Inference and Extraction Attacks on Machine Learning Models: A Review

Jetske Beks
University of Amsterdam
11065249

Willemijn Beks
University of Amsterdam
10775110

Mike Schouw
University of Amsterdam
11268751

Abstract

Machine learning has seen an enormous surge in popularity. Nowadays, machine learning models are often placed in cloud environments to make them publicly accessible. This access heightens the privacy and security risk, as an increasing amount of attacks has been launched against these models. This work aims to give a structured review of the literature on three types of attacks: membership inference, model inversion and model extraction. We analyze the prerequisites, methods and defenses as different aspects of these attacks. The main findings from our research are: (1) there is a large privacy-utility trade-off in defending against membership inference and model inversion attacks; (2) more research needs to be done regarding protection against model extraction; (3) model inversion and model extraction attacks require substantial knowledge beforehand. Although not all attacks are directly applicable to real-world scenarios, some do pose clear security risks to publicized machine learning models.

1 Introduction

In the past decade, machine learning (ML) has been used for a diverse set of applications in various domains, such as medical treatments (Fredrikson et al., 2014), text prediction (Carlini et al., 2019) and authentication systems (Al-Rubaie and Chang, 2016). Increasingly, these models are placed in the cloud for easy access and use, giving rise to Machine Learning as a Service (MLaaS).
However, this convenience comes with a cost: access can be abused to extract information from a model that model owners and data contributors would rather keep private. As models are regularly trained on sensitive data, containing for example medical facts or passwords, obtaining any kind of information about the training set often means a serious breach of privacy. This is the case for partial or entire data samples, but even confirmation of membership in the training set could cause a privacy risk. Knowledge about the model internals, while not strictly related to personal privacy, does make such attacks easier to execute. Additionally, business competitors could steal the model internals and training data for their own use.
In this paper the existing literature on these types of attacks is reviewed. We discuss the prerequisites for mounting an attack, the types of methods used, and the effect of various defenses on the efficiency. It is organized as follows: in Section 2, some preliminaries are provided on the machine learning and privacy preservation context, as well as a more precise demarcation of the research space. Then, the literature on the different types of attacks is reviewed in three dedicated sections, one for each type of attack. Lastly, in Section 6 we conclude the discussion and propose some directions for future research in this field.

Preprint. Under review.

2 Background

2.1 On Machine Learning

Machine learning (ML) can be understood as having three main categories: supervised learning, unsupervised learning, and reinforcement learning.
Supervised learning uses labelled data as input, consisting of a set of samples X and corresponding labels Y. The model is trained to learn the function f mapping samples to output labels, $Y = f(X)$. This results in a classification or regression model depending on whether the labels are discrete or continuous.
With unsupervised learning, the algorithm attempts to find some structure in unlabelled training data. An example of this type is a generative adversarial network (GAN) that is used to create new data samples representative for some dataset.
Finally, reinforcement learning works by defining an agent and a world, with rules for the agent to prosper in that world. The agent iterates taking actions that have an effect on the world, receiving feedback on those actions, and improving its decision process. Learning the preferred action of some agent is usually done by neural networks with large parameter sets.

The main ML technique that is used throughout most research for this paper are neural networks (NN), and specifically deep neural networks (DNN). To understand how these pertain to ML in general we will explain briefly what the different techniques are within this field.
A neural network is an architecture consisting of multiple layers of nodes that connect in some sense to the previous and next layer. The first layer is the input layer, the last layer is the output layer, and all possible layers in between are the hidden layers. A neural network with at least one hidden layer is called a *deep* neural network. The nodes calculate their values based on the weighted sum of the previous nodes that are connected to it. To get a non-linear model, an activation function is used. This process repeats at every layer and finally, the output is calculated at the output layer (Chen and Ran, 2019).
The training of such a network happens with stochastic gradient descent (SGD), which applies many of these iterations. First, a forward pass through the network is done with a random sample. Every time an output value is calculated, the result is compared to a known value (supervised) or some other result measure (unsupervised), and the error is back-propagated through the network and the weights at each node are updated. This process starts at the final layer of the model and ends at the starting layer (Chen and Ran, 2019). For each pass, or epoch, a randomly selected 'mini-batch' of data is used to update the gradients so that the training loss is minimized (Ruder, 2016).
Training a network can be done for any preferred amount of iterations. Nonetheless, it is possible to train too little or too much, which can lead to underfitting or overfitting respectively. Underfitting refers to a model that does not capture the subtleties of a dataset, while overfitting refers to a model that also captured the outliers within the training set and thus does not generalise well.
When the training phase is done, the result for a single (new) datapoint can be obtained by entering it in the trained network; this process is called inference. The type of result depends on whether the dataset that the NN was trained on modelled regression or classification. There are many different types of NNs that all have their own domain of application, e.g. convolution neural networks (CNN) which are used for image processing, or recurrent neural networks (RNN) used for time-series problems.

Lastly, there is the concept of collaborative learning. Until recently, it was assumed that ML applications would run centralized and offline (Park et al., 2019). However, recent developments, such as an increase in edge devices, have made collaborative learning a much more common phenomenon. Participants in collaborative learning can keep their training data on a local device, as well as a local version of the machine learning model that is being trained. After iterating one or several rounds on the local model, the model parameters are exchanged between devices (Park et al., 2019). This can be done via a central server, which receives and averages all parameters, or by sending the parameters to all participants and aggregating them locally.

2.2 On Attacks

In this work, we will focus on three types of attacks intended to extract private information from machine learning models. Attacks designed to interfere with the functionality of the model, such as adversarial or poisoning attacks, are not treated in this survey.

Membership Inference attacks attempt to determine membership status. Given a data instance and access to a model, the goal is to detect whether this instance was used to train the model. As this depends on differences in the treatment of training versus test samples, overfitting is a key concept.

Model Inversion is a related attack which also aims to extract information about the training set. The precise goal differs across works, but generally, the adversary wants to obtain new information on the training data: statistical properties, sensitive attributes, or even complete data points can be extracted. The term *data reconstruction* can refer to model inversion attacks as well as to attacks on statistical databases, without the intermediary of a machine learning model. In this survey we only consider the first type.

Model Extraction attacks attempt to recover the specifications of the model itself. Commonly this happens in situations where the adversary only has query access to the model, such as MLaaS (Machine Learning as a Service) applications. Model extraction can also be done via access to the hardware that the model is running on. We will not be covering these types of attacks, as the methods used for this have a different focus.

In reviewing the literature on these three types of attacks, we will use three aspects to frame the discussion: *prerequisites*, *methods*, and *defenses*. The prerequisites consist of a set of information and access needs that should be met before launching an attack. The method describes the algorithm or process of attacking. Lastly, defenses can be used to repel an attack. Some preliminaries on defense will be discussed in Section 2.3.

Access Level

An often-used distinction in the literature, which is an important part of the prerequisites of any attack, is that between *white box* and *black box* access. An adversary with white box access will be able to see the internals of a model, for example in situations where the model can be downloaded or accessed from the server it is running on. In the context of collaborative learning, the participants have white box access by definition, as they are training the model and exchanging model parameters themselves.
Black box access means that the model internals are hidden. Usually in these cases the adversary has query access to the model, sometimes with a limit on the number of queries that can be done, called the *query budget*. On occasion the term *grey box* is used to indicate an intermediate access form where only some information about the model is known. Model extraction attacks are always grey or black box.

2.3 On Differential Privacy

For defending against attacks on the training data, differential privacy (DP) is a very important notion. In short, it is a formal guarantee that the privacy of a single sample in a set is not breached when doing some operation on the set. In the context of this review, that operation is a machine learning algorithm, but it is also used in other contexts. Usually it is defined as follows:

$$Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} Pr[\mathcal{M}(d') \in S]$$

for an operation $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ working on adjacent inputs $d, d' \in \mathcal{D}$ and a subset of outputs $S \in \mathcal{R}$. This formula expresses that the likelihood of getting some specific output for two similar inputs must be smaller than the term e^{ε} to achieve ε-differential privacy. The value of the ε parameter differs per application, but generally it is considered that $\varepsilon \leq 1$ is a strong guarantee (Abadi et al., 2016; Fredrikson et al., 2014).

There are various algorithms that implement differential privacy for machine learning models. One that is often used is that by Abadi et al. (2016), which takes into account an extra parameter δ added to the term. This parameter is then used to define (ε, δ)-DP, where δ stands for the probability that normal ε-DP is broken. The algorithm works by clipping and adding small amounts of noise to the gradients in each step of the training process.

Another type of DP is f-differential privacy, which can be used to find the upper bounds of privacy leakages. This can be used as a tool for hypothesis testing (Dong et al., 2019). It is described as follows:

$$T(M(S), M(S')) \geq f$$

Stochastic Gradient Descent (SGD), the standard optimization technique for training DNNs, has been modified in a couple of ways so it can ensure more privacy. For example, there is Distributed Selective SGD, where sharing of the gradients is done by adding noise to ensure privacy (Shokri and Shmatikov, 2015). There is also the Private SGD (DP SGD) algorithm for GANs (Abadi et al., 2016), which uses norm clipping in combination with noise addition.

3 Membership Inference

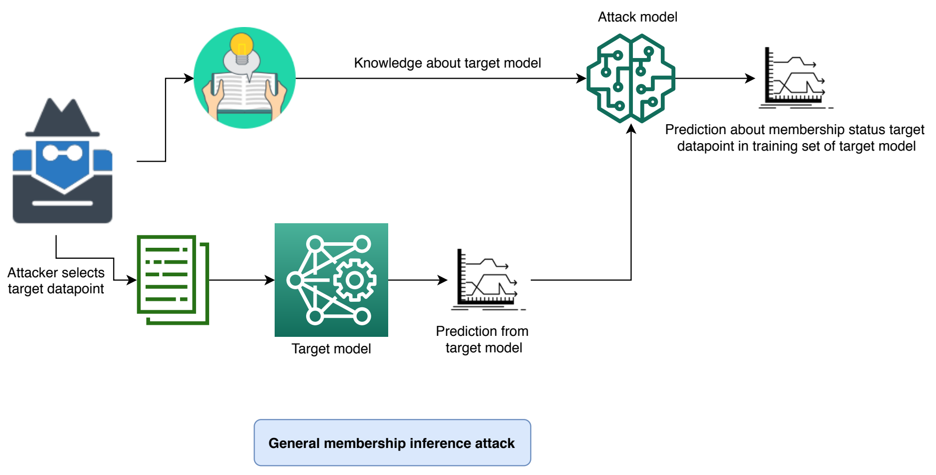
Attack model
Knowledge about target model
Prediction about membership status target datapoint in training set of target model
Attacker selects target datapoint
Target model
Prediction from target model
General membership inference attack

Figure 1: General workflow of a membership inference attack. Knowledge of the attacker depends on the setting (black-box vs white-box). The structure of the attack model varies greatly, depending on the target model type and dataset.

Membership inference attacks can achieve the goal of verifying if a certain datapoint was used to train the machine learning model in question. An unethical party can decide to try and see if a datapoint was used to train a model in a problem setting where privacy is an important aspect. Think of a machine learning model that is used to predict whether or not a patient has symptoms that point towards cancer. A person being a member of this training set could point towards this person having cancer, meaning an invasion of privacy. On a more ethical note, a membership inference attack could also be executed to help confirm whether a datapoint was used to see if the model was trained on data that it was not allowed to use (Pyrgelis et al., 2018). For example when auditing a model to see if it is compliant with GDPR (Song and Shmatikov, 2019).

Similar types of attacks have also been achieved on non machine learning models, where single datapoints have been confirmed to be a part of aggregate data (Calandrino et al., 2011).

Moreover, membership inference attacks can serve as a way to quantify how much models overfit (Hayes et al., 2019) and thus as a measurement tool for safety of releasing MLaaS models to the public.

Membership inference attacks also allow us to learn more about what overfitting means in generative models (Hayes et al., 2019), meaning that it can give new insights to machine learning as a whole.

3.1 Prerequisites

In the current research domain of membership inference attacks on machine learning models there are varying assumptions on the knowledge of the attacker, the attack time, the type of ML model, or even the type of dataset the model was trained on. In the following section we discuss the current range of these parameters.

As was shortly discussed in section 2.2, there are different assumptions that can be made about the knowledge of the attacker possesses. It is a scale from only black-box knowledge towards full white-box knowledge. A strict black box attack would give the attacker only information about the output of the model when he enters an input, as is the case in many MLaaS applications.

In a white box attack the attacker has access at least some of the following: the hyperparameters of the victim model, the type of the model, a subset of samples that belong to the training set, the distribution of the training data, the size of the training set, the training algorithm of the model, or some samples from the training set. There is a substantial body of work that exists somewhere in between strictly black box or fully white box. It is sometimes assumed that in a black-box attack the attacker has access to at least some of the training samples, or the distribution of the training data (for example in Shokri et al. (2017)).

Another relevant factor in attack knowledge is the amount of queries an attacker can afford before risking being found out or running out of money. In MLaaS models, it is probably the case that some alarm bells will go off if a single user submits 100s of queries to a single model in a short time, or it could be pay per query (Hayes et al., 2019).

There are different stages in which a ML model can be targeted by a membership inference attack. A membership inference attack can happen during both training and inference time. If the attack happens during training time, it is in a federated learning setting where the attacker is either one of the training parties or someone who acts as the central update server (Nasr et al., 2019).

The type of model that a membership inference attack targets is of great influence on how that attack can look. Attacks on discriminative models often take into account the confidence score that a model gives in combination with its prediction (Chen et al., 2019). Generative models, however, do not output this kind of confidence score. This complicates the problem and a new approach is needed for this type of model. GANs as well as VAEs have been researched by Hayes et al. (2019) and Chen et al. (2019). Chen et al. (2019) propose a taxonomy to order the types of attack on GANs on the access level that is assumed, ranging from black-box generator, to partial black-box generator, to white-box generator, and finally to accessible discriminator.

The assumption that Shokri et al. (2017) make is that similar models trained on similar data records using the same service will behave similarly. The type of dataset and its distribution influence the success of a membership inference attack. So in addition to an attacker being privy to this information, the information itself is also of importance. Models trained on high dimensional datasets (e.g. on images) are different to attack than models trained on lower dimensional datasets, and models that are trained on bodies of text are different still. Many papers solve this by testing their attack on multiple types of dataset (Chen et al. (2019), Yeom et al. (2018), Salem et al. (2019)).

It is often assumed that what makes models vulnerable to membership inference attacks is decided by whether or not that model overfits and how much (Yeom et al., 2018). However, successful membership attacks are not only determined by whether the target model overfits, but are also influenced by whether or not target attributes have significant influence on the output of a model (Yeom et al., 2018). Even if overfitting looks different in some types of models, it can still play a role in susceptibility to membership inference (Song and Shmatikov, 2019).

3.2 Methods

To perform a membership inference attack, many papers utilise a similar model structure as their target model to achieve their goal. For example, using generative models to attack generative models (Chen et al. (2019), Hayes et al. (2019)). Many papers choose to use a variety of experiments on a variety of datasets. An overview of all papers and their different methods can be found in table 1.

3.2.1 Attacking discriminative models

A very influential paper treating the subject of membership inference attacks was by Shokri et al. (2017). Shokri et al. (2017) made use of ‘shadow models’, as many as there are classes that the target model can output. These are trained on some dataset and return ‘in’ or ‘out’ corresponding to whether or not they are in the training set. These shadow models are then used to train the attack model that learns to distinguish whether an input was ‘in’ or ‘out’ the training set. This model gets fed a correctly labelled input pair together with the prediction vector of one of the shadow models. They find they can stage a successful membership attack, even with minimal levels of assumptions about the model. With SocInf, Liu et al. (2019) use a similar approach as Shokri et al., however they use only one ‘mimic’ model and make use of a GAN to train it. They are successful in staging their attacks on supervised models in combination with social media datasets.

Authors	Model Type	Assumption level	Centralized/Federated
Chen et al. (2019)	Generative	All	Centralized
Farokhi and Kaafar (2020)	Discriminative	White box	Centralized
Hilprecht et al. (2019)	Generative	Black box	Centralized
Hayes et al. (2019)	Generative	Black and White box	Centralized
Hisamoto et al. (2020)	Sequence	Grey box	Centralized
Jayaraman et al. (2020)	Discriminative	Black box	Centralized
Liu et al. (2019)	Discriminative	Black box	Centralized
Melis et al. (2019)	Discriminative	Grey box	Federated
Nasr et al. (2019)	Discriminative	White box	Both
Long et al. (2018)	Discriminative	Black box	Centralized
Pyrgelis et al. (2018)	Discriminative	White box	Centralized
Rahman et al. (2018)	Discriminative	White box	Centralized
Sablayrolles et al. (2019)	Discriminative	White and Black box	Centralized
Salem et al. (2019)	Discriminative	Black box	Centralized
Shokri et al. (2017)	Discriminative	Grey box	Centralized
Shokri et al. (2019)	Discriminative	Black box	Centralized
Song and Shmatikov (2019)	Sequence	Black box	Centralized
Truex et al. (2019)	Discriminative	Black box	Centralized
Yeom et al. (2018)	Discriminative	Black box	Centralized

Table 1: Global overview of all reviewed membership inference papers. Generative or discriminative refers to the target model's structure, and more specifically whether a shadow-model-type attack is possible without adjustments. Black, white, and grey box refer to assumption levels about the target model. Black is no knowledge, white is a lot of knowledge, and grey is somewhere between. Grey is often used if the authors leave their assumption level unspecified. Centralized and federated refer to the way the target model is trained.

Whenever a shadow model or similar needs to be trained, a suitable dataset to train it on has to be found or created. There are multiple approaches used to generate an appropriate dataset. Shokri et al. (2017) propose to use either model-based synthesis, which uses the target model in combination with a hill climbing algorithm, or statistics-based synthesis where feature values are sampled from their marginal distribution to synthesize a training set. Another way to synthesize data is to query the target model repeatedly with possible inputs and use the model output to classify the generated input to create a new dataset (Liu et al., 2019).

In continuation of Shokri et al. (2017), Truex et al. (2019) try to gain insight in the specific factors that make a membership inference successful or not. They use a multitude of models and datasets with an empirical characterization. They try different combinations of attack and victim models and try to measure impact of different kinds of attacker knowledge by investigating the effect of accurate target data and accurate shadow data. Truex et al. (2019) find that in their broader attack setting membership inference attacks can still be successful.

Extending the research of Shokri et al. (2017) by relaxing their two main assumptions Salem et al. (2019) perform a successful membership attack. The two main assumptions made by Shokri et al. (2017) are the need for multiple shadow models as well as the need for the distribution of the training set for the shadow models and the target model to be the same. Salem et al. (2019) show that broader range of attack scenarios can still be successful.

Pyrgelis et al. (2018) propose a new kind of attack on discriminative models. They propose an attack that is meant to distinguish whether or not a datapoint was part of aggregate time-series location data. They propose a few white box settings with differing levels of knowledge and model their attack as a distinguishability game where they train a classifier to decide whether or not a user was part of the training data. In one of their attack settings, a subset of locations of the target are known, and in the other setting the attacker has information about a subset of groups that the target was included in. They train their attack model on a balanced dataset of labeled aggregates over user groups that include or exclude the user of which they want to determine membership status (Pyrgelis et al., 2018).

Sablayrolles et al. (2019) try to find the optimal strategies for membership inference attacks with the help of a probabilistic framework. The optimal attack is not tractable, but can be approximated by relying on the loss of the target model and not the parameters of the target model. With this, they show that a black box attack is as good as a white box attack (Sablayrolles et al., 2019).

Furthering the research into what makes membership inference attacks feasible, Long et al. (2018) show that overfitting is a sufficient condition for this. However, they also find that it is not necessary when a target datapoint has unique influence on the model. They also note that a truly black box attack on a well generalised model has not yet been shown to be successful. To stage their attack, they first build 'reference' models to mimic the target model. Then they select records that should be vulnerable to the membership inference attack. Then they query the target model and run a hypothesis test to decide whether its classification is in line with what the reference models would predict (Long et al., 2018).

Nasr et al. (2019) design multiple attacks to gain more understanding of the privacy preserving capabilities of deep learning models. They design passive and active white-box inference attacks against both centralized and federated learning. They stage their attack by exploiting the SGD optimization algorithm. Nasr et al. (2019) show that models that do not overfit are still susceptible to a membership attack.

Taking a more formal approach are Yeom et al. (2018) by first introducing definitions for membership inference attacks where they replace the training algorithm by a malicious one. They implement this attack under varying knowledge assumptions. They test their proposed attacks on a variety of models and datasets. Yeom et al. (2018) find that models that overfit are the most vulnerable to their attacks, although influence of a target attribute also plays a role.

Looking intro what influences a models' privacy leakage through membership inference, Farokhi and Kaafar (2020) take a theoretical approach with some empirical experiments where they assume white box access to the target model. They use mutual information and Kullback-Leibler divergence to measure information leakage from membership inference. They find that membership information leakage decreases when the size of the training set that is used to train the target model increases.

Contributing to more insight in privacy preserving models and membership inference attacks, Rahman et al. (2018) apply a white box attack on a differential private classification model, with the basic attack method of Shokri et al. (2017). They mention the trade-off between keeping models private and preserving utility.

Looking further into the possibilities of membership inference attacks on differentially private models and non-private models, Jayaraman et al. (2020) show that an attack on such a model can still be successful even with skewed prior probabilities, which is a realistic setting. They consider the case where only a small fraction of the datapoints that are targeted actually belong to the training dataset or the target model, they assume that a training example will be near a local minimum in the loss function. They also consider dynamic inference thresholds that are selected by the attacker. Jayaraman et al. (2020) propose to use the positive predictive value together with the f-differential privacy notion to better measure their imbalanced setting. Only when the privacy loss budget is very high, are the differentially private models vulnerable to their attack (Jayaraman et al., 2020).

The vulnerability of membership inference during federated learning is researched by Melis et al. (2019). Their key observation is that ML models memorize latent features independent of the task being learned. They use an active adversary that can use multi-task learning and they stage an attack against in a federated learning setting with model averaging and are still successful in their goal of property inference for a subset of the class (Melis et al., 2019). They do this by having a participant in the federated learning process be the adversary and he tries to steal training data from another participant. They use a varying number of participants in their experiments and note that the more participants take part in collaborative learning, the less successful attacks become.

The need for transparency and privacy is a key element of trustworthy machine learning (Shokri et al., 2019). However, model explanations can be used to undermine the privacy preserving qualities of said model (Shokri et al., 2019). Shokri et al. (2019) show this by investigating three types of model explanations, namely backpropagation-based, perturbation-based and example-based. For the backpropagation-based and example-based explanations they find that the privacy of the model is not preserved.

3.2.2 Attacking sequence models

Song and Shmatikov (2019) use membership inference to audit ML models and focus on deep learning text generation models. They assume a black box setting with knowledge of the training algorithm of the target model and assume the target model does not output any confidence values or probabilities, they use well-generalized models. An approach with shadow models similar to Shokri et al. (2017) is used. Song and Shmatikov (2019) find that overfitting in RNNs manifest by shifting probabilities in samples that are not often seen, which is a different manifestation of overfitting than what 'usually' happens in ML models. They show that their membership inference attempts are more successful when they include rarer words.

To deepen the understanding about which models are sensitive to membership inference attacks, Hisamoto et al. (2020) research an attack on a sequence-to-sequence model. Carlini et al. (2019) showed that generative sequence models are sometimes sensitive to remembering specific sequences, meaning they could be sensitive to a membership inference attack. They use a machine translation model in combination with the shadow model method from Shokri et al. (2017). They also try some more advanced tricks like trying slight variations of the same input sentence and analyzing the change. Hisamoto et al. (2020) were not yet very successful but their best results were better than random guessing.

3.2.3 Attacking generative models

To attack generative models requires a different approach than when attacking discriminative models. Hayes et al. (2019) do this by leveraging GANs themselves, and training a GAN on data that was generated by their target model. They device a black box setting and a white box setting for their attacks (in the black box setting the adversary is aware of how big the training set of the target model was). The method for the white box setting requires access to the discriminator of the GAN, thus restricting the attack to GANs, this attack once again leverages the confidence values that the discriminator puts out. Their black box attack is also suitable for VAEs and requires a dataset with datapoints that were probably included in the training set of the target model. Hayes et al. (2019) also introduce two sort of grey box attacks, a discriminative one and a generative one. In the discriminative one they leverage overfitting of the target to train a discriminative model to distinct whether or not a sample was in the training set. The generative setting is similar, but they train a GAN to distinguish between training and non-training data.

Chen et al. (2019) propose a generic attack model for their attacks on generative models. They use a Bayesian perspective where they aim to compute the probability of whether or not a sample was a member of the training set. Calculating this is intractable, so they use a kernel density estimationParzen (1962) to approximate it (Chen et al., 2019).

Hilprecht et al. (2019) propose two attacks on generative models, the first attack is based on Monte Carlo integration and applicable for all types of generative models, the second attack is a so-called reconstruction attack that can be used on VAEs (Hilprecht et al., 2019). The Monte Carlo attack is based on their intuition that the generator part of a generative model overfits when the output is close to its training set. They use an appropriate distance measure and heuristic.

3.3 Defenses

Overfitting is one of the most important factors in determining the susceptibility of a model for a membership inference attack (Shokri et al. (2017), Yeom et al. (2018), Liu et al. (2019)). Therefore defenses for a membership attack can consist of making sure your model does not overfit and is trained on a representative training set. Notably, overfitting is not the only aspect of a model that makes it vulnerable in terms of privacy leakage (Shokri et al., 2017). There is also the influence that individual datapoints have on the model. As mentioned by Long et al. (2018), a technique where some samples are excluded from training because of their influence could be of importance. An aspect that is prominent in many of the current body of work on membership inference is the trade-of between utility and privacy. Where a mitigation tactic reduces the accuracy of the model, making it no longer useful in a competitive real-world setting (Rahman et al., 2018).

Other proposed defenses are limiting ow much information the model puts out, by clipping the prediction vector to only output the probabilities of the top k classes (Shokri et al., 2017). Similarly, a defense where the probabilities of the prediction vector of discriminative models are coarsened by decreasing their precision (Shokri et al. (2017), Liu et al. (2019)). Shokri et al. (2017) propose to use a softmax function with temperature, to increase the entropy of the output, which decreases information leakage. Finally, they propose to use regularization to reduce overfitting of the model (Shokri et al., 2017). However, only regularization seems to help significantly against information leakage of the model. (Truex et al., 2019) propose model hardening by model choice, fit control, regularization, and anonymization. They mention the privacy-utility trade-off.

(Salem et al., 2019) mention two main defenses. Just like Melis et al. (2019) they mention dropout, finding that there is a trade-off between utility and privacy, except for when dropout decreases overfitting in the target model. In the case a target model might not be a neural network, they propose model stacking.

Pyrgelis et al. (2018) employ differential privacy to research the trade-off between utility and privacy of their model. They find that differentially private training ensures privacy in some cases, however not in all. They find that an adversary that tries to copy the target model and trains on noisy aggregate data to be partly immune to differentially private mechanisms (Pyrgelis et al., 2018). Moreover, Jayaraman et al. (2020) find that differentially private models can still be vulnerable to membership inference attacks.

Defenses against membership attacks in collaborative learning are similar to those in the centralized setting. Melis et al. (2019) propose to share fewer gradients between participants, a dimensionality reduction of the predictions, which could hurt performance, and they propose dropout (Srivastava et al., 2014) to avoid overfitting. Finally, Melis et al. (2019) use participant-level differential privacy, however they find it hard to still obtain a useful model when enough noise is added to preserve the privacy of the training set. Melis et al. (2019) mention that more research is needed for an effective defense.

(Chen et al., 2019) implement differential private DP SGD algorithm for GANs Abadi et al. (2016). They find that using DP SGD causes slower training, higher computation costs, and worse generation outputs from the model.

In generative models, weight normalization, dropout and differential privacy are also viable options for a defense mechanism (Hayes et al., 2019). Hayes et al. (2019) also propose using their attack as a defense mechanism, by checking vulnerability to a possible attack with a mock-attack. Or alternatively to use the inference attack as a stopping measure for when to halt training, when attack accuracy is still low but model utility is already high enough.

Differential privacy seems to be in the forefront of defenses against the membership inference attack. Next to the mention of the inherent privacy-utility trade-off.

4 Model inversion

A model inversion attack goes a step further than membership inference: beyond testing for membership, this attack can recover new information about the training data, optimally even the entire training set. The name *model inversion* is really an umbrella term for multiple types of attacks, and in the literature, there are various terms used, sometimes interchangeably, for this subject. *Data reconstruction* or *extraction* usually refers to the direct reconstruction of data points from the training set, possibly even the entirety of it. *Attribute inference* is sometimes used for the case in which only one or a few of several attributes of training data are recovered. *Property inference* is also used in this context; however, this term more commonly means that some statistical properties of the training set are extracted that are not necessarily contained directly in the data points themselves.

We will refer to all these types of attacks as model inversion attacks throughout the rest of this section.

Figure 2 shows a general workflow for model inversion. Note that is not a complete model, as there are some exceptions and a lot of detail is omitted. These aspects will be discussed in section 4.2.

4.1 Prerequisites

As described in section 2.2, attacks can be white or black box, depending on whether knowledge about the model internals is required to execute the attack. This is a good starting point for discussing the assumptions made and the access needed prior to starting an attack. Table 2 below shows for

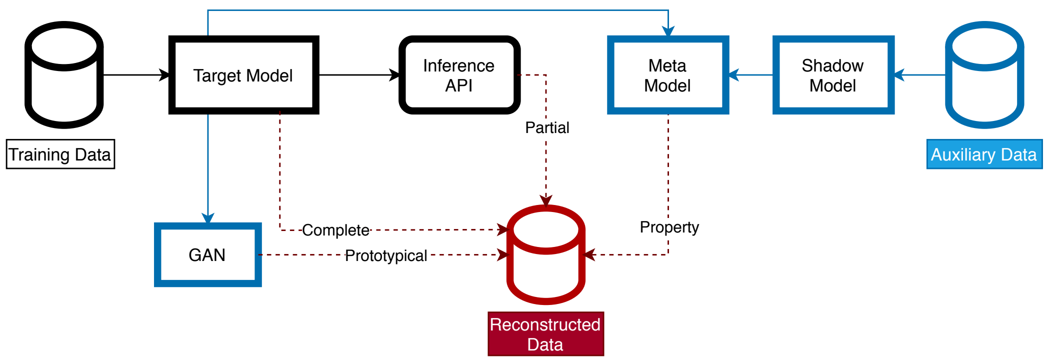

Figure 2: General workflow for multiple types of model inversion attack. Colored **black** for the target architecture, **blue** for the adversary architecture, and **red** for the result. Different components of the adversary architecture are needed for different types of attacks. In the context of collaborative learning, the target model and training data are those of one specific participant.

reference a rough classification of the literature on model inversion attacks in terms of the amount of necessary information.

In general, attacks done in a collaborative setting assume white box access. The participants usually share a common learning objective, which means that they know which model is being trained and which labels are declared. Model parameters or gradients are exchanged during the training phase. The adversary can have access to this training as either a participant (Hitaj et al., 2017; Melis et al., 2019) or a potential central server (Wang et al., 2019). In this last case, the adversary can even actively isolate a participant to make the attack more straightforward.
Some methods do not require knowledge of the common learning objective and achieve success only by using the gradients (Zhu et al., 2019; Zhao et al., 2020). Therefore, an adversary does not have to gain access by participating in the training but could instead intercept the gradients when they are exchanged, which can be easier to achieve in some cases.

There are also some white box attacks that use a centralized setting, which are of course done during the inference phase. A few of these exploit the parameters or other information explicitly (Gambs et al., 2012; Zhang et al., 2020). Others use the information about model internals to create *shadow models* that have the same functionality (Ateniese et al., 2015; Ganju et al., 2018).

Not all white box attacks need an exact copy of the target model to work. For attacks that use shadow models, it could be enough to have an approximation of the model, which can be acquired by performing a model extraction attack. In fact, some authors even explicitly state which model extraction attack could be used as a preparation for their model inversion (e.g. Ganju et al. (2018); Salem et al. (2019)).

Shadow models can also be used in black box attacks. These methods are weaker and usually do not get results quite as good as their white box counterparts. Nonetheless, some attacks work fine when using just the same type of model if that information is known, or even an arbitrary model that can be trained to have the same functionality as the target (He et al., 2020). This is very useful, as a white box attack is often not feasible in real-world situations.
However, using shadow models does mean—in black as well as white box attacks—that an auxiliary dataset to train these models on is essential. For some methods it is acceptable to use an arbitrary dataset, but for others the data needs to come from the same domain (Ateniese et al., 2015) or even the same distribution (Salem et al., 2019) as the data that is being extracted. Property inference attacks with this method also require the data to have samples both with and without the property that is being inferred (Ateniese et al., 2015; Ganju et al., 2018; Melis et al., 2019). How to create or obtain these auxiliary datasets, apart from simply grabbing suitable data from public repositories, is not discussed.

Other black box attacks have less stringent information needs. Knowing the distribution of some or all of the attributes could be important (Fredrikson et al., 2014, 2015), but not always necessary (Shokri et al., 2019; He et al., 2020). For the attribute inference attacks by Fredrikson et al. (2014); Yeom et al. (2018), the values of some non-sensitive attributes are also required, which might not be feasible depending on the context of the attack. Although if it *is* feasible, many works have a way to use this extra information to improve their results. The attack by Yeom et al. (2018) also works with access to a membership oracle instead of to the model, which is interesting even if it will in most cases be simpler to use the model itself.

For Carlini et al. (2019) and Wilkinson and Legg (2020), who perform attacks on text prediction models, it is essential to know the pattern or format of the data that is targeted. However, although this depends on the target and other situational details, such a format could be as simple as "My social security number is [. . .]".

A few other attacks can be done using only query access to the target model, although auxiliary information will improve the results (Al-Rubaie and Chang, 2016; Aïvodji et al., 2019; He et al., 2020).

Lastly, an interesting new avenue is the use of model explanations (Shokri et al., 2019). Especially example-based explanations are relevant for model inversion, although different types of explanations can be used for other attacks.

Black Box			**White Box**	
nothing	**Auxiliary Dataset**	**Attribute Distributions**	***nothing***	**Auxiliary Dataset**
Al-Rubaie and Chang (2016)	Salem et al. (2019)	Shokri et al. (2019)	Gambs et al. (2012)	Zhang et al. (2020)
Aïvodji et al. (2019)		Fredrikson et al. (2014)	Zhu et al. (2019)	Ateniese et al. (2015)
Yeom et al. (2018)		Fredrikson et al. (2015)	Zhao et al. (2020)	Ganju et al. (2018)
Carlini et al. (2019)			Hitaj et al. (2017)	Melis et al. (2019)
Wilkinson and Legg (2020)			Wang et al. (2019)	
He et al. (2020)				

Table 2: Classification of model inversion attacks by level of prerequisites. They are first split by access level to the model, then split by type of auxiliary information needed. If there are multiple levels of prerequisites covered in a work, it is categorized by the minimum required level.

4.2 Methods

As we established at the start of this section, there are various different terms used in the literature to refer to the different model inversion attacks. These terms commonly indicate the type of result that is obtained, and roughly map to the different methods used for the attacks as pictured in figure 2, although not all methods match this diagram completely. A classification is made of the attacks based on these terms, which is shown in table 3 and will also be used here. We first make a distinction between the methods extracting properties and those constructing data samples. This last category is further split into methods that generate new samples, methods that extract partial samples, and methods that extract complete samples.

Property Inference

Some methods aim to infer information about the dataset that is not directly an attribute or class label, but rather a statistical property. The environment that the data was produced in or a class that it belongs to unrelated to the learning task are examples of such properties. These can be extracted from the global dataset or from a single user or class.
As discussed in the previous section, there are multiple attacks that use shadow models with the same functionality as the target model. These are mostly employed in the context of property inference. The method by Ateniese et al. (2015) uses multiple such shadow models, trained on data that is similar to the target data but partitioned on some statistical property. Property inference is then performed by training a meta-classifier on these shadow models. This meta-classifier will be able to classify a target model on the statistical property that distinguishes the data partition used to train it. The precision and recall of this method as tested by Ateniese et al. (2015) are around 0.95, although

they only test relatively simple models like HMM and SVM.
Ganju et al. (2018) extends this attack by adding support for fully connected NNs. This is not very practical, as most NNs are not completely connected, but it is a step towards attacks that work on general NNs. Their precision and recall scores are also good, being all above 0.85. However, both of these attacks do inference only on binary properties, and it is unclear how well they would work on multi-class properties.
Melis et al. (2019) uses a similar method with a meta-classifier in the context of collaborative learning, where its input consists of the gradients that are exchanged during an update. This allows for property inference on the iteration level, as the meta-classification is done per batch. Melis et al. (2019) also test aggregating gradients across more participants, and reducing the number of samples in the batch that have the property. Although the performance goes down in these more difficult conditions, generally it performs as well as the centralized attacks.

Prototypical Reconstruction

Another type of attack uses Generative Adversarial Networks (GANs) to construct data points representative of some class or, in a collaborative setting, of some participant. These attacks produce non-existing data that is *prototypical* for that class, but not a direct reconstruction. Thus, a limitation of this method is that the samples of the targeted class should be similar enough for the generated prototype to be coherent.
Hitaj et al. (2017) perform a GAN attack in a collaborative setting. The adversary is a participant in the learning process, and alongside their local training model they use a GAN to generate data belonging to some class A that the target participant has training data of. The adversary can, as a participant, actively influence the target to release more data of that class by using their generated data as some other class B, which means that the model is less able to differentiate data of class A. However, this influence could cause the global model utility to decrease.
The results are only evaluated visually and therefore difficult to compare, but it is clear that the generated data is of lower quality than the training set.
Wang et al. (2019) apply the same method but have the adversary control the central server instead, which gets more accurate results without having to influence the model utility. However, in a real-world setting it is less likely that the adversary has access to this server than to one of the participants. The results are compared with attacks like those of Hitaj et al. (2017) and Fredrikson et al. (2015), and it does generate better results, visually as well as using a quality measurement called *inception score* (see Salimans et al. (2016)).
Aïvodji et al. (2019) apply the method in a centralized setting. Interestingly, this is the only paper on model inversion attacks that mentions a query budget, although they set it at over a million queries and it is never reached. The results are visually evaluated using a survey with 13 participants, but no mention is made about the background of the participants or the setting of the survey. Most of the results are unrecognizable, scoring at best 55% correct guesses in the survey.
Zhang et al. (2020) adds a public knowledge distillation phase that involves training the GAN on a similar or partial dataset to improve its generation of realistic data. They introduce various evaluation metrics that measure different aspects of the result quality. These metrics are useful in comparing the different attack settings, although they do not give a readable quality estimate.
Salem et al. (2019) attacks a model in an online learning scenario, which is constantly updated with small batches. A shadow model using online learning is used to train the GAN, which can then be applied to the target model. The input for this training is the posterior difference of the online learning model. This is the difference between the model outputs before and after an update. To get the best possible quality result, a large amount of samples are generated and then clustered. Based on this clustering the optimal samples can be chosen. The results are good on the MNIST image set, where all images have a black background with a white digit in the middle, but other more heterogeneous datasets result in fuzzy and unrecognizable images.
All GAN-based attacks discussed here are applied on image data, often even using the same datasets like MNIST. Nonetheless, the attacks cannot be fully compared, as there is no standard metric used for the evaluation, and some of the papers have no reproducible way of evaluation at all.

Partial Reconstruction

An attack aiming for *partial* reconstruction of the training data, also called *attribute inference*, targets only one or a few sensitive attributes from the data. Fredrikson et al. (2014, 2015) were the first to

define such an attack, called the MAP (Maximum A Posteriori) estimator. Given some non-sensitive attributes of a data point and query access to the target model, it reconstructs the other attributes by repeatedly querying the model with possible candidate inputs. Candidates are generated using auxiliary information about possible values for the unknown attributes. Feeding them to the model returns confidence scores, and by maximizing these scores, the most likely values for the sensitive attributes can be found.
The original attack by Fredrikson et al. (2014) relies on knowledge of the output value to choose the most likely candidate. By using the confidence scores, Fredrikson et al. (2015) overcome this limitation. However, the method requires that the sensitive feature is drawn from a small and thus discrete set of values. Also important to note is that only using white box access do the results not contain false positives. A black box attack could perform, depending on the dataset, no better than random guessing, which is especially relevant for an assessment of privacy risks.
Yeom et al. (2018) formalize the MAP attack and introduce the notion of *attribute advantage*, defining the risk to the training data specifically, as compared to the general population. They show that overfitting by the target model is a factor in the effectiveness of the attack, as is the influence of the target attribute on the model output.

A related attack which is not shown in figure 2 is the membership oracle attack, created by Yeom et al. (2018). In this setting, the candidate inputs are given to a membership inference oracle instead of to the target model directly. The membership oracle tests it for membership in the training set. It should then yield 'true' when the right sensitive attribute value is found, because only in that case will the complete candidate input be a correct member of the training set. This attack is theoretically interesting, as it shows that attribute inference is 'harder' than membership inference, but in practice, it is less feasible than the original MAP attack.

Lastly, He et al. (2020) apply various different methods to steal links from a graph dataset. Without any auxiliary information, they simply use distance metrics on the posteriors. When some attribute information or the partial training set is available, a MLP model is trained on features built from this information, again using distance metrics and some entropy measures. Lastly, using an auxiliary dataset, a shadow GNN model can be trained, and a MLP model can be trained in turn on features built from the auxiliary data (instead of from real data). These three methods can even be combined if multiple types of information are available. Most of the attacks outperform traditional link prediction, and especially attacks using the partial training set or shadow models achieve good results, with an AUC score around 0.95 depending on the auxiliary dataset.

Complete Reconstruction

This type of attack aims to extract complete data points from the training set, optimally recovering it entirely.

The first reconstruction attack was inspired by attacks on statistical databases. Gambs et al. (2012) walk through the structure of a decision tree model and collect the attribute values and leaf counts that the tree contains. These are then used to produce what they call a *probabilistic* version of the training dataset, which is the set or range of possible values that the data points in the set could have. It is a very simple attack that depends completely on white box access to the decision tree model type.

Complete reconstruction is also performed by Fredrikson et al. (2015), extracting images from a NN using the MAP estimator. The candidate inputs are generated by a gradient descent computation. Without access to the gradients, the attack could still be achieved by approximating them, but depending on the architecture of the target model this could take more than a month to execute. However, the reconstructed images are only approximations and do not reach the original quality. The images are evaluated by Mechanical Turk workers, with reproducible settings that are clearly described. All attack settings achieve over 55% correct guesses, with 80% for the least complex NN. Al-Rubaie and Chang (2016) apply a similar attack to extract gesture data. Instead of gradient descent, a randomization algorithm generates the candidate inputs. Their attack executes in 8 minutes without requiring white box access. The results are not evaluated on intrinsic similarity, but again the quality is suboptimal.

In a distributed setting, a different method by Zhu et al. (2019) exploits the gradients exchanged during the training process. These gradients are used to update 'dummy' inputs and labels, minimizing the distance between the 'dummy' gradient and the target gradient. Zhao et al. (2020) extends this method by first analytically extracting the labels, and this information in turn improves the reconstruction of the data points. The results from this attack, applied on image data, are very close to the original data, only differing in some artifact pixels. However, the larger the 'batch size' used for producing the gradient, the more time is required to produce this quality. Unfortunately only a number of iterations is given, with no mention of the time each iteration takes.

Carlini et al. (2019); Wilkinson and Legg (2020) attack generative text models, extracting sensitive words and phrases with a distinct pattern. Both use a type of shortest path search on the list of predictions combined with the pattern to search for. Carlini et al. (2019) test both word and character-level prediction on a variety of models and datasets. Additionally, they use multiple regularization methods to show that overfitting is not the cause of this success. The *exposure* or level of memorization of the pattern can be calculated, and this metric indicates whether the attack will succeed. It does take multiple hours to finish the attack.
Wilkinson and Legg (2020) test the extraction of a single 16-character password, without using a surrounding context pattern. The attack has a success rate of 90%, which drops when the size of the text corpus is increased. However, it is uncertain whether the attack will work on passwords with a shorter or unknown length. It is also unclear whether this result will carry over to real-world settings, as the dataset and model type are not representative of this.

Lastly, Shokri et al. (2019) perform their attack based on model explanations. Specifically, they use example-based explanations, which given a data point return the k points in the training set that were the most influential in deciding its label. This means that by traversing this 'influence graph' in an effective way, most or all of the training set can be recovered. A shadow model, trained on a similar dataset, results in a shadow 'influence graph' that can help determine a more effective query strategy.

Property Inference	**Data Reconstruction**		
	Prototypical	**Partial**	**Complete**
Ateniese et al. (2015)	Hitaj et al. (2017)	Fredrikson et al. (2014)	Gambs et al. (2012)
Ganju et al. (2018)	Wang et al. (2019)	Fredrikson et al. (2015)	Al-Rubaie and Chang (2016)
Melis et al. (2019)	Aïvodji et al. (2019)	Yeom et al. (2018)	Carlini et al. (2019)
	Salem et al. (2019)	He et al. (2020)	Wilkinson and Legg (2020)
	Zhang et al. (2020)		Zhu et al. (2019)
			Zhao et al. (2020)
			Shokri et al. (2019)

Table 3: Classification of model inversion attacks by type of result. **Property Inference** extracts statistical properties of the dataset; **Prototypical** reconstruction generates new data samples representative of the set; **Partial** reconstruction extracts some sensitive attributes of the data samples; **Complete** reconstruction extracts complete data samples.

4.3 Defenses

Depending on the exact method and the prior assumptions made, it is possible to defend against model inversion attacks in various different ways. As described in section 2.3, differential privacy is used to protect single data points from membership inference. It can also be applied in some cases against data reconstruction, but in other cases and especially for property inference attacks this will likely have no effect. Other methods to defend these attacks are needed and indeed sometimes implemented in the literature. An added complication is the fact that the use of differential privacy, as well as some other defense methods that rely on adding noise to some element of the model or training process, could result in a decrease in utility. We will examine the various defenses that have been theorized or tested for the attacks discussed above.

Carlini et al. (2019) implement DP on a text prediction model using the algorithm by Abadi et al. (2016). This causes a slight utility loss, but does defend completely against the attack. Interestingly, even very high ε values which should offer no meaningful protection already defend well enough to make extraction impossible. The very similar attack by Wilkinson and Legg (2020) could theoretically also be mitigated by DP, although the authors have not tested this.

Fredrikson et al. (2014) implement DP on a pharmacogenetics model, which does defend against their MAP attack with low values of ε. However, they also tested its influence on the utility of the model by simulating clinical trials. The results from this test indicate that the utility needed to use the model in medical contexts, without increasing risk to the patients, can only be achieved with an

$\varepsilon \geq 20$, which is too high to be effective.
This does not mean, of course, that DP is ineffective for this model type. In a different situation, these measures could very well be a good solution. However, for medical and other contexts where a decrease in utility could mean an increase in risk, other defenses should be investigated.

On the attack of Zhu et al. (2019) and Zhao et al. (2020), which is based on gradient leakage in a collaborative setting, DP could theoretically be useful. Zhu et al. (2019) applies various distributions of noise to the gradients, and the results indicate that the attack is only stopped at a point where the noise seriously affects the utility of the model. However, they do not connect this to a DP framework, and it is possible that some implementation of DP would succeed without impeding utility too much. Zhu et al. (2019) does test some other defenses, and finds that compressing or encrypting the gradients successfully deters the attack; as well as increasing the batch size or image quality, which the attack in its current form cannot handle.

In other cases, even without considering utility, DP will not work at all. Property inference attacks fail to be prevented by DP measures, as Ateniese et al. (2015) show. As this type of attack extracts statistical properties that apply on the entire training set, protecting single samples in it will not have any effect on the property itself. Ganju et al. (2018), whose attack is based on this one, do test some other defenses, most of which are not generally feasible. They theorize without testing about one defense that could work against their method, which consists of encoding some extra arbitrary information into the model parameters of deep NNs. This would confuse the meta-classifier that they use but not affect the utility of the target model.
Using this type of attack in a collaborative learning context means that there are some possible defenses related to the logistics of that. Though most of the defenses Melis et al. (2019) test are insufficient, they do introduce an interesting notion: DP at the *participant-level*. Based on their experiments, models trained using this version of DP will only converge in settings with thousands of participants. Nonetheless, it could be valuable to explore this avenue, to possibly adapt it to a more practical setting.

Attacks that use GANs to reconstruct training data are also not averted by the usual DP methods. Hitaj et al. (2017) find that record-level DP is not sufficient, but do theorize that DP with coarser granularities – such as device, class, or participant level – would protect against their attack. Zhang et al. (2020) and Aïvodji et al. (2019) also test and reject DP, which solidifies the notion that DP will not have an effect against this attack. Though Aïvodji et al. (2019) only test one ε value, which would not have been enough to say something useful about DP.
This result is consistent with that on property inference attacks. The GAN-based attacks do not reconstruct single samples, but rather class or participant representatives.

It is clear that differential privacy can be used to defend against attacks aiming to recover (attributes of) exact data points from the training set. This conforms to the definition of DP itself, which attempts to minimize the privacy risk to single records. Correspondingly, attacks that are aimed at statistical properties or class averages of the training data are not impeded by implementing DP.
Nevertheless, against these types of attacks other defense methods can be used, as multiple authors have shown. Some of these are very specific to one attack, such as for instance the encoding of arbitrary information in the model parameters protecting against property inference attacks. Others, though, do protect more generally: rounding or completely excluding confidence scores for instance will give an adversary less information to go on no matter what method is used.

For white-box attacks, where access to the model internals is needed, sometimes the best option is to shut off that access. Query access is usually possible, and in cases where the model needs to be downloaded, it should be encrypted. However, model extraction is of course a possibility, and often in the literature for model inversion, no distinction is made between legal white box access and model extraction done prior to the attack. This is not always a valid assumption, as we'll discuss in section 5, but approximations of the model might be enough to work with. Still, in some cases protection against model extraction could be sufficient also against these white box model inversion attacks.

Collaborative learning is often already partly privacy preserving. Participants do not have to share their training data but instead exchange model parameters or gradients. Using differential privacy with participant-level granularity could improve the protection substantially, as could operations that encode, compress, or otherwise protect the gradients themselves.
Differential privacy with other granularities could also be a solution for centralized models. Datasets

where participants contribute multiple samples exist in this setting as well, and perhaps defense against GAN attacks on a class-level is possible.

Lastly, there are many works in which no attention is paid to defenses. For the subject of model inversion, this is the case for as many as half the papers considered. Some of these mention defense as a topic for future research, and give suggestions for what type might work best, whereas others do not mention it at all. This is an oversight: to know how feasible it is to execute an attack in the real world, possible defenses are an important piece of information. Additionally, authors often have good intuitions for methods to deter their own attacks, and it would be useful to mention if not test these ideas.

5 Model Extraction

The goal of model extraction techniques is to fully recreate a model that is not yours. The general flow of model extraction techniques is depicted in Figure 3. To commence the model extraction attack, there is an adversary model that is usually pre-trained with similar data to the victim model. Then, the adversary queries samples to the victim models and trains itself on the returned labels. Sometimes, there is also some selection technique available that determines which set of samples is the most will provide the most effective results when querying the victim model. When the process is complete, the adversary will obtain a model that is ideally fully equivalent with the victim model. In the following sections, we will discuss the prerequisites for model extraction techniques (Section 5.1), discuss the methods used for various model types (Section 5.2) and which defense mechanisms are tested against the proposed model extraction attacks (Section 5.3). Table 4 gives a global overview of these three sections. Each section will contain a more in depth table specific for that theme. For the *Prerequisites* column, we have looked at the results of Table 5 and summarised the total prerequisites in three categories: Low, Medium and High. When a paper is classified as Low, the paper does not require any or much knowledge beforehand. When the paper is classified as Medium, it is comparable with the average of other papers. When the paper is classified as High, it requires more knowledge beforehand compared to the other the papers on average.

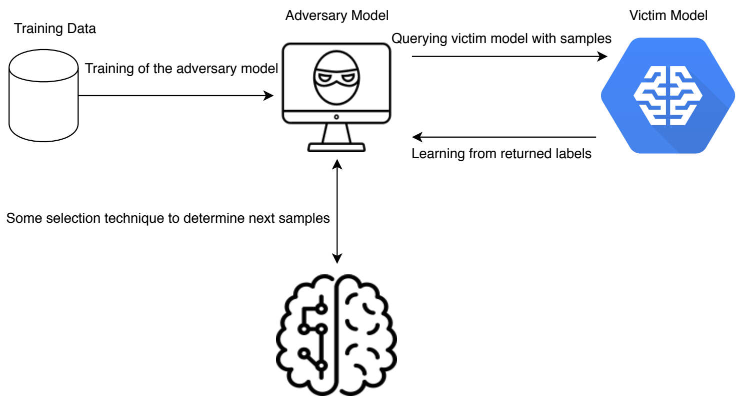
Training Data
Adversary Model
Victim Model
Training of the adversary model
Querying victim model with samples
Learning from returned labels
Some selection technique to determine next samples

Figure 3: General workflow for model extraction attacks. Adversary trains on some dataset and then queries the victim model with some selection technique to determine the samples. The labels of the victim model are learnt.

5.1 Prerequisites

As described in Section 2, there are almost always some prior assumptions when executing attacks. Therefore we will group papers with different assumption characteristics together in this section. We will focus on characteristics that are based on assumption of the papers. Firstly, there are assumptions about the models. Some experiments assume prior knowledge about the model type, such as a DNN or SVM and some papers even assume knowledge about the architecture of the victim model. Furthermore, assumptions are often made about the dataset that is used by the victim. We make a distinction here between knowledge of the dataset itself, so the distribution of the dataset for example, and knowledge about the problem domain, so images of traffic signs for example. Lastly, there can

Authors	Model Type	Attention To Defense	Prerequisites	Novelty
Shi et al. (2018)	All models	No	Medium	Active learning with query budget
Pal et al. (2019)	DNN	Yes	Medium	Extract DNN without domain knowledge with query budget
Yu et al. (2020)	DNN	Yes	High	Near-perfect performance on DNN extraction with significant less queries
Atli et al. (2020)	DNN	Yes	Medium	Reproducing and validating Knockoff nets and introducing new defense method
Pal et al. (2020)		Yes	Medium	Universal thief dataset from unannotated public data, evading PRADA.
Kariyappa et al. (2020)	DNN	Yes	High	MAZE, does not require any data but creates synthetic data
Mosafi et al. (2019)	DNN	Yes	Low	Generating composite images for model extraction, impervious to watermarking
Tramèr et al. (2016)	DNN, DT, LR, SVM	Yes	Medium	Simple, effective attacks with near-perfect fidelity
Duddu et al. (2018)	NN	Yes	High	Side channel leakage
Duddu and Rao (2019)	NN	Yes	Low	Side channel leakage
Jagielski et al. (2019)	NN	Yes	High	Fidelity as performance measure for model extraction
Oh et al. (2019)	NN	No	High	Exposing attributes of neural networks
Correia-Silva et al. (2018)	CNN	No	Low	Copycat network by using random non-labeled data
Orekondy et al. (2019)	CNN	Yes	Low	Creating Knockoff nets by querying random images
Roberts et al. (2019)	CNN	No	High	Framework for extracting model parameters by training a new model on random impulse response pairs
Krishna et al. (2019)	BERT	Yes	High	Model extraction in NLP
Wallace et al. (2020)	MT	Yes	Medium	Stealing machine translation (MT) systems
Chen et al. (2020)	DRL	Yes	Medium	Extracing DRL models
Reith et al. (2019)	SVM, SVR	No	Medium	Highly accurate SVM and SVR extraction with feasible and cheap attack
Chandrasekaran et al. (2018)	SVM, DT, RF	Yes	High	Formalizing model extraction and drawing parallels between model extraction and active learning
Shi et al. (2017)	NB, SVM	No	Medium	Inferring Naive Bayes and SVM with deep learning
Takemura et al. (2020)	RNN, LTSM	No	High	Infer RNN and LTSM
Wang and Gong (2018)	RR, LASSO, KRR, SVM, L1-LR, L2-LR	Yes	High	Stealing hyperparameters

Table 4: Global overview of all reviewed model extraction papers. Papers are categorized by: the model type that has been extracted (Section 5.2), if there was any attention to defense mechanisms (Section 5.3), how many prerequisites are needed (Section 5.1) and the novelty of the paper.

be assumptions about the amount of queries that are needed for the attacks. Some papers use queries as an basis of efficiency and some authors even impose query budgets on their experiments. We will make a distinction between acknowledging the amount of queries needed for the attack and enforce query budgets. The assumptions are summarised in Table 5.

Most paper discussed in this review have a prerequisite that the model type is known before commencing the attack. The only papers that do not assume this are from Shi et al. (2018, 2017); Pal et al. (2020); Reith et al. (2019); Wallace et al. (2020); Tramèr et al. (2016). Almost half of the papers discussed do not require any knowledge of the model architecture. We assume that not knowing the model type beforehand greatly complicates the model extraction technique. There is a big difference when attacking a RNN or DT for example. Therefore, having the model extracting attack work on multiple model types which are unknown beforehand requires an excellent structure of the attack.

Pal et al. (2019) is an good example where no domain knowledge is required of the model to launch the attack. The authors do know the victim's model type, specifically a DNN. However, the model structure and specifics is unknown to the attackers. Furthermore, Pal et al. (2019) executed their experiments with various query budgets: 10K, 15K, 20K, 25K and 30K respectively. Pal et al. (2019) implemented two model extraction techniques, one for image classification and one for text classification. They conclude that in general there is a substantial increase of agreement between two models when the query budget is increased, with 30K bringing the best results most of the time.

In the paper of Shi et al. (2018), the authors require no knowledge of the target's architecture and type beforehand. Furthermore, the distribution of the training data is also unknown to the attackers. The only information the attackers have is how to properly call the online API of the model. With regards to any query budget, the model explores two options. In the first option, they are limited in their amount of API calls allowed per day, thus operating on a tight query budget. This attack is combined with active learning methods, as described in Section 5.2.6. In the other attack the assumption of the query budget is relaxed and a large amount of API calls is permitted.

Shi et al. (2017) is partly similar to Shi et al. (2018), Shi et al. (2017) also do not require any knowledge of the victims model type or architecture before launching the attack. However, the problem domain and the dataset for training by the victim is known. First, the victim is trained on a part of the dataset and afterwards, the adversary tries to extract the victim's model with the remaining part of the dataset. There is some mention about the amount of queries that have been used to extract the model, since this is the remaining part of the training data set of the victim. Although, there is no real restriction of the amount of queries used, except for the size of the remaining dataset. Therefore, we conclude that there was no query budget imposed during the experiments.

Additionally, there is the paper of Pal et al. (2020). Pal et al. (2020) describe two different methods. In both methods, the training dataset is kept secret. With the first method, there is full knowledge of victim model architecture and with the second method, there is no knowledge about the victim model architecture. Similarly to Pal et al. (2019), Pal et al. (2020) have the same query budget restrictions for their attacks. The results are also similar, the higher the query budget, the higher the agreement between the victim and the stolen model. However, it is noted that *ActiveThief* performs similar when comparing between a query budget of 30K (30% of the dataset available) and no query budget (100% of the dataset available).

In the case of Atli et al. (2020), *Knockoff nets* are evaluated against other complex models. The experiments of the *Knockoff nets* are reproduced and the results are the same as the original paper, by Orekondy et al. (2019). Orekondy et al. (2019) assume no access to the distribution of the training data set of the victim, no knowledge of the model architecture and no knowledge of the problem domain. Furthermore, the experiments are allowed to collect unlimited API calls. Atli et al. (2020) iterate on this and conduct extra experiments, where access to the victim's dataset is assumed.

Chen et al. (2020) try to extract Deep Reinforcement Learning (DRL) models by using a Recurring Neural Network (RNN). Since the authors are focused on extracting DRL models, the authors obviously have knowledge of the model type beforehand. However, the authors do not require any information regarding the model structure. Further, the authors do not have any considerations about the amount of queries and therefore also do not have a query budget. Lastly, the authors do require information about the problem domain, but the authors do not have any access to the dataset that has been used by the victim.

Correia-Silva et al. (2018) have two different approaches regarding creating their fake dataset comprised of images that is used to steal the model. In their first approach they use random images that are non-labeled while the problem domain is unknown. In their second approach, the problem domain is known, and images from the same problem domain are used to create the dataset. Additionally, the structure of the victim model is not known to the attackers. Since both approaches have no query budget, the authors explore the costs creating the two fake datasets. The dataset where the problem domain is known is financially feasible. The first dataset was created for less than 10 dollars and the second dataset needed some more queries, therefore costing over 300 dollars. The authors conducted four experiments: facial expression classification, object classification, crosswalk classification and the Microsoft Azure Emotion API. The authors launched three attacks: dataset with problem domain, dataset without problem domain and a combination of the two.

Yu et al. (2020) have created *FeatureFool* that adopts internal representation for generating a subset of malicious samples. Yu et al. (2020) assume black-box access to the victim model, meaning that the attacker has no knowledge of the victim model architecture, hyperparameters, weights and training data. Yu et al. (2020) have implemented their model extraction technique based on image classification and have also experimented with the query budget. In their experiments, they created three different budgets: A, B and C. For their four datasets, Budget A ranged from 0.43K - 0.68K, Budget B ranged from 1K to 1.53K and Budget C ranged from 1.05K to 2.15K. The influence of the query budgets on the accuracy of the stolen models varies per experiment. In one of their experiments, the accuracy results from Budget A to Budget C increase up to almost 4 times. In other experiments, the results increase only up to 1.3 times.

Kariyappa et al. (2020) propose a method, *MAZE*, that does not require any dataset to extract the victim model, instead it "creates synthetic data using a generative model". *MAZE* also requires knowledge of the victim's architecture, which is similar to Yu et al. (2020). The authors also proposed an extension of the *MAZE* algorithm, *MAZE-PD*. This can be used when the attacker has small partial access to the victims training dataset. Kariyappa et al. (2020) have imposed query budget for their *MAZE* algorithm, albeit being relatively high. Their *MAZE* algorithm requires a query budget of 30M to reach an accuracy of 0.90. The reason for this high query budget is that *MAZE* does not require any data to be present for stealing the model, since the data is synthetically created. *MAZE-PD* has a query budget reduction between 2 and 24 times for the same accuracy as *MAZE*. Despite the high large query budgets, the algorithms still requires problem domain knowledge before the commence of the attack.

Duddu and Rao (2019) propose a theoretical framework for extracting machine learning models via side channel leakage. The authors assume that the target model type is a neural network (NN), whilst the architecture is unknown to the attackers, which is comparable to Pal et al. (2019); Chandrasekaran et al. (2018); Atli et al. (2020); Orekondy et al. (2019). Furthermore, the attacker has no knowledge of the data that the victim has used to train the model, similar to Orekondy et al. (2019); Pal et al. (2019) The attacker must rely on creating a synthetic dataset to train the adversary model, which is similar to Kariyappa et al. (2020). Additionally, since this is only a theoretical framework, and not tested in real world situations, there is no real consideration of a query budget. The authors declare that the minimal amount of queries is desirable. However, we feel that anyone can simply state this and this has no real weight.

Duddu et al. (2018) attempt to steal neural networks by exploiting timing side channels and effectively reconstructing the substitute model architecture. The attacks are proposed in the black-box setting, where the adversary has no knowledge of the victim's architecture and no knowledge of the training data used by the victim. However, Duddu et al. (2018) assume that the adversary has knowledge of the underlying data distribution from which the training data was sampled. Therefore, we decide to label this as having knowledge of the dataset in Table 5, similar to Shi et al. (2017, 2018); Pal et al. (2020); Yu et al. (2020); Reith et al. (2019); Chandrasekaran et al. (2018); Atli et al. (2020); Jagielski et al. (2019); Shi et al. (2017); Takemura et al. (2020); Roberts et al. (2019); Wang and Gong (2018). There if no further mention of the amount of queries that are necessary for the attack. The authors do note that they want to use the least amount of queries possible. However, similar to Duddu and Rao (2019), we don't see this as valuable information about the query budget, and therefore decide that the authors have no consideration for either the query budget and the amount of queries used.

Similarly, Jagielski et al. (2019) show the same characteristics for their attack. Of the victim models, it is known that it is a fully connected neural network with a single hidden layer and between 16 and 512 hidden units. Furthermore, the training set, MNIST and CIFAR-10, are also known before the attack is commenced. However, there is no mention of a query budget that has been imposed. The authors do describe how many queries are needed for the extraction of the ImageNet model, which is around 2^{17} queries.

In the experimenets of Takemura et al. (2020), the knowledge of the model type and architecture is known, Recurrent Neural Networks (RNN). Takemura et al. (2020) also have full knowledge of the training data that is used by the victim, similar to Jagielski et al. (2019). When executing the experiments, they first train the victim model with the MNIST or Air Quality dataset. The dataset is split into five equal parts, the first four parts are used for training the victim model and the last part is used for extracting the victim model by the adversary, which is similar to Shi et al. (2017). Despite having some limitations in the amount of data that is available to the authors, we do not recognize this as a imposed query budget. Since the amount of queries that can be done is restricted by how much data is left after training the victim model. If the authors would use two different datasets for training the victim and extracting the victim, we see no indications that the authors would limit the amount of queries executed by the attack. Therefore, we decide that the authors did not impose a query budget, which is comparable to the paper of Shi et al. (2017).

Similar to Yu et al. (2020); Jagielski et al. (2019); Kariyappa et al. (2020); Takemura et al. (2020), the model type and architecture of the victim is known before the attack is initiated. Furthermore, there is no imposed query budget, the authors assume that they can make unlimited API calls to extract the victims properties. The authors also don't specify how many queries are needed for the attack, with is comparable to the approach of Atli et al. (2020); Orekondy et al. (2019). Moreover, the victim's training datasets are also known to the adversary, and the adversary also has access to these datasets, MNIST and KMNIST. This problem domain is similar to the attacks of Jagielski et al. (2019); Takemura et al. (2020).

Reith et al. (2019) mainly explored attacks on SVMs and SVRs where the victim model was completely known, also known as a white-box model attack. They also describe one kernel agnostic algorithm, also considered a black-box attack.. Furthermore, the datasets that the victims have used for training are known, California Housing, Boston House Prices , UJIIndoorLoc respectively. While the authors did not explicitly impose a query budget for the attacks, the attacks executed with relatively low amount of queries. Most of their attacks completed with the amount of queries between 35 and 400, which does not only mean it is very cheap, but also very fast when applied to an Internet setting. The runtime of these queries range between 3 seconds and 4 minutes. Some of the other attacks completed with between 1700 and 5151 queries. This results in a runtime in the range of 3 minutes to 7 hours. The big increase of the amount of queries is mainly due to the drastic increase of features of the model. While the fast experiments had between 8 and 20 features, the slower experiments that between 100 and 520 features.

Wallace et al. (2020) describe four different methods for launching their attacks for extracting a machine translation system. Similar to Reith et al. (2019), in some of their methods, they assume knowledge of both the architecture, the hyperparameters and the source data. However, in other methods they use different data than the victim and having little to no knowledge about the structure of the model, having more similarities with Shi et al. (2018, 2017). Since the model extraction technique is specific for machine translation systems, the problem domain is always known to the attackers. With regards to the amount of queries for the attacks, the authors do provide the amount of queries that are necessary to extract the machine translation systems. However, there is no real indication of an imposed query budget for the experiments.

Chandrasekaran et al. (2018) has similar assumptions as Reith et al. (2019) with regards to datasets and the amount of queries. Chandrasekaran et al. (2018) have full knowledge of the dataset that has been used by the victim, and there is no real query budget. Nonetheless, the authors realise that having a low amount of queries is beneficial for speed and financial reasons. Therefore is the amount of queries measured as a form of efficiency. With regards to the models that are extracted, the authors have knowledge of the model type, but give no indication of knowledge on the architecture of the victim models. This is comparable with the approach of Pal et al. (2019) with regards to the knowledge of models.

Wang and Gong (2018) are stealing hyperparameters in machine learning. For their attacks, the authors assume knowledge about the training dataset, the ML algorithm that has been used by the victim and (optionally) the learnt parameters of the victim model. This would be motivated by the increasing MLaaS models from Amazon machine learning[1] and Microsoft Azure machine learning[2] for example. The authors do not consider a query budget for the attacks, but the attacks are naturally low in the amount of queries that are necessary for a accurate extraction.

Mosafi et al. (2019) present a novel method for generating composite images for attacking a mentor neural network using a student model. The authors do not require any information regarding model architecture or weights of the mentor's model, comparable with Chandrasekaran et al. (2018); Atli

[1]https://aws.amazon.com/cn/machine-learning

[2]https://azure.microsoft.com/sevices/machine-learning

et al. (2020); Orekondy et al. (2019); Duddu and Rao (2019); Chen et al. (2020). Furthermore, the authors have no expected knowledge of and access to the training data that has been used by the mentor. Moreover, the authors also do not require any information about the problem domain of the mentor, similar to Pal et al. (2019). This issue is solved by using a dataset that is large enough to comprise multiple image domains.

Tramèr et al. (2016) explore attacks against multiple model classes: Logistic Regression, Neural Networks (NN), Decision Trees (DT) and Support Vector Machines (SVM). The attacks are also executed in online cases studies, against BigML and Amazon Web Services. In their attacks, they use multiple methods for extracting the victim model. With some of their attacks, the model type and parameters are already known beforehand by reading the documentation of the MLaaS provider. In other attacks, the model type and parameters are not disclosed by the MLaaS provider, for example with Amazon. When this is the case, certain model characteristics may be narrowed down to a small range. Then, it is tested which characteristics of the models seem the most likely and afterwards this is assumed as correct when the errors are low enough. With regards to the training data of the victim model, the authors may have knowledge beforehand of the training data. However, this information is limited by what is provided by the ML API. For example, attacks against Amazon's system directly leak summary statistics about the training data, while other providers do not. Therefore, we decide that the authors can work with both situations. The authors also introduce a query budget of the form $a * k$, where k is the number of unknown model parameters and a is the budget scaling factor.

Krishna et al. (2019) have developed a method of extracting BERT-based API (Devlin et al. (2018)). For this method, the attackers have full knowledge of the model type and architecture, since BERT only comes in two sizes, *BERT-large* and *BERT-base*. Furthermore, the authors have regulated their experiments with query budgets. With regards to the dataset, the problem domain is already known, since the authors know what the BERT API consists of. Moreover, the authors note that the attacks can commence even without any real training data. More specifically, the attacker does not have to use grammatically or semantically correct queries. The attack can also be commences with random sequences of words coupled with task-specific heuristics form effective queries for model extraction on a diverse set of NLP tasks Krishna et al. (2019)

Oh et al. (2019) shows that multiple attributes of a neural network can be exposed from a sequence of queries. The attack starts with applying a supervised learning classification problem over applied over models. For this to work, the authors first collect a diverse set of white-box models that are similar to the target black-box model as their training set. Then, a new model, the meta-model is trained, to infer the model attributes. The this meta-model takes a model as input and returns the model attributes. In the paper, the model attributes is comprised of: architecture, optimisation process and the training data. Because of this method, we have decided that the authors do have prior knowledge of the model architecture and type. Since the training data of the models must contain white-box models that are similar. If the authors would have no knowledge of the target black-box type and architecture, they also would not know which white-box models to collect for the training data. The same reasoning holds for the training data set, since this attribute is included in the model attributes. With regards to the query budget, the authors do consider a query budget for their experiments.

To conclude this section, we have seen many papers that exhibit different attributes regarding the assumption level of the attacks and potential query budgets. We have categorized two different themes of assumption: assumptions about the target model and assumptions of the data. Moreover, we have also compared the papers on the use of their queries for extracting the desired model. We have seen some papers who require no prior knowledge of the models at all (Shi et al., 2017, 2018; Pal et al., 2020; Reith et al., 2019; Wallace et al., 2020; Tramèr et al., 2016). A limited amount of papers who do not require any knowledge of the data and the problem domain beforehand (Pal et al., 2019; Mosafi et al., 2019). A small amount of papers impose query budgets on the experiments (Pal et al., 2019; Shi et al., 2018; Yu et al., 2020; Kariyappa et al., 2020; Tramèr et al., 2016; Krishna et al., 2019; Oh et al., 2019), while almost all papers mention how many queries are necessary for the attack to be successful.

From this analysis, we can conclude that model extraction is more difficult when the type and the architecture is not known beforehand. Almost always, the model type itself is already known and in almost half of the papers, a grey-box model is assumed. Prior of the knowledge domain is almost always necessary for a successful extraction, while the dataset may remain secret in about half of the papers. Most of the papers have consideration of the amount of queries that are used, which is

Authors	**Model Type Knowledge**	**Model Arch Knowledge**	**Attention to # queries**	**Query Budget**	**Knowledge of Dataset**	**Knowledge of Domain**
Pal et al. (2019)	Yes	No	Yes	Yes	No	No
Shi et al. (2018)	No	No	Yes	Yes	Yes	No
Pal et al. (2020)	Yes & No	Yes & Yes	Yes	No	Yes	No
Correia-Silva et al. (2018)	Yes	No	Yes	No	No	Yes & No
Yu et al. (2020)	Yes	No	Yes	Yes	No	Yes
Reith et al. (2019)	Yes & No	Yes & No	Yes	No	Yes	Yes
Chandrasekaran et al. (2018)	Yes	No	Yes	No	Yes	Yes
Atli et al. (2020)	Yes	No	No	No	Yes	Yes
Orekondy et al. (2019)	Yes	No	No	No	No	Yes
Jagielski et al. (2019)	Yes	Yes	Yes	No	Yes	Yes
Shi et al. (2017)	No	No	Yes	No	Yes	Yes
Wallace et al. (2020)	Yes & No	Yes & No	Yes	No	Yes & No	Yes
Kariyappa et al. (2020)	Yes	Yes	Yes	Yes	Yes & No	Yes
Takemura et al. (2020)	Yes	Yes	Yes	No	Yes	Yes
Roberts et al. (2019)	Yes	Yes	No	No	Yes	Yes
Duddu and Rao (2019)	Yes	No	No	No	No	Unknown
Chen et al. (2020)	Yes	No	No	No	No	Yes
Wang and Gong (2018)	Yes	Yes	Yes	No	Yes	Yes
Mosafi et al. (2019)	Yes	No	Yes	No	No	No
Tramèr et al. (2016)	Yes & No	Yes & No	Yes	Yes	Yes & No	Yes
Duddu et al. (2018)	Yes	No	No	No	Yes	Yes
Krishna et al. (2019)	Yes	Yes	Yes	Yes	No	Yes
Oh et al. (2019)	Yes	Yes	Yes	Yes	Yes	Yes

Table 5: Prerequisites per paper. Papers are categorized on the knowledge required with regards to model type and model architecture. If there is attention to # queries and if there is a query budget. If knowledge of the dataset or domain is required.

valuable. More specifically, when there is no mention of the amount of queries necessary, the reader cannot assume that the attack is feasible in reasonable time. The attack may as well be brute-forcing the imitation phase by submitting tens of millions of queries. Imposing a query budget is even better, since now the attack does not have unlimited time to extract the model at an acceptable level.

5.2 Methods

In this section, we will analyze the model type distributions and the different attacks for these models. Most noticeable is that most of the papers extract some from of Neural Network (NN). The different types of NNs that are extracted are: Deep Neural Network (DNN), Graph Neural Network (GNN), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). Papers attacking similar models often have similar attack methods, often building on earlier versions. This section illustrates the similarities but also the differences between these attacks. To give a clear overview on which papers extract which type of model, this information is summarised in Table 6. We will structure this section per model type. For this, we have grouped similar models, such as all forms of NNs, together to allow for more comparison between extraction techniques.

5.2.1 SVM and SVR

The first model types that we will be discussing are the SVM and SVR models. Reith et al. (2019); Chandrasekaran et al. (2018); Shi et al. (2017); Wang and Gong (2018); Tramèr et al. (2016) all have developed extraction techniques that extract SVM or SVR models.
Tramèr et al. (2016) developed attacks for multiple types of classifiers, which will also be discussed in further parts of this section. For the SVM classifier, the authors based their approach on the approach of Lowd and Meek (2005), which is a black-box attack with membership queries that only return the predicted class label. The extension of Tramèr et al. (2016) adds three types of retraining to these attacks. Firstly, *retraining with uniform queries*. This is a baseline strategy where points are uniformly sampled at random and then presented as queries. The second approach, *line-search retraining*, can be seen as a model-agnostic generalization of the Lowd-Meek attack. Here, samples close to the decision boundary are found by using adaptive queries to the victim using line search techniques. The last approach, *adaptive retraining* is based on active learning. Over multiple rounds, new points are chosen along the decision boundary of which the adversary model is the least certain

Authors	Type of Model Stolen
Shi et al. (2018)	Can be all models
Pal et al. (2019)	DNN
Yu et al. (2020)	DNN
Atli et al. (2020)	DNN
Kariyappa et al. (2020)	DNN
Mosafi et al. (2019)	DNN
Tramèr et al. (2016)	DNN, DT, LR, SVM
Duddu et al. (2018)	NN
Duddu and Rao (2019)	NN
Jagielski et al. (2019)	NN
Oh et al. (2019)	NN
Correia-Silva et al. (2018)	CNN
Pal et al. (2020)	CNN, RNN
Orekondy et al. (2019)	CNN
Roberts et al. (2019)	CNN
Krishna et al. (2019)	BERT
Wallace et al. (2020)	MT
Chen et al. (2020)	DRL
Reith et al. (2019)	SVM, SVR
Chandrasekaran et al. (2018)	SVM, DT, RF
Shi et al. (2017)	NB, SVM
Takemura et al. (2020)	RNN, LTSM
Wang and Gong (2018)	RR, LASSO, KRR, SVM, L1-LR, L2-LR

Table 6: Global overview of model type stolen. Abbreviations are as follows: *DNN* = Deep Neural Network, *NN* = Neural Network, *RNN* = Recurrent Neural Network, *CNN* = Convolutional Neural Network, *GNN* = Graph Neural Network, *SVM* = Support Vector Machine, *SVR* = Support Vector Regression, *DT* = Decision Tree, *RF* = Random Forest, *NB* = Naive Bayes, *MT* = Machine Translation, *LTSM* = Long Short-Term Memory, *RR* = Ridge Regression, *KRR* = Kernel Ridge Regression, *L1-LR* = L1-Regularized Logistic Regression, *L2-LR* = L2-Regularized Logistic Regression, *BERT* = Bidirectional Encoder Representations from Transformers, *DRL* = Deep Reinforcement Learning.

about. These points are then sent to the victim and that output is used for training the adversary model again.
Reith et al. (2019) propose various attacks for the SVM. First, they describe equation solving attacks that can extract exact models for linear and quadratic kernels in a white-box setting. Then, they extend on the attacks described by Tramèr et al. (2016) which are based on the retraining approach. Lastly, they describe an attack to extract a model in a black-box setting. In this last attack, they use the previously described attacks but then for multiple models at once, each having a different kernel. Then, with a test set, the algorithm compares the predictions of each model to the original one. The model with the lowest error is presumed as the correct one and this kernel is assumed to be the kernel of the victim.
Chandrasekaran et al. (2018) also base their methods on the attacks proposed by Tramèr et al. (2016) in combination with active learning strategies from the literature on active learning for SVMs.
Wang and Gong (2018) propose novel methods for extracting SVM with regular hinge loss functions and with squared hinge loss functions (SVM-RHL & SVM-SHL) as well as attacks on the kernel versions of the aforementioned SVMs (KSVM-RHL & KSVM-SHL). Furthermore, the authors also propose attacks on L1-regularized logistic regression and L2-regularized logistic regression (L1-LR & L2-LR) as well as the kernel versions of the aforementioned LRs (L1-KLR & L2-KLR). Shi et al. (2017) has launched a three-step attack on the SVM. First, the victim is polled with input data. That data is observed and lastly, this data is then served as input data for the adversary model.

5.2.2 MT and BERT

There are two papers that only extract text based models: Wallace et al. (2020); Krishna et al. (2019). The paper of Wallace et al. (2020) focuses on Machine Translation (MT) and the paper of Krishna et al.

(2019) focuses on a diverse set of NLP tasks by extracting Bidirectional Encoder Representations from Transformers (BERT). With regards to MT, the attack that the authors have proposed is pretty straightforward. In particular, the authors assume access to a corpus of monolingual sentences and query the victim model with these sentences. The output is recorded and this is used for learning the new model. Krishna et al. (2019) have a similar approach, but they are working with a bit more restrictions. Firstly, the authors do not assume any access to training data. The training data is randomly generated and the sentences do not have to be grammatically or semantically correct. Then, the same approach is taken as by Wallace et al. (2020). However, since this attack is not based Machine Translation systems, but on BERT-based APIs, four NLP tasks are defined on which the model will be attacked: (1) binary sentiment classification using SST2 (Socher et al. (2013)), where the input is a single sentence and the output is a probability distribution between positive and negative; (2) ternary natural language inference (NLI) classification using MNLI (Williams et al. (2018)), where the input is a pair of sentences and the output is a distribution between entailment, contradiction and neutral; (3) extractive question answering (QA) using SQuAD 1.1 (Rajpurkar et al. (2016)), where the input is a paragraph and question and the output is an answer span from the paragraph; and (4) boolean question answering using BoolQ (Clark et al. (2019)), where the input is a paragraph and question and the output is a distribution between yes and no.
To summarise the two papers of Krishna et al. (2019); Wallace et al. (2020), both papers have similar approaches, while Krishna et al. (2019) have some more restrictions regarding the dataset that can be used. Furthermore, Krishna et al. (2019) define four NLP tasks on which the model can be extracted. Wallace et al. (2020) only extract the model for translation purposes.

5.2.3 NN and variations

Many papers extract some form of NN: Pal et al. (2019); Shi et al. (2018); Pal et al. (2020); Correia-Silva et al. (2018); Yu et al. (2020); Atli et al. (2020); Orekondy et al. (2019); Jagielski et al. (2019); Kariyappa et al. (2020); Takemura et al. (2020); Roberts et al. (2019); Duddu et al. (2018); Chen et al. (2020); Mosafi et al. (2019); Tramèr et al. (2016); Duddu and Rao (2019); Oh et al. (2019). In general, most of the papers have the same general approach for stealing a neural network related model. The attacker picks some subset of initial seed samples of the training dataset of the adversary. Then, the victim model is queried with these samples and the obtained labels are saved. Next, the adversary model is trained with these observed labels. After some iterations, the adversary model is trained on the remaining samples of the training set. This attack method is also similar to the flow depicted in Figure 3. While in general, the approaches of all the papers are similar, there are some differences between the attacks. One of these differences can be the evaluation metrics on which the closeness between the victim and adversary model is measured, which can influence how many training data needs to be queried to the victim model before deciding that enough queries are sent and the adversary model can be trained on the remaining dataset. Other differences can be the deployment of active learning strategies Pal et al. (2020, 2019); Shi et al. (2017); Yu et al. (2020); Chandrasekaran et al. (2018); Tramèr et al. (2016), which will be discussed shortly in Section 5.2.6. For the NN extraction methods, we will discuss the papers that had interesting techniques compared to other papers and we will discuss papers that have noteworthy discoveries. The other papers do not differentiate themselves with regards to the chosen technique for model extraction.
Kariyappa et al. (2020) have created an attack with their *MAZE* algorithm where synthetic data is generated for the model extraction attack. Although the authors do not call their method active learning, it has some key similarities. First, the *MAZE* algorithm trains a Generator to produce queries that maximizes the disagreement between the predictions of the victim and adversary by maximizing the KL-divergence. Then, using these generated queries, the adversary model is trained to match the predictions of the victim. Thus obtaining a highly accurate clone model.
Jagielski et al. (2019) has defined various goals that the adversary might have when attempting model extraction. First is the *functionally equivalent extraction*, where the goal is to create such a network that has the same output as the victim model for every input, which is the hardest one of all three goals. The second goal can be *fidelity extraction* where the difference between both the mistakes and correct labels are minimised. The last goal can be *task accuracy extraction* where the goal is to match or exceed the accuracy of the target model. Jagielski et al. (2019) deploy two attacks in their paper, one focusing one being a functionally equivalent extraction and one having fidelity extraction and task accuracy extraction as goal. When considering this classification of goals, other papers can be categorized similarly according to Jagielski et al. (2019). More specifically, Tramèr et al. (2016); Chandrasekaran et al. (2018); Lowd and Meek (2005) have functionally equivalent extraction as goal,

Pal et al. (2019) has fidelity extraction as goal, Orekondy et al. (2019) has task accuracy extraction as goal and Tramèr et al. (2016); Correia-Silva et al. (2018) have both task accuracy and fidelity extraction as main goal for their proposed attacks.
Since most of the papers focus on model extraction for classification tasks, there is less attention for regression tasks. However, Takemura et al. (2020) have created attacks for the regression models and noted that designing attacks for classification and regression tasks is not the same. One of the biggest differences is that a softmax function cannot be used as loss function for the regression tasks. This is important because the output of the softmax will be in the interval (0, 1), while a loss function with the norm, such as L1-loss and L2-loss does not have any restrictions in the range of output. This means that when using softmax, a corrupted prediction will not do much harm to the adversary model, while the L2-loss may be affected greatly by a corrupted prediction from the victim model. Therefore, Takemura et al. (2020) use the method as described by Chen et al. (2017) which utilizes the outputs as an upper bound to be achieved for the adversary model.
Lastly, Tramèr et al. (2016) have defined their attacks against logistic model with confidence values as *equation solving attacks*. Many models that the authors have researched directly compute class probability as continuous function of the input x. When an API does provide these class probabilities, the adversary is presented with samples (x, f(x)) that can be viewed as equations in the unknown parameters. The authors claim that for a large class of models, these equations can be solved efficiently.

5.2.4 DRL

Chen et al. (2020) is the only paper that extracts a form of Deep Reinforcement Learning (DRL) model. Chen et al. (2020) observes that attacks on DRL are often more difficult because DRL often have more complicated and deeper network structures to handle complex tasks. Therefore, the author has designed two stages for the attack on DRL networks. First, a RNN classifier is built which can identify the algorithm of the target DRL based on its run time actions. Secondly, imitation learning is used to replicate the behaviors of the target model based on the extracted algorithm.

5.2.5 All remaining models

For the remaining models, the papers described below encompass attacks on the following models: Decision Trees (DT), Logistic Regression (LR), Random Forests (RF), LASSO and KRR. The attack on a DT introduced by Tramèr et al. (2016) is a *path-finding* attack. The key idea behind the attack is to extract the path that has been traversed by the input in the tree. This can be done by using the information that is provided by the API as a *pseudo-identifier*. By varying the value of each input feature, the authors can find the predicates that have to be satisfied for an input to follow a given path in the tree.
The attacks on DT and RF by Chandrasekaran et al. (2018) are based on the idea of *importance weighted active learning* (IWAL), as described by Beygelzimer et al. (2010). In the results, the attacks based on IWAL is compared with the path-finding attack as described by Tramèr et al. (2016) for the decision trees, with all four models having different datasets. For 2 out of 4 attacks, the attacks by Chandrasekaran et al. (2018) needed 12 times more queries with almost similar results. For one out of the 4 attacks, Chandrasekaran et al. (2018) performed 40.3% better than Tramèr et al. (2016) despite needing only half the amount of the queries. In the last attack Chakraborty et al. (2018) outperformed Tramèr et al. (2016) by only 2.99% while needing 1.5 times the amount of queries. Therefore, we conclude that the attacks proposed by Tramèr et al. (2016) for the decision trees are better than the ones proposed by Chandrasekaran et al. (2018). Wang and Gong (2018) introduce attacks for LASSO and Ridge Regression (RR) models. Similar to the approach with SVMs by Wang and Gong (2018), the attack of RR models has also been implemented for the Kernel version of RR, KRR. However, since this paper is the only paper that we evaluate which proposed methods for these models, we will not discuss this in further detail.

5.2.6 Active Learning

Active learning is a subset selection strategy that chooses the optimal subset of queries to be send to the victim model. Active learning performs better than sending random queries since it provides data samples that are uncertain with respect to the original inferred classifier (Shi et al. (2018)). So in the case of attacks on SVMs, new points are chosen along the decision boundary of which the adversary model is the least certain about. Some papers have used some form of active learning or methods that were based on active learning techniques: Pal et al. (2019); Shi et al. (2018); Pal et al. (2020); Yu et al. (2020); Chandrasekaran et al. (2018); Kariyappa et al. (2020); Tramèr et al. (2016). Although we find this technique interesting and promising, we will not be going into detail because of the limited adoption in the reviewed papers.

5.3 Defenses

There are defense techniques that can prevent model extraction attacks or make them less efficient. In this section, we will compare the various defense mechanisms that have been applied by the authors on their own attacks, if any. The results of this comparison can be found in Table 7.

Authors	**Attention to defense techniques**
Pal et al. (2019)	PRADA, Xu et al. (2018); Kesarwani et al. (2018)
Duddu and Rao (2019)	PRADA, Kesarwani et al. (2018), Lee et al. (2018)
Yu et al. (2020)	PRADA, DefenseNet
Pal et al. (2020)	Lee et al. (2018)
Atli et al. (2020)	Detecting out-of-distribution samples
Tramèr et al. (2016)	Prediction API minimization, Rounding
Chandrasekaran et al. (2018)	Data randomization
Duddu et al. (2018)	Adding noise to execution time, Adversarial machine learning
Kariyappa et al. (2020)	Remove probabilities, Output probabilities with noise, Limit amount of queries
Chen et al. (2020)	Perturbing or removing output probability, returning only class output, Query pattern analysis & Watermarking
Mosafi et al. (2019)	Watermarking, Removing probability values, Perturbing output probability,
Jagielski et al. (2019)	Removing probability values
Krishna et al. (2019)	Membership Classification, Watermarking
Orekondy et al. (2019)	Top-k, Rounding
Wang and Gong (2018)	Rounding
Wallace et al. (2020)	Naive Defense
Reith et al. (2019)	-
Shi et al. (2018)	-
Roberts et al. (2019)	-
Oh et al. (2019)	-
Correia-Silva et al. (2018)	-
Shi et al. (2017)	-
Takemura et al. (2020)	-

Table 7: Summary of defense mechanisms evaluated per paper

Firstly, the papers who have not gave any attention to possible defense mechanisms for their attacks: Yu et al. (2020); Correia-Silva et al. (2018); Reith et al. (2019); Shi et al. (2017); Roberts et al. (2019); Oh et al. (2019); Takemura et al. (2020). While this is not particularly wrong, we feel that the validity of the model is slightly decreased when the proposed attack is not tested on the most basic defense mechanisms. When the most basic defense system already makes the proposed attack irrelevant, then one must ask oneself if your proposed attack is really that great.

One popular defense mechanism is PRADA (Juuti et al. (2019)), which is a generic and effective detection of DNN model attacks. PRADA analyzes the distribution of consecutive API queries and will raise an alarm when this distribution deviates from benign behavior (Juuti et al. (2019)). The authors claim that PRADA can detect all prior model extraction attacks with no false positives. While Duddu and Rao (2019) mentions PRADA as defense mechanism, but does not apply PRADA to its own attacks, Pal et al. (2019) speculate that their attacks will not be detected by PRADA. Due to the fact that PRADA expects the distribution of pairwise distances between benign samples to fit a bell curve. However, Pal et al. (2019) speculate that their universal thief datasets will also produce samples where the distribution will fit the bell curve. Therefore, PRADA is speculated to be not effective against the attacks by Pal et al. (2019). Yu et al. (2020) prove that their attacks can be evaded by PRADA. More specifically, by selecting some parameter M, Yu et al. (2020) can simulate a normal distribution of query samples without significantly degrading the adversary model performance. Therefore, this attack will evade PRADA.

Similar to PRADA, there is the proposal by Kesarwani et al. (2018), which "quantifies the extraction status of models by continually observing the API query and response streams of users". The authors propose two strategies, which either measures the information gain or the feature space spanned by the user queries. This can be used to estimate the learning rate of the user. Pal et al. (2019) evaluated this defense mechanism in their research and they speculated that their attack will probably get caught by the methods proposed by Kesarwani et al. (2018). However, they argue that they could tweak their active learning subset strategy to decrease the information gain or cover only a limited portion of the feature space spanned by the queries, making them less suspicious.

Another popular defense mechanism is watermarking, as described by Zhang et al. (2018); Uchida et al. (2017), where a digital watermarking technique is applied at the training stage DNNs to verify the ownership of the DNN models. Watermarking has no influence on the performance of the model since the watermark is embedded when training the DNN. Uchida et al. (2017) defines a certain 'hidden' which can be a certain shape or noise in the training data. When this model would receive this hidden key, the model would output an unrelated label with high certainty. Consequently, this model of digital watermarking can also work through APIs. When a stolen model is queried through an API with this hidden key and the unrelated label is returned, this model can be assumed as stolen. Various papers have considered watermarking as potential defense mechanisms against their attacks, Chen et al. (2020); Mosafi et al. (2019); Krishna et al. (2019). Chen et al. (2020) claim that their attack is not venerable against watermarking since their attack requirements are minimal. In the paper of Mosafi et al. (2019) it is claimed that the method of Uchida et al. (2017) does not work against the attacks proposed, since their training is based on random combinations of input. Therefore, the authors find the chances of triggering the hidden key negligible. Krishna et al. (2019) note that watermarking works against their attacks. However, the authors note that watermarking does have some limitations. Namely, watermarking assumes that the attacker will publish the stolen model online, after which it can be detected by watermarking. Therefore watermarking is irrelevant if the stolen model is never published. Secondly, when the attacker anticipates watermarking, one can take steps to prevent being detected. The authors give three methods to prevent detection: differentially private training on extraction data (Abadi et al. (2016); Dwork et al. (2014)), fine-tuning or re-extracting an extracted model with different queries (Chen et al. (2019); Szyller et al. (2019)), or issuing random outputs on queries exactly matching inputs in the extraction data (Krishna et al. (2019)).

Subsequently, there is rounding which is used by Tramèr et al. (2016); Orekondy et al. (2019); Wang and Gong (2018) and also described by Fredrikson et al. (2015). Tramèr et al. (2016) explore the effects of rounding on softmax models. The results show that rounding class probabilities to 4 or 5 decimals, as is done by BigML, have no influence on the attacks success. When rounding to 2 or 3 decimals, the attack is weakened slightly but still outperforms when compared to outputting class labels only. For regression trees, rounding has no effect and for classification trees no influence is noted until rounding up to 2 decimals. For their other models, "rounding confidences 3 or 4 decimal places severely undermines our attack" (Tramèr et al. (2016)). Orekondy et al. (2019) examine that rounding has little effect on their attacks, with rounding up to 2 decimals still providing 99% original performance. When rounding up to 1 decimal, the performance seems to decrease more. Wang and Gong (2018) also examine that rounding is not effective enough to prevent certain attacks on ML algorithms. To summarise, although rounding is a relatively simple mechanism to implement by the victim side, it does not provide real protection against the threats of model extraction attacks.

Lee et al. (2018) propose a method that applies perturbation to the predicted softmax probability scores to dissuade model extraction. Duddu and Rao (2019); Pal et al. (2020) have both considered this defense mechanism for their attacks. Pal et al. (2020) note that this attack will not influence the working of their attacks since their attacks only requires the predicted label to work accordingly. Duddu and Rao (2019) explained the workings of the defense mechanism by Lee et al. (2018) but did not elaborate on the effects on their attacks. Since no other papers discusses before have evaluated this defense mechanism, we cannot make any assumptions about the influence of the method by Lee et al. (2018) on the model extraction attacks discussed in this research.

Atli et al. (2020) have developed their own method for detecting queries by *Knockoff nets*. Their goal is to detect queries that do not correspond to the main classification task of the victim model. For this, a binary classifier is proposed based on the ResNet architecture. This binary classifier can be used as filter in front of the prediction API. The authors claim that their defense method correctly detects up to 99% of the adversary queries, which is more than the other state-of-the-art out-of-distribution detectors.

Jagielski et al. (2019) mention in their research that they have considered one defense mechanism, which is removing access to probability values. They mention that they functionally-equivalent attack is completely broken by this defense mechanism and do not provide further information on how to evade such defense mechanisms.

As discussed previously, Duddu and Rao (2019) have considered defense mechanisms such as PRADA and methods described by Kesarwani et al. (2018); Lee et al. (2018). However, the authors have not shown that these methods have been tested against the proposed attack. Therefore, it is difficult to say whether the proposed attack gets caught or becomes less effective by the aforementioned defense mechanisms.

Chandrasekaran et al. (2018) proposes data randomization as defense mechanism. More specifically, they propose two versions, *data-independent randomization* and *data-dependent randomization*. In general method, a wrong output is given instead of the correct output, with some probability. For the data-independent randomization, this probability is not dependent on the data that the user is querying, so effectiveness of this method will not be guaranteed. However, for the data-dependent method, the probability of providing the wrong output is based on the query input and the labeling function. When this is done correctly, the effectiveness of the adversary's attack is not guaranteed anymore.

The remaining defense mechanisms is not discussed in this research paper, since they are not used very often, therefore being not that relevant for the large scope of this research paper.

6 Discussion

Membership Inference

Various of the discussed papers propose that overfitting is a sufficient condition for vulnerability against membership attacks. However, overfitting is not always necessary for an attack to succeed. Some papers also looked into the influence of certain datapoints on the final model as a connection to vulnerability to membership inference attacks. These two factors seem to be the best predictors of susceptibility of a model.

Many of the attacks that were proposed follow the strategy from Shokri et al. (2017), many of which are really successful even at the more limited assumption levels. The current body of research suggests that there are many ways to stage a successful membership attack, even on well-generalized models. Successful attacks have been staged against many type of deep-learning models, from classifiers to generative models, and on a variety of datasets.

As successful attacks are certainly shown to be possible, there is not a conclusive answer on why exactly some attacks work as well as they do. Why exactly models leak information about their training set is still an active question, even though a couple of useful steps have been done in this direction.

The main defenses against this attack seem to stem from regularization techniques and differential privacy. Although there is always a trade-off between privacy of the training set and utility of the model. Sometimes to such an extent that it would make the model no longer competitive if the privacy loss is low.

Model Inversion

The various model inversion attacks discussed in this work are aiming for a diverse set of results. Some succeed in reconstructing complete data samples from the training data, while others extract only attributes or properties. Regardless, all these methods can result in a loss of privacy for the people contributing data to the training set. But as most of these attacks are applied in a controlled environment, it is difficult to determine how much of this privacy risk carries over to real-world contexts.
The amount of information necessary to execute most attacks is quite high. Half of the methods require white box access to work at all. Some attacks should theoretically work under minimal

prerequisites, but then prove to be effective only when auxiliary information is used. How to obtain this information is usually not mentioned. However, model extraction could be used to gain white box access if necessary, and depending on the specificity demands on the auxiliary data, it is sometimes possible for the adversary to find or even to generate such a dataset. Thus concerns for privacy are in most cases legitimate.
To offer protection, some implementation of DP is usually sufficient, taking into account granularity and utility. Finding techniques to determine the optimal DP algorithm and ε value to use in a given context is a topic for further research. In a collaborative setting, the most effective defense is to further reduce or hide the information contained in the gradients. This could be done by adding noise, clipping, compressing, or encrypting them. However, these more involved methods of protection often take more time to implement and could decrease the utility of the model critically, which may motivate model owners especially in business contexts not to use these defenses. Therefore, using techniques as simple as rounding the confidence scores could be valuable as well.
Something that is often missing in the literature on model inversion is an analysis of the cost, in terms of time and money required for the attack. Query budgets are regularly used to constrain and evaluate model extraction attacks and would be just as useful in this context. Making some estimate on the time needed to execute an attack, parameterized by the size of the training set or other suitable aspects, is also very valuable.

Model Extraction

A lot of discussed model extraction techniques are aimed at NN models. We suspect that this is from the rising popularity of NN models throughout the past years. Another reason might be that NNs and variations of NNs are widely used by MLaaS providers for image classification for example. When comparing the methods of the attacks between these NN papers, most models follow the same paradigm. We have also looked at other categories of ML models, such as SVMs, Decision Trees and Logistic Regression.

Furthermore, we have discussed the prerequisites for every paper in detail. The results are similar to model inversion techniques, where the amount of information necessary is generally quite high. From our findings we can conclude that most papers do require knowledge of the model type beforehand, while the architecture of the model is not always required. With regards to domain knowledge, there is a limited set of papers that do not require any domain knowledge before commencing the attack. Knowledge of the database on the other hand is less frequent required. When comparing query budgets, less than half of the discussed papers have imposed some query budget on their attack. Moreover, almost half of the papers have discussed the amount of queries needed to complete the queries to the desired standard. From this we can conclude that a small subset of the papers think query budgets are important for the feasibility of their attack, while almost half of the papers assume that they can send unlimited queries to the victim to complete their attack.

With regards to defense methods, there are some papers that have not considered testing some defense mechanisms against their attacks. While this is not particularly wrong, it does not show that the attack is really valid in real-world settings, where these defense mechanisms might be deployed. Therefore, we would strongly advise to consider at least some of the defense mechanisms mentioned in Section 5.3. More than half of the papers have considered some form of defense mechanisms. And most of these papers show that their proposed method evades this particular defense mechanism. Thus, the authors only show the defense mechanisms that had no influence on their proposed attack. Therefore, we feel that authors are picky with which defense mechanisms they decide to evaluate, only showing the ones which have no influence on their attack. We judge that attacks would be even more valuable when they are tested on multiple defense mechanisms, and not only report the defense mechanisms that can be evaded.

Conclusion

To conclude this review, we state a few general observations and provide some directions for future research.
Firstly, the amount and variety of literature on this subject proves that there are still many security problems with current machine learning models. Even without implementing differential privacy or other complex methods, there are some simple modifications that can be applied to most model types and which significantly lower the risk of a successful attack.

However, differential privacy is an important notion for achieving good protection. It is not necessary to use this in all cases, but for sensitive domains such as medical data, models should be watertight. In this light, we find that more research still needs to be done on different implementations of DP. From the papers we have discussed, it is clear that the trade-off between privacy and utility is not yet understood well enough, and future works should take this into account.
With regards to model extraction, we judge that more research has to be done on defense mechanisms imposed on the extraction attacks. There has to be a more thorough selection and testing of multiple defense mechanisms to ensure validity in real-world scenarios. To a lesser extent, this also applies to the other attacks: many papers do not discuss defense well enough or at all.
Additionally, the connection between membership inference and model inversion could be investigated more. The methods and defenses that are effective for membership inference often also work for model inversion, specifically for data reconstruction.
Lastly, it could be useful to employ a more theoretical perspective to these attacks. There are some attempts to define general notions or algorithms, but most works have an empirical approach, which means that it is unclear whether the attack works on models or data that have not yet been tested.

References

Abadi, M., A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318.

Abadi, M., H. B. McMahan, A. Chu, I. Mironov, L. Zhang, I. Goodfellow, and K. Talwar (2016, 7). Deep learning with differential privacy. In *ACM Conference on Computer and Communications Security*, Volume 24-28-Octo, pp. 308–318.

Aïvodji, U., S. Gambs, and T. Ther (2019). GAMIN: An adversarial approach to black-box model inversion.

Al-Rubaie, M. and J. M. Chang (2016). Reconstruction Attacks Against Mobile-Based Continuous Authentication Systems in the Cloud. *IEEE Transactions on Information Forensics and Security 11*(12), 2648–2663.

Ateniese, G., L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici (2015). Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks 10*(3), 137–150.

Atli, B. G., S. Szyller, M. Juuti, S. Marchal, and N. Asokan (2020). Extraction of Complex DNN Models: Real Threat or Boogeyman? In O. Shehory, E. Farchi, and G. Barash (Eds.), *Engineering Dependable and Secure Machine Learning Systems*, pp. 42–57. Springer International Publishing.

Beygelzimer, A., D. J. Hsu, J. Langford, and T. Zhang (2010). Agnostic active learning without constraints. In *Advances in neural information processing systems*, pp. 199–207.

Calandrino, J. A., A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov (2011). "You might also like:" Privacy risks of collaborative filtering. In *IEEE Symposium on Security and Privacy*, pp. 231–246. IEEE.

Carlini, N., C. Liu, U. Erlingsson, J. Kos, and D. Song (2019). The secret Sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, pp. 267–284.

Chakraborty, A., M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay (2018). Adversarial Attacks and Defences: A Survey.

Chandrasekaran, V., K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan (2018). Exploring Connections Between Active Learning and Model Extraction. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1309–1326.

Chen, D., N. Yu, Y. Zhang, and M. Fritz (2019). GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 343–362.

Chen, G., W. Choi, X. Yu, T. Han, and M. Chandraker (2017). Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, pp. 742–751.

Chen, J. and X. Ran (2019). Deep Learning With Edge Computing: A Review. *Proceedings of the IEEE 107*(8), 1655–1674.

Chen, K., T. Zhang, X. Xie, and Y. Liu (2020). Stealing Deep Reinforcement Learning Models for Fun and Profit.

Chen, X., W. Wang, Y. Ding, C. Bender, R. Jia, B. Li, and D. Song (2019). Leveraging unlabeled data for watermark removal of deep neural networks. In *ICML workshop on Security and Privacy of Machine Learning*.

Clark, C., K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova (2019). Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936.

Correia-Silva, J. R., R. F. Berriel, C. Badue, A. F. De Souza, and T. Oliveira-Santos (2018). Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data. In *Proceedings of the International Joint Conference on Neural Networks*. IEEE.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Dong, J., A. Roth, and W. J. Su (2019). Gaussian differential privacy.

Duddu, V. and D. V. Rao (2019). Quantifying (Hyper) Parameter Leakage in Machine Learning. In *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, pp. 239–244. IEEE.

Duddu, V., D. Samanta, D. V. Rao, and V. E. Balas (2018). Stealing Neural Networks via Timing Side Channels.

Dwork, C., A. Roth, et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science 9*(3-4), 211–407.

Farokhi, F. and M. A. Kaafar (2020). Modelling and Quantifying Membership Information Leakage in Machine Learning.

Fredrikson, M., S. Jha, and T. Ristenpart (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM Conference on Computer and Communications Security*, Volume 2015-Octob, pp. 1322–1333.

Fredrikson, M., E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart (2014). Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *Proceedings of the 23rd USENIX Security Symposium*, pp. 17–32.

Gambs, S., A. Gmati, and M. Hurfin (2012). Reconstruction attack through classifier analysis. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Volume 7371 LNCS, pp. 274–281.

Ganju, K., Q. Wang, W. Yang, C. A. Gunter, and N. Borisov (2018). Property inference attacks on fully connected neural networks using permutation invariant representations. In *ACM Conference on Computer and Communications Security*, pp. 619–633.

Hayes, J., L. Melis, G. Danezis, and E. De Cristofaro (2019). LOGAN: Membership Inference Attacks Against Generative Models. In *Privacy Enhancing Technologies*, Volume 1, pp. 133–152.

He, X., J. Jia, M. Backes, N. Z. Gong, and Y. Zhang (2020). Stealing Links from Graph Neural Networks.

Hilprecht, B., M. Härterich, and D. Bernau (2019). Monte Carlo and Reconstruction Membership Inference Attacks against Generative Models. In *Privacy Enhancing Technologies*, Volume 4, pp. 232–249.

Hisamoto, S., M. Post, and K. Duh (2020). Membership Inference Attacks on Sequence-to-Sequence Models : Is My Data In Your Machine Translation System ? *Transactions of the Association for Computational Linguistics 8*, 49–63.

Hitaj, B., G. Ateniese, and F. Perez-Cruz (2017). Deep Models under the GAN: Information leakage from collaborative deep learning. In *ACM Conference on Computer and Communications Security*, Volume 1, pp. 603–618.

Jagielski, M., N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot (2019). High Accuracy and High Fidelity Extraction of Neural Networks. In *29th USENIX Security Symposium (USENIX Security 20)*.

Jayaraman, B., L. Wang, K. Knipmeyer, Q. Gu, and D. Evans (2020). Revisiting Membership Inference Under Realistic Assumptions.

Juuti, M., S. Szyller, S. Marchal, and N. Asokan (2019). Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 512–527. IEEE.

Kariyappa, S., A. Prakash, and M. Qureshi (2020). MAZE: Data-Free Model Stealing Attack Using Zeroth-Order Gradient Estimation.

Kesarwani, M., B. Mukhoty, V. Arya, and S. Mehta (2018). Model extraction warning in mlaas paradigm. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 371–380.

Krishna, K., G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer (2019). Thieves on Sesame Street! Model Extraction of BERT-based APIs.

Lee, T., B. Edwards, I. Molloy, and D. Su (2018). Defending against model stealing attacks using deceptive perturbations.

Liu, G., C. Wang, K. Peng, H. Huang, Y. Li, and W. Cheng (2019). SocInf: Membership Inference Attacks on Social Media Health Data with Machine Learning. *IEEE Transactions on Computational Social Systems 6*(5), 907–921.

Long, Y., V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen (2018). Understanding Membership Inferences on Well-Generalized Learning Models.

Lowd, D. and C. Meek (2005). Adversarial learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 641–647.

Melis, L., C. Song, E. De Cristofaro, and V. Shmatikov (2019). Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, Volume 2019-May, pp. 691–706. IEEE.

Mosafi, I., E. O. David, and N. S. Netanyahu (2019). Stealing Knowledge from Protected Deep Neural Networks Using Composite Unlabeled Data. In *International Joint Conference on Neural Networks*. IEEE.

Nasr, M., R. Shokri, and A. Houmansadr (2019, 5). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy*, Volume 2019-May, pp. 739–753. Institute of Electrical and Electronics Engineers Inc.

Oh, S. J., B. Schiele, and M. Fritz (2019). Towards Reverse-Engineering Black-Box Neural Networks. In *ICLR 2018*, pp. 121–144.

Orekondy, T., B. Schiele, and M. Fritz (2019). Knockoff nets: Stealing functionality of black-box models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Volume 2019-June, pp. 4949–4958.

Pal, S., Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy (2019). A framework for the extraction of Deep Neural Networks by leveraging public data.

Pal, S., Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy (2020). ActiveThief: Model Extraction Using Active Learning and Unannotated Public Data. *Proceedings of the AAAI Conference on Artificial Intelligence 34*(01), 865–872.

Park, J., S. Samarakoon, M. Bennis, and M. Debbah (2019). Wireless network intelligence at the edge. *Proceedings of the IEEE 107*(11), 2204–2239.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics 33*(3), 1065–1076.

Pyrgelis, A., C. Troncoso, and E. D. Cristofaro (2018). Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *Network and Distributed Systems Security (NDSS) Symposium 2018*.

Rahman, M. A., T. Rahman, R. Laganière, N. Mohammed, and Y. Wang (2018). Membership inference attack against differentially private deep learning model. *Transactions on Data Privacy 11*(1), 61–79.

Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang (2016). Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Reith, R. N., T. Schneider, and O. Tkachenko (2019). Efficiently stealing your machine learning models. In *ACM Conference on Computer and Communications Security*, pp. 198–210.

Roberts, N., V. U. Prabhu, and M. McAteer (2019). Model Weight Theft With Just Noise Inputs: The Curious Case of the Petulant Attacker.

Ruder, S. (2016). An overview of gradient descent optimization algorithms.

Sablayrolles, A., M. Douze, Y. Ollivier, C. Schmid, and H. Jegou (2019). White-box vs Black-box: Bayes optimal strategies for membership inference. In *36th International Conference on Machine Learning, ICML 2019*, Volume 2019-June, pp. 9780–9790.

Salem, A., A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang (2019). Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1291–1308. USENIX Association.

Salem, A., Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes (2019). ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed Systems Security Symposium 2019*. Internet Society.

Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016). Improved techniques for training gans. *Advances in neural information processing systems 29*, 2234–2242.

Shi, Y., Y. Sagduyu, and A. Grushin (2017). How to steal a machine learning classifier with deep learning. In *2017 IEEE International Symposium on Technologies for Homeland Security, HST 2017*.

Shi, Y., Y. E. Sagduyu, K. Davaslioglu, and J. H. Li (2018). Active Deep Learning Attacks under Strict Rate Limitations for Online API Calls. In *2018 IEEE International Symposium on Technologies for Homeland Security, HST 2018*.

Shokri, R. and V. Shmatikov (2015). Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321.

Shokri, R., M. Strobel, and Y. Zick (2019). Privacy Risks of Explaining Machine Learning Models.

Shokri, R., M. Stronati, C. Song, and V. Shmatikov (2017, 6). Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy*, pp. 3–18. Institute of Electrical and Electronics Engineers Inc.

Socher, R., A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642.

Song, C. and V. Shmatikov (2019). Auditing data provenance in text-generation models. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 196–206.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*(1), 1929–1958.

Szyller, S., B. G. Atli, S. Marchal, and N. Asokan (2019). Dawn: Dynamic adversarial watermarking of neural networks.

Takemura, T., N. Yanai, and T. Fujiwara (2020). Model Extraction Attacks against Recurrent Neural Networks.

Tramèr, F., F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart (2016). Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium*, pp. 601–618.

Truex, S., L. Liu, M. E. Gursoy, L. Yu, and W. Wei (2019). Demystifying Membership Inference Attacks in Machine Learning as a Service. *IEEE Transactions on Services Computing X*(c), 1–1.

Uchida, Y., Y. Nagai, S. Sakazawa, and S. Satoh (2017). Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pp. 269–277.

Wallace, E., M. Stern, and D. Song (2020). Imitation Attacks and Defenses for Black-box Machine Translation Systems.

Wang, B. and N. Z. Gong (2018). Stealing Hyperparameters in Machine Learning. In *IEEE Symposium on Security and Privacy*, Volume 2018-May, pp. 36–52. IEEE.

Wang, Z., M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi (2019). Beyond Inferring Class Representatives: User-Level Privacy Leakage from Federated Learning. In *IEEE INFOCOM*, Volume 2019-April, pp. 2512–2520. IEEE.

Wilkinson, G. and P. Legg (2020). "What did you say?": Extracting unintentional secrets from predictive text learning systems. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*.

Williams, A., N. Nangia, and S. R. Bowman (2018, June). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, pp. 1112–1122. Association for Computational Linguistics.

Xu, H., Y. Su, Z. Zhao, Y. Zhou, M. R. Lyu, and I. King (2018). Deepobfuscation: Securing the structure of convolutional neural networks via knowledge distillation.

Yeom, S., I. Giacomelli, M. Fredrikson, and S. Jha (2018). Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE Computer Security Foundations Symposium*, Volume 2018-July, pp. 268–282. IEEE.

Yu, H., K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin (2020). CloudLeak: Large-Scale Deep Learning Models Stealing Through Adversarial Examples. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*.

Zhang, J., Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy (2018). Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 159–172.

Zhang, Y., R. Jia, H. Pei, W. Wang, B. Li, and D. Song (2020). The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 253–261.

Zhao, B., K. R. Mopuri, and H. Bilen (2020). iDLG: Improved Deep Leakage from Gradients.

Zhu, L., Z. Liu, and S. Han (2019). Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems*, pp. 14774–14784.

Contributions

We started this project by finding all available literature on the subject. Some papers were dropped in this phase on account of irrelevance. Generally this work was divided equally. We each chose one of the three attacks to focus on and did a first pass through the papers corresponding to our topics. In this phase some extra papers were selected, mostly based on connections between the papers, and some others were dropped. Succeeding this phase, we started writing the first draft. The main sections on the attacks were written individually, and Jetske wrote the first drafts of the introductions and background. We gave feedback on each others drafts after this round and proceeded with writing the final version. During this phase we again were individually responsible for one of the main sections on the attacks. The background section was extended by Willemijn, partly based on a version in an earlier report also written by her. After writing the main part, we discussed our findings and wrote individual discussion sections. Then Jetske fine-tuned this and wrote the conclusion. We all edited the introduction and background and Mike wrote the abstract.