's decision boundary but preserves key characteristics of that data point. For example, adversarial examples in the image domain are perceived by humans correctly while fooling the ML model, and adversarial examples in the antivirus domain are still malicious code that is misclassified as benign. Any model that is used in security-critical areas where a misclassification can be harmful should be robust, in addition to being privacy-preserving. In the following, we study robustness in centralized learning.

7.4.1 Robustness against Adversarial Examples

Lecuyer *et al.* [299] are inspired by differential privacy to develop a method to ensure model robustness against adversarial examples. Their proposed mechanism, however, is not differentially private. The authors use differentially private noise to make multiple predictions at test time, which allows estimation of the label distribution. Each class label gets a different probability mass in this distribution after multiple predictions. The authors analyze the difference in probabilities for the largest and second largest probability mass. If the difference is larger than what an attacker can maliciously change, the sample is benign. Otherwise, the model refuses to make any prediction in order to avoid misclassifying a (suspected) adversarial example. This reasoning uses the fact that adversarial examples cannot be arbitrarily different from a true data point, but must preserve their properties, thus, the original and manipulated data point are "close". The authors implement their defense mechanism as a noise layer in the neural network. As reasoning about the sensitivity is easier the earlier in the network the noise layer is added, they place it as the first layer. The evaluation shows that with the network's option to reject adversarial examples, protection is better than protections being state-of-the-art at the time of writing, however, this comes at the cost of increased computational costs at test time. It should be checked carefully whether adaptive attacks can circumvent this defensive measure.

Pinot *et al.* [300] do not modify differential privacy, but use the definition of Renyi-DP in neural networks directly. They derive theoretically that this flavor of differential privacy can ensure robustness against adversarial examples. Experiments are not described.

In summary, the randomness introduced for privacy by differentially private learning or similar approaches can increase protection of models against adversarial examples at test time.

7.4.2 Robustness against Poisoning Attacks

Du *et al.* [301] use differential privacy to identify outliers as well as poisoned data that cause backdoors in the model. They first argue theoretically why differential privacy can achieve this. Differentially private learning makes the model less dependent on single data points, which leads to underfitting of outliers and poisoned data points. This underfitting can be detected by a reduced confidence of the model into the data point. Moreover, the authors prove the dependency between the number of outliers and the amount of noise that is added via differential privacy. The more noise is added, the more outliers can be removed, but the more outliers are in the data, the more noise is required to detect them with a given performance. Experiments with real-world data sets on autoencoders show that even small amounts of noise suffice for significantly improving the detection compared to the non-private baseline.

The results are confirmed by Ma *et al.* [302]. Their threat model is clearly described by the authors. They consider a powerful attacker with knowledge of the entire training data set, the ML algorithm and privacy protection being used as well as the noise distribution (i.e., the exact noise value is unknown). The adversary can change features and labels as well as add or remove entire training data points. The authors study different attack goals and strategies for minimizing attack costs. The attacker may target specific labels, i.e., target a small set of data to be misclassified, or a large set of data to be misclassified. Also parameter-targeting is studied where the misclassifications should remain close to original predictions to avoid detection. The attacker can use different strategies for selecting which data points are worth poisoning. Heuristic strategies based on the largest initial gradient norm or the influence of the data point can be used as well as optimizations of the attack cost via stochastic gradient descent, which is applicable to models with objective perturbation (such as Abadi *et al.* [263]) or output perturbation. Their empirical evaluation shows an exponential decay in robustness with the amount of perturbed data points as well as selection strategies via stochastic gradient descent optimization being most successful. Unsurprisingly, more privacy during training makes all attacks less successful.

Both papers show that privacy protection as well as robustness against data poisoning attacks both require the model to be relatively independent of single data points, thus privacy and robustness issues can be addressed with the same tools.

8 Provision of ML Models to External Users: Recommendations

In the following, we describe steps to design an ML system that quantifies and reduces the risks of privacy attacks (membership inference attacks, data reconstruction attacks and model stealing attacks). The steps consider the model training phase as well as deployment and usage of the learned model. The information gathered in the process such as considerations, choices and empirical observations, serve as a basis for a documentation. Such a documentation can be the basis for a discussion on privacy risks with stakeholders.

Disclaimer

While the provided recommendations can help to minimize the risk of privacy loss via ML applications, it is important to be aware of the limitations. The strategy described below has been derived on a purely conceptual level, taking into account literature reviews and practical experiments conducted in the course of this study. However, the strategy has not been applied to a real-world use case. Practitioners should check carefully whether the same strategy applies in their individual use case and if additional steps need to be taken. Moreover, it needs to be stressed that the security of AI systems is an active and fast developing field of research and that new developments need to be considered continuously.

8.1 Understanding Attack Vectors

The first step in deciding on a mitigation of privacy attacks for any given situation is the identification of attack vectors that are applicable.

A detailed explanation of attack vectors against a trained model can be found in Section 5.1.1. In the case of a model that is provisioned to external users, the model itself can be targeted by an attacker in the form of a model stealing attack to gain unlimited access to the model or a model very similar to the targeted model. Alternatively, as training data is often IP-protected and contains private information, attacks against the training data via the exposed or distributed model, such as membership inference and data reconstruction attacks, need to be strongly considered.

Additionally, the following choices and factors influence the feasibility of potential attacks against any provisioned model, and must be taken into consideration:

- Data type and featurization method
- ML architecture
- Release mechanism (e.g., shared model / model API access)

However, not all theoretically possible (or practically demonstrated) attacks are applicable to the situation at hand.

The type of data and task influences which attack types are applicable and the risk that these attacks succeed. A model inversion attack might be possible, but it does not have to pose a privacy threat, if the labels are not privacy-sensitive and therefore the "leaked" prototypical class example is not privacy relevant. Many concrete attacks assume the attacker to have access to data from the same or a similar distribution as the training data, for example, for training shadow models as part of membership inference attacks. The availability of both data and metadata are pertinent when assigning risk in these instances: Restrictions may be made to limit the availability of column headers, data ranges, data types etc. However, aspects of this data may still be able to be inferred from context, such as publications of involved authors, the goal of the project, general domain-specific expertise, etc. While it is plausible that the attack has, or may infer, such data for image classifications tasks as used in research papers, that assumption might not hold for niche data types. In the case where the attacker has *no prior knowledge* about the domain, a target for the membership inference attack needs to be created from scratch. It is possible that the probability of generating a meaningful data point, or even a training data point, is so low that it may take too long to actually generate a positive membership inference result. This is particularly relevant if the input space is sufficiently vast, even if the targeted neural network itself is leaky. Take for example a 1000-bit binary input vector: With no prior knowledge about the probability distribution of the input bits, there are about 10^{301} possible inputs, and only a tiny fraction of those may be used during training of the target classifier. Notice, however, that if the attacker does have prior knowledge about the domain, membership inference attacks can be targeted to relevant subsets of the domain and might be feasible.

Comparably, it might be that these vectors are much shorter and within the possibility of an exploratory attack, but are the product of a one-way hashing algorithm, which might prevent reverse-engineering attacks because the reversed model input is meaningless. That can be either due to the function being a one-way function that cannot be reversed, for instance, a cryptographically secure one-way function. Or the function can be reversed, but not with the knowledge the attacker has available. Similarly, the architecture of the model may mitigate certain attack vectors. For example, graph-based neural networks that take a graph representation as an input instead of a feature vector. In many implementations of graph-based neural networks, a part of the preprocessing infrastructure converts a graph representation into a fixed-length embedding – a process which is necessarily a many-to-one process and therefore lossy. Such a non-reversible process may in itself be sufficient to prevent model inversion attacks.

However, the model architecture can also increase the attack risks. Support vector machine (SVM) models by default embed training data within their support vectors. We recommend using a different model type, or privacy-preserving SVMs like described by Lin *et al.* [303]. Similarly, Random Forest models can be easily converted to tables of decision rules [304] which may disclose fragments of training data domain.

Some attacks, such as model stealing attacks but also the membership inference attack by Choquette-Choo [238] need to query the model. Thus, the feasibility of such attacks depends on how the model can be queried. Authorization before usage of the model or rate-limiting might delay the attacks or even render them infeasible. Note that we assume throughout this case that black-box access to the model is given. If instead the model is released to potential attackers (white-box access), attacks might get easier for the adversary [239].

A comprehensive understanding of the relevant environmental factors to the data and model ecosystem is the first requirement towards designing a privacy-preserving architecture, alongside an understanding of the applicability of all applicable attack vectors. Note that also the combination of different applicable attack vectors should be taken into account. For example, the attacker is given black-box access to the model which makes the attack against the training data infeasible, however, once the model is successfully stolen, the attack against the training data can become feasible.

8.2 Understanding Additional Constraints

Theoretical attack vectors that have been identified are not the only consideration when selecting a defense solution for deployment. Some use cases may require additional considerations, some examples of which may be:

- Conforming to legal requirements
- Conforming to parties' contractual requirements
- Ensuring that models are explainable
- Ensuring that models meet certain fairness metrics

Legal or contractual requirements for certain standards may influence the selection of available defense solutions against attacks. Moreover, requirements that all parties must participate in certain authentication protocols or logging may provide trust. For example, authentication and logging of excessive polls against a model could uncover model stealing attacks that require more queries than what is expected from legitimate usage.

It may be that local governmental or legal requirements dictate that all relevant data is sufficiently anonymized that even the isolation of some singular data set entries does not constitute a data breach. In this instance, some attack vectors might be disregarded, notably costly data reconstruction attacks, since the data set itself has been hardened against such an attack, offering protection of the underlying data. However, often anonymization is not trivial and requires expert knowledge.

Conversely, collaboration requirements may require explainable models that may prohibit certain defense solutions by nature or mechanism. An explanation mechanism that de-

pends upon gradient information may be mutually incompatible with defenses against model inversion attacks [296, 297].

8.3 Risk Analysis and Attack Plausibility

Once the decisions with respect to the model itself, the applicable attack vectors, and the additional constraints have been concluded, the severity of the attack vectors themselves should be evaluated. Attacks that are entirely theoretical, or that are not feasible to implement practically, might be excluded. In this case a thorough risk analysis should be done, which takes into account:

- The applicability of the attack
- The current feasibility of the attack
- The anticipated future feasibility of the attack
- The anticipated lifetime of the system
- The anticipated growth of the system (complexity, scalability, trustworthiness of future users)
- The value of the system and the motivation of a potential attacker
- The access-level of potential attackers

A well-known or a mature attack vector should be given high priority if it pertains to the user situation. The more widely-known, accessible, or well-developed the attack, the more likely it is to be used adversarially against the platform. This heavily ties-in to the feasibility of the attack, which depends upon required expertise and availability of example implementations. If an existing proof-of-concept attack can be easily repurposed against a user implementation, especially by a low-skilled adversary, then a high priority should be given to defenses against such an approach. For example, if the respective attack can be found in open-source attack libraries, it is more likely to be used against the system compared to an attack that was just recently described in a research paper without concrete implementation.

Similarly, consideration should be given to the future. If the system is expected to have a lifetime of several years, consideration should be given to the maturation not just of the system itself, but also towards the potential future attack vectors against it. An attack which might currently be esoteric or theoretical can quickly mature into a proof-of-concept and from that very rapidly propagate into the wild as a feasible attack. Similarly, a small deployment amongst trusted users released to a broader audience can quickly outgrow an honest or semi-honest ecosystem into a malicious one, rapidly changing the scope of required defense mechanisms. Additionally, if the expansion of the system adds significant value to it, then even

if the trustworthiness of the participants does not change, the system may become a more desirable target, making attacks that were previously considered infeasible, feasible. On the other hand, if the system is planned to be updated regularly, additional defenses can be made part of those updates whenever the attack vectors change. That reduces the burden to predict future developments during the initial system design phase.

The access-level of attackers is also an important factor to consider: A closed system might negate attacks from malicious attackers that require access, or an attack might require ownership of a compute node that could be owned by a third party. In distributed systems however, the growth of a system over time and users adds not just potential target value and more mature attacks, but also a wider audience to consider: Earlier stages of deployment usually mean better-trusted participants and a higher level of accountability, but as the system grows, attention should be focused on the corresponding growth in attack surface.

When evaluating the impact of an attack, care should be taken not to lightly rule out attacks based on quick arguments such as "There is no monetary gain from the attack" or "That information cannot be used to harm individuals". These kind of thoughts are often biased and error-prone, leading to an underestimation of the actual risk.

8.4 Mitigating Attack Vectors

Based on the system and model architecture as well as potential attack vectors, the next step is to select defense mechanisms to mitigate these attacks. We presented a more comprehensive outline of potential defense solutions in Section 5.1.2, and describe in the following the process of selecting suitable solution(s) for the concrete case.

8.4.1 Selection of Defense Method

Once a suite of solutions for the relevant attack vectors has been identified as described in the previous step, consideration must be made when selecting the specific implementations. The following points serve as starting point and general guideline. Concrete use-cases may need prioritize these items differently.

- Simpler methods should be preferred over equivalent solutions that come with a higher complexity.
- The earlier in the deployment chain in which a method can be implemented, the more preferential it is to an equivalent later approach.
- Methods that preserve or increase utility should be preferred over equivalent methods that incur a loss in utility.

- Methods with a formal proof of security should generally be preferred over equivalent methods that demonstrate empirical security.

An example for the first point of preferring simpler methods over complex methods is protection against membership inference attacks. Due to the increased vulnerability of poorly generalizing models to these attacks [236, 237, 240], it is recommended to solve the generalizability problem first (e.g., by regularization or early stopping). Improving the model's generalization is both simpler and increases utility compared to differential privacy.

On the other hand, differential privacy is a method with a formal security proof that should be preferred over empirical membership inference protections like Nasr *et al.* [271], as the former is easier to justify.

Examples for solutions that are implemented in earlier steps of the development chain are improving data quality and using reasonable anonymization techniques before using dedicated data reconstruction and membership inference defenses. A data preprocessing step containing anonymization can increase generalization abilities of the network and improve protection against membership inference attacks. Making sure that there are enough data samples even for smaller subgroups (e.g., minorities in a data set about the population) in the data also increases generalizability, and thus increases utility while keeping the remainder of the development chain unchanged.

8.4.2 Evaluation and Implementation of Selected Solutions

With the selection of defense solutions, they should be revisited from an aggregate perspective. Not only should it be confirmed that all applicable attacks are counteracted, but further that system performance or maintenance cost is not prohibitively compromised. Solutions may together add a computational overhead that makes end-user performance unusable, or increase code-base complexity to the point that it is impractical to maintain or debug or deploy. In this case, it may make sense to select defense mechanisms that mitigate more than one attack, or to replace more comprehensive solutions with multiple smaller, simpler, or faster solutions. For example, noise can protect against membership inference attacks and data poisoning attacks (see Section 7.4). No one-size-fits-all solution exists, so consideration should be given with regards to:

- Utility/privacy trade-off
- Computational overhead
- Applicability to attack vector(s)
- Code-base maintainability
- Implementation complexity (how long would it take to train a new engineer to the solution)

- Availability of model or API to participants

The background of solutions or solution implementations is an important consideration. Well-maintained or active code-bases should be preferred over orphaned or infrequently-updated code-bases, as potential security flaws are more likely to be noticed or addressed in a timely manner in the first case. The trustworthiness of a source is also an important consideration.

Many privacy-preserving solutions involve a notable trade-off, to either the utility of the end model (e.g., Differential Privacy) or the computational overhead (e.g., homomorphic encryption). The impact of these trade-offs depend heavily on the use case, and thus, the ranking of potential solutions similarly depends on the use case. A model architecture that pushes computational saturation of cutting-edge hardware may prefer a trade-off on utility over a compromise on computational performance. Conversely, a smaller network may prefer slower computation than to sacrifice the utility of the end model.

At this stage, the selected suite of possible defenses should be evaluated - not just individually but as a complete implementation. Solutions against one attack vector could feasibly disrupt or simplify another attack vector, and a holistic view of the chosen end solutions must be considered. A (mock) implementation should be made to test the system empirically, from both a security perspective as well as a performance perspective.

8.5 Deployment and Maintenance

The defense possibilities, risk assessments of attacks, reasoning behind the chosen solution, and the respective disadvantages, should all be documented with their corresponding justification. This transparent process builds trust in the chosen solutions, and also allows auditors to verify the absence of mistakes in the process.

Multiple abstractions of the chosen solution should be made for review by developers, for audit by relevant auditor parties, and for abstraction to users. Consideration should be given to each with respect to the technical knowledge of the respective audience, the applicability of the information (e.g., users likely do not require a formal proof of security, whereas auditors likely do), and the succinctness of the documentation.

After the system has been deployed, regular checks should be made regarding literature and state-of-the-art attacks, as well as user growth, user trustworthiness, attacker motivation, and protection mechanisms.

Part III

Provision of Datasets to External Users

Hagestedt, Inken (Apheris AI GmbH)
Withnall, Michael (Apheris AI GmbH)
Höh, Michael, Dr. (Apheris AI GmbH)
Zimmer, Raphael, Dr. (BSI)
Sennewald, Britta (BSI)

9 Provision of Data Sets to External Users: Literature Overview

In this part, untrusted third parties should be enabled to learn on confidential data sets. They may use one or several of these confidential data sets from different sources for their training. To further define the case, we assume that the confidential data set cannot be handed out directly. Instead, we discuss two ways to provide data access: Either by setting up a federated learning process or by handing out synthetic data generated from the confidential data set. In short, **federated learning (FL)** means that at least two parties jointly train an ML model but do not pool their data in the process. Privacy-preserving synthetic data generation aims at mitigating possible privacy attacks and providing data of sufficient quality such that the receiving third party can train an ML model with similar results compared to a model trained directly on the confidential data. We provide more details on these two approaches in Sections 9.1.1 and 9.1.4. Furthermore, an additional technique called homomorphic encryption is discussed in the course of this section. It encompasses a set of methods for encrypting the data to be shared in a way that allows for conducting calculations on the encrypted version of the data. Hence, it enables various parties to share their encrypted confidential data (e.g., on a centralized server), where the ML procedure can be conducted and corresponding models can be trained.

9.1 Broad Discussion of Relevant Research Field

9.1.1 Background on Federated Learning

Federated learning [235] is based on the idea that multiple parties do not directly share their training data, but only the insights obtained from the data that are necessary for learning. Intuitively, this increases privacy as no data directly leaves the data owners' secure storage. Whether this intuition matches reality will be discussed later after this general introduction to federated learning.

Fig 47 gives an overview of the most common federated learning setup. The multiple parties, often called clients, who train the model together always fit the model to their local data set. After the local training, the local models need to be combined. This can either be done by a central server or by one of the clients in a peer-to-peer setup. The result is a global model with insights from all clients that can be used by the clients again for further local training. The local training and aggregation of results are repeated until some convergence criteria are met, for example, if the accuracy on a held-out validation data set stops increasing or after a fixed number of rounds or computation time. Aggregation can either be done by computing a (weighted) average of the local models' learned parameters, often referred to as federated averaging [305] in the literature. Or, if the model is trained with stochastic gradient descend (SGD), it is also possible to share the loss of the gradients [306] instead, such that the gradients are averaged and not the learned weights. This is called federated SGD in the literature. In special cases, where the clients do not have enough computation power for learning and bandwidth constraints limit the sharing of raw data such as in IoT devices, split learning can be used. The deep neural network is split between clients and server such that the clients train the initial layers and the server the final layers [307, 308].

The number of available clients and their data distribution is crucial for convergence, model quality and privacy. Therefore, we distinguish between cross-silo federated learning with 2-10 clients and cross-device federated learning with several hundred or thousands of clients [309]. Note that in the literature there is no strict boundary between the two. In cross-silo federated learning, as depicted in Figure 47, there are only a few data contributors with large data sets each. Our scenario described later in this section on chemical models are cross-silo setups. Cross-device federated learning has many more clients, each of them with a much smaller data set. An example of these is federated learning of a next-word predictor for smartphone keyboards [310]. In contrast to cross-silo learning, cross-device learning has to be robust to unreliable clients that drop out of the learning process, and learning needs a small memory and energy footprint to be run on end devices such as smartphones.

Another important distinction in federated learning setups is between horizontal and vertical learning. This distinction refers to how the data is split between the clients. In horizontal learning, all clients share the same feature space but have data generated from different entities, as depicted in Figure 47. An example for horizontal learning would be the credit scoring task as presented previously (Section 9): data sets of different clients would typically contain data of different customers. In the case of vertically distributed data, all clients share data about the same entities but have different feature spaces.

Federated learning has unique challenges, such as non-independent and not identically distributed data (non-i.i.d. data), hyperparameter optimization and model selection. These challenges are not fully solved in research yet [309]. With non-i.i.d. data, we refer to data set splits across the clients that violate the assumption on independently and identically distributed sampled data. This standard assumption in ML tasks is often violated in practice due to the nature of the FL problem. Labels or features might be distributed differently in the data sets of the clients, the quality of the data may differ, data might not be sampled independently

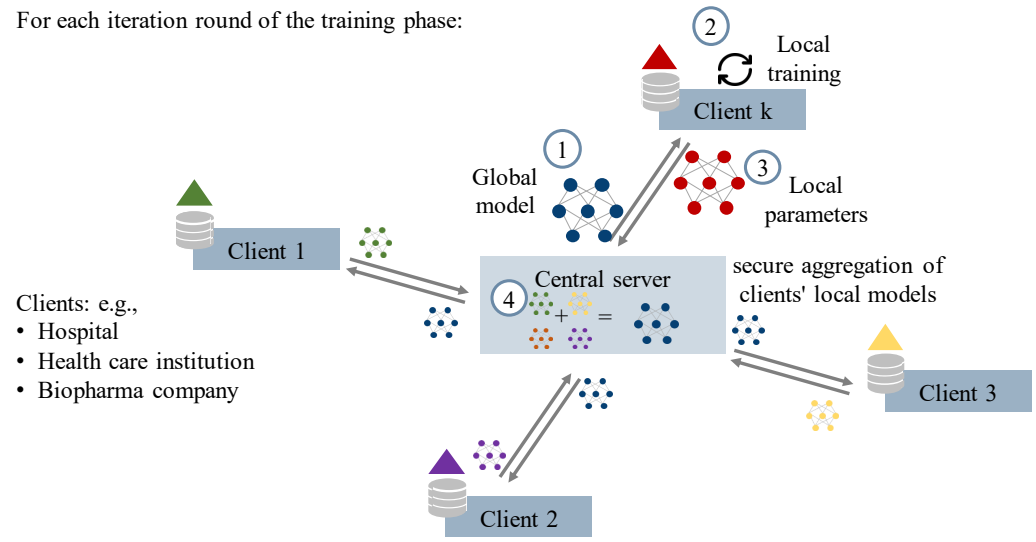


Figure 47: An overview of the most common federated learning setup. We depict horizontal federated learning in cross-silo setting here. A learning round starts by downloading the global model (step 1), then clients further train the model locally on their data (step 2) and upload the new model parameters (step 3) to the server, which aggregates all clients' parameters into a new global model (step 4).

or there might be temporal shifts in the data collection. Further examples and a formal definition can be found in the summary paper by Kairouz *et al.* [309]. Federated learning algorithms have additional hyperparameters such as when and how to share gradients or weights, the number of local steps before aggregation. Traditional hyperparameters of the model such as the number of epochs to train a neural network and the learning rate are relevant to federated learning as well. Note that hyperparameter optimization is more costly in a federated setting, especially in a cross-device setting due to resource scarcity, while at the same time the optimization is not only for model performance, but also for communication and computing efficiency. Model selection is also more challenging in federated learning compared to centralized machine learning since the data cannot be directly inspected.

9.1.2 Attack Vectors

The types of attacks against federated learning at training time are similar to centralized learning explained previously in Section 5.1.1. However, we exclude model stealing attacks as we assume that the server and all the clients have white-box access to the model and, thus, no benefit of stealing the model. As described in Section 9, membership inference attacks determine whether a given data point was used during training or not. Reconstruction attacks exploit the model to reconstruct missing feature values.

Compared to centralized learning, the attack surface of a federated learning system is more extensive. Not only can users of the final model be malicious, but any client that participated during training as well as the server can also be malicious. An attack can happen at any time during training. We distinguish between passive and active attacks and focus on privacy attacks even though poisoning and adversarial attacks are applicable as well. In passive attacks, the malicious party follows the federated learning protocol correctly but extracts privacy-sensitive information in the process. In cryptography, such a passive attack is referred to as "honest-but-curious". In active attacks, the malicious party manipulates their outputs to maximize information extraction, sometimes at the cost of overall accuracy. Nevertheless, too much manipulation could be detected and result in abortion of the training process before the attack has succeeded. In our literature review, we focus on papers introducing new attacks. Summaries of existing attacks can be found, for example, in the following survey papers: [311, 312, 313, 314, 315, 316, 317, 318].

One of the first attacks that was proposed against federated learning was such an active attack by a client, described by Hitaj *et al.* [319]. They show that a malicious client can reconstruct prototypical examples of another client's data, so the attack is a model inversion attack. The attack is active as the malicious client inserts artificial data with the wrong label to force the model to release more information about the actual data. The necessary artificial data is generated by a GAN where the discriminator uses the currently learned federated model to judge how realistic the generation of the fake sample currently is.

Melis *et al.* [320] define a property inference attack carried out by clients during federated learning where the inferred property should be unrelated to the label. They point out that the previous attack only works when the label itself is sensitive (e.g., the name of the person depicted in the images). In the example of images, such a property could be whether a person wears glasses if the label is the person's gender. This is a form of an attribute inference attack. All the information needed for the attribute inference attack and an additional membership inference attack is extracted from shared gradients. The passive forms of their attacks only observe the gradients. The active property inference attacks add a loss term for the property of choice and report back gradients towards the joint loss. This forces the model to learn more separable representations of the label-related and property-related information, making the attack more likely to succeed.

However, the attacks all assume gradients over small batches of training data to be shared

instead of a local model trained for several epochs over the whole training data of the client. This results in higher communication overhead compared to federated averaging and is by construction inferior in terms of privacy. A similar setup of gradients over small batches of training data is used by Zhu *et al.* [321]. The evaluation of defense methods shows that only increased batch sizes are suitable mechanisms. This finding suggests that the attack does not carry over to federated averaging settings where weights of trained local models are shared instead of gradients computed from small data batches. The idea of optimizing gradients of a guessed input for reconstruction was also used by Geiping *et al.* [322] under the same assumption. The formalization of privacy leakage using gradient optimization by Wei *et al.* [323] is based on that assumption as well.

Nasr *et al.* [271] show that even in the more realistic setup of federated averaging, gradients differ between members and non-members of the training data. Since every client in the federated learning experiment has white-box access to the current average model, every client can test the model on data and inspect gradients. A passive membership inference attacker can, thus, observe gradients over multiple rounds of federated learning. An active attacker can do a gradient ascend step towards the chosen target point and inject that information in the model update they upload to the server. If the target point is indeed in the victim's training data set, the "wrong" information is corrected by the victim in the next training round, resulting in a steeply descending gradient that can be picked up even easier. Any client could launch such an active or passive membership inference attack.

Both passive and active attacks of clients change the training algorithm, either to store data, or to actively manipulate the output of the algorithm. To protect against these malicious code changes, trusted execution environments could be used. In practice, the usage is limited by the memory capacity of their hardware. Thus, realistically sized neural networks do not fit into the memory and cannot be protected. To overcome this drawback, Mo *et al.* [324] want to move the training of the most sensitive layers into the trusted execution environment, while keeping the less sensitive layers in the standard environment. This requires measuring which layers are most privacy-sensitive. The authors develop metrics to measure the information content of gradients across different layers of the neural network. These metrics could potentially be used in combination with other protection mechanisms beyond trusted execution environments.

A malicious server is considered by Wang *et al.* [325] to reconstruct all clients' prototypical samples. The server starts with a guess and then optimizes the guess to match the shared model update. The synthesized guess is further optimized with a GAN to yield more realistic outputs. They describe a passive attack as well as an active attack. In the latter, the server isolates the targeted client by not forwarding other clients' gradients, which results in less noise during reconstruction and thus a more powerful attack.

9.1.3 Privacy Mechanisms and Defenses

Since the attack vectors on federated learning are similar to centralized learning, it is not surprising that similar methods and tools are used as a defense. However, the tools need to be adjusted to the federated setting.

Federated learning alone is not sufficient to achieve privacy protection, as Zheng *et al.* [326] show. They focus on membership inference attacks and compare federated learning without further privacy preservation to direct data sharing with local differential privacy. The authors conclude from their experiments that membership inference attacks succeed when applied to federated learning, which can be prevented by sharing differentially private data. However, data sharing leads to decreased performance. Therefore, we focus in the following on how to add further privacy preserving mechanisms to federated learning to reach both high utility and privacy protection.

The first approach to bring differential privacy to federated learning was made by Shokri and Shmatikov [327]. Instead of sharing all updated weights of the locally trained model, a client shares mostly those that have changed during the latest local training round and are therefore most relevant to share with the server. Their algorithm is based on the sparse vector technique that, intuitively, allows cherry-picking only a few informative items with almost no cost of finding them. The sparse vector technique uses carefully crafted noise and a noisy comparison to preserve privacy in the process and result. The paper was later criticized for its loose privacy bounds, leading to high privacy parameter values ϵ for reasonable amounts of noise, resulting in low formal privacy guarantees. Re-using our formalization from Section 5.1.2, they define the difference between the "neighbouring" databases D, D' as one data point being changed in one of the clients. This is referred to as "central" DP in the literature.

The work by Truex *et al.* [328] can be seen as an extension of that idea. They allow clients to differ in their privacy parameters for perturbation of the neural network weights before they are sent to the central server that computes the federated average. A random selection of a subset of the clients helps to scale down the noise required in each round. Additional tricks on distributing the privacy budget over the neural network layers and parameters are used in their experiments to further reduce noise and improve utility. Note, however, that Truex *et al.* do not base their formal privacy guarantee on the sparse vector technique, but on condensed local differential privacy.

The moments accountant method developed by Abadi *et al.* [263] for centralized learning is known to have tight privacy bounds with differential privacy. McMahan *et al.* [310] show that this can be directly applied to federated learning in some settings. The server adds the necessary noise during SGD on the global model. This yields user-level privacy, thus, the resulting model is almost the same even if one of the users is replaced. In their use case of learning a next word predictor for smartphone keyboards, this setup was reasonable due to many clients contributing their data. However, in a cross-silo setting with only a few clients, the clients' individual training samples are still vulnerable to membership inference attacks. Us-

ing the previous formalization (cf. Section 5.1.2), D, D' differ by the contribution of all data points of one client, which is called "local DP". This paper shows how challenging it is to migrate privacy tools from a centralized to a federated setting.

Choudhury *et al.* [329] use objective perturbation instead of DP-SGD to achieve differentially private models. Concretely, they perturb the classifiers' decision function directly. Their case study on healthcare data sets shows that too much noise needs to be added, destroying the utility of the model. Their work points out the importance of choosing a suitable DP mechanism to maintain utility while preserving privacy.

Goryczka *et al.* [330] show how differentially private noise can be added by multiple participants during an addition such that the noise sums up to the value that is required to protect the underlying data. Combined with encryption mechanisms, they achieve secure and privacy-preserving additions. Such additions can be used as a building block for aggregation on the server during federated learning.

9.1.4 Privacy-Preserving Synthetic Data Generation

Instead of training jointly on private data, a one-time effort can be made to train a model that can generate an unlimited amount of synthetic data. Such synthetic data should have similar characteristics to the private training data but be generated in a privacy-preserving way. There are several situations in which privacy-preserving synthetic data generation is a suitable option – this is, for example, the case when only one party is involved and federated learning is not applicable. Even if multiple parties are involved, they might not be comfortable to make their data available for learning at a later point or they face network and time restrictions to be online and participate in the learning process. Additionally, in some situations, the task is to inspect raw data instead of training a model for a specific purpose. In these situations, raw data should be given out, but handing out the data directly leaves it vulnerable. Here, synthetic data generation is a solution approach. Furthermore, synthetic data can be generated in larger quantities and is beneficial in situations where the downstream (learning) task is more data-hungry than the synthetic data generation process itself.

Synthetic data generation can be achieved using generative adversarial networks (GAN) or autoencoders. GANs are composed of two neural networks that are trained jointly, as shown in the left side of Figure 48. The generator synthesizes data, and the discriminator judges how realistic the synthesis is by learning to distinguish synthesized and real training data. A good prediction accuracy of the discriminator is fed back into the generator as a penalty, teaching the generator to synthesize more realistic outputs which the discriminator cannot differentiate from real data. Only the generator is released which is able to synthesize realistic data points given just randomly generated noise as input. Autoencoders learn a low-dimensional representation of the input, as shown in the right side of Figure 48. The first part of the autoencoder reduces the input's dimensionality (encoding), and the second part reconstructs the original input (decoding). Only the decoder is released and synthesizes data points. Synthesis

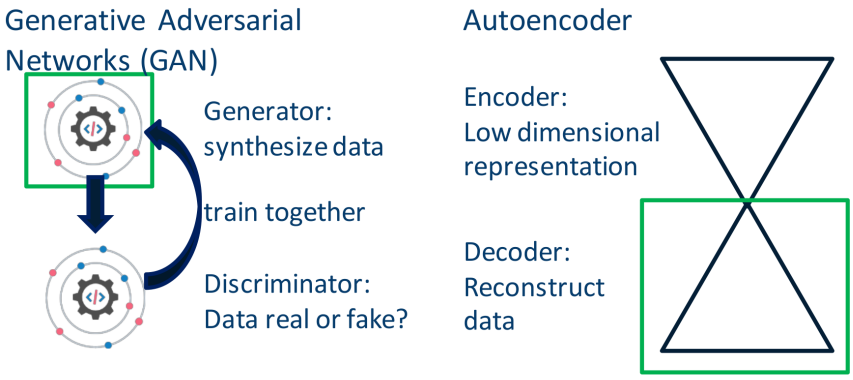


Figure 48: Synthetic data generation via GANs (left) or autoencoders (right). The part of the respective model in the green box is released to the user after training.

of data is either based on actual data points known by the receiver of the autoencoder or uses Gaussian noise if it is a variational autoencoder.

As a generative model is handed out, only membership inference attacks apply, which can be mitigated with differential privacy. We still need to protect the training data against membership inference attacks, that is, any inference whether a given data point is part of the training data set. Reconstruction attacks do not apply because it is the purpose of the generator to reconstruct plausible data points, which means that it needs to capture all dependencies between features and labels.

Hayes *et al.* [331] pointed out that generative models such as GANs can be vulnerable to membership inference attacks as well. They claim overfitting to be causal for membership leakage and notice that overfitting prevention is more challenging for generative models compared to deep neural network classifiers. Standard regularization techniques such as weight normalization and dropout showed some level of privacy protection in their experiments. For more information on regularization see standard ML literature such as Goodfellow *et al.* [260]. Differential privacy prevents the attacks only at the cost of utility, that is sample quality in this case. The differentially private GAN is generated by inserting a Gaussian noise layer into the discriminator, the only part of the system that accesses actual data.

Similar observations were made by Chen *et al.* [332] in their systematic study on the leakage of GANs of membership inference information under different assumptions of attacker knowledge. Their experiments support the argument of overfitting contributing to membership inference leakage and observe that larger training data sets lead to less overfitting and, therefore, to less leakage. Unsurprisingly, an attacker with more knowledge, for example, with control over the input of the generator or with white-box knowledge of the generator's weights increases membership inference vulnerability. The reason is that more knowledge allows for a better estimation of the probability distribution learned by the generator, since a

higher probability of reconstructing a given data point indicates that the data point was in the training data set. Chen *et al.* [332] add a calibration technique to the attack that takes into account that some data points are harder to reconstruct than others. Thus, the reconstruction error can be split up into two parts: one due to the relative hardness of data reconstruction and one due to membership of the data point. The authors also experimented with DP-SGD for privacy protection and observe that in order to maintain utility, that is, the quality of the generated data, low amounts of noise are required resulting in incomplete attack mitigation.

Chen *et al.* [333] train autoencoders and variational autoencoders. Autoencoders learn to generate new data points given a data point by first encoding the input and then decoding it into another, similar output. Variational autoencoders do not need actual data as input since a Gaussian noise vector suffices. The authors use the well-established technique of adding noise during gradient descend developed by Abadi *et al.* [263]. Their evaluation shows that membership inference attacks can be mitigated with differentially private training.

For a GAN setup with generator and discriminator, Triastcyn *et al.* [334] added a Gaussian noise layer to the discriminator network and used the moments accountant method [263] to determine privacy bounds. As the generator processes only differentially private data, it is itself differentially private by the post-processing theorem [335]. Note that the setup comes with mild limitations on the network structure before the noise layer, for example, no dropout or no batch normalization can be used.

Xie *et al.* [336] show another way to create differentially private GANs. They clip weights and thereby bound gradients and then add noise to achieve (ϵ, δ)-DP during the training process. Their evaluation on MNIST and on health data shows a larger difference of the generated data distribution and the training data distribution compared to previous work. However, they argue that this does not come at a cost of utility. The reconstruction of features given the remaining features was even easier on the generated data set, thus the relationships among data dimensions was preserved by the GAN.

Also, the differentially private student-teacher approach by Papernot *et al.* [264, 265] can be applied to GANs as Jordon *et al.* [337] showed. The one discriminator network is replaced by several teachers and one student model accordingly. A good balance must be found between too few teachers resulting in high noise and too many teachers resulting in poor teacher performance due to too little training data. As only the generator is released, which is trained on top of differentially private outputs, the post-processing theorem applies and states that the generator is differentially private without further adoptions.

Beaulieu-Jones *et al.* [338] also use the differentially private gradient descend by Abadi *et al.* [263], but to train the discriminator of a generative adversarial network (GAN). Recall that the GAN's discriminator part is the only part with access to the private training data. They use the generator to synthesize patient data for research on blood pressure.

Phan *et al.* [339] used an approach that differs from DP. They approximated the autoencoder's reconstruction function by a polynomial in order to use the Functional

Mechanism [340], which adds Laplace noise to the polynomial's coefficients. Their comparison with other DP mechanisms using the example of health data shows better utility for the same privacy levels. However, their paper lacks evaluation of privacy attacks.

9.1.5 Homomorphic Encryption

Another possible solution for providing confidential data to third parties is the use of encrypted computation. The main idea of encrypted computation is that users share their data after they have encrypted it in a way such that it cannot be decrypted (i.e., not viewed by random stakeholders on the internet). Subsequently, the processing, which includes the training and updating of ML models, takes place on this encrypted data and the corresponding results are extracted based on the shared user inputs. In the *Full Homomorphic Encryption* setting explained below, the decrypted result is the same as if the computation would have been performed directly on the plaintext.

Encrypted computation can be useful in the context of cloud computing, federated learning and transfer learning, and can be applied to various different scenarios. An example are intrusion detection systems (IDS) which need to look into the traffic passing through the IDS device and analyze it, without breaking potential encryption and compromising the privacy of the end users [341, 342]. Therefore, it is reasonable to apply different privacy preserving [343] and encrypted computation ML techniques in the scope of intrusion detection and prevention for end users and data center networks [344].

Homomorphic Encryption (HE) encompasses a set of encryption methods that allow to perform computations on encrypted data[25]. Specifically, in homomorphic encryption schemes, the plaintext message is encrypted and computation is performed on the ciphertext, so that the plaintext message remains secret. The result obtained after computation is decrypted. The first *Fully Homomorphic Encryption* (FHE) scheme was developed by Craig Gentry in 2009 [346] and can conduct arbitrary operations on the encrypted data (i.e., addition and multiplication), based on lattice-based crypto systems. FHE can also perform the computation of complex operations that may be composed of multiple additive and multiplicative layers. Some of the FHE schemes discussed in the literature are Brakerski-Gentry-Vaikuntanathan (BGV) [347], Brakerski-Fan-Vercauteren (BFV) [348], Gentry-Sahai-Waters (GSW) [349] and Cheon-Kim-Kim-Song (CKKS) [350].

[25] A further trend from the past years is given by the notion of secure enclaves [345]. The idea behind secure enclaves is to achieve an isolation of the computational process at the hardware level. This involves special security mechanisms in the CPU as well as storing data for the secure process in encrypted memory areas, such that other programs and applications cannot access the confidential information. In this regard, secure enclaves is to be seen as a complementary concept that can be used on various devices, e.g. in smartphones or on central servers to isolate and additionally secure some confidential processing tasks.

The Role of Homomorphic Encryption in Machine Learning The general architecture of utilizing HE in current distributed systems is depicted in Figure 49. The figure illustrates a client that sends its encrypted data to a server, where complex and computationally intensive operations are performed (e.g., the training or evaluation of an ML model). The encryption and decryption of the data is facilitated by a corresponding *public key* and *secret key*, while the training and evaluation of the ML model is based on the execution of required operations (e.g., additions and multiplications) on the encrypted data. For this process an *evaluation key* needs to be generated in the preparation step of the overall setting.

Figure 50 presents a distributed/decentralized view on the HE process and architecture. In this case, multiple clients aim at sharing their secret data with a central server, on which the training and evaluation of an ML model is executed based on the overall set of inputs from the involved clients. Also in this setting, a corresponding set of *public keys*, *secret keys* and *evaluation keys* are required per client, in order to ensure that the HE scheme can be operated across the distributed system. Indeed, complex cryptographic key setups[26] are required, which are generated by so called Distributed Key Generation Algorithms such as the one introduced in the study of Pedersen [351].

HE can also be used in the context of federated learning. This setting consists of multiple clients that use the HE ingredients from the distributed HE scenario (cf. Figure 50), in order to share specific parameters of the locally trained ML models. In parallel, the clients receive global aggregated parameters from the central server, which is responsible for consolidating, aggregating and handling the overall ML model. An example for parameters which are exchanged based on the HE scheme is given by the weight and gradient updates that are computed in the course of a gradient decent based training algorithm, for example, backpropagation in deep neural networks.

A further related concept is the idea of secure multi-party computation (SMPC). Within SMPC, multiple parties aim to compute a function together without revealing the specific inputs that they provide for computing the global output. This goal can be achieved by utilizing different protocols (e.g., Garbled Circuit Evaluation [352] or Linear Secret Sharing [353]), with FHE [354] playing an important role as one of the possible solutions. If FHE is used with a corresponding set of keys, each involved party encrypts its data and shares it with the others (e.g., over different centralized servers) or as part of a peer-to-peer protocol, such that this encrypted data can be used for calculating the overall target function in an SMPC setting.

Homomorphic Encryption Methods for Machine Learning: Models and Performance Aspects Training on homomorphically encrypted data sets has been the focus of many researchers, and accordingly, many techniques have been published based on the different types of HE schemes, HE libraries, ML algorithms, and data sets used in practice. Researchers made

[26] In this context, a set of keys is required that allows all clients to encrypt their data, while the server should correspondingly be able to perform operations (with an *evaluation key*) on data aggregated from the different involved clients.

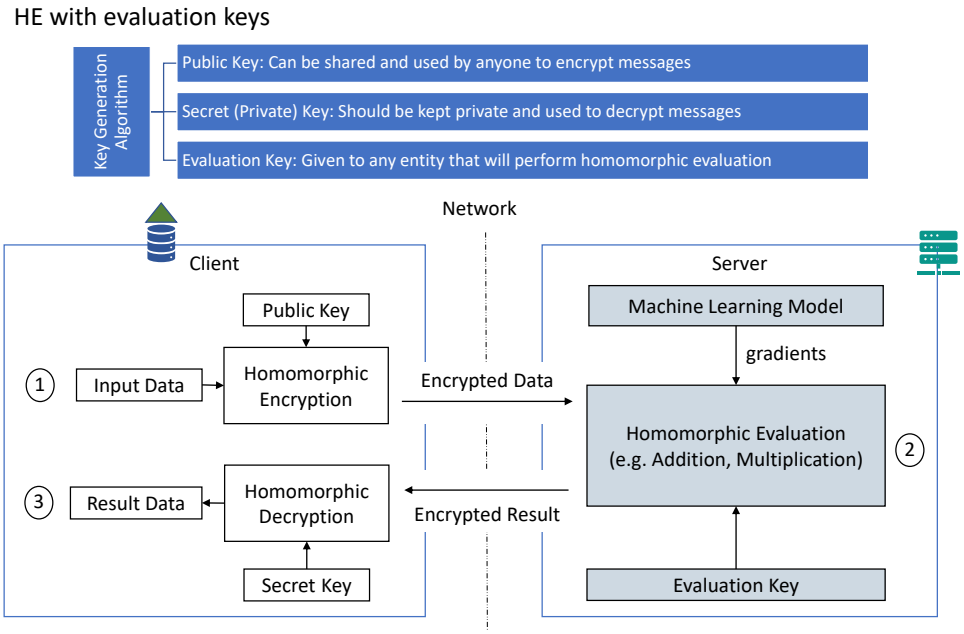


Figure 49: The architecture of the homomorphic encryption process including relevant cryptographic keys.

an initial attempt in [355] where they use a *Somewhat Homomorphic Encryption* scheme[27] along with linear means (LM) and a Fisher's linear discriminant (FLD) classifier for classification of the Wisconsin breast cancer data set from the UCI machine learning repository [28]. Cheon *et al.* [356] use the CKKS HE scheme based HEAAN library[29] in combination with ensemble gradient descent, a variant of gradient descent, as an optimization algorithm of the logistic regression for training on the MNIST [357] data set. Using similar tools, Park *et al.* [358] evaluated the training of a support vector machine on 8 different data sets, including the above mentioned Wisconsin breast cancer data set.

The authors of [359] utilize an improved Paillier crypto-scheme, in order to enable the sharing of gradients for the updates of ML models. In this regard, this research follows the typical structure of federated learning on top of a homomorphic encryption layer. Thereby, each client encrypts the local gradients and sends them to a server, which in turn performs

[27] A *Somewhat Homomorphic Encryption* fulfills the requirement of being able to correctly decrypt and evaluate/execute operations (e.g., addition and/or multiplication) on ciphertext. However, the compactness of the overall scheme is not guaranteed meaning that especially the evaluation of operations on the ciphertext can lead to results which grow arbitrary in size or influence the ciphertext output negatively in a way that a decryption is not possible afterwards.

[28] `http://archive.ics.uci.edu/ml`

[29] `https://github.com/snucrypto/HEAAN`

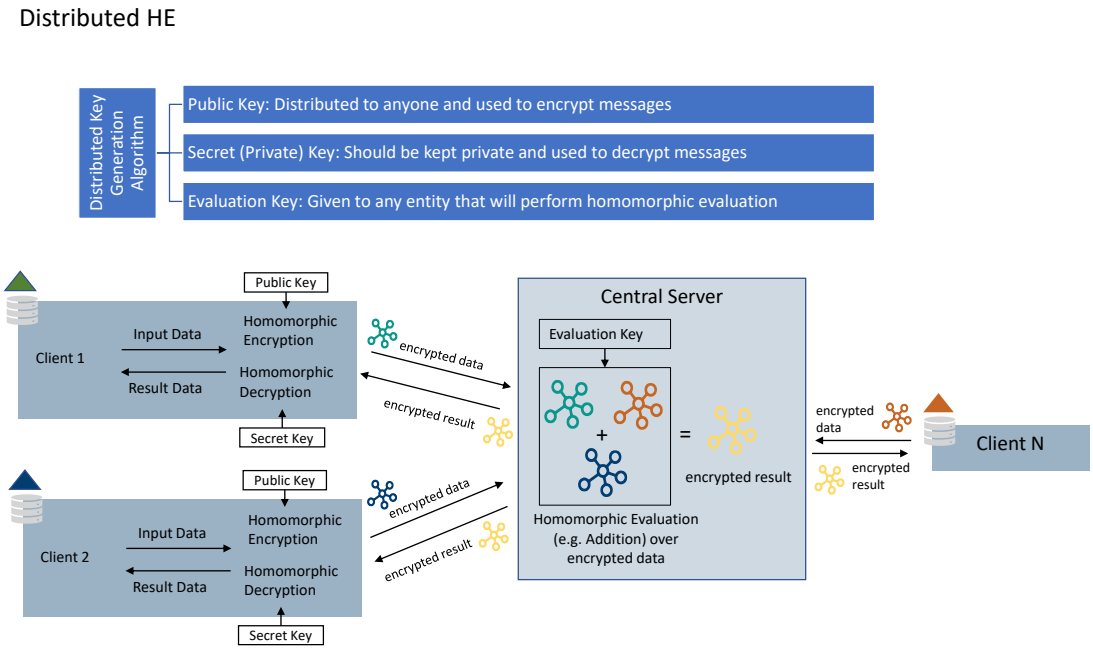


Figure 50: Illustration of the process of distributed homomorphic encryption.

the operations within the scope of the Paillier-based HE scheme and sends the updated global gradient back to each client for updating the local model instance. The evaluations and comparative results are performed on the MNIST data set and reveal that with increased complexity of the encryption scheme (basically increasing the key length) the processing and training overhead increases. This is not a surprising result and illustrates the impact of the underlying encryption scheme on the HE based processing and computation.

With Cheetah [360], Reagen *et al.* recently proposed a framework for optimizing and accelerating HE processes and operations. The utilized HE scheme is the BFV FHE [348] and, in summary, Cheetah proposes a number of optimizations in terms of software profiling, GPU acceleration, instruction execution and kernel optimizations, in order to achieve improved results for convolutional and other deep neural networks. The results show a substantial speed-up (up to 30 times) compared to other research works relating to HE and machine learning. The evaluations are conducted over different models and various data sets and can be considered as meaningful indicators for the potential to speed-up HE based machine learning.

In [361], an approach to implementing privacy-preserving machine learning with deep neural networks on top of the RNS-CKKS[30] FHE scheme is presented. The authors state that despite FHE being a known concept for a while, most of the used ML models in this context

[30] RNS-CKKS stands for Residue Number System-CKKS

have to be modified such that only simple operations are used, and, for instance, the full power of the available transfer and activation functions in the scope of neural networks cannot be utilized. As a result, the accuracy of these models is decreased compared to other similar models operating on unencrypted data. By using the RNS-CKKS scheme with bootstrapping[31], the authors managed to use the ReLU (Rectified Linear Unit) activation function in CNNs and achieve performance rates in terms of accuracy which are comparable (over 95% accuracy) to those of CNNs trained on the unencrypted data. Inference over homomorphically encrypted data sets can be realized through the implementation of Cryptonets, which constitute neural networks that can be applied to encrypted data [362]. This work demonstrated a 99% accuracy on the MNIST data set with almost 51,000 predictions per hour. However, in order to achieve acceptable amortized performance, the proposed framework requires huge batches of inference data, making it less viable for use cases that assess a single instance of data.

It is important to mention that the reported numbers of the implementations above cannot be easily compared with each other, as the experimental setups (HE parameters, data sets etc.) differ significantly between the studies. Nevertheless, the studies may provide the reader with an intuition regarding the current state of the research and the experiments that are conducted in the era of HE and ML.

Limitations of Homomorphic Encryption for Machine Learning We state limitations of state-of-the-art methods in the domain of machine learning and HE.

1. Evaluating deep neural networks on homomorphically encrypted data is expensive in terms of communication, memory as well as computation [363], making it a challenge to apply it in particular to deep learning which is characterized by its need for big data.

2. The computational overhead increases with the message size (i.e., the size of the ciphertext) and to the length of the keys that are being used for the cryptography.

3. Similarly, the communication overhead between client and server increases with the length of the ciphertext.

4. A cryptographic infrastructure for key generation and management is required in order to enable HE-based ML processes.

5. Each conversion from plaintext to ciphertext has some noise associated with it. This noise can continue to increase each time that the ciphertexts are added or multiplied. As a result, the final ciphertext might no longer be decryptable. Accordingly, the ML models should be relatively simple in terms of their architecture (i.e., neural networks should be rather shallow), in order to obtain reasonable results at the output.

[31] Bootstrapping in a homomorphic encryption context stands for a process that is used to reduce noise and potential deformations of the evaluation outputs for operations on the ciphertext.

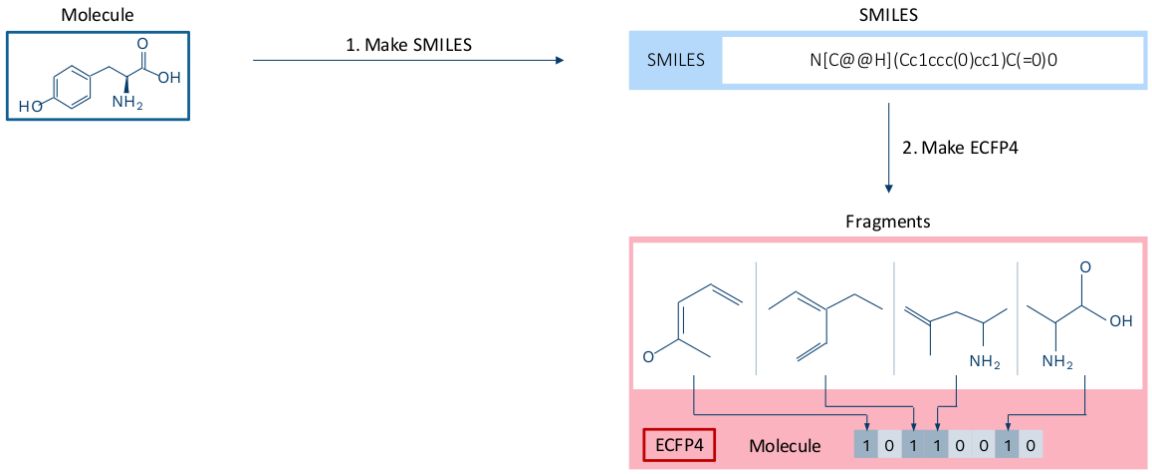


Figure 51: Visualization of an exemplary molecule and its ECFP fingerprint. For visualization purposes, we use a length of 4 to match substructures against, while in practice, larger sizes are used.

In particular the last mentioned shortcoming is a limitation for using modern ML algorithms via HE.

9.2 Deep Analysis of the Selected Application Scenario

9.2.1 Application Scenario: Molecule Structure Modeling

Material scientists and drug developers for medicine are involved in tasks where they have to design molecules. They need to know whether these newly designed molecules are toxic, soluble in water and which other properties they have in order to know whether the molecules are useful. Synthesizing and experimentally testing molecules is an expensive process only worth pursuing for promising drug candidates; Wouters *et al.* [288] estimate costs of $314 million to $2.8 billion. To avoid high costs, instead, Quantified Structure Activity Relationship (QSAR) modeling can be used to predict the properties of molecules based on their structure. If unwanted characteristics of drug-like molecules become evident by QSAR modeling, drug candidates can be excluded from the multi-stage drug discovery pipeline.

A data set that realistically models QSAR data generated during the drug discovery process is described by Sun *et al.* [364] and is publicly available. In general, data for QSAR models is high-dimensional, and there are various possibilities for representing the molecules' structure. Labels in this task are various chemical properties such as toxicity or water solubility as well as interactions with the therapeutic target. Therefore, QSAR models are often multi-task models that predict multiple labels at a time. Translated to the domain of image classifica-

tion, a single-task model could predict whether the image contains a smiling face or not. A multi-task model could predict whether the depicted face is smiling or not, whether the person wears glasses as well as the eye color.

Molecules can be represented as SMILES strings [287] that describe the atoms and bindings as a text string, as shown in the upper part of Figure 51. Note that we displayed a small molecule for better understanding. These strings describing molecules differ in length, which is challenging as input to neural networks. Therefore, a common preprocessing step is to generate fingerprints, so-called ECFPs [289], which are binary vectors of fixed length. The entries are determined by checking whether the molecule has a predefined substructure and setting the respective entry to one, comparable to the one-hot encoding of words occurring in a sentence, as shown in the lower part of Figure 51. The transformation of SMILES to ECFP is lossy since different molecule (sub-)structures can lead to the same ECFP representation. This makes ECFPs comparable to hashes, however, reverting the fingerprint is an active field of research, and the latest results show that reversion is neither straightforward nor completely impossible [365], while properly defined hash functions cannot be inverted. In contrast to the aforementioned credit scoring scenario, molecule data is hard to interpret even for domain experts. The data dimensionality is high, and there are complex dependencies between the features. The dimensionality is high due to the large molecules with many connections in the raw input data. The dependency between the features is due to overlapping submolecule structures for which the molecule is scanned in the ECFP generation process. For example, the NH_2 substructure of our molecule in Figure 51 has contributed to the last two fragments being found and two entries of the ECFP fingerprint being set to one.

Data generated during the drug discovery process is highly biased. Only data points related to the disease that have passed the previous pipeline steps, such as the modeling of their characteristics, are experimentally measured. Additionally, the theoretical chemical space is vast compared to the data that can be collected. In order to generalize to unseen chemical spaces and reduce bias, learning across different institutions is beneficial. However, the bias and the small size of each institutions's data set with respect to the data universe are a few of the reasons for privacy risks. If a competitor can reverse-engineer part of the experimental data, it is possible to infer further information about the drug discovery process that generated the data. That information could be abused to replicate and advance the process leading to a faster patenting of a drug than the victim. Therefore, the victim looses the benefit of investing into the experiments that are required to find suitable drug candidates as part of the costly drug discovery process.

9.2.2 Application of Attack Vectors and Privacy Mechanisms

In general, both membership inference and reconstruction attacks can be launched against federated learning in this scenario.

Successful membership inference attacks against one of the clients in chemical modeling

may lead to the loss of a drug candidate. As discussed above, the data is usually skewed due to the experimental process generating it and therefore, further information about the experimental efforts of a single institution could be gained, which may eventually lead to stealing a drug candidate. Since federated learning allows clients to profit from the data of others indirectly, clients with too little data to generate a meaningful model alone are motivated to join the federated training. Theoretically, their data is also at risk of being reverse-engineered, and a small data set size combined with the inevitable bias may increase the risk.

Whether a successful reconstruction attack against a client is a privacy risk depends on the setup. Recall that model inversion attacks aim at generating prototypical examples for each client and label. For chemical data, in the worst case for the victim, the chemical space and a blueprint molecule (scaffold) could be generated, but in the best case, such an average molecule for a given chemical property is meaningless. Similarly, an attribute inference attack that reconstructs part of the molecule may in the worst case leak the chemical space or the scaffold, but in the best case that reconstruction is meaningless. Whether one of these extreme worst or best cases is true or something in between can only be answered experimentally based on the data, the labels and the model. Note that the scenario does not have a private label-input relation as facial images and names of the depicted people used in many research papers.

As we discussed in the concrete attacks above, we find it unrealistic that federated SGD is used, that means, gradients are shared over single data batches. It is more realistic that federated averaging is used in which model updates are shared, generated from at least one pass over the whole training data set, which rules out several attacks [319, 321, 322]. As the attacks by Melis *et al.* [320] and Nasr *et al.* [271] work in the more realistic scenario, they are applicable and threatening the privacy. The attack by Wang *et al.* [325] which assume the presence of a malicious server is applicable to our scenario. Their attack reconstructs prototypical examples, but whether these are a threat in our concrete scenario is debatable (see two paragraphs above).

The defense by McMahan *et al.* which uses differential privacy [310] is limited to a cross-device setting and cannot be used in scenarios with much fewer clients. Differential privacy can be used as a defense mechanism following Shokri and Shmatikov [327], but due to the criticism of the loose privacy bound, attacks should be modeled and checked in parallel. The same holds for the extension proposed by Truex *et al.* [328].

The methods for privacy-preserving addition developed by Goryczka *et al.* [330] seem promising for our scenario. Whether the additional overhead by encryption is acceptable in practice and whether the introduced noise does not decrease utility are crucial questions that can only be answered with experimental evaluations. In the case where full encryption cannot be used due to performance issues, putting only the most vulnerable layers under costly protection mechanisms like trusted computation environments as suggested by Mo *et al.* [324] might be an option worth considering. But again, attacks should be modeled in parallel to ensure that the protection is strong enough. Additionally, one should remember that the differ-

ential privacy is by design only constructed to protect against membership inference attacks, not against reconstruction attacks.

Synthetic data generation is in general applicable to our scenario as well. Differentially private GANs can be trained as described by Chen *et al.* [332] or Triastcyn *et al.* [334], or autoencoders as described by Chen *et al.* [333]. It is also possible to use the differentially private student-teacher model approach for GANs by Jordon *et al.* [337]. Whether synthetic data can actually be successfully generated with either method can only be answered with extensive experiments and most certainly depends on the quality and quantity of data available for training. Especially the generalizability is expected to be challenging due to the huge chemical space. On the other hand, a large supply of high-quality synthetic data has the potential to advance the chemical field in which traditional data generation is costly and thus a major bottleneck.

As QSAR based data sets, models, or predictions are a result of aggregating decentralized data inputs, also homomorphic encryption is applicable in the context of QSAR. An incentive of the participating parties to use homomorphic encryption is to protect their data (or model) and keep it, partially or completely, confidential.

Hashimoto *et al.* [366] present a privacy-preserving protocol that is closely related to FHE performed in the scope of neural networks. The privacy-preserving protocol [366] includes: encrypting the data locally, sending the encrypted data to the server to calculate the output of the hidden neurons of the neural network, and sending the output to the local device decrypting the received output and calculating the output class label. Hence, this approach follows the generic architecture and process of HE as described in Figure 49 and Figure 50, and can be applied in a similar way to QSAR.

Although HE for QSAR is likely to be too slow for data sets with millions of features, recent research on encrypted computation techniques, as detailed in Section 9.1.5, is providing new opportunities to compute over confidential data sets through ML models in a much faster way than it was possible in the past. Hence, the research field continues to develop and various open source tools[32] are available in addition to first cloud offers allowing to gain experiences with programming and using HE for machine learning.

[32] e.g., FHE C++ Transpiler (`https://github.com/google/fully-homomorphic-encryption`), the (`https://github.com/tfhe/tfhe`) and others presented on the *Awesome Homomorphic Encryption* list (`https://github.com/jonaschn/awesome-he`)

10 Provision of Data Sets to External Users: Problems

We focus in this section on federated learning as one way to allow external users to make use of a confidential data set. Previously mentioned bias (Section 7.2) and explainability issues (Section 7.3) apply to federated learning as well, but are not covered here by specialized literature. Instead, we cover the topic of poisoning attacks in a federated learning setup and give an overview over open challenges in the field of federated learning.

10.1 Poisoning Attacks in a Federated Learning Setup

In this section, we first discuss works that answer the question whether poisoning attacks are effective in federated learning. As they are, we then discuss whether local differential privacy can also be used to mitigate such attacks.

Poisoning Attacks in Federated Learning Tolpegin *et al.* [367] study targeted data poisoning in which data points of only one or several targeted classes are misclassified. They describe a label flipping attack where attackers only flip labels of the training data points. Attackers manipulate several clients during the federated learning process. The authors' experiments show that only a few clients suffice for a significant drop in accuracy in the targeted classes, and adding more clients to the attack yield even higher accuracy loss. Federated learning contains several rounds in which the model is refined, and the later the attack is launched in this process, the larger is the impact on model accuracy. The authors also propose a defense mechanism against poisoning attacks. The server can do a principal component analysis (PCA) on the received gradients from all clients. Gradients that do not "fit in" to the PCA dimensions can be rejected as malicious.

Huang [368] goes one step further and describes dynamic backdoor attacks in which the poisoned data set is changed in each federated learning round to evade detection. This also achieves higher accuracy of the backdoor compared to related work. Federated meta-learning is even more vulnerable to this type of attack. The goal of federated meta-learning is to learn multiple tasks at once. Simultaneously, easy fine-tuning to additional tasks after training is achieved by federated meta-learning.

Model poisoning and data poisoning attacks are studied by Burkhalter *et al.* [369], who additionally assume an honest-but-curious server. To protect against the curious server, secure aggregation protocols are used in federated learning, that means, only encrypted model updates are exchanged with the server. Nevertheless, poisoning attacks are possible. The authors propose norm-bounding the client updates such that clients only have limited impact. Due to encryption, this is challenging to implement. The authors use non-interactive zero-knowledge proofs and hiding non-interactive commitments to define this norm bound. They claim that this approach scales to real world settings with federated learning.

In summary, poisoning attacks are applicable to federated learning and require a defense mechanism. Next, we revisit the idea to use differential privacy for protection, this time against poisoning attacks.

Local Differential Privacy as Defense against Poisoning Attacks We focus here on local differential privacy, that means, the noise is added by each participant of the federated setup.

Cheu *et al.* [370] study federated protocols in general and argue that they are highly vulnerable to adversarial manipulation. They argue that messages sent from clients to the server are required to be almost independent of the data to ensure the local differential privacy property. Thus, there is only a small signal of the data to be picked up by the server. To pick up this small signal, the server needs to be highly sensitive to small changes. This fact can be abused by an attacker who sends few messages to the server with large changes. These changes are larger than the sensitivity computed for the protocol, as the attacker is malicious and does not respect these sensitivity bounds. This reasoning is applicable to many problems, including gradient averaging which is relevant for federated learning. The effect of adversarial manipulation is compared against two types of baselines, including no manipulation, or manipulation only of the data which respects the sensitivity bounds. The authors study different protocols for local differential privacy and observe they are all vulnerable. However, some protocols are more vulnerable, while others are more stable to adversarial manipulation. Thus, a careful choice of the protocol can reduce the effect of adversarial manipulations.

The vulnerability results, as well as different degrees of vulnerability of different protocols, are confirmed by Cao *et al.* [371]. They focus on heavy hitter attacks in federated learning: the goal of the attacks is to be perceived as a heavy hitter having the most frequent data point among a group of users. Being perceived as such influences how the learned weights of the user are further processed in the federated learning protocol. Experiments on synthetic and real-world data sets support the findings. The authors also propose defense mechanisms against heavy-hitter attacks. One defense mechanism is to normalize the data frequency to a probability distribution (i.e., probabilities are non-negative and sum up to one). Additionally, malicious users can be detected by the server as fake data leaves observable patterns. In case only one data point is targeted at a time, the authors claim this is detectable for the server as well.

While these two papers point out increased vulnerability to poisoning attacks when using local differential privacy, other authors claim the opposite.

Sun *et al.* [372] find that even small amounts of Gaussian noise suffices to protect against backdoor attacks without compromising the utility. The noise is differentially private, but too small for significant privacy protection. During their experiments, the authors observe that the required norm clipping alone already has significant protection potential. However, adaptive attackers who know about this defense mechanism can craft better attacks. Instead of manipulating just one iteration of the federated learning protocol, multiple iterations can be used

with smaller perturbation in each step that is not well enough protected against with noise.

Finally, Naseri *et al.* [373] study local as well as central differential privacy for protection against membership inference attacks and backdoor attacks. While local differential privacy is added by the clients during stochastic gradient descent and achieves record-level differential privacy, central differential privacy is added by the server after aggregation and achieves client-level differential privacy. Both methods use Gaussian noise, clipping and the moments accountant by Abadi *et al.* [263] for estimation of the privacy guarantee. Naseri *et al.* [373] argue similar to Du *et al.* [301] discussed previously that differential privacy limits the influence of poisoned data points and thereby protects against backdoor attacks. In their evaluation, they first define a utility goal and then check whether a differential privacy setting achieving this goal does mitigate membership inference and backdoor attacks in image and text classification tasks.

In summary, we observe the effect of differential privacy on poisoning attacks in federated learning is disputed in the literature, some papers find differentially private models more vulnerable to attacks while others find them protecting against privacy and poisoning attacks simultaneously.

10.2 Open Challenges in Federated Learning

In this section, we summarize the arguments of two large overview papers on federated learning by Kairouz *et al.* [309] and Wang *et al.* [374] on additional challenges and open problems in federated learning.

Non-Malicious Failures Kairouz *et al.* [309] point out that there are several reasons why clients fail to complete the training round in FL tasks that are not maliciously motivated and why this is problematic. Already during measurement, noise might be introduced into the data, and if the data is further transmitted for processing, the transmission itself can add more noise. If noise is an issue, preprocessing or robust aggregation might be options.

Additionally, data preprocessing scripts may fail at some clients, which is hard to detect for the FL operator. Some failures leaves corrupted data that is invisible because no data leaves the client's device, other failures lead to error messages, but also that might be hard to debug without data access.

Other failures during the learning process might lead to clients not reporting back to the server, for example due to sudden loss of network access or insufficient time for processing. This unexpected dropping out of clients is especially challenging for secure aggregation protocols that are designed to share randomness that must cancel out upon summation at the server. Especially in cross-device learning settings, secure aggregation protocols that are robust to drop-offs should be used.

Finally, the central server for aggregation is a single point of failure and can be a bottleneck. If FL operators worry about this being the case, peer-to-peer distributed learning protocols are available, but decentralized stochastic gradient descent (SGD) comes with its own challenges such as convergence.

Some of the above arguments can also be found in the summary by Wang *et al.* [374], they add that computational constraints clients face, especially on mobile devices, might be another reason for clients dropping out of an FL round. Just skipping these clients might lead to too few clients completing the round, which not only increases noise during training but also harms convergence. Wang *et al.* [374] write that choosing the optimal number of clients per round is an open problem.

Computational constraints might not only be insufficient processing or memory resources, but also requirements like the mobile phone being idle, plugged in, or being on an unmetered network to avoid making the device unusable for the owner. Kairouz *et al.* [309] use the example of this requirement being enforced at night, which systematically excludes phones of night shift workers. This example shows that bias issues in the model might be the result of such constraints.

Non-i.i.d. Data and Convergence Machine learning in general usually assumes data samples being i.i.d., which means, independently sampled from the same underlying distribution. This may make sense for a centralized data collection, but there are many examples where this is not the case in federated learning. Kairouz *et al.* [309] summarize reasons for non-i.i.d. data such as a skew in the feature distribution, concept shifts or a skew in data quality. If this is the case, the FL operator should consider carefully whether fitting a global model makes sense in such situations. Also, many aggregation schemes treat all clients similarly, which again might not make sense in skewed client distribution. One main requirement in FL algorithms in general is that they should converge to a result. Convergence, however, is usually studied for i.i.d. data. Kairouz *et al.* [309] also summarize papers that study non-i.i.d. data with an assumed bound on dissimilarity, which also needs critical evaluation whether this is the case in the FL operator's situation. When direct comparison between i.i.d. convergence and non-i.i.d. convergence is possible in theory, convergence is slowed down. Wang *et al.* [374] point out that training objectives of clients with imbalanced data sets suffer, thus they may have different local minima. This explains the complexity in finding a global minimum. Solutions to this problem can be probabilistic approaches.

Finally, Wang *et al.* [374] point out that asymptotic convergence rates might be misleading, as the number of FL rounds is strictly limited in realistic settings due to time and resource constraints. Also the strategies of how clients are sampled are assumed to be uniform or unbiased in theory, but often biased in practice, as we discussed above in the non-malicious failures paragraph.

They also shed light on the consequences of choosing an aggregation schema. The

natural choice for a weighted aggregation would be to weight each clients' contribution by the amount of training data they possess to put more weight onto those that have more data. However, that might be harmful for privacy. Also robustness to outliers and fairness issues might be worsened with weighted aggregation and would therefore profit from uniform weighting. Wang *et al.* [374] write that more research is needed in this area to provide sufficient information to practitioners how to aggregate clients' weights best.

Communication Overhead Kairouz *et al.* [309] point out that the communication overhead in federated learning is often the major bottleneck. To address this, researchers have described numerous compression methods to reduce communication. However, some privacy technologies such as secure aggregation or DP are not designed to work with compressed gradients. On the other hand, Wang *et al.* [374] write that privacy might require efficient communication. Additionally, convergence might fail especially when using biased compression methods that yield high compression rates. Further to this, aggregation protocols might not be compatible with all compression algorithm choices.

In summary, choosing the right compression method is key to make an FL algorithm scale, but the algorithm must be chosen carefully to be compatible with aggregation and privacy requirements.

11 Provision of Data Sets to External Users: Recommendations

The process of designing and implementing an FL system or a generative model is similar to the previously described process for centralized learning (see Section 8). Therefore, we follow the same structure, and focus on additional considerations for these applications.

Disclaimer

While the provided recommendations can help to minimize the risk of privacy loss via ML applications, it is important to be aware of the limitations. The strategy described below has been derived on a purely conceptual level, taking into account literature reviews and practical experiments conducted in the course of this study. However, the strategy has not been applied to a real-world use case. Practitioners should check carefully whether the same strategy applies in their individual use case and if additional steps need to be taken. Moreover, it needs to be stressed that the security of AI systems is an active and fast developing field of research and that new developments need to be considered continuously.

11.1 Understanding Attack Vectors

When designing FL systems or generative models for data-sharing, the first step in deciding upon a privacy solution is the identification of attack vectors. A more comprehensive outline of potential attack vectors is presented in Section 9.1.2. Conversely to the centralized learning case (Section 8), however, a number of additional factors must be taken into specific consideration when designing secure systems for federated learning and generative models, that we discuss in the following.

11.1.1 Attacks on Federated Learning Systems

In addition to membership inference, data reconstruction and model stealing attacks against the final learned model, an FL system can also be attacked at training time by malicious client(s). In centralized systems with a server for aggregation, a malicious server could launch additional training-time attacks. It suffices to cover membership inference and data reconstruction attacks as potential attack vectors at training time. Model stealing attacks do not need to be considered, as the intermediate is shared with all participating parties during the learning process. Additionally, backdoor attacks and poisoning attacks might be relevant for some use cases. As explained in Section 9.1.2, these training time attacks can be active or passive. While passive attacks only observe the information available during the FL learning process without manipulation, active attacks manipulate the information shared and thereby have larger impact, but also larger risk of detection.

With respect to training-time attacks, FL systems present a unique spread of potential vulnerabilities which require specific consideration, given that multiple parties of potentially unknown trustworthiness can contribute updates to the end model. For FL systems, when considering attack vectors concern must be given especially towards:

- The number of participating clients.
- The trustworthiness of each client.
- The communication and aggregation protocols.
- Whether the system is centralized or decentralized.

For example, it might be that the use case involves hundreds of thousands of non-colluding edge devices with an anticipated minuscule fraction of malicious participants, so poisoning attacks might not be enough of a risk to be worth mitigating. Another example is the architecture decision of the aggregation method, which has a high impact on privacy: Based on the literature review in earlier sections, it seems that Federated SGD is less privacy-preserving compared to federated averaging, therefore the latter should be preferred if possible. More details can be found in Section 9.1.2 and Section 9.1.3. As attacks can target the local learning or the aggregation process, both centralized and decentralized systems might need protection against malicious or honest-but-curious aggregators in addition to protection against malicious learners.

It is crucial to accurately model the attacker(s) with respect to the points given to prepare a solid foundation for the choice of defense mechanisms. If the attacker is underestimated, defenses might not suffice. And if the attacker is overestimated, the deployed defenses might adversely impact utility or system complexity without any corresponding payoff with respect to privacy.

11.1.2 Attacks on Generative Models

Generative models, too, can present their own spectrum of challenges to training data privacy, especially membership inference attacks. These are dependent on the type of the generative model (GAN or Autoencoder), whether the generative model itself is shared, the generated data is shared, or the model can be queried via a public API. Regardless of the mechanism by which the model is shared, a generative solution will provide examples from within its training domain, so attention must be given towards:

- The anonymization/featurization of the input data
- The specificity of the domain-space of the training data
- Resilience against adversarial attacks

- The architecture itself
- The release mechanism

One way to support generative models in their task is to anonymize, aggregate, or abstract the data as much as possible already before training the model. Similarly to the defense against attacks on models in the case of centralized learning (see Section 8), if the input data comes from a computationally intractable domain or a transformed vector cannot be reversed to the original data, this strengthens model security.

The anonymization of data itself is a non-trivial task. There has been much work upon the topic (e.g., Carlini *et al.* [277]) when it comes to respecting i.i.d. in a general data ecosystem. When considering specific cases, caution is required to ensure that a balance is struck between removing enough information to preserve anonymity and thus privacy, and not removing too much information to preserve the learnable signal. Similarly, naive approaches to data generation can be detrimental to resultant data sets, instead of simply non-contributing.

To illustrate this with an example, let us assume a generative model trained on an unbiased population that creates a data set of predisposition to a certain disease with respect to previous health features and residential location. For the sake of the example we assume that there is a natural correlation between the residential location and the socio-economic status, concretely, that some parts of a city are inhabited predominantly by poorer/richer parts of the population. Additionally, we assume that the socio-economic status impacts general health. Anonymizing the location by using coarse-grained information or removing it entirely, thus, might remove important information from the learnable signal. Similarly, generating fake locations might even introduce unanticipated bias in the data. Therefore, a better selection of features might be to directly include the correlated feature (socio-economic status in this example) whenever possible.

A highly-specialized model trained on a very narrow domain of data may be vulnerable to a spectrum of attacks that reveal this domain. Generally in this situation, the domain itself is disclosed alongside the model, since it is commonly understood that the model will not produce a meaningful signal outside of this domain. However, consideration should still be made regarding the data domain - especially when training sets may be aggregates of multiple distant, highly-specialized domains. The HIV data set used in the practical investigations, see Section 6.2.1, is an example of such an aggregate of distant, specialized domains, where we selected small sets of similar data points from the large space of chemical molecule representations.

Especially in situations where the model itself is released, or an unmetered public API is provided, the model should be evaluated against a simulated adversary. Note that any defense mechanism against membership inference attacks may also increase the generalizability of the model itself. In particular in highly-specific data domains or sub-optimal training approaches, generative models can easily learn idiosyncrasies of the training data which can result in information leakage or potential privacy exploits.

The release mechanism also influences the attack surface of a model. A model which is wholly released to the public, for example, is much more vulnerable to many attacks than a model which only releases a handful of generated data sets, or which is available through a strictly rate-limited or output-limited API. The latter presents implementation concerns, but can be considered more privacy-secure than a released model. The specific considerations to be made in such an instance pertain to enforcing the limitations of the API and preventing a bypass of such restrictions.

11.2 Understanding Additional Constraints

Similarly to provisioning ML models to external users, other considerations must be made beyond potential attack vectors. For federated learning and generative model systems, these may include:

- Conforming to legal requirements.
- Conforming to parties' contractual requirements.
- Conforming to standards for communication protocols, storage solutions, authentication methods, etc.
- Ensuring that models are explainable.
- Ensuring that models meet certain fairness metrics.

Legal requirements may require that input data undergoes stringent anonymization, which might reduce the scope of potential attacks and simplify the privacy scope of the project, for example. Moreover, contractual and organizational measures might be taken to mitigate security risks.

All parties participating in federated learning need to use compatible protocols, which might reduce the available options due to hardware or software limitations. For example, when learning on diverse, resource-scarce devices such as smartphones, protocols need to be used that are compatible with sufficiently many devices and are lightweight. Otherwise, some devices and thus user types would be systematically excluded, which might lead to undesired bias in the data, more details on that can be found in Section 10.2.

The following general challenges influence attacks and defense mechanisms:

- Handling of non-i.i.d. data
- Communication efficiency, especially the use of compression methods and their impact on convergence and compatibility with defensive methods (details in Section 10.2)
- Selection of clients for each training round, especially with regards to efficiency, data quality, and data confidentiality

Whenever non-i.i.d data requires more aggregation rounds and less local training rounds, the probability that the malicious client is chosen and can launch the training time attack is increased. However, smaller updates are expected, so large adversarial manipulations (e.g., large changes of the weights) have an increased risk of detection.

In cross-device federated learning where only a fraction of global clients participate in each round, a selection based on reliability, availability and data quality might be preferred, using these historical data as a trust metric to inform the selection decision. On the other hand, data privacy would benefit from a random selection, for example to scale down the differentially private noise, and to reduce the risk of adversarial manipulations in general.

11.3 Risk Analysis and Attack Plausibility

With the knowledge of the system architecture, the attack surface, and the other sundry constraints, a risk assessment of each relevant attack vector should be performed. Similarly to Section 8, the assessment should take into account:

- The applicability of the attack
- The current feasibility of the attack
- The anticipated future feasibility of the attack
- The anticipated lifetime of the system
- The anticipated growth of the system (complexity, scalability, trustworthiness of future users)
- The value of the system and the motivation of a potential attacker
- The access-level of potential attackers

While the list of considerations bears strong similarity to those of Section 8, the details differ. The lifetime of a FL system, if extensive, can open the model up to more subtle or diluted attacks over a longer time that are harder to detect due to their subtle changes. Or in the case of generative models available through an API, a long lifetime of the deployment can bypass less sophisticated rate-limiting defenses.

Similarly, the access level of potential attackers as well as the anticipated growth of the system is worthy of much more consideration in a FL domain than in the case of centralized learning. While an initial private deployment among trusted partners could suffice for a pilot project, if that project gains traction and value then the sensible next step may be to open the system to additional users or parties. In that case the resilience of the model regarding user trustworthiness or access scope may need to be reconsidered, as for example an attacker initially modeled as honest-but-curious might now better be described as malicious and launching active instead of passive attacks.

For generative models, the value of the system may change over time, or it might be that the system contains a mix of value: The average data value might be low, but a specific data points might hold high value to an adversary. Consider, for example, the pharmaceutical sector: Physical-chemical data about a general data set of compounds might be of relatively low value, but knowledge that a participant is focusing on a specific family of experimental compounds could result in companies being beaten to patent on a new drug, resulting in potentially billions of USD in lost profits. Conversely to this, the value of data in the model (assuming it is not frequently retrained) will decay over time, so while attacks may become more feasible over time the value of the system as a target may decrease.

11.4 Mitigating Attack Vectors

Similarly to Section 8, the system architecture and model architecture choices influence the attack vectors, which are the basis for choosing defense mechanisms to mitigate these attacks. We focus here on the process to select which solutions are suitable for the given system, an overview of available potential solutions can be found in Section 9.1.3.

11.4.1 Selection of Defense Methods

Once a suite of solutions has been identified, consideration must be made when selecting the specific implementations. Again, the following points serve as guideline and starting point, while concrete use-cases might prioritize the items differently:

- Simpler methods should be preferred over equivalent solutions that come with a higher complexity.
- The earlier in the deployment chain in which a method can be implemented, the more preferential it is to an equivalent later approach.
- Methods that preserve or increase utility should be preferred over equivalent methods that incur a loss in utility.
- Methods with a formal proof of security should generally be preferred over equivalent methods that demonstrate empirical security.

Defenses against test time attacks were designed for centralized learning and need adaption to FL protocols. Before such a potentially costly adaptation is made, it should be checked whether a similar defense against the same training time attack vector offers sufficient protection. For example, differentially private noise that is designed to protect against malicious clients at training time might offer sufficient protection against test time membership inference attackers. Another example is the hardening of generative models that should first focus on ensuring the model generalizes well, as that directly reduces membership inference

risks [331, 332]. This is both is simpler and less complex compared to Differential Privacy solutions against membership inference attacks [33].

11.4.2 Evaluation and Implementation of Selections

Once the solutions for defense against the perceived attack scope have been decided, they should be considered together. All pertinent attacks should be addressed, and overhead should not be prohibitive. Again, similarly to Section 8, maintenance cost and code-base complexity should remain practical for deployment, and the suite of solutions should be evaluated from a holistic perspective. For federated learning, additional considerations should be made for convergence and communication efficiency, for details, we refer to Section 10.

But the other previously discussed considerations also apply to FL systems and generative models:

- Utility/privacy trade-off
- Computational overhead
- Applicability to attack vector(s)
- Code-base maintainability
- Implementation complexity (how long would it take to train a new engineer to the solution)

For generative models, the concerns primarily apply to the utility/privacy trade-off, especially with respect to resilience against membership inference attacks. However, the availability of the model or API to participants may be a strong secondary consideration, both with regards to usability as well as privacy.

For federated learning, the usage of well-maintained and independently audited privacy libraries could increase trust of clients in the system and thereby encourage them to participate, which allows the system to benefit from their data that would otherwise not be shared due to privacy risks. Moreover, FL systems are often considered to be harder to debug compared to centralized learning [309], thus good maintainability and low implementation complexity are preferred to decrease the risk of errors.

11.5 Deployment and Maintenance

Documentation for the respective user groups as well as regular checks for new attacks, defenses and shifted assumptions about the systems as outlined in Section 8 apply to federated

[33] Note that Differential Privacy might still be needed in some cases, for example, if generalizability alone provides insufficient protection.

learning and generative models as well. Note that the audience of documentation is larger for federated learning, as potential new clients may need a different type of explanation than model users or code auditors. This mainly depends on whether clients are envisioned to be device users (e.g., smartphone users for federated learning on smartphones) or professional data providers (e.g., chemical companies with experimental chemical data).

Summary and Conclusion

As machine learning applications continue to proliferate, there exists a growing demand for computational resources, pretrained models and large training data sets. The availability of such resources, for example in open source repositories, makes machine learning development increasingly accessible and efficient. At the same time, research in the field of adversarial machine learning in recent years reveals major vulnerabilities of machine learning systems to attacks. Considering security concerns, thus, is important when developing machine learning systems, especially when using external resources in the development process. At the same time, developers who release their models or data sets to untrusted third parties should consider the realistic threat of sensitive information leakage, among other risks. This study discussed security threats that could arise from two use cases: (i) outsourcing part of the model training (Part I) and (ii) releasing models (Part II) or data sets (Part III) to untrusted third parties.

Security vulnerabilities that exist in use case (i), transfer learning, arise from the difficulty to ascertain the security of externally-provided teacher models used to train a student model. As not only the performance but also security vulnerabilities transfer from a teacher model to a student model, using teacher models from untrusted sources opens the door to poisoning, backdoor and transferred adversarial attacks on the student model. Our practical investigations on a case study on medical imaging illustrated that the common practice of freezing layers of the teacher model when training the student model facilitates the transfer of backdoors from teacher to student. Training methods and defenses that adjust the weights or architecture of the student could reduce the likeliness of vulnerabilities transferring from the teacher model. Based on the literature review and the practical investigations, we offered recommendations for mitigating and defending against backdoor attacks. Each option comes with benefits and drawbacks. We advise a developer who wishes to use external sources to thoroughly assess the associated risks and benefits before proceeding.

In the second and third use case, security risks arise from the ability of untrusted external parties to access trained machine learning models (specifically black-box access) or to learn from sensitive data. Sharing information comes with the risk that untrusted parties could steal information about the model's inner workings or confidential data, leading to data leakage and facilitating privacy attacks. Using Differential Privacy methods might reduce some of the risks, but it is crucial that the parameters are tailored to the specific use case (see Section 6). Other risks require additional methods that we discuss in Section 5.1.2. Federated learning, synthetic data generation, or methods allowing for training on encrypted data could allow for untrusted parties to learn from sensitive data with a reduced risk of data privacy violations.

None of the discussed mitigation strategies was found to ensure full protection against privacy attacks or malicious manipulations of models and data sets throughout the machine learning lifecycle. Therefore there is a need for further research in the area of AI security. For a user developing an ML application, it is recommended to thoroughly assess potential security

risks and put in place defense mechanisms as necessary. The selection of appropriate defense mechanisms is challenging. The following general considerations may be taken into account:

- Generally, it is better to mitigate security risks as early in the ML lifecycle as possible. For instance, it can be considered safer to ensure at an early stage that a teacher model is free of backdoors than trying to filter out malicious input to the deployed system at operation time.
- It is important to be aware of the specific weaknesses and limitations of selected defense mechanisms. If possible, the limitations should be tackled, for instance, by complementing a defense mechanism with another defense strategy, while carefully considering potential interaction effects between mitigation strategies.

A major limitation for many defense methods is that they provide no guarantees of security. The reason is that the vast majority of defense methods are based on heuristics that might be bypassed, for instance, by adaptive attacks. Certification and verification methods are actively investigated, but often not applicable to widely-used deep learning architectures.

As ongoing research in adversarial machine learning might reveal new attack vectors in the near future or break existing defense mechanisms, AI model providers and developers need to continuously consider new research results. For practitioners this is a challenging and time-consuming task, and there is a need for easily accessible and regularly updated publications that provide an overview over relevant attack vectors and mitigation options. A general problem for creating such comprehensive surveys is that each research study usually applies the implemented attacks or defenses to specific use cases. Differences in the made assumptions and in the experimental setups such as the used model architecture make it challenging to compare the effectiveness of methods between different studies. As a result, it is often unclear whether the results of one study conducted in a specific setting transfer to more general settings. Currently, to reliably estimate system security, it is necessary to evaluate available methods and implementations individually for a specific setting.

Moreover, there is the possibility that different mitigation methods applied together might influence each other making it difficult to predict the whole system security. Further research is required for thoroughly investigating such interaction effects. One approach to enable secure machine learning in the future, could be systematic and tool-aided benchmarking that makes it possible to compare attacks and defenses from the literature and to study interaction effects between several defense mechanism applied together.

The results derived in this study are an important contribution for supporting ML practitioners to develop secure ML models by giving a thorough overview of relevant attack vectors and defenses as well as providing guidance on how to proceed.

Glossary

active attack	A term used in the context of federated learning privacy attacks to describe a malicious client or server that returns wrong outputs of their algorithm, in contrast to passive attacks (see below).
adversarial attack	An attack algorithm which slightly modifies input data to a model at run time in order to cause misclassification (in the literature also sometimes referenced as evasion attacks).
adversarial example	A specific input that was optimized in order to cause a misclassification when fed to a trained machine learning model.
attribute inference attack	A privacy attack that abuses a machine learning model in order to reconstruct (a sensitive) part of a data point's feature values given the remaining feature values as well as the label and access to the machine learning model.
API access	Same as black-box access (see below).
autoencoder	A generative machine learning model that jointly trains an encoder and a decoder. The encoder maps the input data to a low-dimensional vector representation, and the decoder restores the input from the low-dimensional vector representation.
backdoor	A built-in weakness of a neural network model that causes the network to misclassify input data in the presence of a certain trigger.
backdoor attack	An attack algorithm which implants a backdoor into a neural network model with the aim to cause misclassification for a certain trigger. A backdoor can be implanted, e.g., by poisoning the training data set.
benign example	A data point, e.g. from the training data set, in its original form that has not been tampered with.
black-box access	An access mode where the machine learning model can only be used for inference, but its inner workings (e.g., weight terms) cannot be inspected.

capacity The ability of a model to fit to the training data given a learning algorithm. The higher the capacity, the more complex the model's task can be. Capacity is sometimes used synonymously with model complexity and usually given by the model architecture.

centralized learning Standard learning on a centralized server or PC, used in this study as a term to point out the contrast to federated learning.

client If used in the context of federated learning, client refers to an entity with a local dataset that takes part in the federated learning process.

CNN Abbreviation for convolutional neural network, a network type using convolution filters of the input which is typically used for processing image input data.

convergence We say a machine learning model converges if the loss function reaches a global minimum. Intuitively, if the model does converge, it reaches it's learning goal.

cross-device A federated learning setup characterized by many clients with small data sets each, for example, smartphones training a next word predictor.

cross-silo A federated learning setup characterized with few clients with large data sets each, for example, three hospitals training on patient data.

decision boundary The boundary between (two) regions in the feature space that lead to different labels predicted by the ML model. Each model trained on the same data set has a slightly different decision boundary.

DL Abbreviation for deep learning, a subset of machine learning algorithms using deep neural networks which contain a large number of network layers.

DP Abbreviation of differential privacy, a system that implements data privacy by ensuring that individual data points do not significantly affect the output, thus, hindering membership inference attacks.

DP-SGD Abbreviation of a differentially private implementation of stochastic gradient descend.

fairness	There are multiple formal fairness definitions trying to capture what "fair" means with respect to the machine learning output, for example, that the error of the ML model is similar for both dominant and minority groups within the data.
feature extraction transfer learning	A type of transfer learning where all existing layers are frozen and only the classification layer is replaced for student training such that the teacher model serves as a feature extractor.
feature space	The multidimensional space defined by all possible feature values for a given task. The feature space depends on the featurization method as well as the domain for each feature.
federated averaging	One way to aggregate knowledge from the clients in federated learning. The trained model weights (usually weights of a neural network) are averaged. Possibly, weighted averaging is performed to increase the influence on the average of clients with more data or data of better quality.
federated learning (FL)	A machine learning technique where training is performed in a decentralized way via multiple participating clients who contribute to the model training with their own set of training data without revealing the data itself.
federated SGD	One way to aggregate the knowledge from the clients in federated learning. The clients share their locally computed gradients with each other.
fully homomorphic encryption (FHE)	An encryption scheme where the decrypted result of arbitrary computations performed on the ciphertext is the same as if the computation would have been performed directly on the plaintext.
GAN	Abbreviation for generative adversarial network, a generative neural network structure composed of a discriminator and a generator that are fitted simultaneously. The generator learns to output realistic data samples while the discriminator tries to distinguish real data from data of the generator, forcing the generator to output more realistic data.
generalizability	The ability of a machine learning model to generalize to unseen data and provide high-quality predictions especially in areas underrepresented in the training data.

homomorphic encryption (HE)	An encryption method that allows to perform computations on the encrypted data.
horizontally distributed data	Data sets (at least two) that share the same features, but have different entities, for example, MNIST images where one party owns the first half of the data set and the second party the other half.
i.i.d.	Data that is sampled independently from the same distribution (identicaly distributed). It is a common assumption that ML data is i.i.d., however, in practice, i.i.d. is hard to reach.
IP	Abbreviation for Intellectual Property.
LSTM	Abbreviation for long short-term memory neural networks, a recurrent neural network structure used for time series prediction, mostly in the context of language and speech processing.
membership inference attack	An attack launched on an ML model that allows the attacker to estimate whether a certain data instance was part of the model training dataset.
ML	Abbreviation for machine learning.
MLaaS (Machine Learning as a Service)	Machine-Learning-as-a-Service is an umbrella term for cloud offerings which support customers in training ML models for their own use cases. The services might include computational resources, pre-trained ML models and tools for training and testing ML models as well as training data management and preprocessing. Motivations to use such services can be - among other things - missing local computational resources or the providers support in model selection or parameter tuning. The charges for download or usage of trained models are often proportional to the computational effort for training. In transfer learning, often a teacher model is purchased via MLaaS platforms.
MLOps	A set of practices for efficient and reliable development of machine learning applications.
model architecture	Usually used for neural networks, where the number of layers and the number of nodes in each layer are referred to as model architecture. Also the choice of activation functions is contained in the model architecture.

model extraction attack	An attack on the model's inner working, same as model stealing attack.
model poisoning attack	An attack that performs malicious modifications on the model, for example, to cause targeted misclassifications at test time. In the context of federated learning, data poisoning and model poisoning attacks are synonyms as both can be achieved by manipulating the data.
model stealing attack	An attack on the model's inner working with the aim to steal the model's decision function and use it without the model owner's oversight.
NLP	Abbreviation for natural language processing. Neural language processing contains all tasks involving language as input, such as machine translation, next word prediction etc.
noise	A general term referring to all "small" perturbations generated from a noise distribution, for example, a Gaussian or Laplace distribution. Notice that not all noise is differentially private. For noise to be differentially private noise, it needs to be generated from a distribution parameterized according to the differential privacy definitions applied to the concrete task.
outlier	A data point that is an "exception" to the dataset. Datasets are usually viewed as sampled from a distribution, and outliers are defined as out-of-distribution samples, or samples that are occurring rarely in the distribution. Outliers occur naturally in datasets and need to be handled accordingly, e.g., by filtering them out, by increasing support in that region of the input space, or by proper regularization of the classifier to not overfit to outliers.
overfitting	Property of a machine learning model that it fits the training data too closely, lacking generalizability and potentially leaking information about the training data. Overfitting can typically be diagnosed by a small training error but a large testing error, i.e., it causes a lack of generalization ability.
passive attack	A term used in the context of federated learning privacy attacks to describe a malicious client or server that follows the algorithm correctly and only observes all information available for the privacy attack. This is similar to the term "honest-but-curious" in cryptography.

peer-to-peer When used in context of federated learning, peer-to-peer characterizes an architecture where aggregation is done without a central server.

performance A measure of how well the model's predictions on a given dataset match the ground-truth labels. Different learning objectives, numbers of classes and datasets require different performance metrics. For example, the *accuracy* measures how often the model's prediction is correct, which is suitable for balanced problems with an approximately equal number of data points in each class. For imbalanced problems, the *area under receiver operating curve (AUC)* is easier to interpret, as it takes false positives, true positives, false negatives and false positives into account. For regression problems where the output is not a label but a number, neither of the above is applicable, and for example metrics like *means squared error (MSE)* are used.

poisoning attack An attack which is based on "poisoning" the data set, i.e., by changing the data vectors or the class labels with the purpose to cause misclassifications in a machine learning model which uses this data for training.

QSAR Abbreviation for quantified structure activity relationship, a mathematical function that approximates the relationship between a molecule structure and its characteristics (activities).

reconstruction attack A privacy attack that abuses the machine learning model with the goal of reconstructing part of the data point or the whole data point using the data point's label and access to the model. Reconstruction attacks are **attribute inference attacks** and **model inversion attacks**.

saliency map A map that describes for each input feature the amount of impact that this feature has on the classification result during the forward-pass of a machine learning model for one specific input sample.

secure multi-party computation (SMPC) A protocol that enables multiple parties to compute a function together without revealing their inputs necessary for computing the global output.

self-supervision learning	A machine learning approach similar to supervised learning, but instead of using labels that were provided explicitly by human experts, labels are extracted from the data itself using implicit features of the data (e.g., in NLP, the context of how a word is used in a sentence can be utilized as a feature).
SGD	Abbreviation for stochastic gradient descent, an iterative optimization method for ML models that approximates gradient descent stochastically in order to gain computational efficiency.
student	In the context of transfer learning, the student model is the model that is trained on the new (student) task while reusing the weights of a model trained on a related (teacher) task (teacher model).
substitute model / shadow model / surrogate model	A neural network model that performs the same tasks and behaves similarly to the victim model. A substitute model has a similar structure, but not necessarily exactly the same structure (e.g., it may differ in the number of layers), and is trained using the same data set as the victim model, or a data set drawn from the same data distribution. The terms are equivalent, but differ in the domains in which they are commonly used: Whereas the term substitute (or surrogate) model is used mainly in the context of adversarial attacks, the term shadow model is commonly used in the context of membership inference and model reconstruction attacks.
target class	A class of a neural network classifier which is targeted by an attack. Usually, in the literature, the target class indicates the class *to which* the attacker aims to misclassify input data.
target data point	The data point targeted by the attack, for example, the membership inference attack that attempts to figure out whether this particular data point was part of the data set.
target model	Same as victim model
teacher	In the context of transfer learning, the teacher model is a model trained on a related (teacher) task which a user reuses in order to train a new model (student model) with a new (student) task.

transferability A property of adversarial examples. An adversarial example that was optimized for one machine learning model is said to be transferable to another machine learning model if it causes a misclassification also in the new model. Older publications often use the term generalize between models to describe that adversarial examples transfer.

transfer learning (TL) A machine learning approach for effectively training deep learning models, in particular, if little training data is available, by using a pretrained model that was trained on a related task. The parameters (e.g., weights) of the pretrained model (teacher model) are transferred to the new model (student model) and the new model is fine-tuned on the target data set. Depending on how many layers of the model are retrained during the student model training process, feature extraction transfer learning and weight initialization transfer learning can be differentiated.

trigger An input pattern that can be added to an input sample in order to activate the backdoor of a neural network which, in turn, would cause a misclassification.

Trojan (backdoor) A specific case of a backdoor of a neural network which is optimized for the given network model and usually invisible for a human observer.

utility A notion of usefulness of the model, usually used in context of differential privacy. Often, the model performance is used as utility metric.

vertically distributed data Data sets (at least two) that share the same entities, but have different features, for example, a medical data from a hospital and a general practitioner over the same patients.

VGG A set of ”very deep” convolutional neural network architectures consisting of multiple sets of convolutional and pooling layers followed by a fully-convolutional layer. VGG network architectures are commonly used for training image classification tasks. Pretrained on ImageNet, they serve as a common teacher model for transfer learning. Different VGG versions (e.g., VGG-16 and VGG-19) differ in the depth of the network.

victim model A term referring to the machine learning model under attack; equivalent to the term “target model”.

weight initialization transfer learning	A type of transfer learning where the student model is initialized with the parameters of the teacher model but (almost) all layers are retrained.
white-box access	An access mode where the machine learning model is fully known, including its inner parameters such as weights and bias terms for neural networks.

Literature

[1] N. Carlini. *Development of the number of adversarial example papers between 2014 and 2022*. https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html.

[2] W. Tao et al. "Real-Time Assembly Operation Recognition with Fog Computing and Transfer Learning for Human-Centered Intelligent Manufacturing". In: *Procedia Manufacturing* 48 (2020). DOI: 10.1016/j.promfg.2020.05.131.

[3] S. J. Pan and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009).

[4] K. Weiss, T. M. Khoshgoftaar, and D. Wang. "A survey of transfer learning". In: *Journal of Big data* 3.1 (2016).

[5] F. Zhuang et al. "A comprehensive survey on transfer learning". In: *Proceedings of the IEEE* 109.1 (2020).

[6] X. Liu et al. "Bagging based ensemble transfer learning". In: *Journal of Ambient Intelligence and Humanized Computing* 7.1 (2016).

[7] W. Dai et al. "Boosting for transfer learning". In: *Proceedings of the 24th international conference on Machine learning*. 2007.

[8] M. Oquab et al. "Learning and transferring mid-level image representations using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[9] M. Rivière et al. "Unsupervised pretraining transfers well across languages". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020.

[10] A. Baevski, M. Auli, and A. Mohamed. "Effectiveness of self-supervised pre-training for speech recognition". In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1911.03912.pdf.

[11] T. Evgeniou and M. Pontil. "Regularized multi–task learning". In: *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2004.

[12] J. Gao et al. "Knowledge transfer via multiple model local structure mapping". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008.

[13] X. Qin et al. "A novel relational-based transductive transfer learning method for PolSAR images via time-series clustering". In: *Remote Sensing* 11.11 (2019).

[14] J. Yosinski et al. "How transferable are features in deep neural networks?" In: *arXiv preprint* (2014). eprint: https://arxiv.org/pdf/1411.1792.pdf.

[15] M. Raghu et al. “Transfusion: Understanding transfer learning for medical imaging”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1902.07208.pdf.

[16] Y. Cui et al. “Large scale fine-grained categorization and domain-specific transfer learning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

[17] S.-J. Huang, J.-W. Zhao, and Z.-Y. Liu. “Cost-Effective Training of Deep CNNs with Active Model Adaptation”. In: *IEEE Women in Engineering Magazine* (2018). eprint: https://arxiv.org/pdf/1802.05394.pdf.

[18] B. Liu et al. “A cost-effective manufacturing process recognition approach based on deep transfer learning for CPS enabled shop-floor”. In: *Robotics and Computer-Integrated Manufacturing* 70 (2021). DOI: 10.1016/j.rcim.2021.102128.

[19] McKinsey. *Driving impact at scale from automation and AI*. https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Driving%20impact%20at%20scale%20from%20automation%20and%20AI/Driving-impact-at-scale-from-automation-and-AI.ashx. 2019.

[20] R. Ashmore. “Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1905.04223.pdf.

[21] A. Mathis et al. “DeepLabCut: markerless pose estimation of user-defined body parts with deep learning”. In: *Nature neuroscience* 21.9 (2018).

[22] C. Doersch and A. Zisserman. “Sim2real transfer learning for 3D human pose estimation: motion to the rescue”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1907.02499.pdf.

[23] P. Madhu et al. “Enhancing Human Pose Estimation in Ancient Vase Paintings via Perceptually-grounded Style Transfer Learning”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2012.05616.pdf.

[24] *Zenia app: Your personal Yoga studio*. https://zenia.app/.

[25] Y.-C. Huang et al. “Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications”. In: *Proceedings of 6th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2019.

[26] M. Archana and M. K. Geetha. “Object detection and tracking based on trajectory in broadcast tennis video”. In: *Procedia Computer Science* 58 (2015).

[27] K. Rangasamy et al. “Hockey activity recognition using pre-trained deep learning model”. In: *ICT Express* 6.3 (2020).

[28] A. Esteva et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *Nature* 542.7639 (2017).

[29] S. Sasikala. “Towards Improving Skin Cancer Detection Using Transfer Learning”. In: *Bioscience Biotechnology Research Communications* 13 (2020). DOI: 10.21786/bbrc/13.11/13.

[30] H. Zunair and A. Ben Hamza. “Melanoma detection using adversarial training and deep transfer learning”. In: *Physics in Medicine & Biology* 65.13 (2020). ISSN: 1361-6560. DOI: 10.1088/1361-6560/ab86d3.

[31] F. Liao et al. “Estimation of the Volume of the Left Ventricle From MRI Images Using Deep Neural Networks”. In: *IEEE Transactions on Cybernetics* 49.2 (2019). DOI: 10.1109/TCYB.2017.2778799.

[32] N. Behl. *White paper: Deep Resolve Mobilizing the power of networks*. Tech. rep. Siemens Heathcare, 2020.

[33] *Siemens Healthineers. Press release: Siemens Healthineers introduces AI-based assistants for magnetic resonance imaging*. https://www.corporate.siemens-healthineers.com/press/releases/pr-ai-rad-companions-mri.html. 2019.

[34] S. U. H. Dar et al. “A transfer-learning approach for accelerated MRI using deep neural networks”. In: *Magnetic resonance in medicine* 84.2 (2020).

[35] F. H. Fitzek, F. Granelli, and P. Seeling. *Computing in Communication Networks: From Theory to Practice*. Academic Press, 2020.

[36] J. M. Valverde et al. “Transfer Learning in Magnetic Resonance Brain Imaging: A Systematic Review”. In: *Journal of Imaging* 7.4 (2021). ISSN: 2313-433X. DOI: 10.3390/jimaging7040066.

[37] C.-L. Chen et al. “Generalization of diffusion magnetic resonance imaging–based brain age prediction model through transfer learning”. In: *NeuroImage* 217 (2020). ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2020.116831.

[38] S. Lu, Z. Lu, and Y.-D. Zhang. “Pathological brain detection based on AlexNet and transfer learning”. In: *Journal of Computational Science* 30 (2019).

[39] C. Cai et al. “Transfer learning for drug discovery”. In: *Journal of Medicinal Chemistry* 63.16 (2020).

[40] B. Qiang et al. “Target prediction model for natural products using transfer learning”. In: *International journal of molecular sciences* 22.9 (2021).

[41] E. Betzig et al. “Imaging intracellular fluorescent proteins at nanometer resolution”. In: *Science* 313.5793 (2006).

[42] O. Z. Kraus et al. “Automated analysis of high-content microscopy data with deep learning”. In: *Molecular systems biology* 13.4 (2017).

[43] J. M. Newby et al. “Convolutional neural networks automate detection for tracking of submicron-scale particles in 2D and 3D”. In: *Proceedings of the National Academy of Sciences* 115.36 (2018). ISSN: 0027-8424. DOI: 10.1073/pnas.1804420115. URL: https://www.pnas.org/content/115/36/9026.full.pdf.

[44] M. Saha, C. Chakraborty, and D. Racoceanu. “Efficient deep learning model for mitosis detection using breast histopathology images”. In: *Computerized Medical Imaging and Graphics* 64 (2018). ISSN: 0895-6111. DOI: 10.1016/j.compmedimag.2017.12.001.

[45] Z. Han et al. “Breast cancer multi-classification from histopathological images with structured deep learning model”. In: *Scientific reports* 7.1 (2017).

[46] Y. Zheng et al. “Feature extraction from histopathological images based on nucleus-guided convolutional neural network for breast lesion classification”. In: *Pattern Recognition* 71 (2017).

[47] C. Mazo et al. “Transfer learning for classification of cardiovascular tissues in histological images”. In: *Computer Methods and Programs in Biomedicine* 165 (2018). ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2018.08.006.

[48] N. Das, E. Hussain, and L. B. Mahanta. “Automated classification of cells into multiple classes in epithelial tissue of oral squamous cell carcinoma using transfer learning and convolutional neural network”. In: *Neural Networks* 128 (2020). ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.05.003.

[49] K. S. Beevi, M. S. Nair, and G. Bindu. “Automatic mitosis detection in breast histopathology images using convolutional neural network based deep transfer learning”. In: *Biocybernetics and Biomedical Engineering* 39.1 (2019).

[50] Y. Song et al. “Adapting fisher vectors for histopathology image classification”. In: *Proceedings of 14th International Symposium on Biomedical Imaging (ISBI 2017)*. IEEE. 2017.

[51] Y. Song et al. “Supervised intra-embedding of fisher vectors for histopathology image classification”. In: *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017.

[52] F. Kogan et al. “Winter wheat yield forecasting in Ukraine based on Earth observation, meteorological data and biophysical models”. In: *International Journal of Applied Earth Observation and Geoinformation* 23 (2013).

[53] J. Gallego, E. Carfagna, and B. Baruth. “Accuracy, objectivity and efficiency of remote sensing for agricultural statistics”. In: *Agricultural survey methods* (2010).

[54] Z. Yin and T. L. Williams. “Obtaining spatial and temporal vegetation data from Landsat MSS and AVHRR/NOAA satellite images for a hydrologic model”. In: *Photogrammetric Engineering and Remote Sensing* 63.1 (1997).

[55] S. Foerster et al. “Crop type mapping using spectral–temporal profiles and phenological information”. In: *Computers and Electronics in Agriculture* 89 (2012).

[56] A. Nowakowski et al. “Crop type mapping by using transfer learning”. In: *International Journal of Applied Earth Observation and Geoinformation* 98 (2021).

[57] J. Chen et al. “Using deep transfer learning for image-based plant disease identification”. In: *Computers and Electronics in Agriculture* 173 (2020).

[58] M. Abdallah et al. “Anomaly Detection through Transfer Learning in Agriculture and Manufacturing IoT Systems”. In: *arXiv preprint* (2021). eprint: https://arxiv.org/pdf/2102.05814.pdf.

[59] S. Coulibaly et al. “Deep neural networks with transfer learning in millet crop images”. In: *Computers in Industry* 108 (2019).

[60] X. Wen et al. “A rapid learning algorithm for vehicle classification”. In: *Information sciences* 295 (2015).

[61] W. Wu et al. “Detection of Moving Violations”. In: *Comput. Vis. Imaging Intell. Transp. Syst* 1 (2017).

[62] R. Marikhu et al. “Police eyes: real world automated detection of traffic violations”. In: *Proceedings of 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. IEEE. 2013.

[63] M. Peppa et al. “URBAN TRAFFIC FLOW ANALYSIS BASED ON DEEP LEARNING CAR DETECTION FROM CCTV IMAGE SERIES.” In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 42.4 (2018).

[64] D. Acharya, W. Yan, and K. Khoshelham. “Real-time image-based parking occupancy detection using deep learning.” In: *Proceedings of Research@Locate*. 2018.

[65] J. Shashirangana et al. “Automated license plate recognition: a survey on methods and techniques”. In: *IEEE Access* 9 (2020).

[66] A. Brunetti et al. “Computer vision and deep learning techniques for pedestrian detection and tracking: A survey”. In: *Neurocomputing* 300 (2018).

[67] D. Temel, M. Chen, and G. AlRegib. “Traffic Sign Detection Under Challenging Conditions: A Deeper Look into Performance Variations and Spectral Characteristics”. In: *IEEE Trans. Intell. Transp. Syst.* 21.9 (2020). DOI: 10.1109/TITS.2019.2931429.

[68] J. Yu and L. Petnga. “Space-based collision avoidance framework for autonomous vehicles”. In: *Procedia Computer Science* 140 (2018).

[69] M. H. Alkinani, W. Z. Khan, and Q. Arshad. “Detecting human driver inattentive and aggressive driving behavior using deep learning: Recent advances, requirements and open challenges”. In: *IEEE Access* 8 (2020).

[70] T. Liu et al. “Driver distraction detection using semi-supervised machine learning”. In: *IEEE transactions on intelligent transportation systems* 17.4 (2015).

[71] E. L. Manibardo, I. Laña, and J. Del Ser. “Transfer Learning and Online Learning for Traffic Forecasting under Different Data Availability Conditions: Alternatives and Pitfalls”. In: *Proceedings of the 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020.

[72] S. Y. Jo et al. “Transfer learning-based vehicle classification”. In: *Proceedings of the International SoC Design Conference (ISOCC)*. IEEE. 2018.

[73] S. Akhauri, L. Zheng, and M. Lin. “Enhanced Transfer Learning for Autonomous Driving with Systematic Accident Simulation”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[74] X. Cao et al. “Transfer learning for pedestrian detection”. In: *Neurocomputing* 100 (2013).

[75] R. Ayachi et al. “Pedestrian detection for advanced driving assisting system: A transfer learning approach”. In: *Proceedings of the International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. IEEE. 2020.

[76] M. E. Peters et al. “Deep contextualized word representations”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1802.05365.pdf.

[77] A. Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019).

[78] J. Devlin et al. “BERT: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1810.04805.pdf.

[79] J.-S. Lee and J. Hsiang. “Patent classification by fine-tuning BERT language model”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1906.02124.pdf.

[80] M. Mozafari, R. Farahbakhsh, and N. Crespi. “A BERT-based transfer learning approach for hate speech detection in online social media”. In: *Proceedings of the International Conference on Complex Networks and Their Applications*. Springer. 2019.

[81] C. Sun et al. “How to fine-tune BERT for text classification?” In: *Proceedings of the China National Conference on Chinese Computational Linguistics*. Springer. 2019.

[82] I. Beltagy, K. Lo, and A. Cohan. “SciBERT: A pretrained language model for scientific text”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1903.10676.pdf.

[83] D. Zhang et al. “E-BERT: A Phrase and Product Knowledge Enhanced Language Model for E-commerce”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2009.02835.pdf.

[84] J. Lee et al. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics* 36.4 (2020).

[85] P. Schrempf et al. “Paying Per-label Attention for Multi-label Extraction from Radiology Reports”. In: *Interpretable and Annotation-Efficient Learning for Medical Image Computing*. Springer, 2020.

[86] Y. Cui et al. “Pre-training with whole word masking for Chinese BERT”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1906.08101.pdf.

[87] B. Maschler, T. Knodel, and M. Weyrich. “Towards Deep Industrial Transfer Learning for Anomaly Detection on Time Series Data”. In: *arXiv preprint* (2021). eprint: https://arxiv.org/pdf/2106.04920.pdf.

[88] H. I. Fawaz et al. “Transfer learning for time series classification”. In: *Proceedings of the International Conference on Big Data (Big Data)*. IEEE. 2018.

[89] Q.-Q. He, P. C.-I. Pang, and Y.-W. Si. “Transfer learning for financial time series forecasting”. In: *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*. Springer. 2019.

[90] Q.-Q. He, P. C.-I. Pang, and Y.-W. Si. “Multi-source Transfer Learning with Ensemble for Financial Time Series Forecasting”. In: *arXiv preprint* (2021). eprint: https://arxiv.org/pdf/2103.15593.pdf.

[91] Y. Gao et al. “Deep learning and transfer learning models of energy consumption forecasting for a building with poor information data”. In: *Energy and Buildings* 223 (2020).

[92] T. Le et al. “Multiple electric energy consumption forecasting using a cluster-based strategy for transfer learning in smart building”. In: *Sensors* 20.9 (2020).

[93] P. Gupta et al. “Transfer learning for clinical time series analysis using deep neural networks”. In: *Journal of Healthcare Informatics Research* 4.2 (2020).

[94] P. Xiong et al. “Application of transfer learning in continuous time series for anomaly detection in commercial aircraft flight data”. In: *Proceedings of the International Conference on Smart Cloud (SmartCloud)*. IEEE. 2018.

[95] S. Gunduz, U. Ugurlu, and I. Oksuz. “Transfer Learning for Electricity Price Forecasting”. In: *arXiv preprint* (2020). eprint: `https://arxiv.org/pdf/2007.03762.pdf`.

[96] N. Laptev, J. Yu, and R. Rajagopal. “Reconstruction and regression loss for time-series transfer learning”. In: *Proceedings of the SIGKDD Workshop on Mining and Learning from Time Series*. 2018.

[97] R. Ye and Q. Dai. “Implementing transfer learning across different datasets for time series forecasting”. In: *Pattern Recognition* 109 (2021).

[98] U. Sarawgi et al. “Multimodal Inductive Transfer Learning for Detection of Alzheimer’s Dementia and its Severity”. In: *arXiv preprint* (2020). eprint: `https://arxiv.org/pdf/2009.00700.pdf`.

[99] W. Pan. “A survey of transfer learning for collaborative recommendation with auxiliary data”. In: *Neurocomputing* 177 (2016).

[100] D. Cevher, S. Zepf, and R. Klinger. “Towards multimodal emotion recognition in german speech events in cars using transfer learning”. In: *arXiv preprint* (2019). eprint: `https://arxiv.org/pdf/1909.02764.pdf`.

[101] S. Singhal et al. “Spotfake+: A multimodal framework for fake news detection via transfer learning (student abstract)”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 10. 2020.

[102] M. Abadi et al. “Tensorflow: A system for large-scale machine learning”. In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016.

[103] A. Paszke et al. *Automatic differentiation in PyTorch*. `https://pytorch.org/`. 2017.

[104] F. Chollet et al. *Keras*. `https://keras.io`. 2015.

[105] A. Kurakin, I. Goodfellow, and S. Bengio. “Adversarial machine learning at scale”. In: *arXiv preprint* (2016). eprint: `https://arxiv.org/pdf/1611.01236.pdf`.

[106] I. J. Goodfellow, J. Shlens, and C. Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint* (2014). eprint: `https://arxiv.org/pdf/1412.6572.pdf`.

[107] N. Carlini and D. A. Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *Proceedings of the 2017 Symposium on Security and Privacy*. IEEE Computer Society, 2017. DOI: 10.1109/SP.2017.49.

[108] C. Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint* (2014). eprint: https://arxiv.org/pdf/1312.6199.pdf (cs.CV).

[109] N. Papernot et al. “Practical black-box attacks against machine learning”. In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 2017.

[110] M. Sharif et al. “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition”. In: *Proceedings of the 2016 acm sigsac conference on computer and communications security*. 2016.

[111] A. N. Bhagoji et al. “Practical black-box attacks on deep neural networks using efficient query mechanisms”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[112] T.-W. Chin, C. Zhang, and D. Marculescu. “Improving the adversarial robustness of transfer learning via noisy feature distillation”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2002.02998.pdf.

[113] S. Rezaei and X. Liu. “A target-agnostic attack on deep models: Exploiting security vulnerabilities of transfer learning”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/arXiv:1904.04334.pdf.

[114] A. Abdelkader et al. “Headless Horseman: Adversarial Attacks on Transfer Learning Models”. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020.

[115] B. Wang et al. “With great training comes great vulnerability: Practical attacks against transfer learning”. In: *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*. 2018.

[116] S. Rezaei and X. Liu. “Security of deep learning methodologies: Challenges and opportunities”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1912.03735.pdf.

[117] O. M. Parkhi, A. Vedaldi, and A. Zisserman. “Deep face recognition”. In: *Proceedings of the British Machine Vision Conference 2015 (BMVC 2015)*. British Machine Vision Association, 2015.

[118] K. He et al. “Deep residual learning for image recognition”. In: *Proceedings of the 2016 Conference on Computer Vision and Pattern Recognition*. IEEE. 2016.

[119] J. Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *Proceedings of the 2009 Conference on Computer Vision and Pattern Recognition*. IEEE. 2009.

[120] Y. Gao et al. “Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2007.10760.pdf.

[121] Y. Li et al. “Backdoor learning: A survey”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2007.08745.pdf.

[122] A. Nguyen and A. Tran. "Input-aware dynamic backdoor attack". In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2010.08138.pdf.

[123] T. Gu, B. Dolan-Gavitt, and S. Garg. "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain". In: *arXiv preprint* (2017). eprint: https://arxiv.org/pdf/1708.06733.pdf.

[124] K. Doan et al. "Lira: Learnable, imperceptible and robust backdoor attacks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

[125] M. Goldblum et al. "Data Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses". In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2012.10544.pdf.

[126] W. Guo et al. "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in AI systems". In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1908.01763.pdf.

[127] C. Guo, R. Wu, and K. Q. Weinberger. "Trojannet: Embedding hidden Trojan horse models in neural networks". In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2002.10078.pdf.

[128] X. Huang, M. Alzantot, and M. Srivastava. "NeuronInspect: Detecting Backdoors in Neural Networks via Output Explanations". In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1911.07399.pdf (cs.CR).

[129] X. Chen et al. "Targeted backdoor attacks on deep learning systems using data poisoning". In: *arXiv preprint* (2017). eprint: https://arxiv.org/pdf/1712.05526.pdf.

[130] A. Shafahi et al. "Poison frogs! targeted clean-label poisoning attacks on neural networks". In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1804.00792.pdf.

[131] A. Turner, D. Tsipras, and A. Madry. *Clean-label backdoor attacks*. https://people.csail.mit.edu/madry/lab/cleanlabel.pdf. 2018.

[132] Y. Liu et al. "Trojaning attack on neural networks". In: *Proceedings of Network and Distributed Systems Security (NDSS) Symposium 2018*. 2017. URL: https://weihang-wang.github.io/papers/tnn_ndss18.pdf.

[133] R. Tang et al. "An Embarrassingly Simple Approach for Trojan Attack in Deep Neural Networks". In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2006.08131.pdf (cs.CR).

[134] A. Saha, A. Subramanya, and H. Pirsiavash. "Hidden Trigger Backdoor Attacks". In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1910.00033.pdf (cs.CV).

[135] Y. Liu et al. "Reflection backdoor: A natural backdoor attack on deep neural networks". In: *European Conference on Computer Vision*. Springer. 2020.

[136] Y. Liu, Y. Xie, and A. Srivastava. "Neural trojans". In: *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE. 2017. DOI: 10.1109/ICCD.2017.16.

[137] R. Pang et al. “TROJANZOO: Everything you ever wanted to know about neural backdoors (but were afraid to ask)”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2012.09302.pdf.

[138] Y. Yao et al. “Latent backdoor attacks on deep neural networks”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019.

[139] C. Zhu et al. “Transferable clean-label poisoning attacks on deep neural nets”. In: *International Conference on Machine Learning*. PMLR. 2019.

[140] H. Aghakhani et al. “Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability”. In: *arXiv preprint* (2021). eprint: https://arxiv.org/pdf/2005.00191.pdf (cs.LG).

[141] R. Pang et al. “A Tale of Evil Twins: Adversarial Inputs versus Poisoned Models”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2020. ISBN: 9781450370899. DOI: 10.1145/3372297.3417253.

[142] A. Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*. 2018. URL: https://openreview.net/forum?id=rJzIBfZAb.

[143] N. Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1 (2014). ISSN: 1532-4435.

[144] U. Shaham et al. “Defending against adversarial images using basis functions transformations”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1803.10840.pdf.

[145] A. Chakraborty et al. “A survey on adversarial attacks and defences”. In: *CAAI Transactions on Intelligence Technology* 6.1 (2021). DOI: https://doi.org/10.1049/cit2.12028. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/cit2.12028. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cit2.12028.

[146] N. Das et al. “Shield: Fast, practical defense and vaccination for deep learning using jpeg compression”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018.

[147] A. Buades, B. Coll, and J.-M. Morel. “A non-local algorithm for image denoising”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 2. IEEE. 2005.

[148] N. Kapoor et al. “From a Fourier-Domain Perspective on Adversarial Examples to a Wiener Filter Defense for Semantic Segmentation”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2012.01558.pdf.

[149] N. Carlini and D. Wagner. “Adversarial examples are not easily detected: Bypassing ten detection methods”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017.

[150] H. Hosseini et al. “Blocking transferability of adversarial examples in black-box learning systems”. In: *arXiv preprint* (2017). eprint: https://arxiv.org/pdf/1703.04318.pdf.

[151] W. He et al. “Adversarial Example Defense: Ensembles of Weak Defenses are not Strong”. In: *Proceedings of the 11th USENIX Workshop on Offensive Technologies (WOOT 2017)*. USENIX Association, 2017. URL: https://www.usenix.org/conference/woot17/workshop-program/presentation/he.

[152] S. Rezaei and X. Liu. “Deep learning for encrypted traffic classification: An overview”. In: *IEEE communications magazine* 57.5 (2019).

[153] N. Baracaldo et al. “Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach”. In: *AISec'1: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017. DOI: 10.1145/3128572.3140450.

[154] Y. Liu et al. “ABS: Scanning neural networks for back-doors by artificial brain stimulation”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019.

[155] B. Chen et al. “Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1811.03728.pdf (cs.LG).

[156] B. Tran, J. Li, and A. Madry. “Spectral Signatures in Backdoor Attacks”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1811.00636.pdf (cs.LG).

[157] E. Chou, F. Tramer, and G. Pellegrino. “Sentinet: Detecting localized universal attacks against deep learning systems”. In: *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE. 2020.

[158] Y. Gao et al. “STRIP: A Defence Against Trojan Attacks on Deep Neural Networks”. In: *Proceedings of the 35th Annual Computer Security Applications Conference*. ACSAC '19. San Juan, Puerto Rico, USA: Association for Computing Machinery, 2020. ISBN: 9781450376280. DOI: 10.1145/3359789.3359790.

[159] B. Wang et al. “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks”. In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019.

[160] H. Chen et al. “DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 2019. DOI: 10.24963/ijcai.2019/647.

[161] S. Udeshi et al. “Model Agnostic Defence against Backdoor Attacks in Machine Learning”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1908.02203.pdf (cs.LG).

[162] K. Liu, B. Dolan-Gavitt, and S. Garg. “Fine-pruning: Defending against backdooring attacks on deep neural networks”. In: *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer. 2018.

[163] D. Blalock et al. “What is the State of Neural Network Pruning?” In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2003.03033.pdf (cs.LG).

[164] M. Fredrikson, S. Jha, and T. Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015.

[165] R. R. Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017.

[166] The CyberPeace Institute. *Playing with Lives: Cyberattacks on Healthcare are Attacks on People*. 2021. URL: `https://cyberpeaceinstitute.org/report/2021-03-CyberPeaceInstitute-SAR001-Healthcare.pdf`.

[167] TrapX Research Labs. *TrapX Investigative Report MEDJACK.4: Medical Device Hijacking*. 2018. URL: `https://www.trustdimension.com/wp-content/uploads/2015/02/MedJack.4-ilovepdf-compressed.pdf`.

[168] BBC. “Irish cyber-attack: Hackers bail out Irish health service for free”. In: (2021). URL: `https://www.bbc.com/news/world-europe-57197688`.

[169] L. Coventry and D. Branley. “Cybersecurity in healthcare: a narrative review of trends, threats and ways forward”. In: *Maturitas* 113 (2018).

[170] M. S. Jalali and J. P. Kaiser. “Cybersecurity in hospitals: a systematic, organizational perspective”. In: *Journal of medical Internet research* 20.5 (2018).

[171] M. Mozaffari-Kermani. “Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare”. In: *IEEE Journal of Biomedical and Health Informatics* 19.6 (2015).

[172] C. Macrae. “Governing the safety of artificial intelligence in healthcare”. In: *BMJ Quality & Safety* 28.6 (2019). ISSN: 2044-5415. DOI: `10.1136/bmjqs-2019-009484`.

[173] A. I. Newaz et al. “Adversarial attacks to machine learning-based smart healthcare systems”. In: *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE. 2020.

[174] P. Vepakomma et al. “Split learning for health: Distributed deep learning without sharing raw patient data”. In: *arXiv preprint* (2018). eprint: `https://arxiv.org/pdf/1812.00564.pdf`.

[175] E. Bagdasaryan et al. “How to backdoor federated learning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020.

[176] A. Qayyum et al. “Secure and Robust Machine Learning for Healthcare: A Survey”. In: *arXiv preprint* (2020). eprint: `https://arxiv.org/pdf/2001.08103.pdf` (cs.LG).

[177] S. G. Finlayson et al. “Adversarial attacks on medical machine learning”. In: *Science* 363.6433 (2019).

[178] N. Karimian et al. “Unlock Your Heart: Next Generation Biometric in Resource-Constrained Healthcare Systems and IoT”. In: *IEEE Access* 7 (2019).

[179] Y. Mirsky et al. “Malicious tampering of 3D medical imagery using deep learning.” In: *28th USENIX Security Symposium, USENIX Association*. 2019.

[180] A. Rahman et al. “Adversarial examples – security threats to COVID-19 deep learning systems in medical IoT devices”. In: *IEEE Internet of Things Journal* 8 (2020).

[181] M. J. Chuquicusma et al. “How to fool radiologists with generative adversarial networks? A visual turing test for lung cancer diagnosis”. In: *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE. 2018.

[182] X. Ma et al. “Understanding adversarial attacks on deep learning based medical image analysis systems”. In: *Pattern Recognition* 110 (2021). ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107332.

[183] T. Sipola, S. Puuska, and T. Kokkonen. “Model fooling attacks against medical imaging: a short survey”. In: *Information & Security: An International Journal* 46.2 (2020).

[184] T. Sipola and T. Kokkonen. “One-Pixel Attacks Against Medical Imaging: A Conceptual Framework.” In: *WorldCIST (1)*. 2021.

[185] J. Korpihalkola, T. Sipola, and T. Kokkonen. “Color-Optimized One-Pixel Attack Against Digital Pathology Images”. In: *2021 29th Conference of Open Innovations Association (FRUCT)*. IEEE. 2021.

[186] L. C. Chu et al. “The potential dangers of artificial intelligence for radiology and radiologists”. In: *Journal of the American College of Radiology* 17.10 (2020).

[187] M. Nwadike et al. “Explainability Matters: Backdoor Attacks on Medical Imaging”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2101.00008.pdf.

[188] A. Jain, S. Nundy, and K. Abbasi. “Corruption: medicine’s dirty open secret”. In: *BMJ* 348:g4184 (2014).

[189] S. Wang et al. “Backdoor attacks against transfer learning with pre-trained deep learning models”. In: *IEEE Transactions on Services Computing* (2020). DOI: 10.1109/TSC.2020.3000900.

[190] A. Holzinger et al. “Causability and explainability of artificial intelligence in medicine”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9.4 (2019).

[191] M. A. Ahmad, C. Eckert, and A. Teredesai. “Interpretable machine learning in healthcare”. In: *Proceedings of the 2018 ACM international Conference on Bioinformatics, Computational Biology, and Health Informatics*. 2018.

[192] I. D. Apostolopoulos and T. A. Mpesiana. “Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks”. In: *Physical and Engineering Sciences in Medicine* 43.2 (2020).

[193] H. S. Maghdid et al. “Diagnosing COVID-19 pneumonia from X-ray and CT images using deep learning and transfer learning algorithms”. In: *Multimodal Image Exploitation and Learning 2021*. Vol. 11734. International Society for Optics and Photonics. 2021.

[194] N. E. M. Khalifa et al. “Detection of coronavirus (COVID-19) associated pneumonia based on generative adversarial networks and a fine-tuned deep transfer learning model using chest X-ray dataset”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2004.01184.pdf.

[195] M. E. Chowdhury et al. “Can AI help in screening viral and COVID-19 pneumonia?” In: *IEEE Access* 8 (2020).

[196] T. Rahman et al. “Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images”. In: *Computers in biology and medicine* 132 (2021).

[197] D. Kermany, K. Zhang, M. Goldbaum, et al. “Labeled optical coherence tomography (oct) and chest X-ray images for classification”. In: *Mendeley data* 2.2 (2018).

[198] T. Rahman et al. “Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray”. In: *Applied Sciences* 10.9 (2020).

[199] A. Veldanda and S. Garg. “On Evaluating Neural Network Backdoor Defenses”. In: *arXiv preprint* (2020). eprint: `https://arxiv.org/pdf/2010.12186.pdf`.

[200] E. Wenger et al. “Backdoor attacks against deep learning systems in the physical world”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

[201] S. Moosavi-Dezfooli et al. “Universal Adversarial Perturbations”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017. DOI: `10.1109/CVPR.2017.17`.

[202] J. Steinhardt, P. W. W. Koh, and P. S. Liang. “Certified defenses for data poisoning attacks”. In: *Advances in neural information processing systems* 30 (2017).

[203] X. Huang et al. “Safety verification of deep neural networks”. In: *International conference on computer aided verification*. Springer. 2017.

[204] M. Shafique et al. “Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead”. In: *IEEE Design & Test* 37.2 (2020).

[205] K. Doan, Y. Lao, and P. Li. “Backdoor Attack with Imperceptible Input and Latent Modification”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[206] K. Chen et al. “Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models”. In: *arXiv preprint* (2021). eprint: `https://arxiv.org/pdf/2110.02467.pdf`.

[207] Google. *Google Cloud ML transfer learning tutorial*. 2016. URL: `https://cloud.google.com/blog/products/gcp/how-to-classify-images-with-tensorflow-using-google-cloud-machine-learning-and-cloud-dataflow`.

[208] C. Basoglu et al. *Microsoft CNTK transfer learning tutorial*. 2018. URL: `https://docs.microsoft.com/en-us/cognitive-toolkit/Build-your-own-image-classifier-using-Transfer-Learning`.

[209] O. Eberhard and T. Zesch. “Effects of Layer Freezing when Transferring DeepSpeech to New Languages”. In: *arXiv preprint* (2021). eprint: `https://arxiv.org/pdf/2102.04097.pdf`.

[210] M. C. Kruithof et al. “Object recognition using deep convolutional neural networks with complete transfer and partial frozen layers”. In: *Optics and Photonics for Counterterrorism, Crime Fighting, and Defence XII*. Vol. 9995. International Society for Optics and Photonics. 2016.

[211] F. Qi et al. “Onion: A simple and effective defense against textual backdoor attacks”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2011.10369.pdf.

[212] S. Li et al. “Deep learning backdoors”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2007.08273.pdf.

[213] M. Xue et al. “One-to-N & N-to-one: Two advanced backdoor attacks against deep learning models”. In: *IEEE Transactions on Dependable and Secure Computing* (2020).

[214] M. Xue et al. “Imperceptible and Multi-channel Backdoor Attack against Deep Neural Networks”. In: *arXiv preprint* (2022). eprint: https://arxiv.org/pdf/2201.13164.pdf.

[215] L. Zhu et al. “CLEAR: Clean-Up Sample-Targeted Backdoor in Neural Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

[216] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. “Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples”. In: *arXiv preprint* (2016). eprint: https://arxiv.org/pdf/1605.07277.pdf.

[217] J. Chen and M. I. Jordan. “Boundary Attack++: Query-Efficient Decision-Based Adversarial Attack”. In: *CoRR* abs/1904.02144 (2019). eprint: https://arxiv.org/pdf/1904.02144.pdf.

[218] N. Papernot et al. “Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks”. In: *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016. DOI: 10.1109/SP.2016.41.

[219] A. Kurakin, I. J. Goodfellow, and S. Bengio. “Adversarial examples in the physical world”. In: *CoRR* abs/1607.02533 (2016). arXiv: 1607.02533. URL: http://arxiv.org/abs/1607.02533.

[220] H. Zhang et al. *The Limitations of Adversarial Training and the Blind-Spot Attack*. 2019. eprint: http://arxiv.org/abs/1901.04684.

[221] A. Shafahi et al. “Adversarially robust transfer learning”. In: *CoRR* abs/1905.08232 (2019). arXiv: 1905.08232. URL: http://arxiv.org/abs/1905.08232.

[222] A. Demontis et al. “Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks”. In: *Proceedings of the 28th USENIX Security Symposium*. USENIX Association, 2019. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/demontis.

[223] S. A. Seshia et al. *Formal Specification for Deep Neural Networks*. Tech. rep. UCB/EECS-2018-25. EECS Department, University of California, Berkeley, 2018. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-25.html.

[224] F. Liu, C. Zhang, and H. Zhang. “Towards Transferable Unrestricted Adversarial Examples with Minimum Changes”. In: *CoRR* abs/2201.01102 (2022). arXiv: 2201.01102. URL: https://arxiv.org/abs/2201.01102.

[225] P. Schwerdtner et al. “Risk Assessment for Machine Learning Models”. In: *CoRR* abs/2011.04328 (2020). arXiv: 2011.04328. URL: https://arxiv.org/abs/2011.04328.

[226] J. Smith. “Formal Verification of the Adversarial Robustness Property of Deep Neural Networks Through Dimension Reduction Heuristics, Refutation-based Abstraction, and Partitioning”. In: (2020). DOI: 10.26076/E0B1-6FA3. URL: https://digitalcommons.usu.edu/etd/7934.

[227] F. Tramèr et al. “The Space of Transferable Adversarial Examples”. In: *arXiv preprint* (2017). eprint: https://arxiv.org/abs/1704.03453.

[228] F. Tramèr et al. “On Adaptive Attacks to Adversarial Example Defenses”. In: *CoRR* abs/2002.08347 (2020). eprint: https://arxiv.org/pdf/2002.08347.pdf.

[229] F. Assion et al. “The Attack Generator: A Systematic Approach Towards Constructing Adversarial Attacks”. In: *CoRR* abs/1906.07077 (2019). arXiv: 1906.07077. URL: http://arxiv.org/abs/1906.07077.

[230] L. Visengeriyeva et al. *ml-ops.org*. en-us. URL: https://ml-ops.org/ (visited on 06/15/2022).

[231] A. Zolfi et al. “Adversarial Mask: Real-World Adversarial Attack Against Face Recognition Models”. In: *arXiv preprint* (2021). eprint: https://arxiv.org/pdf/2111.10759.pdf.

[232] C. Berghoff. “Protecting the integrity of the training procedure of neural networks”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2005.06928.pdf.

[233] C. Galusha. “Getting started with IT asset management”. In: *IT Professional* 3.3 (2001).

[234] M. Stone et al. “IT Asset Management”. In: *NIST special publication* 5 (2018).

[235] M. Rigaki and S. Garcia. “A survey of privacy attacks in machine learning”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2007.07646.pdf.

[236] R. Shokri et al. “Membership inference attacks against machine learning models”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017.

[237] A. Salem et al. “Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1806.01246.pdf.

[238] C. A. Choquette-Choo et al. “Label-Only Membership Inference Attacks”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2007.14321.pdf.

[239] M. Nasr, R. Shokri, and A. Houmansadr. “Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning”. In: *Proceedings of the IEEE Symposium on Security and Privacy*. 2019. ISBN: 9781538666609. DOI: 10.1109/SP.2019.00065.

[240] J. W. Bentley et al. “Quantifying Membership Inference Vulnerability via Generalization Gap and Other Model Metrics”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2009.05669.pdf.

[241] Y. Long et al. “Understanding Membership Inferences on Well-Generalized Learning Models”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1802.04889.pdf.

[242] M. Yaghini, B. Kulynych, and C. Troncoso. “Disparate Vulnerability: On the unfairness of privacy attacks against machine learning”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1906.00389.pdf.

[243] G. Ateniese et al. “Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers”. In: *International Journal of Security and Networks* 10.3 (2015). ISSN: 17478413. DOI: 10.1504/IJSN.2015.071829.

[244] S. Yeom et al. “Privacy risk in machine learning: Analyzing the connection to overfitting”. In: *Proceedings of the IEEE Computer Security Foundations Symposium*. 2018. ISBN: 9781538666807. DOI: 10.1109/CSF.2018.00027.

[245] M. Fredrikson et al. “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing”. In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014.

[246] Y. Zhang et al. “The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1911.07135.pdf.

[247] Z. Yang, E. C. Chang, and Z. Liang. “Adversarial neural network inversion via auxiliary knowledge alignment”. In: *arXiv preprint* (2019). ISSN: 23318422. eprint: https://arxiv.org/pdf/1902.08552.pdf.

[248] S. Hidano et al. “Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes”. In: *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE. 2017.

[249] X. Wu et al. “A methodology for formalizing model-inversion attacks”. In: *Proceedings of the IEEE Computer Security Foundations Symposium*. 2016. ISBN: 9781509026074. DOI: 10.1109/CSF.2016.32.

[250] K. Ganju et al. “Property inference attacks on fully connected neural networks using permutation invariant representations”. In: *Proceedings of the ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2018. ISBN: 9781450356930. DOI: 10.1145/3243734.3243834.

[251] F. Tramèr et al. “Stealing Machine Learning Models via Prediction APIs”. In: *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*. 2016. ISBN: 9781931971324.

[252] J. R. Correia-Silva et al. “Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data”. In: *Proceedings of 2018 International Joint Conference on Neural Networks (IJCNN)*. 2018. DOI: 10.1109/IJCNN.2018.8489592.

[253] T. Orekondy, B. Schiele, and M. Fritz. "Knockoff Nets: Stealing Functionality of Black-Box Models". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. DOI: 10.1109/CVPR.2019.00509. URL: https://ieeexplore.ieee.org/abstract/document/8953839.

[254] M. Juuti et al. "PRADA: Protecting against DNN Model Stealing Attacks". In: *Proceedings of 2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2019. DOI: 10.1109/EuroSP.2019.00044.

[255] M. Jagielski et al. "High Accuracy and High Fidelity Extraction of Neural Networks". In: *Open access to the Proceedings of the 29th USENIX Security Symposium is sponsored by USENIX*. 2020. ISBN: 9781939133175. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/jagielski.

[256] V. Chandrasekaran et al. "Exploring Connections Between Active Learning and Model Extraction". In: *Open access to the Proceedings of the 29th USENIX Security Symposium is sponsored by USENIX*. 2020. ISBN: 9781939133175. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/chandrasekaran.

[257] J. Zhao, Y. Chen, and W. Zhang. "Differential privacy preservation in deep learning: Challenges, opportunities and solutions". In: *IEEE Access* 7 (2019).

[258] Z. Ji, Z. C. Lipton, and C. Elkan. "Differential privacy and machine learning: a survey and review". In: *arXiv preprint* (2014). eprint: https://arxiv.org/pdf/1412.7584.pdf.

[259] T. Ha et al. "Differential privacy in deep learning: an overview". In: *2019 International Conference on Advanced Computing and Applications (ACOMP)*. IEEE. 2019.

[260] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Cambridge, Massachusetts, USA: MIT Press, 2016.

[261] C. Dwork. "Differential privacy: A survey of results". In: *International conference on theory and applications of models of computation*. Springer. 2008.

[262] C. Dwork et al. "Calibrating noise to sensitivity in private data analysis". In: *Theory of cryptography conference*. Springer. 2006.

[263] M. Abadi et al. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016.

[264] N. Papernot et al. "Semi-supervised knowledge transfer for deep learning from private training data". In: *arXiv preprint* (2016). eprint: https://arxiv.org/pdf/1610.05755.pdf.

[265] N. Papernot et al. "Scalable private learning with pate". In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1802.08908.pdf.

[266] B. Jayaraman and D. Evans. "Evaluating Differentially Private Machine Learning in Practice". In: *28th USENIX Security Symposium*. 2019. ISBN: 9781939133069. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman.

[267] M. A. Rahman et al. “Membership inference attack against differentially private deep learning model”. In: *Transactions on Data Privacy* 11.1 (2018). ISSN: 20131631.

[268] K. L. van der Veen et al. “Three Tools for Practical Differential Privacy”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1812.02890.pdf.

[269] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov. “Differential privacy has disparate impact on model accuracy”. In: *Advances in Neural Information Processing Systems*. 2019.

[270] D. Bernau et al. “Assessing differentially private deep learning with Membership Inference”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1912.11328.pdf.

[271] M. Nasr, R. Shokri, and A. Houmansadr. “Machine learning with membership privacy using adversarial regularization”. In: *Proceedings of the ACM Conference on Computer and Communications Security*. 2018. ISBN: 9781450356930. DOI: 10.1145/3243734.3243855.

[272] J. Jia and N. Z. Gong. “AttriGuard: A practical defense against attribute inference attacks via adversarial machine learning”. In: *Proceedings of the 27th USENIX Security Symposium* August (2018).

[273] T. A. Alves, F. M. França, and S. Kundu. “MLPrivacyGuard: Defeating Confidence Information based Model Inversion Attacks on Machine Learning Systems”. In: *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. 2019.

[274] J. Jia et al. “Memguard: Defending against black-box membership inference attacks via adversarial examples”. In: *Proceedings of the ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2019. ISBN: 9781450367479. DOI: 10.1145/3319535.3363201.

[275] J. Wang et al. “Private model compression via knowledge distillation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019.

[276] V. Shejwalkar and A. Houmansadr. “Reconciling Utility and Membership Privacy via Knowledge Distillation”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1906.06589.pdf.

[277] N. Carlini et al. “Is private learning possible with instance encoding?” In: *Proceedings of the IEEE Symposium on Security and Privacy* 2021-May (2021). ISSN: 10816011. DOI: 10.1109/SP40001.2021.00099.

[278] H. Jia et al. “Entangled Watermarks as a Defense against Model Extraction”. In: *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*. 2020. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/jia.

[279] C. Song and A. Raghunathan. “Information Leakage in Embedding Models”. In: *Proceedings of the ACM Conference on Computer and Communications Security* (2020). ISSN: 15437221. DOI: 10.1145/3372297.3417270. eprint: 2004.00053.

[280] S. Hisamoto, M. Post, and K. Duh. "Membership Inference Attacks on Sequence-to-Sequence Models: Is My Data In Your Machine Translation System?" In: *Transactions of the Association for Computational Linguistics* 8 (2020). DOI: 10.1162/tacl_a_00299. URL: http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00299/1923547/tacl_a_00299.pdf.

[281] N. Carlini et al. "The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks." In: *Proceedings of the 28th USENIX security symposium*. 2019. ISBN: 9781939133069. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/carlini.

[282] K. Krishna et al. "Thieves on Sesame Street! Model Extraction of BERT-based APIs". In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1910.12366.pdf.

[283] N. Carlini et al. "Extracting training data from large language models". In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2012.07805.pdf.

[284] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998).

[285] Z. Liu et al. "Deep learning face attributes in the wild". In: *Proceedings of the IEEE international conference on computer vision*. 2015.

[286] D. M. Bernard, J. S. Banthin, and W. E. Encinosa. "Wealth, income, and the affordability of health insurance". In: *Health Affairs* 28.3 (2009).

[287] D. Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *Journal of Chemical Information and Computer Sciences* 28.1 (1988). DOI: 10.1021/ci00057a005.

[288] O. J. Wouters, M. McKee, and J. Luyten. "Estimated research and development investment needed to bring a new medicine to market, 2009-2018". In: *JAMA* 323.9 (2020).

[289] D. Rogers and M. Hahn. "Extended-Connectivity Fingerprints". In: *Journal of Chemical Information and Modeling* 50.5 (2010). DOI: 10.1021/ci100050t.

[290] D. Butina. "Unsupervised data base clustering based on daylight's fingerprint and Tanimoto similarity: A fast and automated way to cluster small and large data sets". In: *Journal of Chemical Information and Computer Sciences* 39.4 (1999).

[291] B. Wang and N. Z. Gong. "Stealing Hyperparameters in Machine Learning". In: *Proceedings - IEEE Symposium on Security and Privacy* 2018-May.May (2018). ISSN: 10816011. DOI: 10.1109/SP.2018.00038. arXiv: 1802.05351.

[292] T. Zhu et al. "More Than Privacy : Applying Differential Privacy in Key Areas of Artificial Intelligence". In: *arXiv preprint* (2020). arXiv: https://arxiv.org/abs/2008.01916v1.

[293] J. Ding et al. "Differentially private and fair classification via calibrated functional mechanism". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 01. 2020.

[294] A. Beutel et al. “Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations”. In: *arXiv preprint* (2017). arXiv: https://arXiv.org/abs/1707.00075v2.

[295] B. Zhang et al. “Privacy for All : Demystify Vulnerability Disparity of Differential Privacy against Membership Inference A ack”. In: *arXiv preprint arXiv:2001.08855* (2020). arXiv: arXiv:2001.08855v1.

[296] S. Milli et al. “Model Reconstruction from Model Explanations”. In: *arXiv preprint* (2018). arXiv: https://arxiv.org/abs/1807.05185.

[297] R. Shokri, M. Strobel, and Y. Zick. “On the Privacy Risks of Model Explanations”. In: *AIES 2021 - Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society* (2021). DOI: 10.1145/3461702.3462533. arXiv: https://arxiv.org/abs/1907.00164.

[298] B. Baron and M. Musolesi. “Interpretable Machine Learning for Privacy-Preserving Pervasive Systems”. In: *IEEE Pervasive Computing* 19.1 (2020). ISSN: 15582590. DOI: 10.1109/MPRV.2019.2918540. arXiv: https://arxiv.org/abs/1710.08464.

[299] M. Lecuyer et al. “Certified robustness to adversarial examples with differential privacy”. In: *Proceedings - IEEE Symposium on Security and Privacy* 2019-May.February (2019). ISSN: 10816011. DOI: 10.1109/SP.2019.00044. arXiv: 1802.03471.

[300] R. Pinot et al. *A unified view on differential privacy and robustness to adversarial examples*. Tech. rep. 2019. eprint: https://arxiv.org/abs/1906.07982v1.

[301] M. Du, R. Jia, and D. Song. *Robust Anomaly Detection and Backdoor Attack Detection Via Differential Privacy*. Tech. rep. 2019. eprint: https://arxiv.org/abs/1911.07116v1.

[302] Y. Ma, X. Zhu, and J. Hsu. “Data Poisoning against Differentially-Private Learners: Attacks and Defenses”. In: *arXiv e-prints* (2019). arXiv: 1903.09860. URL: http://arxiv.org/abs/1903.09860.

[303] K.-P. Lin and M.-S. Chen. “On the design and analysis of the privacy-preserving SVM classifier”. In: *IEEE transactions on knowledge and data engineering* 23.11 (2010).

[304] P. Strecht. “A survey of merging decision trees data mining approaches”. In: *Proc. 10th Doctoral Symposium in Informatics Engineering*. 2015.

[305] B. McMahan et al. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017.

[306] J. Dean et al. “Large scale distributed deep networks”. In: *Advances in neural information processing systems* 25 (2012).

[307] Y. Kang et al. “Neurosurgeon: Collaborative intelligence between the cloud and mobile edge”. In: *ACM SIGARCH Computer Architecture News* 45.1 (2017).

[308] J. Hauswald et al. “A hybrid approach to offloading mobile image classification”. In: *2014 International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014.

[309] P. Kairouz et al. “Advances and open problems in federated learning”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1912.04977.pdf.

[310] H. Brendan McMahan et al. “Learning Differentially Private Recurrent Language Models”. In: *arXiv preprint* (2017). eprint: https://arxiv.org/pdf/1710.06963.pdf.

[311] V. Mothukuri et al. “A survey on security and privacy of federated learning”. In: *Future Generation Computer Systems* 115 (2021).

[312] N. Bouacida and P. Mohapatra. “Vulnerabilities in Federated Learning”. In: *IEEE Access* 9 (2021).

[313] P. M. Mammen. “Federated Learning: Opportunities and Challenges”. In: *arXiv preprint* (2021). eprint: https://arxiv.org/pdf/2101.05428.pdf.

[314] T. Li et al. “Federated learning: Challenges, methods, and future directions”. In: *IEEE Signal Processing Magazine* 37.3 (2020).

[315] L. Li et al. “A review of applications in federated learning”. In: *Computers & Industrial Engineering* (2020).

[316] C. Zhang et al. “A survey on federated learning”. In: *Knowledge-Based Systems* 216 (2021).

[317] M. S. Jere, T. Farnan, and F. Koushanfar. “A taxonomy of attacks on federated learning”. In: *IEEE Security & Privacy* 19.2 (2020).

[318] L. Lyu, H. Yu, and Q. Yang. “Threats to federated learning: A survey”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2003.02133.pdf.

[319] B. Hitaj, G. Ateniese, and F. Perez-Cruz. “Deep models under the GaN: Information leakage from collaborative deep learning”. In: *Proceedings of the 2017 ACM SIGSAG Conference on Computer and Communications Security* 1 (2017).

[320] L. Melis et al. “Exploiting unintended feature leakage in collaborative learning”. In: *Proceedings of the IEEE Symposium on Security and Privacy* 2019-May (2019). ISSN: 10816011. DOI: 10.1109/SP.2019.00029. eprint: 1805.04049.

[321] L. Zhu, Z. Liu, and S. Han. “Deep leakage from gradients”. In: *Federated learning*. Vol. 12500. Lecture Notes in Computer Science book series. Springer, 2020.

[322] J. Geiping et al. “Inverting Gradients - How easy is it to break privacy in federated learning?” In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2003.14053.pdf.

[323] W. Wei et al. “A framework for evaluating gradient leakage attacks in federated learning”. In: *arXiv preprint* (2020). ISSN: 23318422. eprint: https://arxiv.org/pdf/2004.10397.pdf.

[324] F. Mo et al. “Layer-wise Characterization of Latent Information Leakage in Federated Learning”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/pdf/2010.08762.pdf.

[325] Z. Wang et al. “Beyond Inferring Class Representatives: User-Level Privacy Leakage from Federated Learning”. In: *Proceedings of the IEEE INFOCOM* 2019-April (2019). ISSN: 0743166X. DOI: 10.1109/INFOCOM.2019.8737416.

[326] H. Zheng, H. Hu, and Z. Han. “Preserving User Privacy for Machine Learning: Local Differential Privacy or Federated Machine Learning?” In: *IEEE Intelligent Systems* 35.4 (2020). ISSN: 19411294. DOI: 10.1109/MIS.2020.3010335.

[327] R. Shokri and V. Shmatikov. “Privacy-Preserving Deep Learning”. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015. ISBN: 9781450338325.

[328] S. Truex et al. “LDP-Fed: Federated learning with local differential privacy”. In: *Proceedings of the 3rd ACM International Workshop on Edge Systems, Analytics and Networking, Part of EuroSys 2020*. 2020. ISBN: 9781450371322. DOI: 10.1145/3378679.3394533.

[329] O. Choudhury et al. “Differential Privacy-enabled Federated Learning for Sensitive Health Data”. In: *arXiv preprint* (2019). eprint: https://arxiv.org/pdf/1910.02578.pdf.

[330] S. Goryczka and L. Xiong. “A Comprehensive Comparison of Multiparty Secure Additions with Differential Privacy”. In: *IEEE Transactions on Dependable and Secure Computing* 14 (5 2017). DOI: 10.1117/12.2549369.Hyperspectral.

[331] J. Hayes et al. “LOGAN: Membership inference attacks against generative models”. In: *Proceedings on Privacy Enhancing Technologies* 1 (2017). ISSN: 23318422. DOI: 10.2478/popets-2019-0008.

[332] D. Chen et al. “GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models”. In: *Proceedings of the ACM Conference on Computer and Communications Security* 1 (2020). ISSN: 15437221. DOI: 10.1145/3372297.3417238.

[333] Q. Chen et al. “Differentially Private Data Generative Models”. In: *arXiv preprint* (2018). ISSN: 23318422. eprint: https://arxiv.org/pdf/1812.02274.pdf.

[334] A. Triastcyn and B. Faltings. “Generating Differentially Private Datasets Using GANS”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1803.03148v1.pdf.

[335] C. Dwork, A. Roth, et al. “The algorithmic foundations of differential privacy.” In: *Found. Trends Theor. Comput. Sci.* 9.3-4 (2014).

[336] L. Xie et al. “Differentially Private Generative Adversarial Network”. In: *arXiv preprint* (2018). eprint: https://arxiv.org/pdf/1802.06739.pdf.

[337] J. Jordon, J. Yoon, and M. Van der Schaar. “PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees”. In: *Proceedings of International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=S1zk9iRqF7.

[338] B. K. Beaulieu-Jones et al. “Privacy-preserving generative deep neural networks support clinical data sharing”. In: *Circulation: Cardiovascular Quality and Outcomes* 12.7 (2019).

[339] N. H. Phan et al. “Differential privacy preservation for deep auto-encoders: An application of human behavior prediction”. In: *30th AAAI Conference on Artificial Intelligence, AAAI 2016* (2016).

[340] J. Zhang et al. “Functional mechanism: Regression analysis under differential privacy”. In: *Proceedings of the VLDB Endowment* 5.11 (2012). ISSN: 21508097. DOI: 10.14778/2350229.2350253.

[341] A. Fischer et al. “Computation on Encrypted Data using Data Flow Authentication”. In: *arXiv preprint* (2017). eprint: https://arxiv.org/pdf/1710.00390.pdf.

[342] N. Sengupta. “Designing Encryption and IDS for Cloud Security”. In: *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing*. ICC ’17. Cambridge, United Kingdom: Association for Computing Machinery, 2017. ISBN: 9781450347747. DOI: 10.1145/3018896.3018954.

[343] Y. Lin et al. “Privacy-preserving deep packet filtering over encrypted traffic in software-defined networks”. In: *Proceedings of the 2016 IEEE International Conference on Communications (ICC)*. 2016. DOI: 10.1109/ICC.2016.7510993.

[344] N. Chatzis. “Securing the Internet by analysing and controlling DNS traffic: email worm detection and mitigation”. PhD thesis. Berlin, Germany: Berlin Institute of Technology, 2010. URL: https://www.depositonce.tu-berlin.de/bitstream/11303/2944/1/Dokument_9.pdf.

[345] S. Zhao et al. “SecTEE: A Software-Based Approach to Secure Enclave Architecture Using TEE”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. London, United Kingdom: Association for Computing Machinery, 2019. ISBN: 9781450367479. DOI: 10.1145/3319535.3363205.

[346] C. Gentry. “Fully Homomorphic Encryption Using Ideal Lattices”. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. STOC ’09. Bethesda, MD, USA: Association for Computing Machinery, 2009. ISBN: 9781605585062. DOI: 10.1145/1536414.1536440.

[347] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. “Fully Homomorphic Encryption without Bootstrapping”. In: *Cryptology ePrint Archive, Report 2011/277* (2011).

[348] J. Fan and F. Vercauteren. “Somewhat Practical Fully Homomorphic Encryption”. In: *Cryptology ePrint Archive, Report 2012/144* (2012).

[349] C. Gentry, A. Sahai, and B. Waters. “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based”. In: *Cryptology ePrint Archive, Report 2013/340* (2013).

[350] J. H. Cheon et al. “Homomorphic Encryption for Arithmetic of Approximate Numbers”. In: *Proceedings of Advances in Cryptology (ASIACRYPT 2017), 23rd International Conference on the Theory and Applications of Cryptology and Information Security*. Vol. 10624. Lecture Notes in Computer Science. Springer, 2017. DOI: 10.1007/978-3-319-70694-8_15.

[351] T. P. Pedersen. “A Threshold Cryptosystem without a Trusted Party”. In: *Proceedings of Advances in Cryptology (EUROCRYPT ’91)*. Ed. by D. W. Davies. Springer Berlin Heidelberg, 1991. ISBN: 978-3-540-46416-7.

[352] V. Nikolaenko et al. “Privacy-Preserving Ridge Regression on Hundreds of Millions of Records”. In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy, (SP 2013)*. IEEE Computer Society, 2013. DOI: 10.1109/SP.2013.30.

[353] R. Ma et al. “Secure multiparty computation for privacy-preserving drug discovery”. In: *Bioinformatics* 36.9 (2020). DOI: 10.1093/bioinformatics/btaa038.

[354] P. Mukherjee and D. Wichs. "Two Round Multiparty Computation via Multi-key FHE". In: *Proceedings of Advances in Cryptology (EUROCRYPT 2016) of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016. DOI: 10.1007/978-3-662-49896-5_26.

[355] T. Graepel, K. Lauter, and M. Naehrig. "ML Confidential: Machine Learning on Encrypted Data". In: *Proceedings of Information Security and Cryptology (ICISC 2012)*. Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-37682-5.

[356] J. H. Cheon et al. "Ensemble Method for Privacy-Preserving Logistic Regression Based on Homomorphic Encryption". In: *IEEE Access* 6 (2018). DOI: 10.1109/ACCESS.2018.2866697.

[357] L. Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research". In: *IEEE Signal Processing Magazine* 29 (2012).

[358] S. Park et al. "HE-Friendly Algorithm for Privacy-Preserving SVM Training". In: *IEEE Access* 8 (2020). DOI: 10.1109/ACCESS.2020.2981818.

[359] H. Fang and Q. Qian. "Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning". In: *Future Internet* 13.4 (2021). ISSN: 1999-5903. DOI: 10.3390/fi13040094. URL: https://www.mdpi.com/1999-5903/13/4/94.

[360] B. Reagen et al. "Cheetah: Optimizing and Accelerating Homomorphic Encryption for Private Inference". In: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 2021. DOI: 10.1109/HPCA51647.2021.00013.

[361] J. Lee et al. "Privacy-Preserving Machine Learning with Fully Homomorphic Encryption for Deep Neural Network". In: *arXiv preprint* (2021). eprint: https://arxiv.org/pdf/2106.07229.pdf.

[362] R. Gilad-Bachrach et al. "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy". In: *Proceedings of the 33nd International Conference on Machine Learning (ICML 2016)*. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016. URL: http://proceedings.mlr.press/v48/gilad-bachrach16.html.

[363] A. Benaissa et al. "TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption". In: *arXiv preprint* (2021). eprint: https://arxiv.org/pdf/2104.03152.pdf.

[364] J. Sun et al. "ExCAPE-DB: an integrated large scale dataset facilitating Big Data analysis in chemogenomics". In: *Journal of Cheminformatics* 9.1 (2017).

[365] T. Le et al. "Neuraldecipher - Reverse-Engineering ECFP Fingerprints to Their Molecular Structures". In: *chemrXiv preprint* (2020). DOI: 10.26434/chemrxiv.12286727.v2.

[366] M. Hashimoto and Q. Zhao. "An ELM-Based Privacy Preserving Protocol for Implementing Aware Agents". In: *Proceedings of the 3rd IEEE International Conference on Cybernetics (CYBCONF 2017)*. IEEE, 2017. DOI: 10.1109/CYBConf.2017.7985756.

[367] V. Tolpegin et al. “Data poisoning attacks against federated learning systems”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 12308 LNCS. Springer Science and Business Media Deutschland GmbH, 2020. ISBN: 9783030589509. DOI: 10.1007/978-3-030-58951-6_24. arXiv: https://arxiv.org/abs/2007.08432.

[368] A. Huang. “Dynamic backdoor attacks against federated learning”. In: *arXiv preprint* (2020). eprint: https://arxiv.org/abs/2011.07429. URL: http://arxiv.org/abs/2011.07429.

[369] L. Burkhalter et al. “RoFL: Attestable Robustness for Secure Federated Learning”. In: *arXiv preprint* (2021). eprint: https://arxiv.org/abs/2107.03311.

[370] A. Cheu, A. Smith, and J. Ullman. “Manipulation Attacks in Local Differential Privacy”. In: *arXiv e-prints* (2019). arXiv: https://arxiv.org/abs/1909.09630v1.

[371] X. Cao, J. Jia, and N. Z. Gong. “Data poisoning attacks to local differential privacy protocols”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021.

[372] Z. Sun et al. “Can You Really Backdoor Federated Learning?” In: *arXiv preprint* (2019). eprint: http://arxiv.org/abs/1911.07963.

[373] M. Naseri, J. Hayes, and E. De Cristofaro. “Toward Robustness and Privacy in Federated Learning: Experimenting with Local and Central Differential Privacy”. In: *arXiv preprint* (2020). eprint: http://arxiv.org/abs/2009.03561.

[374] J. Wang et al. “A Field Guide to Federated Optimization”. In: *arXiv preprint* (2021). eprint: http://arxiv.org/abs/2107.06917.