I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences

DARYNA OLIYNYK, SBA Research, Austria
RUDOLF MAYER, SBA Research & Vienna University of Technology, Austria
ANDREAS RAUBER, Vienna University of Technology, Austria

Machine-Learning-as-a-Service (MLaaS) has become a widespread paradigm, making even the most complex Machine Learning models available for clients via e.g. a pay-per-query principle. This allows users to avoid time-consuming processes of data collection, hyperparameter tuning, and model training. However, by giving their customers access to the (predictions of their) models, MLaaS providers endanger their intellectual property such as sensitive training data, optimised hyperparameters, or learned model parameters. In some cases, adversaries can create a copy of the model with (almost) identical behaviour using the the prediction labels only. While many variants of this attack have been described, only scattered defence strategies that address isolated threats have been proposed. To arrive at a comprehensive understanding why these attacks are successful and how they could be holistically defended against, a thorough systematisation of the field of model stealing is necessary. We address this by categorising and comparing model stealing attacks, assessing their performance, and exploring corresponding defence techniques in different settings. We propose a taxonomy for attack and defence approaches and provide guidelines on how to select the right attack- or defence strategy based on the goal and available resources. Finally, we analyse which defences are rendered less effective by current attack strategies.

CCS Concepts: • **Security and privacy → Vulnerability management**; • **Computing methodologies → Machine Learning**.

Additional Key Words and Phrases: Machine Learning, Model Stealing, Model Extraction

ACM Reference Format:
Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. 2023. I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences. *ACM Comput. Surv.* 1, 1, Article 1 (January 2023), 38 pages. https://doi.org/10.1145/3595292

1 INTRODUCTION

Training a Machine Learning model can be very complex and time- as well as resource-consuming. To safeguard their intellectual property, owners may opt to keep their models secret, allowing external users to access them only by input-output queries over a predefined API. However, black-box access to a model does not imply a *protected* model. Recent work has shown how an adversary can *steal* (extract) such models [1–3]. The technique of model stealing (also called "model extraction") aims at obtaining e.g. training hyperparameters, the model architecture, learned parameters, or an approximation of the behaviour of a model, all of which to the detriment of the lawful model owner.

The number of domains where model stealing attacks are successful has dramatically risen over the last few years. Dozens of attacks were executed regarding attack image classification [3], text classification [4], natural language processing [5], and reinforcement learning [6]. Jagielski et al. provide a preliminary taxonomy based on the attackers'

Authors' addresses: Daryna Oliynyk, SBA Research, Vienna, Austria, doliynyk@sba-research.org; Rudolf Mayer, SBA Research & Vienna University of Technology, Vienna, Austria, rmayer@sba-research.org; Andreas Rauber, Vienna University of Technology, Vienna, Austria, rauber@ifs.tuwien.ac.at.

goals, thus classifying different types of attacks [7]. However, the authors focus on a specific subset of attack patterns that address behaviour stealing and target only neural networks. Therefore, a comprehensive analysis of the potential and abilities of model stealing remains an important open task.

There are two main approaches for protecting a Machine Learning model against a model stealing attack: attack detection [8] and attack prevention [9]. The first approach cannot protect the model on its own, but informs the owner that somebody tries to steal the model or that it has already been stolen. The second approach should prevent the attack or at least make it less effective. Unfortunately, a lot of the defences can either be fooled [10–12] or work only under specific conditions [13]. Hence, studying existing approaches and investigating new ones is of the utmost importance. To the best of our knowledge, there is no work that comprehensively compares defences against model stealing. A systematisation of defence approaches will lead to a better understanding of success criteria and, subsequently, to new, more effective defences – for instance, by combining defences to cover multiple attack models at once.

Our contributions in this paper are the following:

- We collect and describe approaches to model stealing attacks as well as defence techniques. We explore how, when, and for which goals they were created, and unify reported performance measures of known attacks.
- We provide novel a taxonomy of model stealing attacks and -defences based on goal, methodology, and target model type. Following this taxonomy, we classify attacks and defences.
- We compare *query-based* attacks by their effectiveness and efficiency; based on this comparison we develop recommendations for how to design and evaluate model stealing attacks.
- We provide two guidelines for model stealing attacks and -defences, illustrated with diagrams. Following those guidelines, one can decide which attack- or defence strategy suits best in a given setting.

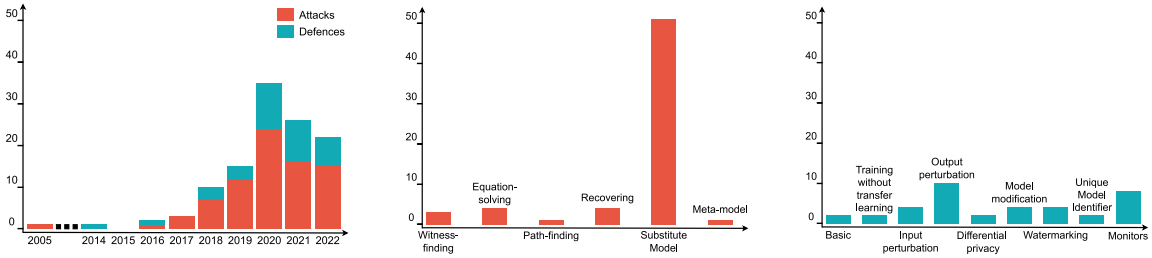
The rest of the paper is structured as follows: Section 2 describes related work, before Section 3 details the methodology used in our survey and systematisation. Section 4 provides the reader with the background knowledge required to understand this paper, while Section 6 introduces important concepts of model stealing. Section 5 introduces our novel taxonomy on model stealing attacks, followed by Sections 7 and 8 which describes known attack approaches and provides the corresponding classification of the attacks as well as an overview of the performance of individual attacks. Section 9 describes proposed defence strategies and provides a respective taxonomy; subsequently, Section 10 presents two guidelines for choosing the best attack- or defence strategy under certain conditions and compares the effectiveness of defences against known attacks. Finally, Section 11 provides conclusions and an outlook for future work.

2 RELATED WORK

To the best of our knowledge, there is no systematisation that provides a comprehensive research of model stealing attacks and defence techniques. Jagielski et al. [7] were the first to categorise model stealing attacks in terms of two objectives: accuracy and fidelity. The authors compared the goals of different attacks and argued about the importance of fidelity, which is a valuable basis for this work. However, they focus only on a specific subset of attacks, i.e. behaviour stealing of neural networks, and do not include defence strategies. In our work, we propose a comprehensive taxonomy and systematisation of both attacks and defences. Given the dynamics of the field, we are also able to consider a significantly larger number of papers (more than 100 on attacks and on defences, as opposed to 9 on attacks).

Gong et al. [14] provide an overview of six model stealing attacks as well as six defences. The authors categorised them based on specific characteristics, e.g. an ability to steal/protect a deep neural network (DNN). However, the paper covers only a fraction of relevant works, and consequently the taxonomy and categorisation comprises only a subset

Manuscript accepted in ACM Computing Surveys (CSUR): https://doi.org/10.1145/3595292



(a) Literature across years for attacks on and defences of ML models

(b) Papers per attack type (cf. Sections 7 and 8 for a description of attack types)

(c) Papers per defence type (cf. Section 9 for a description of defence types)

Fig. 1. Literature statistics

of the field. In contrast, a significant number of surveys regarding privacy and security in Machine Learning have been published [15–18]. In these works, model extraction attacks are usually only briefly mentioned as one sub-field of adversarial Machine Learning, while the main focus is on e.g. evasion (adversarial examples) and data poisoning attacks. Furthermore, there are publications which explore attacks and defences, including model stealing, in specific settings like reinforcement learning [19] or edge-deployed neural networks [20]. We go beyond these studies and focus on model stealing as a crucial issue of Machine Learning security, presenting a comprehensive, structural view on the broad range of attacks as well as defences.

3 METHODOLOGY

Our paper is based on an extensive literature research, including *formal*, peer-reviewed literature such as conference papers or journal articles as well as *grey* literature, i.e. works that did not undergo a peer-review process; the latter primarily includes pre-prints published on the arXiv repository.

We defined the following criteria to identify the most relevant literature regarding model stealing. Our inclusion criteria are: (1) Literature which proposes a method to perform model stealing attacks (2) Literature which proposes a defence against model stealing attacks. (3) Literature which evaluates or compares earlier schemes.

Our exclusion criteria are: (1) (Near) Duplicates; if the titles are different, but the content is very similar, we consider the most comprehensive or peer-reviewed version, and cite only that version. (2) Literature which only *applies* earlier model stealing attacks as vehicle, without introducing novel attacks or -defences. This includes e.g. using model stealing to transform black-box access to a model into white-box access to a substitute model for an evasion attack.

This resulted in a total of more than 100 papers on model stealing attacks and -defences for our in-depth investigation. Figure 1a provides an overview on how the field evolved over the years. There is very early work from 2005 ([21]) which covers model stealing, but the main body of literature has been published since 2016. First, more work was published on attacks; however, the volume of literature on defence strategies has caught up since 2018. It has to be noted that some publications cover an attack and the respective defence as well as propose a new defence immune to that attack. Regarding the different types of attacks depicted in Figure 1b, we can see that the vast majority of works focuses on substitute model training attacks. Since there are many attacks utilising hardware (HW) or software (SW) side channels (SCA), it must be noted that these attacks often exploit multiple, different types of side-channels, thus forming a rather inhomogeneous group of attacks. From Figure 1c, we can see that output perturbation and monitors are the most

prominent defence techniques. It should also be noted that some approaches such as monitoring and watermarking are reactive and thus aimed at *detecting* an attack, while others are proactively trying to *prevent* an attack.

4 BACKGROUND

In this section, we briefly summarise important concepts, terminology and notation required for the rest of this paper.

4.1 Machine Learning

So far, model stealing literature primarily targets *supervised-* and *reinforcement* learning. In supervised learning, each sample x_i from the input data set X has a corresponding label y, and the goal is to learn a model that approximates the real mapping function $f(x) = y$ for a given problem P to eventually predict labels $\hat{y}$ for unlabelled data. If the labels are discrete values, this is a classification problem; if they are continuous values, the problem is called regression. In reinforcement learning, *agents* are learning to make the best decision in a given situation so that the reward of a performed action is maximal. They act in a particular environment in order to achieve a predetermined goal.

Successfully learning a Machine Learning model requires different resources and domain knowledge. First, the quality of the model depends on the quality of the training dataset. This includes sample gathering and data labelling, often requiring human experts' knowledge which can be very resource-consuming. Before learning the model, training hyperparameters, such as the learning rate for a Neural Network or the architecture of the model (e.g. the number of layers in a Neural Network), have to be set. Selecting fitting values requires expert knowledge and experience. Finally, model training itself can be very compute- and time consuming and may require many refinement cycles of hyperparameter setting and training in order to arrive at the most representative model. The need for large datasets, expert knowledge and compute resources are the main reasons for the emergence of the MLaaS paradigm.

In the following, we introduce Machine Learning concepts relevant for this paper.

In an **Active Learning** learning [22] process, an *oracle* receives data samples and returns the corresponding labels. Since data is labelled dynamically, one can choose the samples most useful to building the model, thus reducing the required amount of labelled data. Thus, active learning is a strategy often employed to reduce the number of queries required during model stealing. Generally, the "oracle" is considered to be a *human (domain) expert* who is asked to provide an ad hoc ground truth; however, it can be any other information source. In model stealing, the *target model*, which can label samples, is thus considered to be the oracle. This connection between active learning and model stealing has been explored by Chandrasekaran et al. [23]; other works [24–27] use active learning to improve attack efficiency.

Knowledge Distillation [28] is a model compression [29] method that allows to train a smaller version (student network) of an already trained larger (teacher) network, without decreasing accuracy. The main idea is that the student network is learning to duplicate the outputs of the teacher network on *each*, and not only the final layer. In model stealing, these ideas form the basis for some attacks (e.g. Kariyappa et al. [30]) as well as defences (e.g. Xu et al. [31]).

Machine Learning as a Service (MLaaS) refers to cloud-based computing platforms that offer Machine Learning tools. These services allow users to remotely train their models, evaluate them, or use pre-trained models via a *pay-per-query* principle. Providers such as Amazon[1], Microsoft (Azure)[2], or Google[3] offer these services. Models supplied by MLaaS are usually only available for input-output interaction without revealing the model architecture and -parameters. If a model is trained on a cloud-based server by a user, its parameters and training hyperparameters may

[1] https://aws.amazon.com/machine-learning
[2] https://azure.microsoft.com/en-in/services/machine-learning
[3] https://cloud.google.com/products/ai

Manuscript accepted in ACM Computing Surveys (CSUR): https://doi.org/10.1145/3595292

be revealed afterwards; however, some MLaaS keep also the user's models secret, making it impossible to transfer the models to the user's device. Amazon and Microsoft Azure e.g. provide two modes for model training: (1) A user does not specify training hyperparameters, but the server spends time searching optimal values for them. The MLaaS does not disclose these after the training. (2) Specified hyperparameters are required; therefore, it takes less time for running and costs less. Wang et al. [32] have shown how to exploit the first mode for stealing the training hyperparameters.

We now briefly describe the methods most frequently targeted in model stealing attacks. Naive Bayes (NB) applies Bayes' theorem with the naive assumption of conditional independence between every pair of features. It uses the maximum a posteriori estimation to obtain the likelihood of a class for a given input. A Decision Tree (DT) is a tree-structured model in which internal (decision) nodes represent conditions on the values of input features, branches represent the decision rules, and leaf nodes represent the outcome. If used for regression, the trees are called Regression Trees (RT). Logistic Regression (LogReg) computes the odds of a class as a linear combination of the features and uses the logistic function to model a binary target variable; it can be extended to multi-class settings (MLogReg). A Support Vector Machine (SMV) constructs a hyperplane that maximises the distance to the nearest training data point. For problems that are not linear separable, a kernel function maps the input samples into a higher-dimensional space, hoping that separation is possible there. Kernels include the linear (SVM-lin), quadratic (SVM-quad), or the radial basis function (SVN-RBF). Some works subsume all linearly separating, binary-class models as *linear binary model* (LBM).

4.2 (Deep) Neural Networks and Deep Learning

Many works in model stealing specifically address *(Artificial) Neural Networks* ((A)NN) which consist of neurons that are organised in layers. The first is called the *input layer*, the last the *output layer*, and all in between are *hidden layers*. The parameters of NNs usually are called *weights*. Architectures frequently considered in model stealing research include:

- Deep Neural Networks (DNNs), i.e. neural networks with at least two hidden layers; often, these are fully-connected feed-forward networks: neurons from one layer can be connected only to neurons on the next layer.
- *Convolutional Neural Networks* (CNNs) are a special case of DNNs often applied to image data. They do not require feature extraction as a data preprocessing step, but can extract local spatial features in an end-to-end learning fashion [33]; computationally, this feature extraction is relatively cheap. CNNs usually contain three types of layers: (1) *Convolutional* layers apply filters to the layer's input and perform spatial feature extraction. (2) *Pooling layers* are used for dimensionality reduction. (3) *Fully connected* layers are performing the classification.
- *Recurrent Neural Networks* (RNNs) allow cyclic connections and support sequential data (of variable length), e.g. handwriting and speech recognition tasks. RNNs have an internal memory considering previous states.
- *Generative Adversarial Networks* (GANs) [34] can be used to generate data; they consist of two networks competing with each other: the *generator* learns to generate samples indistinguishable from the training samples, whereas the *discriminator* learns to distinguish between original and generated data samples.
- *Graph Neural Networks* (GNN) process graph structures [35], e.g. for social network analysis. GNNs can perform node classification, link prediction, or complete graph classification.

4.3 Adversarial Machine Learning

Barreno et al. [36] are amongst the first to explore security issues of Machine Learning and distinguish e.g. between attacks in the the model's training stage versus attacks on a trained model. Biggio and Roli categorised adversarial attacks based on the attacker's goal and capabilities [37] (see Table 1). The goal of an attack can be the model's confidentiality,

Table 1. Attacks against Machine Learning, adapted from [37]

Attacker's capability	Attacker's goal		
	Integrity	**Availability**	**Privacy/Confidentiality**
Test data	Evasion (e.g., adversarial examples)	-	Model extraction/stealing, model inversion, membership inference, …
Train data	Poisoning for subsequent intrusions - e.g., backdoors	Poisoning to maximise error	-

Table 2. Disambiguation of model stealing terminology. The first column gives the term primarily used in the literature and, thus, also in this paper. The second column lists other, equivalent terms used across the literature.

Terminology used in this paper	**Other designations with the same meaning**
Model stealing (extraction) attack [1]	Reverse-engineering attack [21], copy attack [10], exploratory attack [4], inference attack [42], duplication attack [43], mimicking attack [12], model approximation attack [44]
Target model [1]	(Target) oracle [1, 26], classifier (model) under attack [4], secret model [25], victim model [3], original model [45], proprietary model [23], mentor model [12], source model [46]
Substitute model [2]	Adversarial classifier (model) [4], copycat network [10], knockoff model [3], surrogate model [11], extracted model [45], inferred classifier (model) [42], model approximation [23], student model [12], stolen model [47], replicated model [48], clone model [49]
Attacker's data	Fake dataset [10], thief dataset [25], attacker set [8], transfer set [3], proxy data [50], surrogate dataset [51]
Fidelity [7]	Extraction accuracy [1], label prediction match [2], similarity [26], agreement [25], approximation accuracy [45]

integrity, or availability (the so-called "CIA triangle"). Confidentiality attacks are aimed at training data (e.g. model inversion) or the model as intellectual property (architecture and hyper(parameters)). Integrity attacks raise the number of false negatives. The goal of availability attacks is to make the model irrelevant by increasing prediction errors.

- *Poisoning* attacks [38] poison the training data, for instance by flipping the labels or adding some malicious data into the training set. As a consequence, the trained model's accuracy is lower or it can be fooled by samples modified in the same manner as the training data.
- *Evasion* attacks target the prediction phase. By e.g. applying small perturbations to original data [39], an adversary can obtain an adversarial example that is most of the time indistinguishable for humans, but misclassified by the model. These attacks have become prominent for images, but were first executed on email (i.e. text data) [40].
- *Membership inference* attacks [41] determine if a given sample belongs to the training data or not. To do this, an adversary tries to distinguish the differences in the predictions of inputs in the training set and outside of it.
- *Model stealing* (model extraction) reveals a model's hyperparameters resp. learned parameters or steals model behaviour and, thereby, the intellectual property a model constitutes. Model stealing is the focus of this work.

5 TAXONOMY OF MODEL STEALING ATTACKS

In this section, we first provide a unified terminology (Section 5.1), followed by a comprehensive taxonomy of model stealing attacks (Section 5.2).

Manuscript accepted in ACM Computing Surveys (CSUR): https://doi.org/10.1145/3595292

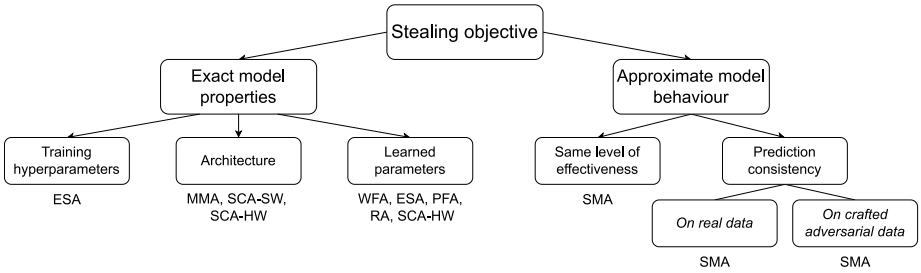


Fig. 2. Taxonomy of model extraction attacks.

5.1 Terminology and Notation

We present a unified terminology in Table 2. We identify the most widely used terms in the literature in the first column and adhere to them in our paper. The second column indicates alternative terms along with a list of works that utilise them. A model that an adversary aims to steal is called the *target model* and is denoted as f. The adversary can use this model as an oracle to collect the *attacker's data* that consists of pairs (x, y). The input x is a data sample that the attacker sends to the oracle. The output y is the prediction of the target model, i.e. $f(x) = y$. One such interaction with the target model is called a *query*. If outputs are the only information one can obtain from f, we assume that an adversary has *black-box* access to the target model, or that f is a *black box*. If the architecture and parameters of the target model are known, we assume *white-box* access to the model, or that f is a *white box*. Any in-between state is called *grey box*. If the attacker obtains a (possibly approximate) copy of the target model, we denote that model with $\hat{f}$.

5.2 Objectives of Model Stealing Attacks

By their objective, as depicted in Figure 2, attacks can be divided into two categories: (1) stealing *exact model properties* (Section 5.2.1), and (2) stealing *approximate model behaviour* (Section 5.2.2).

5.2.1 Stealing Exact Model Properties. Depending on the considered task, the stealing of exact properties can further be distinguished by the stolen *assets*: the learned parameters (e.g. the learned weights of a neural network), training hyperparameters (e.g. a regularisation parameter utilised during training), or architecture (e.g. the arrangement of nodes and layers in a neural network).

Training Hyperparameters. In this category of attacks, an adversary tries to reveal a hyperparameter responsible for the training process. Wang and Gong [32] proposed an equation-solving approach for stealing the regularisation hyperparameter from ridge and logistic regression, SVM and NN (see Section 7.2). Oh et al. trained a *meta-model* that can predict some of the training hyperparameters, such as batch size or optimisation algorithm [52] (c.f. Section 7.7).

Architecture. An architecture stealing attack is usually applied to neural networks, as most other models vary only on training hyperparameters and have a fixed architecture. In this case, "architecture" means the set of hyperparameters that defines the target model structure. In particular, the number of layers, layer type and its characteristics like the size of a kernel are parts of a CNN architecture. Two main approaches to architecture stealing have been proposed in the literature. The first one is the aforementioned meta-model attack [52] that predicts the architecture of the target model by utilizing queries. Other works [53–58] exploit side-channel access to the model. We provide a taxonomy of side-channel attacks, describe the difference from query-based attacks, and introduce key techniques in Section 8.

Learned Parameters. A parameter stealing attack aims to extract parameters of the target model whose structure (architecture) is known. Lowd and Meek were the first to propose a model extraction attack for stealing the weights of a binary classifier [21]. Later, Tramèr et al. [1] extended the idea and introduced equation-solving attacks (see Section 7.2) which allow to extract the exact parameters of (multi-class) logistic regression and Multi-Layer Perceptron. Reith et al. [45] presented an equation-solving attack also for support vector regression with linear- or quadratic kernels. Generally, learned parameter extraction is highly related to other attack types. For instance, it can be applied after an architecture stealing attack to steal a target model with an unknown model type. By performing a successful extraction of model parameters, an adversary automatically obtains *identical* behaviour.

5.2.2 Stealing Model Behaviour. Jagielski et al. [7] classified a subset of those attacks that aim to steal the model behaviour based on their accuracy- and fidelity performance. We generalise from concrete metrics to the goals of obtaining the *same level of effectiveness* as the target model, or trying to be *consistent with the predictions* of the target model; we further distinguish two cases for the latter, depending on what they are tested on. In Section 6.3, we detail concrete, frequently employed metrics to measure these goals, and in Tables 4 and 5, we analyse model stealing attacks based on their performance objectives.

Same Level of Effectiveness. This category covers attacks that aim at approximate stealing and that focus on getting a copy of the target model that reaches the same level of effectiveness. Given the target model f, an attacker aims to create a model $\hat{f}$ that performs similarly to f on the original data. As a result, the attacker can use $\hat{f}$ for solving the same task as f without restrictions (e.g. daily caps or fees). To get a stolen model with similar effectiveness performance, an adversary can use the same model architecture as in the target model [1, 10, 23], the same model type but with a different structure [3, 27], or use a completely different class of models [4, 59].

Prediction Consistency. The second category that aims at approximate behaviour stealing covers attacks that produce a model $\hat{f}$ that predicts outputs consistently with f. Consistent predictions means that for any sample x, the stolen model prediction should coincide with the target model prediction $f(x) = \hat{f}(x)$. Hence, if f misclassifies a sample from the original data, we want $\hat{f}$ to also misclassify it. Depending on the domain of x, we distinguish two sub-categories: consistency on real data and consistency on crafted adversarial data. For the real-data case, we will get a model that has the same effectiveness as the target model and makes the same mistakes, e.g. on the original data. In that sense, f and $\hat{f}$ are more similar than in the previous category. Prediction consistency on crafted adversarial data can be the goal of a model stealing attack which opens the black-box target model for further white-box attacks.

6 MODEL STEALING: THREAT MODEL

A common way to understand the mechanics of a security attack is to model potential threats. Hence, we specify the attacker's motivation (incentives) to perform a model stealing attack and describe potential consequences for the model owner. Then we analyse how the attacker may execute an attack. This step is also helpful for modelling potential defences as it reveals the attack's weaknesses. Finally, we formulate concrete goals for model stealing attacks and define metrics to measure the level of success.

6.1 Attacker's Incentives

We distinguish the following two reasons for a model stealing attack.

Exploit a (partial) copy of the target model. If the target model is only available via API as a black box, there might be some restrictions that prevent API users from unlimited model querying – for instance, daily caps or fees. If

the attacker wants to overcome these, obtaining a copy of the model would be the solution. Another motivation is e.g. stealing a novel architecture, which could help an adversary to get a better model for another task (i.e. not necessarily the one that the target model solves). In this case, the attacker does not steal the model itself, but one of its components.

"Open" the target model for further white-box attacks. An attacker may want to perform an attack that requires white-box access to the target model. In this case, a model stealing attack can be used as an intermediate step. By obtaining a copy of the target, the attacker "opens" the black box to perform some white-box attack on it. Several works exploit model stealing to enable evasion- or poisoning attacks, for example [60–62]. As mentioned in Section 3, we did not include these papers in our classification since – in contrast to those mentioned in Table 3 – they are focused on the subsequent white-box attacks; here, model stealing is not actually studied, but rather used as a preparatory step.

6.2 Attacker's Capabilities

We consider three main aspects regarding the attacker's capabilities: knowledge about the target model and the data it was trained on (the *original* data), actions that the attacker can perform, and the resources available to them.

Attacker's Knowledge. As mentioned in Section 5.1, an attacker might have one of the three types of access to the target model: white-box, grey-box, or black-box. Having white-box access means that there is no reson for stealing the model as it is already known. However, such a model can still be exploited to reveal its training hyperparameters [32]. Grey-box access refers to the situation when the architecture of the target model is known, which is required for some types of attacks (see Section 7, Tables 4 and 5 for more details). However, the default assumption for model stealing attacks is that there is only black-box access, i.e. the only information revealed are model outputs. Some of the model stealing attacks are data-agnostic, not requiring any data at all (see Section 8) or at least no meaningful data (see, for instance, Section 7.2). However, there are many attacks (Section 7.5) for which the quality of the data is important. We focus on the following categories: original data, problem-domain data, non-problem domain data, and artificial data. We describe these categories in detail in Section 7.5.2.

Attacker's Actions. *Queries* are the basic interactions between the attacker and the target model (cf. Section 5.1). We call attacks that use only this type of action as information source *query-based* attacks. These attacks are therefore suitable in an MLaaS setting. An overview of query-based attacks is presented in Section 7. If the attacker has hardware- or software access to the computing resource on which the model is deployed, this opens an additional possibility for a model stealing attack. In these settings, the attacker can exploit *side-channel leakages*, thus performing so-called *side-channel attacks* (SCAs). We present an overview of side-channel attacks in Section 8. SCAs can optionally also use queries as an additional source of information.

Attacker's Resources. As discussed in Section 6.1, there might be restrictions that affect the number of queries an attacker can perform. Hence, (query-based) model stealing attacks are usually considered with regards to their *query budget* – i.e. the number of queries that an attack requires. It is an important task to find a trade-off between the attack performance and its query budget.

Based on their capabilities, we can call attackers *weak* or *strong*. For instance, an adversary who knows about the architecture of the target model and the original training data is stronger than one without that knowledge. It is not always possible to say which capabilities make a stronger attacker (i.e less knowledge and more resources v. more knowledge and fewer resources). However, we can differentiate within the categories knowledge, actions, and resources.

6.3 Attacker's Goals

Every model stealing attack aims to copy the target model or some of its aspects. As discussed in Section 5, on the top level, we separate the attacks into two categories: (1) stealing exact model properties, and (2) stealing approximate model behaviour (Figure 2). In this section, we focus on defining metrics which estimate if a certain goal was reached.

Effectiveness. Depending on the stealing objective, the effectiveness of the attack is measured differently.

- Effective *exact model properties extraction* means that the extracted values are very close or equal to the corresponding target values. Thereby, the most common way to measure the effectiveness is to calculate the absolute difference between target- and stolen values.
- The effectiveness of model stealing attacks that *aim to steal behaviour* is usually measured with one or several metrics; below, we define accuracy, fidelity and transferability and describe their relevance for model stealing attacks. Additionally, different error rates can be calculated, but they are inverse to the metrics mentioned above; hence, we do not focus on them.
  - *Accuracy* shows how close model predictions are compared to the ground-truth values. It is calculated on both target- and stolen models, and results are expected to be similar. However, even equal performance does not mean that the stolen model simulates the original model perfectly – models can still yield different predictions for single data points, and averaged identical accuracies can just be a coincidence. This metric is used to evaluate approximate stealing attacks which aim to reach the same level of effectiveness as the target model.
  - *Fidelity* is calculated as the accuracy of the substitute model when the target model predictions are considered the ground-truth. This metric shows how well the stolen model simulates the original. Furthermore, fidelity does not require ground-truth labelling since it uses only the labels of the target model, which can be observed through querying the model. Consequently, it can be calculated on any data from any distribution without losing its relevance. Fidelity can be used to evaluate the success of an attack which aims to create a model that consistently makes the same predictions as the target model.
  - *Transferability* shows how many adversarial examples generated for the stolen model $\hat{f}$ are also adversarial for the target model f. In other words, let x be a real data sample, $f(x) = \hat{f}(x) = y$, and x^* be an adversarial example for $\hat{f}$, so $\hat{f}(x) \neq \hat{f}(x^*)$. Having $f(x) \neq f(x^*)$ then means that there is transferability between the stolen model and the target model. To measure transferability numerically, one can create a test set of adversarial examples crafted for $\hat{f}$ and measure how many of them are misclassified by f. This metric is used when an adversary wants to reach high prediction consistency on crafted adversarial data for, e.g., targeting a black-box model with an evasion attack. Papernot et al. [63] showed that adversarial examples, crafted via exploiting a stolen black-box model, lead to a high misclassification rate on the target model.

Efficiency. To measure the efficiency of an attack, two metrics are usually used: the number of queries, i.e. the *query budget*, and the *time* needed to carry out the attack. We focus on *the number of queries per parameter* which is a metric often employed for analysing equation-solving attacks (Section 7.2), but is relevant for most query-based attack types.

- The *number of queries* (query budget) corresponds to the price an adversary pays for performing an attack, and is usually calculated only for query-based attacks. A drawback of this metric is that it can only be compared for models of the same size, as the amount of required queries generally increases with the learned parameters
- To account for that, the *number of queries per parameter* is calculated as the query budget divided by the number of learned parameters of the target model, and thus allows to compare attacks across different target model.

- *Timing* is measured less frequently, and it usually means the time of preparation for an attack. For instance, if an API provides 1,000 queries per day for free and an adversary has no budget but wants to apply an attack that requires 3,000 queries, three days will be spent just on data collection. Since this metric can depend on API- or computational resources of an attacker, we do not consider it in this paper (cf. Tables 4 and 5).

7 QUERY-BASED MODEL STEALING ATTACKS

In this section, we discuss query-based model stealing approaches. We group them by the stealing method and analyse each of the methods separately. In particular, we describe the adversary's capabilities for each attack and analyse their efficiency and effectiveness. Table 3 presents the taxonomy of query-based attacks, using the categorisation described in Section 5.2.1. If a paper only proposed an improvement of a known attack and does not perform the attack itself, we did not include it (e.g. [43]). If authors claim their attack to be a behaviour stealing attack, but actually

Table 3. Taxonomy of query-based attacks. Note that the attack goal of stealing learned parameters in most cases also implicitly provides behaviour stealing.

Attack goal	Stealing method	Data domain	Target model	Papers
Training hyperparameters	Meta-model	Image	DNN, CNN	[52]
	Equation-solving	Tabular	RR, LR, SVM, NN	[32]
Learned parameters	Witness-finding		LBM, SVM-poly	[1, 21, 45]
	Equation-solving		LR, MLR, MLP, SVR-lin/quad	[1, 13, 45]
	Path-finding		DT, RT	[1]
	Recovery	Image	ReLU-DNN	[7, 64–66]
Architecture	Meta-model		DNN, CNN	[52]
	Recovery		ReLU-DNN	[65]
Level of effectiveness	Substitute model	Image, Tabular, Text, Sequential, Graph, RL environment	LBM, MLogReg, DT, RF, SVM, NN, CNN, BERT, DRL, chip, GNN, GAN, Encoder	[3, 5, 10–12, 23, 30, 47–51, 59, 64, 67–86]
Prediction consistency	Substitute model	Image, Tabular, Text, Graph, RL enviroment, Recommendation	(M)LogReg, kNN, DT, LGBM, SVM, SVR, NB, NN, CNN, RNN, BERT, DRL, GNN, GAN, SRS	[1, 2, 4–6, 8, 24–27, 42, 44, 45, 47, 68, 70, 75, 81, 84, 86–90]

provide a method for stealing parameters, we define their goal as parameter stealing in Table 3. Such classification does not contradict the one defined by the authors since a high-performing parameter stealing attack usually leads to a model with equivalent behaviour. We define the goal of an attack as the "level of effectiveness" if the performance of the attack is evaluated using accuracy. If the performance is evaluated using fidelity or transferability, the goal of the attack is defined as "prediction consistency". In some of the papers both accuracy and fidelity (or accuracy and transferability) were measured. Those papers belong to both categories simultaneously. Success measures for these methods are discussed below in Tables 4 and 5; this section focuses on the description of the approaches.

7.1 Witness-finding Attack

Lowd and Meek presented the earliest model stealing attack which aims to steal parameters of linear binary models (LBMs) [21]. Considering one positive and one negative sample, they changed the feature values of the positive sample one by one until they found *sign witnesses* - i.e. a couple of samples which are identical except one feature value f and belong to different classes. They set the corresponding weight w_f to 1 or -1 depending on sign witness values

and then used a line search to reveal the relative weight of other features. Since the main step of the attack is to find sign witnesses, we call it the *witness-finding attack* (WFA). An adversary needs the target model architecture and two data samples (one positive and one negative) to perform this attack. Since the attack allows the exact extraction of weights, it produces a model with the same performance as the target model. The main drawback is the inefficiency of the attack: it takes at least 11 queries per parameter (weight) to steal a model (cf. Table 4), which can be problematic for large models. More than a decade later, Tramèr et al. and Reith et al. adapted the witness-finding attack to Support Vector Machines (SVMs) and Support Vector Regression Machines (SVRs) [1, 45].

7.2 Equation-solving Attacks

We define an attack as *equation-solving attack* (ESA) if it is based on setting up a system of equations and solving it. The solution corresponds to the values an adversary wants to extract. Thereby, an ESA appears when the extraction goal is the exact values of the target model – more specifically, either the learned parameters or the training hyperparameters.

ESAs were first utilised by Tramèr et al. for stealing learned parameters of (Multi-class) Logistic Regression, and Multi-Layer Perceptron [1]. They sent data samples $x_1, \ldots, x_n$ to the target model f_w with learned parameters w and used the outputs $y_1, \ldots, y_n$ to construct the system of equations $f_w(x_i) = y_i$, $i = 1, \ldots, n$. The solution of the system reveals the values of the parameters w. Similarly to the witness-finding attack, it reaches perfect extraction scores (cf. Table 4). However, an ESA is more efficient, requiring 1 to 4 queries per parameter depending on the target model type. The attack also requires the architecture of the target model to be known and data samples to query the model. However, since queries are only used to construct a system of equations, the attacker can use samples which are not necessarily real or meaningful. Reith et al. applied this attack for stealing parameters of SVRs with linear or quadratic kernels [45]. Yan et al. used an ESA to steal an MLP under a differential privacy defence [13] which adds noise to the outputs close to the decision boundary [91] (cf. Section 9.2.3). They duplicated queries and, by observing different outputs for the same inputs, created a system of equations, the solution to which approximates the outputs of the target model.

Wang and Gong used an ESA to steal a regularisation hyperparameter, used in the objective function to balance between a loss function and a regularisation term [32]. The learned parameters of the target model should minimise the value of the objective function. Hence, the gradient of the objective function, calculated on the model's parameters, should be (close to) 0. Based on this, an adversary first computes gradients of the objective function and sets them to 0. An obtained over-determined system can be solved by using the linear least square method. To perform this attack, an adversary needs white-box access to the target model. Thereby, for extracting training hyperparameters with having only black-box access, one must first perform an architecture and learned parameters extraction attack.

7.3 Path-finding Attacks

The *path-finding* attack (PFA) was presented by Tramèr et al. for stealing Decision Trees (DTs) and Regression Trees (RTs) [1]. This attack requires prediction labels and an identifier of the leaf that outputs the label. An adversary sends an input x to the target tree t and collects an output $t(x)$ and a leaf identifier id. Then, by varying values of features, the adversary uncovers the conditions that an input sample has to satisfy in order to reach the leaf id. The leaf identifier is required to be able to distinguish between different leaf nodes that lead to the same label being returned, and thus would be indistinguishable by that information alone. Besides the predictions, this attack also recreates the conditions a sample has to satisfy in order to be classified by a specific leaf. If all leafs have unique ids, the stolen tree has identical behaviour to the original one. For regression trees, the authors achieved perfect fidelity scores as all leaves had unique identifiers. For classification tasks, this was not the case and the performance was worse (cf. table 4). Compared with

the aforementioned attacks, the PFA is the most inefficient, requiring 66-317 queries per parameter depending on the target tree. The authors tried to optimise the attack by allowing samples with only some of the features, so-called "non-complete queries". As a result, the number of queries per parameter decreased to 44-91.

7.4 Recovering Attacks

Recovering attacks (RAs) are designed to reveal the weights or even the architecture of (D)NNs with (at least partially) linear activation functions. All works we identified focus on (D)NNs with ReLU activation functions (ReLU-(D)NNs). Milli et al. were the first who theoretically described a recovering attack for stealing parameters of ReLU-(D)NNs with two layers [64]. They claimed that the ReLU network's weights could be viewed as separating hyperplanes. By finding input points that lie on these hyperplanes, one can recover the weights up to their signs. These points are also called critical- [7, 66] and boundary [65] points. Finally, one can recover the sign of the weights by querying samples and solving a system of equations. Jagielski et al. implemented such an attack and showed that the stolen model has very high fidelity and perfect transferability [7] (see Table 4). Rolnick et al. extended this approach for stealing a model of arbitrary depth and revealing its architecture [65]. Carlini et al. compared the model stealing problem with crypto-analysis of block ciphers [66]. They proposed an attack that reveals weights of ReLU networks and requires fewer queries than Rolnick et al. (see Table 4). RAs are the least efficient for small models among the attacks presented in Table 4. It takes approximately 312 queries per parameter to steal a model with 210 parameters [66]. For models with tens of thousands of parameters, this score decreases to 12, still leaving these attacks among the most inefficient ones. However, since a successful RA leads to the exact copy of the target model, it fully reproduces its behaviour.

7.5 Substitute Model Training

This approach has been widely used over the past years by numerous authors (see Table 3 and Figure 1b). The idea is to train a substitute model (cf. Table 2 for alternative terminology used throughout the literature, e.g. "surrogate model") using data labelled by the target model, i.e. using the target model as an oracle for the labels. The substitute model can have the same architecture as the target model; however, this is not necessary and usually not the case. The main condition is rather a syntactical input-output correspondence between models, i.e. the substitute model has to accept the same format of inputs and return outputs in the same representation as the target model.

The first substitute model attack, besides other attacks presented, was introduced by Tramèr et al. [1]. They called the attack "retraining", but in fact they trained a substitute model for LBMs, (multi-class) Logistic Regression, MLP and SVM with RBF kernel, assuming that an adversary knows which target model is used. Reith et al. used the same approach for stealing SVMs and SVRs with linear and quadratic kernel [45]. Another early form of this attack was presented by Papernot et al. [2]. They proposed two substitute models: a more complex DNN and the simpler LogReg for stealing DNN, LogReg, SVM, Decision Trees, and k-NN. The main goal was to train a model with a decision boundary similar to the one of the original model in terms of transferability, i.e. that the an approximation of the original model allows to craft adversarial examples that will, with high probability, fool the target model. The substitute model can thus be used to carry out evasion attacks against the target model.

In the following, we discuss different aspect to substitute model training, namely (i) the substitute model architecture, (ii) the type and domain of training data, and (iii) the strategies to pick the samples and thus required number of queries. Attacks against specific model types and domain-specific models are discussed in Section 7.6.

7.5.1 Substitute Model Architecture. In general, to perform a substitute model attack an adversary first has to pick a model's architecture. Usually, this decision is influenced by the type of model inputs. For instance, if a model is an image classifier, a CNN can be a good choice. Recent works have shown that for a higher stealing success rate, an adversary's model has to be at least as deep (complex) as the target model [3–5, 8, 69]; this applies also to language models and LSTMs [5, 69]. Shi et al. experimented with stealing Naive Bayes and SVM using DNNs, and vice versa [4]. Their results also showed that using a more complex model (DNN) results in a better-performing substitute model.

7.5.2 Substitute Model Training Data. Another important aspect of substitute model training attacks is the dataset used for training. This dataset is often unlabelled, and thus, the target model is first queried with it to observe corresponding labels. The data and obtained labels then form the training data for the substitute model. We can distinguish various scenarios regarding the domain of the problem or data. While in regards to model stealing a clear definition of "domain" is lacking, this concept has been explored in detail in the context of Transfer Learning[92] where "domain" is characterised as consisting of two components [92]: the feature space and a marginal probability distribution. If two domains are different, then they may differ in feature space or the marginal probability distribution. We will use these components to characterise different settings in model stealing. As described in Section 6.2, we distinguish four categories of data: original, problem domain, non-problem domain, and artificial.

Original data is the data that was actually used to train the target model. While some attacks assume the availability of this data, and it corresponds to an attacker with the strongest data knowledge, it may not be a realistic scenario.

Problem Domain (PD) data [2] is data drawn from a distribution that closely resembles the original dataset – for example, using images depicting human faces to steal a model trained for face recognition. This would be data where the feature space is the same, and the marginal probability distribution might be quite similar, but not identical. In most cases, this data is obtained from public data repositories. Depending on the domain, getting such data could still be difficult and expensive. Having problem domain data results in weaker knowledge than original data.

Non-Problem Domain (NPD) data [10] is data sampled from the same type of content as the target model's input, e.g. image data for image models and text data for text models. This data has the same syntactic type, potentially the same feature space, but a rather different marginal probability distribution. If there is any public data of the same modality as the original model, any attacker can use it and hence we can consider NPD data as the weakest knowledge.

Artificial data includes e.g. data produced by GANs [30], data sampled from standard probability distributions [1], noise [67], and data obtained as the result of optimisation of the input space without using any natural samples [47, 88]. Depending on the method used, artificial data can be more or less valuable than, e.g. NPD data. Hence, we can not unequivocally say how strong an attacker is with artificial data without knowing the properties of this data.

Papernot et al. experimented with stealing a model trained on the MNIST dataset [2]. They performed an attack using a handcrafted digit dataset for model querying, assuming that the original data is not available. Correia-Silva et al. proposed an attack called "Copycat" which trains a substitute for a target CNN using NPD data [10]. Orekondy et al. also used NPD data to train a substitute model (a "Knockoff net") for stealing CNNs [3]. In their results, a substitute model trained on the original data performs better than the one trained on NPD data. Later, Zhang et al. explored how the attacker's knowledge affects the attack performance if the attacker's dataset only covers a few classes of the original dataset, or if there is only non-problem domain data available [76].

Gong et al. proposed to leverage a model inversion attack in their SMA called InverseNet [88]. First, they trained a simple substitute model on data selected from public datasets. Based on this model, they selected samples with high

confidence scores as starting point for a model inversion attack to obtain representative samples for each class. After being augmented, these samples are used to query the target model to train a final substitute model.

Having data with a distribution similar to the original dataset can be crucial for the substitute model performance, especially for complex classification tasks. However, since it might be challenging to obtain such data, several works consider so-called "data-free" scenarios, where an attacker creates artificial data, assuming only little or no available natural data. Kariyappa et al. [30] proposed a model stealing attack called "MAZE", which uses a generative model that works similar to GANs, but learns to generate those samples on which attacker and target models disagree the most. They also considered a case in which an adversary has access to a small subset of original training data. In these settings, they trained a Wasserstein GAN to generate artificial samples. This attack performed better and required significantly fewer queries than the initial attack. Yuan et al. proposed an attack called "ES Attack" [47]. It consists of two key steps: estimation of the parameters of the substitute model and data synthesising. The authors presented two methods for crafting artificial samples: the first uses an Auxiliary Classifier GAN for data generation, and the second operates directly in the input space.

Truong et al. launched a data-free model extraction (DFME) attack to steal CNNs [51]. They trained a generator that produces samples in which the target and the substitute models disagree the most. Miura et al. introduced MEGEX – an adaptation of the DFME attack for the case when confidence scores and gradient-based explanations are returned for each query [49]. Sanyal et al. introduced DFMS-HL, an SMA that requires no data and, in contrast to other data-free attacks [30, 49, 51] uses only top-1 labels for stealing CNNs [83]. From a (randomly) initialised GAN, they iteratively generate samples that are labelled by the target model, and are used for both training the substitute model and improving the GAN. The generator was trained using adversarial loss and class diversity losses, whereby the latter allows reaching an almost uniform distribution of generated samples across all classes. Xie et al. also used a GAN to produce samples for their attack called GAME [84]. However, they assumed that (N)PD data is available and used it to train an auxiliary classifier GAN (AC-GAN). Through active learning they determined the most promising classes and then used the AC-GAN to generate samples from these classes to train a substitute model.

While attacking a classification model, an adversary might obtain different levels of detail from the target model, e.g., confidence scores for each class, or just top-1 labels. While confidence scores might contain important information for an attack, but are not always available. A few works proposed how to imitate them having only top-1 labels available. Wang et al. proposed the Black-Box Dissector, an SMA which operates with NPD data [81]. They showed how to estimate the confidence scores of the target model by erasing parts of images (selected by, e.g. the Gradient-weighted Class Activation Mapping (Grad-CAM) method [93]) and aggregating predictions for them. Wang and Lin proposed another method of emulating class probabilities [78]. For each class, they first created a prototypical representation, and then set the probability of belonging to a certain class based on the distance to the corresponding representations.

Roberts et al. provided an SMA using only noise as input for querying [67]. They experimented with noise coming from different distributions: Uniform, Standard Normal, Standard Gumbel, Bernoulli, and Ising. The authors claim that their attack is a parameter stealing attack; however, since they are not stealing model parameters directly but instead observe them by substitute model training, we classify this attack as behaviour stealing attack.

Mosafi et al. proposed an approach using a composite data generation method [12]. They created a new dataset from a public one by superimposing two randomly selected images and used it to train a substitute model. The authors showed that a substitute model trained using the superimposed images – even if only predicted labels are available from the model – performs better than a model trained on regular data with confidence scores, i.e. a more detailed output.

7.5.3 Number of Queries. Another important aspect is the number of samples sent to the black-box model for labelling (i.e. the number of queries), since this is one of the most critical metrics in terms of the attack's efficiency – and also a potential way for a defender to detect attacks. The strongest assumption in this case means having an attacker with no limits on the number of queries. In weaker settings, an adversary has a limited number of queries available and hence applies different techniques to reduce them. These techniques include, for instance, picking the most informative samples for querying, making samples more informative by crafting adversarial examples, or augmenting the attacker's dataset. Where available, we have gathered information on the number of queries (absolute as well as relative to the number of parameters of the model) in Table 5.

Three optimisation techniques have frequently been proposed to pick the most optimal samples for queries: active learning, reinforcement learning, and evolutionary algorithms. Active learning is one of the most widely explored techniques for optimising the querying process. Tramèr et al. were the first to propose optimising queries [1], suggesting two optimisation strategies: besides line-search (samples laying close to the decision boundary), they use adaptive retraining based on active learning. Later Reith et al. used adaptive retraining to steal SVRs [45]. Chandrasekaran et al. [23] showed how two active learning approaches, namely *probably approximately correct* (PAC) and *query synthesis* (QS), can be applied to steal DTs, RFs, LBMs, SVMs. Shi et al. [27] used an active learning approach that reveals uncertain samples which are in turn used as queries to steal FNN (MLP). Pengcheng et al. [26] compared a non-optimised random selection strategy with two active learning methods: least confidence and margin-based. Pal et al. explored active learning strategies such as uncertainty, K-center and DeepFool-based Active Learning (DFAL) to identify the most meaningful samples and, subsequently, use them for their "Activethief" attack [25]. Several other works used active learning together with non-problem domain or artificial data for stealing image classifiers [24, 78, 84].

Reinforcement learning for picking optimal samples was first utilised by Orekondy et. al. in their Knockoff attack [3]. Zhang et al. demonstrated that an attacker applying reinforcement learning and adversarial examples for the querying process performs better than having non-problem domain samples with no query strategy [76]. Barbalau et al. [50] assumed that for an adversary without problem-domain data, it could be difficult to generate samples that are classified with high confidence by the target model. Hence, they applied an evolutionary algorithm to select from samples generated by a GAN those that will be predicted with high confidence.

Another way to optimise queries is to generate samples that help an adversary train a model with better performance. Adversarial examples were first ustilised by Papernot et al., who used Jacobian-Based Data Augmentation (JBDA) to generate new samples that are close to the decision boundary [2]. Juuti et al. [8] and Pengcheng et al. [26] crafted training samples using white-box adversarial example generation techniques, for instance the fast gradient sign method (FGSM, [94]). Yu et al. combined active learning and adversarial examples to reduce the number of queries in their attack FeatureFool [72] which exploits benign and adversarial examples with different, but low target model confidence scores. We note here that, in some papers, the authors used adversarial examples rather to reach a high transferability of the attack, instead of optimising the number of queries. However, this approach is promising for both goals. We also notice that adversarial crafting was mainly applied to original or problem-domain data, which might mean that this approach only works for an attacker with stronger data knowledge.

Data augmentation techniques enlarge the attacker's dataset while spending less queries. Shi et al. proposed to optimise the number of queries by using GAN-based data augmentation [42]. They first queried the target model with a small number of samples, then used the dataset thus obtained to train a GAN, which was in turn used to produce training data for the substitute model training. We distinguish attack approaches that use generative models to increase the quality of the attacker's data, mentioned in Section 7.5.2, from attacks that aim to augment the attacker's dataset. In

the first case, the number of queries is usually much bigger (see Table 5), and an attacker does not optimise the queries, as in the second case.

Besides query optimisation, a few other strategies for attack improvement were proposed. Joshi et al. suggest to use a gradient-driven adaptive learning rate (GDLR) to make the substitute model learning process more efficient [43]. Aivodji et al. [68] showed how MLPs can be stolen using counterfactual explanations (CFEs) in addition to the regular labels. Counterfactual explanations are very similar to adversarial examples, whereby their intent is not to deceive the model, but rather to explain it [95]. These samples, together with initially queried ones, are used for substitute model training. Wang et al. argue that using CFEs and regular queries to train a substitute model leads to shifting the decision boundary far away from the one of the target model [90], since regular queries usually lay far from the decision boundary, while CFEs are close to it. Thus, if a model tries to separate them, the decision boundary shifts towards the regular samples. To overcome this issue, the authors introduced DualCF, an attack that additionally uses CFEs of CFEs (named CCFEs), thus creating samples close to both sides of the decision boundary. They also theoretically proved that a single couple of CFE and CCFE is enough to extract a linear model with 100% fidelity.

7.6 Substitute Model Training Attacks against Specific Model Types and Domain-specific Models

Substitute model attacks can be widely used for different models and data domains. Below we provide an overview of works that explore these attacks in very specific or highly-focused settings. Since most of the research is dedicated to CNNs and classification, attacks and defences on other types of models are less explored and remain an open topic.

Takemura et al. [69] explored attacks against LSTM for both classification- and regression tasks. They trained an LSTM as well as RNN with lower complexity as a substitute model, and showed that the substitute LSTM performs better than the RNN. Krishna et al. [5] explored model stealing attacks against BERT-based models [96], which are commonly used in natural language processing (NLP). They showed how all of transfer learning, a mismatch between target- and substitute architectures, and the attacker's data source affect the substitute model performance. He et al. also explored SMAs against BERT-based models and evaluated their transferability scores [74]. They also showed that the accuracy of the attack remains high even if there is a mismatch in the architectures.

Behzadan and Hsu [6] investigated a model extraction attack against deep reinforcement learning. They utilised a technique called "Deep Q-Learning from Demonstrations" to develop two attacks that learn an adversarial policy - i.e. an imitation of the target policy. They first tried to predict the training algorithm family based on an action sequence, using an RNN. Then, the authors utilised imitation learning [97] to train a substitute DRL model.

Szyller et al. target image transformation models [73], and stole the functionality of GANs for neural style transfer and super-resolution tasks by training a substitute model on image pairs obtained from queries. Hu and Pang trained a substitute GAN on images generated by the target GAN for two scenarios: high-fidelity- and high-accuracy extraction [75]. The main distinction is that for high-accuracy extraction, they applied an additional step of subsampling high-quality samples by using the discriminator of the target GAN.

Liu et al. launched an SMA (called StolenEncoder) against encoders trained in self-supervised settings using contrastive learning [79]. They queried the target encoder with images to obtain the original embeddings and trained a substitute encoder so that its embeddings coincided with the original. To reach better performance while using fewer queries, they augmented images and used embeddings of corresponding non-augmented images as ground truth. Sha et al. considered the same settings and proposed another attack that uses contrastive learning, called ContSteal [80]. They defined a loss function that minimises the difference between target- and substitute embeddings of the same image and

maximises it for different images. Dziedzic et al. explored three attack scenarios against encoders [85]. In some settings, using the original projection head is beneficial for the downstream classification accuracy.

Several works explore model stealing attacks against graph neural networks (GNNs). Since graphs are a collection of nodes and edges, they generally contain more degrees of freedom and using random data as attacker's data is less effective. DeFazio et al. were the first to introduce a GNN model stealing method [87]. They considered a node classification problem and proposed an attack that allows to steal a 2-layer GNN, if knowing a subset of original training data and having access to a 2-hop subgraph of the original graph. Wu et al. [70] also attack GNNs for node classification problems. However, the authors explored different settings regarding the attacker's knowledge: the adversary may know node attributes, the graph structure and/or have access to shadow (auxiliary) data. He et al. [71] also explored these settings while attacking GNNs; however, in contrast to previous works, they considered link stealing attacks that aim to reveal if two nodes are connected. Shen et al. considered SMAs against inductive GNNs, i.e. GNNs that can infer previously unseen unlabelled data [86]. They studied two attack types: with and without knowledge of the target GNN structure. For each of them, they considered three different attackers depending on the available output information.

Teitelman et al. proposed an attack that steals the functionality of a microchip [59]. They introduced an architecture called Deep Neural Tree, which is a combination of a neural network and a decision tree. This model can learn to distinguish different tasks of the chip and provide a certain level of explainability thanks to the tree-like architecture.

Yan et al. proposed a dual-task model extraction attack (DTMEA) [82] for stealing a model that returns both confidence scores and output explanations by training a multi-task CNN with two classification heads: one solves the classification task, and another learns to imitate explanations. Although stealing explanations is not the primary goal of model stealing, the multi-task substitute model reached higher accuracy than a substitute trained only on confidence scores.

Ali and Eshete [44] explored SMA against malware classifiers for Windows Portable Executables (PEs). In particular, they use different data representations for the target model (features extracted from bytes) and a substitute model (images based on bytes-to-pixels mapping). They also experimented with mismatching architectures for target and substitute models and concluded that a similar architecture is not the best choice. In their experiments, using a pre-trained Inception-V3 as a substitute model resulted in higher fidelity than using a custom MalConv model (CNN for Malware detection) that corresponds to the target model architecture. We speculate, that one of the reasons could be that the size of the substitute training set was only 40% of the size of the target training set; hence, transfer learning could play a crucial role. Yue et al. explored SMA against a sequential recommender system (SRS), aiming to open the black-box target model for performing further profile pollution and data poisoning attacks [89].

Aarts et al. considered a different goal for creating substitute models [77]. Instead of creating a high-performing substitute that completely emulates a target model, they proposed to train a substitute to a certain level of effectiveness and then, depending on its confidence scores, use either the substitute or the target model to obtain predictions. This scenario is feasible if an attacker steals the model to launch a competitive API since, in this case, even by delegating some of the queries to the target model and paying for them, the attacker can turn a profit.

7.7 Meta-model Training Attacks

Oh et al. [52] proposed the meta-model attack (MMA) – the first and so far only query-based attack that can reveal information about the target model architecture. They trained a meta-model that, for a given model, predicts details about the target model structure, training setup, and the amount of training data. As dataset for the meta-model, they used a set of candidate CNNs which vary in architecture parameters (type of activation functions, the number of convolutional and fully-connected layers, etc.), optimisation parameters (type of algorithm, batch size), and data parameters (data

Table 4. Performance and other characteristics of query-based attacks exluding SMAs. N/A indicates that the authors did not provide the information

Attack	Model		Data	TM Properties		Stealing Performance		
	Target (TM)	*Attack (AM)*	**Modality**	*Effectiveness*	*Parameters*	*Query budget*	*Efficiency Score*	*Stealing Effectiveness*
WFA [21]	NB	Same as target	Tabular		23k; 1k; 1k	261k; 25k; 23k	11; 25; 23	Exact extraction
	MaxEnt	Same as target			23k; 1k; 1k	119k; 10k; 9k	5; 10; 9	Exact extraction
WFA [1]	LogReg	Same as target	Tabular		d	50d	50	Exact extraction
WFA [45]	SVM-lin	Same as target	Tabular		d	17d	17	Exact extraction
ESA [1]	(M)LogReg, NN	Same as target	Tabular	N/A	d	d; |classes| × d; 4d	1; |classes|; 4	100% fid; 100% fid; 99.99% fid
ESA [45]	SVR-lin/quad	Same as target	Tabular	N/A	d	d; $\frac{1}{2}$ d^2 + $\frac{3}{2}$ d+1	1; $\frac{1}{2}$ d+ $\frac{3}{2}$	100% fid
ESA (dupl. quer.) [13]	LogReg, NN	Same as target	Tabular	N/A	d	|duplications|×d	|duplications|	88-100% fid; 98-100% acc
PFA (compl. quer.; incompl. quer.) [1]	DT	Same as target	Tabular	N/A	26-318[4]	1.7k-101k; 1.1k-30k	19-318; 17-100	86.4-100% fid; 99.65-100% fid
	RT				49-155[4]	6k-32k; 1.8k-7.4k	122-206; 36-48	100% fid; 100% fid
Recovery [7]	ReLU NN	Same as target	Image	94.3-97.7%	12,5k - 100k	$2^{17.2}$ - $2^{20.2}$	12	99.98-100% fid
Recovery [65]	ReLU DNN	Same as target (isomorphic)	Image, Tabular	N/A	N/A	N/A	250-390 (est)	N/A
Recovery [66]	ReLU DNN	Same as target	N/A	N/A	210 – 100k	2^{16} - $2^{21.5}$	30 - 312	100% fid

split, data size). Then the meta-model was trained to represent the correlation between hyperparameters of a model and its performance on specific test samples. Those samples were subsequently used to reveal the hyperparameters of the target model. One peculiarity of the attack is that to successfully steal a hyperparameter, this hyperparameter should be influential for the target model and its value should appear in the training set. For instance, to steal the number of convolutional layers, an adversary has to be sure that the target model is indeed a convolutional neural network and that in the training set of the meta-model there is a model with the same number of convolutional layers. The attack requires significant computational power- and time resources. For instance, to perform hyperparameter stealing on MNIST classifiers, the authors created 10,000 candidate CNNs which took 40 days of training on a GPU. On average, the attack predicted the correct hyperparameter value in 80.1% of the cases, whereas the average chance to guess is 34.9%. Given that the meta-model attack steals the hyperparameters of the model, an adversary needs an additional parameter-stealing attack to obtain a model that approximates the target model behaviour. For the same reason, this attack cannot be compared with other query-based attacks in terms of effectiveness and efficiency.

7.8 Comparison of Query-Based Attack Performance

In this section, we compare the performance of query-based model stealing attacks in terms of effectiveness and efficiency of the attack, with the help of Tables 4 and 5, whereby the latter covers substitute model attacks and the former other query-based attacks. For each of these attacks, the tables provide the type of models they have been applied to – for both the target model (TM) to be stolen and the model type chosen by the attacker, i.e. the attacker model (AM)) – and the data modality considered in the evaluation.

Further, the tables provide details on the performance of the attacks. We report the effectiveness of the target model as reference (by default in % accuracy) as well as the number of parameters to be stolen, i.e. the number of learned parameters of the target model. Also, the reported number of queries is shown; in several works, multiple settings are evaluated, e.g. an attack with a small, medium and large number of queries (and corresponding other effectiveness measures); these are also shown in the table. If both the number of parameters and queries are given, we can compute the relation of queries per parameter as *efficiency score*. Finally, the tables detail the reported stealing effectiveness, i.e. the accuracy, fidelity or transferability (cf. Section 5.2.2), or other scores, e.g. AUC.

There are two categories of data not provided in the table. (1) The effectiveness score of the target model for WFAs (denoted as grey cells). As WFAs produce an exact copy of the target model, the accuracy of the target model equals the

[4]Since DTs and RTs are non-parametric, we use the number of leaves in a tree.

Table 5. Performance and other characteristics of SMAs. N/A indicates that the authors did not provide the information

Attack	Model		Data	TM Properties		Stealing Performance		
	Target (TM)	*Attack (AM)*	**Modality**	*Effectiveness*	*Parameters*	*Query budget*	*Efficiency Score*	*Stealing Effectiveness*
SMA-*/NN (retraining) [1]	MLogReg; NN; SVM-RBF	Same as target	Tabular	N/A	d	100d×|classes|; 100d; 10d-100d	100×|classes|; 100; 10-100	98.24-100% fid
SMA-* (retraining) [45]	SVM-lin/RBF, SVR-RBF	Same as target	Tabular	N/A	d	d; 20d; d-40d	1; 20; 1-40	99-100% fid
SMA-NN [2]	CNNs, LogReg, SVM, DT, kNN	LogReg, CNNs	Image	92-94.97%	60k (est)	6.4k	9 (est)	61-89% fid, 96-97% tr
SMA-NN [4]	NN, SVM, NB	SVM, NB, NN	Text	85.56-96.51%	N/A	859	N/A	97.44 - 97.9% fid
SMA-CNN (Copycat) [10]	VGG-16	Same as target	Image	88.7-95.8%	138m (est)	3m	2×10^{-2} (est)	93.7-98.6% rel acc
SMA-CNN;RNN (Activethief) [25]	CNN	Same as target	Image	N/A	N/A	10k-120k	N/A	64.2%-95.8% fid (10k)84.99-98.54 fid (120k)
	CNN [98]		Text			10k-89k		75.87-77.69 fid (10k)86.21-90.07 fid (89k)
	RNN (GRU)					89k		89.12-93.01 fid (89k)
SMA-CNN [8]	CNN, VGG-16	Same as target	Image	95-98%	CNN: 486k	102k; 6.4k	2.1×10^{-1}; 1.3×10^{-2}	97.9% fid; 39.3 % tr
SMA-CNN (Knockoff) [3]	ResNet34	Same as target*[5]	Image	78.8%	21M (est)	60k	2.9×10^{-3} (est)	76.2% acc (97% rel acc)
SMA-CNN (Knockoff) [11]	ResNet34	Same as target	Image	71.1-98.1%	21M (est)	1.2m (est)	6×10^{-2} (est)	53.5-94.8% acc (75-97% rel acc)
SMA-NN (DS) [42]	N/A	NN	Tabular	N/A	N/A	100	N/A	69.2-72.6% fid
SMA-NN (AL) [27]	N/A	NN	Tabular	N/A	N/A	1k	N/A	80.77% fid
SMA-CNN (DS + AL) [26]	CNN	CNN (simple)	Image	99.24%	N/A	100-25.6k	N/A	47.64-94.19% fid
	ResNet	VGG-16		91%				53.61-79.75% fid
SMA-* (AL) [23]	SVM-kernel	Same as target	Tabular	N/A	N/A	48-1k	N/A	94.5-98.2% acc
	DT			52.1-86.8%		361-244k		73.1-89.4% acc
SMA-CNN (PD, AL) [24]	CNNs	CNNs	Image	N/A	N/A	10k; 30k; 100k	N/A	64.2-95.8%; 78.36-98.18%; 81.57-98.81% fid
			Text			10k; 30k; 89k		58.6-77.67%; 71.8-87.04%; 77.8-90.07% fid
SMA-CNN (DS) [12]	CNN	VGG-16	Image	90.48%	N/A	N/A	N/A	89.59% acc
SMA-CNN (DS) [47]	LeNet5, ResNet18,34	LeNet5, ResNet18,34	Image	91.12-99.10%	60k-60m (est)	N/A	N/A	80.79-93.97% acc (88.66-94.82% rel acc); 92.14-100% tr
SMA-CNN (membership; gradients) [64]	MLogReg, ReLU-NN, CNN	Same as target, permuted	Image: MNIST	93-99% (est)	N/A	CNN: 1k;10; MLogReg: 784;1; ReLU-NN: 10k;100	N/A	93-99% acc (est)
	CNN, VGG-11, ResNet18		Image: CIFAR10	75-90% (est)	11m-15m (est; excl. CNN)	CNN: 10k;100; VGG/ResNet: 10k;1k	10^{-5}-10^{-3} (excl. CNN)	75-88% acc (est)
SMA-CNN [30]	LeNet, ResNet20	WideResNet22	Image	91.04-97.43%	N/A	5M-30M	N/A	82.9-94.32% acc (91-99% rel acc)
SMA-*/CNN (noisy) [67]	CNN	Same as target	Image	88.62-99.03%	N/A	600k	N/A	10.47-95.93% acc (11.81-96.87% rel acc)
SMA-CNN (DS) [50]	AlexNet	half-AlexNet	Image	82.5%	62M (est)	N/A	N/A	79.0% acc
SMA-CNN [44]	CNN	Inception	Image	93%	N/A	16k	N/A	88.65% fid
SMA-CNN (FeatureFool) [72]	N/A	VGG-19-DeepID	Image	77.93%	N/A	2.15k	N/A	76.05% acc (97.63% rel acc)
SMA-CNN (InverseNet) [88]	CNN	Same as target*	Image	N/A	N/A	30k	N/A	95.88% fid
SMA-CNN (DFME) [51]	ResNet34	ResNet18	Image	95.5%	21m (est)	20m	0.95 (est)	88.1% acc (92% rel acc)
SMA-CNN (MEGEX) [49]	ResNet34	ResNet18	Image	95.5%	21m (est)	20m	0.95 (est)	92.3% acc (97% rel acc)
SMA-CNN (DS) [83]	ResNet34	ResNet18	Image	95.5%	21m (est)	8m	0.38 (est)	93.96% acc
SMA-CNN (NPD, Rl) [76]	N/A	N/A	Image	N/A	N/A	10k	N/A	75.4% acc
SMA-CNN [77]	MobileNetV2	Same as target	Image	N/A	3m (est)	131k	4.4×10^{-2} (est)	100% fid (est)
SMA-CNN [78]	ResNet34	Same as target	Image	N/A	21m (est)	30k	1.4×10^{-3} (est)	80.90% acc
SMA-CNN (Black-box Dissector) [81]	ResNet34	Same as target*	Image	91.56%	21m (est)	30k	1.4×10^{-3} (est)	80.47% acc, 82.14% fid, 76.63% tr
SMA-CNN [82]	ResNet50	CNN	Image	92.03%	24m (est)	50k	2×10^{-3} (est)	85% acc (est)
SMA-CNN (GAME) [84]	AlexNet	Half-AlexNet	Image	98.29%	62m (est)	N/A	N/A	75.88% acc (77% rel acc), 76.74% fid
SMA-* (CFEs) [68]	NN	Same as target	Tabular	84.7%	N/A	1k	N/A	94.89 % fid; 83.97 % acc
SMA-* (DualCF) [90]	MLP	Same as target	Tabular	N/A	N/A	329	N/A	99% fid (est)
SMA-RNN [5]	BERT	BERT, XLNet	Text	76.1-93.1%	345m (est)	9.4k-392.7k	3×10^{-5}-10^{-3} (est)	66.8-91.4% acc (87.78-98.17% rel acc); 72.5-92.8% fid
SMA-RNN [74]	BERT	Same as target	Text	97.1%	110m (est)	N/A	N/A	92.8% acc 76.5% tr
SMA-RNN [69]	LSTM	RNN	Image	97.3%	N/A	N/A	N/A	90-97.5% acc
		LSTM	Sequential	$0.899R^2$				$0.85R^2$ (est)
SMA-GNN [87]	GCN	Same as target	Graph	N/A	N/A	70	N/A	64-80% fid
SMA-GNN [70]	GCN	Same as target	Graph	71.3-81.6%	N/A	60-120	N/A	70.8-79.9% acc; 84.6-89.6% fid
SMA-GNN (ShD) [6] [70]	GCN	Same as target	Graph	69.7-81.6%	N/A	60-120	N/A	70.8-83.2% acc; 73.6-83.7% fid
SMA-GNN [71]	GCN	Same as target	Graph	N/A	N/A	10% of nodes	N/A	0.958-0.999 AUC
SMA-GNN [86]	GIN	GIN	Graph	92.4 %	N/A	5.9k (est)	N/A	87.7% acc, 90.6% fid

accuracy of the stolen model. Hence, the relative accuracy is always 100%, and we do not need to know the effectiveness of the target model. (2) Non-available data, denoted as N/A. It corresponds to data that is not provided by the authors of papers. In some cases, we estimated certain data. For example, we could estimate the number of parameters of the target model from a given model architecture name. These cases, together with corresponding efficiency scores, are indicated

[5]Same as target* means that we report results for the same architecture but the authors also provide results for other substitute architectures.
[6]Shadow data

Manuscript accepted in ACM Computing Surveys (CSUR): https://doi.org/10.1145/3595292

in the table with *est.* We also used this notation for attack performance scores extracted from plots or diagrams when no exact numbers were provided.

One observation from Table 3 is that a comprehensive comparison between the proposed methods is difficult due to the lack of a uniform reporting standard. We highlight two issues:

(1) Effectiveness is reported in multiple ways, e.g. absolute accuracy on the original classification task (which is not comparable among different datasets). Relative accuracy and fidelity are more expressive, as they contrast the effectiveness of the stolen model with the original one. Which of these two measures is more important depends on the exact use case. Reporting both would therefore be the preferred approach. For a more extensive analysis, we also suggest to report the transferability rate of the stolen model since it reveals the similarity between the decision boundaries of the target- and adversary models.
(2) It is difficult to properly assess the efficiency of several of the attacks since the literature very often omits important aspects. For example, the absolute number of queries needs to be rather put in relation to the amount of information that needs to be stolen, such as the number of learned parameters of the target model, to compute an average amount of queries required per parameter. However, in many cases at least one of the required numbers for computing the score is not provided.

We note that the efficiency scores are most useful when comparing different attack variations on similar model types since attacks are not directly comparable across model types. As such, it might be feasible for an attacker to spend 10 queries per parameter when stealing a model with 1,000 parameters in total, but such ratios would be prohibitive if stealing a CNN model with millions of parameters. We also want to point out that comparing attacks should take the adversary's capabilities into account. In Table 3, we highlight some settings like the difference between the target- and the adversary model architecture; however, due to limited space, we are omitting for instance the difference between original- and attacker training data.

8 SIDE-CHANNEL ATTACKS

Side-channel (SC) attacks (SCAs) exploit hardware- or software characteristics to reveal the model. Therefore, their performance strongly depends on the device on which the target model is running. SCAs were initially proposed for key recovery attacks in cryptography, e.g. against RSA [112]. Recent usages of SCAs extend to the model stealing domain, where they have most commonly been employed to extract the model architecture; however, some attacks also target other hyperparameters or learned model parameters. While some authors are calling their attack a "reverse-engineering attack", we will use the terminology defined in Section 5.1 and call it a model stealing attack.

Table 6 provides a classification of side-channel attacks based on access to the model and the exploited channels. Generally speaking, for an SCA, an attacker models possible effects of specific causes, e.g. by generating a set of candidate models and observing how their inference influences the respective side-channel. If the attacker is observing effects of an unknown model on the side-channel, this can then be used to learn information about possible causes, e.g. possible model hyperparameters.

8.1 Software Access

While having software access to the device, an adversary can exploit cache- or timing side channels. Both channels can be used to infer the type of computational operations performed; however, attacks based on this can only extract the architecture of a model. Most cache side channels try to manipulate the contents of a cache (shared or otherwise

Table 6. Taxonomy and attack success of side-channels attacks. ✓indicates that the stealing goal is fully achieved, while ∼indicates partial success.

					Success	
Access	**Channel**	**Stealing goal**	**Target Model**	**Ref**	**Arch**	**Params**
Software	Timing	Architecture	CNN	[99]	∼	
	Cache		VGG-19, ResNet-50	[56]	✓	
			VGG-16, ResNet-50	[55]	∼	
			AlexNet, VGG-13/16	[100]	✓	
			MalConv, ProxalessNAS	[101]	✓	
Hardware	PCIe Bus	Parameters	ResNet-34/101/152,NasNet	[54]	✓	
	Memory	Parameters	quantized CNN (ResNet-18/34, VGG-11)	[102]		∼
		Architecture	Alexnet, VGG-16, Resnet-18/50/101	[103]	∼	
		Architecture, parameters	AlexNet, SqueezeNet	[53]	✓	∼
	EM (electro-magnetic)	Parameters	NN	[104]		∼
			AlexNet, VGG-16, (Wide)ResNet-50, Inception-v1/v3, DenseNet, NasNet, Xception, Inception-ResNet-50-2	[105]		✓
		Architecture	AlexNet, VGG-19	[54]	✓	
		Architecture, parameters	Binary NN	[106]	∼	∼
			NN, CNN	[107]	✓	∼
		Architecture, parameters	Decision Tree	[108]	✓	✓
			Binary NN, CNN, VGG, LeNet, AlexNet	[109]	∼	∼
	Power trace		AlexNet, Inception-v3, ResNet-50/101	[57]	∼	∼
		Parameters	NN	[110]		∼
			Binarized NN	[111]		∼
	PCIe	Architecture, parameters	NN, VGG-16, ResNet-20	[58]	✓	✓

accessible, e.g. via cache conflicts) prior to the running of the target process to force a reload of the cache from memory. The timing difference (memory needed to be reloaded or not) can then be used to infer if the target accessed that cache. Well-known attacks of this kind are e.g. *PRIME + PROBE* [113], and *FLUSH + RELOAD* [114]

Duddu et al. [99] exploited timing side channels to extract the depth of the network. Based on this information, they evaluated a set of candidate architectures and selected the one with the most similar prediction behaviour to the target model. This attack requires a membership inference attack beforehand, as original training data is needed to evaluate (and select from) the candidate architectures. The authors applied reinforcement learning to construct the optimal substitute architecture. Hunt et al. [115] used GPU kernel execution time for predicting classification outputs.

Hong et al. [56] used a Flush+Reload side channel to match an observed architecture to a set of candidate architectures by learning a Decision Tree meta-model on the SC attributes, thus demonstrating the ability to steal VGG-19 and ResNet-50 architectures out of 13 candidates. Hong et al. [101] also exploited a Flush+Reload cache SC to reveal a CNN architecture. They extracted the trace of calls of specific Pytorch- or Tensorflow functions that compose an NN. Observed execution times are then mapped onto a computational graph that corresponds to the target model. They demonstrated the attack performance against MalConv- [116] and ProxylessNAS [117] models. Yan et al. used Prime+Probe and Flush+Reload to extract VGG and ResNet architectures [55]. The authors analysed Generalised Matrix Multiply executions and revealed, with the help of a meta-model, DNN hyperparameters responsible for the network architecture such as the kernel size and number of layers. The proposed attack allows to reduce the search space of architecture candidates. Liu and Srivastava [100] also utilised the information leaked via cache side-channel. They introduced a framework in which DNNs are characterised by the patterns of their access to specific caches over time.

Their architecture stealing attack does not require sharing the memory segment between the attacker and the model, unlike e.g. [56], and allows the exact architecture reconstruction (and not only restricting the search space as in [55]).

8.2 Hardware Access

Hardware access to the device on which the model is executed opens a door for more advanced attacks. These side-channel attacks are based on the observation that all computation running on a certain platform results in unintentional physical leakages. These manifest as physical signatures of reaction time, power consumption, or electromagnetic (EM) emanations while the data is manipulated.

Hua et al. [53] were among the first to implement an attack using hardware-access side-channels. They showed that the architecture and parameters of a CNN can be revealed through the inputs and outputs of the accelerator- and off-accelerator memory access patterns, even if the accelerator has a protected memory access (in an enclave). Rakin et al. adapted a rowhammer memory SCA to steal parameters of a CNN quantised to 8-bit [102]. Wang et al. studied architecture extraction attacks using hardware side channels [103]. They explored how model execution events can be observed through hardware behaviour (calling these observations "Arch-hints"), which side channels can be used, and how one can estimate the effectiveness of a given Arch-hint. Then they applied their observations to launch an attack against Unified Memory, i.e. the memory is shared among all processes running on the machine to track the model traffic and use that information to extract the sequence of layers in the target model.

Hu et al. [54] proposed an architecture stealing attack leveraging electromagnetic (EM) emanations or PCI-express bus events as side channel to infer read/write volume, memory addresses and execution time as features of a CNN. They learned the relation between these features and model internal architecture aspects such as CNN layer types and sizes of layers and kernels. Batina et al. [107] proposed an attack for stealing architecture and parameters of NNs, extracting the activation functions, the number of layers and neurons in the layers, the number of output classes, and parameters via an EM channel. Subsequently, their methodology was used by Jap et al. to attack tree-based algorithms [108].

Yoshida et al. showed a parameter stealing attack on a DNN accelerator implemented on an FPGA (field-programmable gate array) [104]. This work shows that an adversary can extract model parameters by exploiting EM leakage even if they are protected by data encryption. Dubey et al. considered a parameter extraction attack against a Binarised Neural Network (BNN) running on a remote multi-tenant FPGA platform via a power SC [111]. Yu et al. [109] combined side-channel and query-based approaches. They stole the architecture of a NN via an EM side-channel and then trained a substitute model using adversarial examples. Their research was extended by Regazzoni et al. [106] who also studied NN structure identification via EM emanations. Xiang et al. [57] leveraged power traces to reveal the architecture of a DNN, estimated sparsity of parameters, and derived the weights. Zhu et al. [58] identified a new attack surface: unencrypted PCIe traffic to observe GPU-based operations. The proposed attack, called "Hermes Attack", succeeds in stealing a DNN model with identical hyperparameters, parameters, and architecture. Li and Merkel investigated how the availability of power side-channel leakage can improve the transferability of a substitute model [110]. They trained a substitute model and compared its performance with a model that also uses power consumption information. The results showed that power information helps to increase the similarity between weights, but not the transferability.

Breier et al. [105] introduced a parameter extraction attack on DNNs obtained via transfer learning [92], i.e. a known architecture and only a few fine-tuned layers. This is achieved through a fault injection attack, specifically an attack flipping a sign bit. The fault injection requires power- or EM leakage to detect the right time for the attack. Then, from the differences of the original output and the sign-flipped output, model parameters are reconstructed.

9 DEFENCES AGAINST MODEL STEALING

In this section, we provide an overview and systematisation of defences against model stealing attacks. Table 7 shows our proposed taxonomy and classifies defence approaches.

Table 7. Taxonomy of defence techniques.

Defence goal	Method	Papers
Detection	Unique model identifier	[46, 118]
	Watermarking	[119–122]
	Monitor-based	[8, 13, 72, 123–128]
Prevention	Basic	[1, 27]
	Re-training from scratch	[5, 11]
	Differential privacy	[13, 91]
	Input perturbations	[75, 129–131]
	Output perturbations	[9, 75, 131–138]
	Model modification	[31, 139–141]

An important distinction between defences concerns their mode: reactive, e.g. *detection* of an (ongoing or past) attack, or pro-active, i.e. *prevention* of an attack. We can further distinguish reactive defences along two goals as follows: (i) ownership verification tries to prove ownership of a stolen model and is mostly achieved by *unique model identifiers* or *watermarking*; it mainly aims at proving past attacks; (ii) attack *detection* tries to establish whether a model is (currently) being attacked and is mostly achieved by monitoring (in itself another reactive method). If pro-active methods are employed, they aim to mitigate an expected attack and usually modify some aspects of the model – the architecture, the learned parameters, the decision boundary, or the overall effectiveness of the model. An important distinction is on whether the model owner has the possibility to influence the model already during its training stage. If so, and depending on the defence asset, knowledge distillation is one strategy for instance. If the trained model is given, then various approaches can modify the output, weights, or even the architecture in a post-hoc fashion. Reactive methods cannot prevent that a model gets stolen, but inform the model owner about the incident. However, detecting an ongoing attack via a reactive monitor might be a trigger for a pro-active defence to mitigate or even halt the attack. This might be more beneficial than applying pro-active methods upfront, since these generally also have a negative impact on legitimate users, e.g. reduced predictive accuracy. It should be noted that a defence's success is not a binary state, i.e. completely preventing that information gets stolen or failing to do so. In many settings, it is sufficient if the defence can, for example, lower the fidelity of the stolen model so that it becomes useless for the adversary.

9.1 Attack Detection (reactive)

9.1.1 Unique Model Identifier. Unique model identifier (UMI) is a reactive approach to prove the ownership of a model, similar to the concept of device fingerprinting [142]. The idea is to identify a unique model property that will transfer to a substitute model during model stealing. A model owner can then verify that a model was stolen by revealing this property, but in contrast to model watermarking (cf. Section 9.1.2), the owner does not need to actively embed it, as it is model-inherent. Maini et al. proposed a defence called Dataset Inference (DI) [118] which allows checking whether a model was trained on a specific dataset. The defence is based on the idea that training samples have a larger distance to the decision boundary than other samples. The model owner can use a subset of the original training data to measure if those samples are far from the decision boundary for the substitute model. If it holds, the substitute model

contains the identifier of the target model and, hence, can be considered as stolen. However, DI is not applicable if the original dataset is publicly available since other models trained independently on this dataset will be recognised as stolen. This issue was highlighted by Li et al., who showed that DI incorrectly classifies a benign model as stolen if it is trained on data that comes from the same distribution as the original data [122]. They proposed embedding some external knowledge into the target model as an alternative solution (see Section 9.1.2). Lukas et al. aimed to find a specific subclass of transferable adversarial examples – termed *conferrable adversarial examples* – to obtain a unique fingerprint from substitute models [46]. Conferrability means that these adversarial examples transfer to substitute models, but not to other independently trained models. Hence, conferrable examples can be used to check if a particular model is a substitute for the target model.

9.1.2 Watermarking. Model watermarking (WM) is another approach to prove ownership of a (stolen) model [143]. In contrast to UMIs, this is usually achieved by actively embedding hidden information in the model that only the legitimate owner knows how to extract. One possible way is to build secret backdoors into the model: during training, a model learns to predict the predefined values for some outlier samples, i.e. the model overfits to specific outliers. Then, knowing these specific samples, one can query the model and recognise the watermark through its predictions.

To resists model stealing, watermarks need to persist during the attack and appear as well in the stolen model. Jia et al. [119] trained a model to extract common features from problem-domain samples and watermarking samples. This approach guarantees that an adversary who queries the model on the problem-domain data distribution will extract watermarks together with model behaviour. Szyller et al. [120] proposed a strategy called DAWN – instead of applying watermarking during the training process, they change, for a small number of queries, the output of the model, thus using queried samples as (dynamic) watermark carriers. Chakraborty et al. proposed another dynamic watermarking strategy called DynaMarks [121]. In contrast to DAWN, the authors alter the probabilities that the target model returns, thus in most cases preserving more utility of the model. The added perturbations are randomised; hence, an attacker cannot bypass this defence by querying the same sample several times. The watermark is then extracted through comparing the distributions of probabilities per class returned by the substitute model and the original protected model.

Li et al. modified a part of the training set by applying Style-GAN while preserving the original labels [122]. Then, by observing model gradients on a modified image, they were able to say if the knowledge about those samples is present in a given model. However, this defence requires white-box access for ownership verification. A recent survey of further watermarking approaches is given by [143].

9.1.3 Monitor-based. Another reactive defence approach is detecting malicious users by analysing the queries. This approach is called monitor [123] or monitoring-based [13]. Kesarwani et al. [123] implemented a monitor which estimates the data space covered by the issued queries, thus inferring a kind of "extraction completeness status" (ECS). The authors used this approach to detect attacks on decision trees. Juuti et al. [8] proposed a defence technique named PRADA which analyses the distribution of queried samples. Their method is based on detecting a deviation from the normal distribution in the distances between queried samples.

Yu et al. proposed a monitor called DefenseNet [72], which is an NN trained to classify if a sample is adversarial or benign. As input features, DefenseNet takes all outputs from each hidden layer of the target model produced during sample forward propagation. Zhang et al. introduced SEAT, a monitor that aims to defend against attacks using adversarial examples [124]. They trained an encoder that checks whether a current query is too close to any of the previous queries and, thus, likely to be an adversarial example. As soon as the number of such detections exceeds a certain threshold, the corresponding user is blocked. Pal et al. proposed a monitor called VarDetect which uses a

variational autoencoder (VAE) [125]. The monitor collects queries, and if the count reaches a certain number, the monitor checks if those queries are coming from a benign (original or PD data) or malicious (artificial, adversarial PD or NPD data) client. Liu et al. proposed SeInspect, a two-stage monitor for image data that first analyses the last batch of queries sent by a user and, if it seems suspicious, analyses the user's whole query history to detect an attack [126]. The authors showed that although a slightly perturbed image can be indistinguishable from the original, the features for these images on the penultimate layer of the target network differ significantly. They used this observation to detect both adversarial examples and NPD images as malicious queries.

Sadeghzadeh et al. utilised a notion of the hardness of data samples to launch a monitor called HODA [127]. The hardness of a sample is determined by the number of epochs required for its prediction to stabilise. The authors showed that in-distribution samples are generally easier to learn than NPD or adversarial examples, so the latter can be detected by measuring their hardness scores. Dziedzic et al. designed a monitor-based defence that increases the effort to query hard examples based on ideas of the Proof-of-Work (PoW) principle [128]. They exploited differential privacy techniques to quantitatively measure the extracted information of a query and created a proportionally difficult puzzle the attacker needs to solve before the query is answered, effectively slowing down the attacker's querying process.

9.2 Attack Prevention (proactive)

Attack prevention approaches are mostly directed against the effectiveness of the attack, i.e. they do not prevent the attacker from obtaining a model, but aim to render the quality of the stolen model too low for it to be useful.

9.2.1 Basic Defences. This section covers basic defence approaches, all of which are based on simple ideas and can be easily implemented. Tramèr et al. [1] proposed defences with low implementation overhead that can be applied to models that return detailed class prediction information, such as soft-max outputs or logits. One approach is to return only the label predicted for the sample, but no additional information.

9.2.2 Training from Scratch. Atli et al. [11] explored defence techniques against Knockoff nets [3] (cf. Section 7.5). In the original version of the attack, the target models were pre-trained on ImageNet; ImageNet was also used by the attacker to query the target model. Atli et al. trained two models with a specific architecture for the data domain from scratch; attacking them with Knockoff nets was then less successful. A similar approach was explored by Krishna et al. [5] as a countermeasure against their attack on BERT-based models. They trained a model from scratch [92] and observed that the F1 score of the stolen model had also decreased.

9.2.3 Data perturbation. Several studies showed how data perturbation can be used to defend against model stealing (see Table 7). The main idea of this approach is to make the model predict an inexact output while preserving integrity. There are two types of data to perturb: data fed to the model, i.e. input, or the model prediction, i.e. output.

Wang et al. introduced the concept of Information Laundering (IL) [131]. They considered input and output perturbations simultaneously to achieve two goals: (i) hide the predictions of the target model to increase its confidentiality, and (ii) preserve the utility of the model. The work theoretically describes the optimal distribution of input and output perturbations such that both goals are achieved.

Input Perturbation (IP). Grana analytically proved that perturbations added to inputs prevent logistic- and linear regressions from parameter stealing [129]. Guiga and Roscoe [130] protect image models by adding noise to the unimportant pixels selected by the Gradient-weighted Class Activation Mapping (Grad-CAM) method [93]. Hu and Pang proposed two defences for GAN protection [75]. The first defence takes several input queries and replaces each

of them with an input obtained as a result of linear interpolation of two original queries. The second defence applies constraints on inputs such that outputs can belong only to a predefined set (e.g., generating faces with only green eyes).

Output Perturbation (OP). Tramèr et al. [1] provided a basic form of this defence, namely rounding the predicted scores; they state, however, that it is not a promising strategy. A more advanced form of OP was proposed by Orekondy et al. [9] who named their defence Maximising Angular Deviation (MAD). The main idea is to perturb output confidence scores to make the gradient maximally far from the original. Lee et al. [132] proposed to use the reverse sigmoid activation function as a defence. A specific characteristic of that function is that it maps different logit values to the same probability. This leads to wrong gradient values and complicates the stealing process.

Shi et al. [60] proposed a defence against decision boundary stealing. They claim that flipping some labels in the training set can make the model more robust against evasion attacks and, thus, against revealing the decision boundary. Kariyappa and Qureshi [133] proposed a more advanced approach than Shi et al. [60]: wrong predictions are returned only for queries that are out of distribution. Chen et al. utilised an adaptive softmax transformation to perform another OP defence called DAS-AST [134]. By modifying softmax outputs, they changed the distribution of samples obtained by an attacker, misleading them from the decision boundary.

One of the earliest defences was reported by Alabdulmohsin et al. in 2014 [138], when the authors explored the security of SVMs against several types of adversarial attacks. They proposed to train several models and randomly pick one of them to produce an output. Kariyappa et al. extend on that by a defence called Ensemble of Diverse Models (EDM) [135]. They train multiple models that all accurately classify in-distribution samples, but are trained to on purpose predict diverse values for out-of-distribution samples. Since the decision boundaries of those models different, the accuracy of a substitute model trained on out-of-distribution data is decreased.

Lee et al. considered an OP defence called DeepDefence for target models that return probabilities and gradient-based explanations in the form of attribution maps (such as Grad-CAM) [136]. They proposed to perturb the gradients of the target model to make them orthogonal to the original ones, while preserving the order of the top-k probabilities and the values of an attribution map. Mazeika et al. argued that some highly-effective OP defences perturb confidence scores too much, thus harming benign users [137]. In contrast, they introduced a defence called GRAD^2, which adds perturbations that direct the substitute model training in a predefined (non-optimal) direction while preserving the number of total changes below a certain threshold.

Hu and Pang devised a concept of OP for GANs [75]. They proposed to add noise to images, apply the Gaussian filter, and JPEG compression.

Differential Privacy. Zheng et al. [91] used a form of differential privacy [144] against behaviour stealing. Their main idea is to make outputs of all samples that are close to the decision boundary indistinguishable from each other. This is achieved by adding perturbations to these outputs through a so-called "boundary differential privacy layer" (BDPL). Yan et al. [13] broke BDPL with their query-flooding parameter duplication attack (QPD) (cf. Section 7.2) and, subsequently, proposed a new defence called MDP which combines differential privacy (as in [91]) with monitoring to mitigate the QPD attack. If an attack is assumed by a monitor, the amount of noise that should be added to the data is dynamically determined. This makes the perturbation less predictable and, thus, determining the true output more difficult.

9.2.4 Model Modification. Contrary to perturbing the data, which aims at reducing the precision of the stolen models' behaviour, one can modify the model architecture and/or -parameters. The motivation for protecting architectures can e.g. be that an architecture is novel and has certain advantages over others. The main goal of the defender is thus not to

protect one specific trained instance of this architecture (i.e. the learned model parameters) or training hyperparameters, but the general architecture itself, as this should prevent an attacker to apply it to a different domain.

Xu et al. [31] proposed a defence that simulated a CNN feature extractor using a shallow sequential convolutional block and used it to train a smaller model with similar performance, applying ideas from Knowledge Distillation. Lin et al. [139] proposed a strategy inspired by secret sharing in cryptography. A model is first transformed into what they refer to as a *bident model structure*, i.e. a model with two independent branches which merge before the output layer. Each sub-model receives the same input, and the sub-models' outputs are merged into a single output. Given the output and one sub-model, it is impossible to reconstruct another sub-model and, therefore, the whole model. Chabanne et al. [140] investigated a defence against recovering attacks (Section 7.4). The authors added redundant layers to a CNN with ReLU activation functions that do not change the functionality but make the model more complex and, thus, more difficult to steal. They further showed that the modified model's decision boundary differs from the original model's decision boundaries, but keeps a similar functionality. Szentannai et al. [141] implemented a defence for NNs with fully connected layers. The authors proposed to transform a model into a functionally equivalent model, but with so-called "sensitive weights" which make the model less robust and, thus, behaviour stealing more difficult. To do this, they added deceptive neurons to the network which add noise to individual layers, but cancel each other out in the overall effect.

10 ANALYSIS OF ATTACKS AND DEFENCES

In this section, we analyse how attacks and defences compare against each other. To this end, we propose two guidelines: (1) how to steal a model, and (2) how to protect it against model stealing. Further, based on results reported in the literature, we show how model stealing attacks and defences fare against each other.

In related work, Duddu and Vijay Rao [145] proposed a framework based on a Bayesian network to quantitatively estimate information extracted from a target DNN model through model stealing attacks. Based on the results, the most prominent combinations are equation-solving attacks with either a meta-model attack or a side-channel attack. It is worth mentioning that this analysis did not explore if these combinations are possible in practice. For instance, the meta-model attack was applied for CNNs, whereas none of the equation-solving attacks targets CNNs.

10.1 Guideline on How to Steal a Model

As a summary of our attacks analysis, in Figure 3 we provide a guideline for the best attack based on the stealing objectives and attacker's capabilities. Depending on the objective of the attacker, an exact (for architecture, training hyperparameter, or learned parameter stealing) or approximate (for the level of effectiveness or prediction consistency stealing) attack approach is to be chosen. Approximate extraction does not necessarily require information on the target model type, given that model-agnostic generic approaches, denoted as *SMA** are available. Having more detailed information on the domain or type of data can give an indication on which specific model type is well suited. Then, a method specifically designed to steal e.g. recurrent neural networks (*SMA-RNN*) can be employed. In the group of exact extraction attacks, stealing training hyperparameters requires knowledge of the model type and learned parameters. With this information available, attacks addressing specific model types such as logistic regression or SVM can be carried out. Similarly, for stealing model parameters, attacks specialised in specific model types have been proposed, e.g. witness-finding attacks (*WFA*) target linear binary model types, and path-finding attacks can be used to steal Decision or Regression Trees. Stealing the architecture mostly applies to neural networks, where hyperparameters define the layers, neurons, activation functions, etc. It can be achieved in two ways: If the attacker can issue (black-box) queries to the model, a meta-model attack, which builds on a knowledge base of known architectures, can be used. Such a

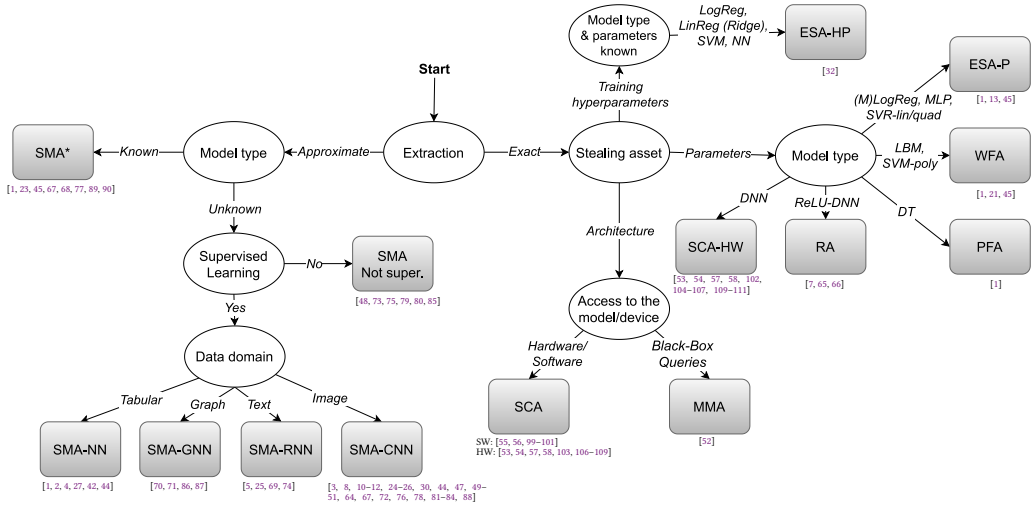


Fig. 3. A comprehensive taxonomy of model stealing attacks, in the form of an "attacker's guide". Abbreviations used: SMA - Substitute Model Attack, SCA - Side-Channel Attack, MMA - Meta-Model Attack, RA - Recovering Attack, PFA - Path-Finding Attack, WFA - Witness-Finding Attack, ESA-(H)P - Equation-Solving Attack - (Hyper)Parameters

knowledge base needs to cover many different architectures, and is thus expensive to obtain. If such an approach is not feasible, but a hardware- or software side-channel access is available, this can be utilised.

10.2 Guideline on How to Protect a Model

Figure 4 provides a guideline on choosing a defence strategy according to particular goals and conditions. Hence, the considered defence taxonomy (Table 7) has two branches on the top level: reactive and pro-active defences. Depending on the availability of the model training stage, there are different approaches how a model owner can mitigate an attack that targets a certain asset. For instance, if the owner wants to defend the architecture of the model, defences that modify the target model architecture are the best option since they "hide" the original architecture. If the owner's primary goal is to track or detect malicious users of an API, then unique model identifier, model watermarking or monitor defences can be applied. However, if an adversary never makes a stolen model public, unique model identifiers and model watermarking are useless. Further, monitors may detect an ongoing attack too slowly and issue a warning of potential danger only when the target model has already been stolen. A combination of different defence techniques could lead to a better protection level. Following the guideline, one can choose suitable defences and combine them into a potentially more powerful defence.

10.3 Attacks and Defences Lineup

Tables 8 and 9 provide a lineup of query-based attacks against reactive and pro-active defences correspondingly, indicating which attack can be mitigated by what defence, and which defence has already been broken by another attack. Rows in Tables 8 and 9 correspond to query-based attacks, while columns correspond to defences. Whenever a defence in a column d was shown to mitigate the attack in row a, we put a ✓ mark in cell (a, d) and a reference to the respective paper. If the mitigation was not shown, but a claim has been made without experimental demonstration, we indicate this with an additional ~. If the authors only speculate about the (in)effectiveness of defences, we denote

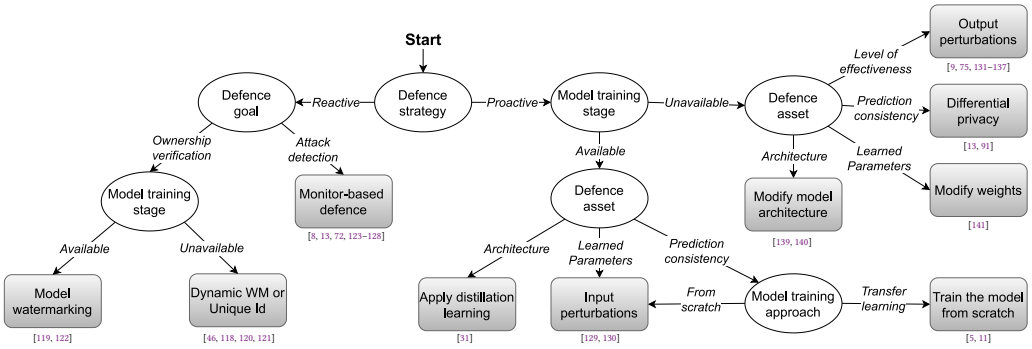


Fig. 4. Comprehensive taxonomy of model stealing defences, in the form of a "model protection guideline"

Table 8. Comparison of attacks and reactive defences. ✓ - attack is mitigated; ↘- defence has limited effect; ✗ - defence is broken; ~- claim made by the authors; ≀- speculation made by the authors

Attack/Defence types	Monitor								WM		UMI	
	ECS [123]	PRADA [8]	DefenseNet [72]	SEAT [124]	VarDetect [125]	PoW [128]	SeInspect [126]	HODA [127]	DAWN [120]	[119]	[46]	DI [118]
ESA-P (dupl. quer.) [13]	↘ [13]											
PFA [1]	✓ [123]											
SMA-* (retraining) [1, 45]		✓ [8]			✓ [125]		✓ [126]					
SMA-NN [2]		✓ [8]		✓ [124]	✓ [125]	✓ [128]	✓ [126]	✓ [127]			✓ [46]	
SMA-CNN Copycat [10]					✓ [125]	✓ [128]						
SMA-CNN (PD, AL) [24]	✓ [24] ≀	✗ [24] ≀										
SMA-CNN (DS) [47]		✗ [47]										
SMA-CNN;RNN (Activethief) [25]	✗ [25] ~	✗ [25]			✓ [125]							
SMA-CNN [8]		✗ [8]		✓ [124]	✓ [125]		✓ [126]	✓ [127]	✓ [120]			
SMA-CNN (Knockoff) [3]					✓ [125]	✓ [128]	✓ [126]	✓ [127]	✓ [120]		✓ [46]	
SMA-RNN [5]									✓ [5]			
SMA-CNN (FeatureFool) [72]		✗ [72]	✓ [72]	✓ [124]								
SMA-CNN (DFME) [51]				✓ [124]		✓ [128]						
SMA-Encoder [85]		↘ [85] ~				↘ [85] ~				✓ [85]		↘ [85] ~

this with an additional ≀. If a defence was proven or claimed to be completely ineffective against an attack, we indicate this with a ✗ mark. A mark ↘ means that a defence is partially effective: either the attack performance has fallen only slightly (less than 5%), or, if the defence was a monitor, the attack was detected too slowly. If a defence has been broken at least for one considered dataset, we mark it as broken. If a cell (a, d) is empty, it means that there is no information available about the effectiveness of the defence d against the attack a, and vice versa.

In this lineup, we did not include papers that present only theoretical results, or do not provide a specific lineup against an attack. For instance, we did not include defence papers which consider their defences against a generic class of attacks without referring to a specific attack paper. We also omit defences which are described in attack papers if they are either attack-specific and not effective, or described in insufficient detail.

From Tables 8 and 9, we can observe that the basic defence method proposed in [1], which relies on returning labels instead of confidence scores, has been broken by several attacks and, thus, seems unreliable. Regarding other defences, several early monitors – while initially successful against early attacks – have since shown to be ineffective against more recent strategies developed to specifically counter them [8, 123]. However, none of the monitors published afterwards is broken. A few defences are, at the time of writing, not broken by specific attacks; however, we note that

Table 9. Comparison of attacks and pro-active defences. ✓ - attack is mitigated; ↘- defence has limited effect; ✗ - defence is broken; ∼- claim made by the authors; ≀- speculation made by the authors

Attack/Defence types	Basic	Data Perturbation									Diff. Privacy		No TL	Model Modification	
	[1]	IP [130]	OP [132]	OP [133]	OP (MAD) [9]	OP (DAS-AST) [134]	OP (EDM) [135]	IP, OP [75]	IP+OP (IL) [131]	OP ($GRAD^2$) [137]	BDPL [91]	MDP [13]	[5, 11]	Distill. [31]	Parasatic [140]
WFA [1, 21, 45]											✓ [91]				
ESA-P [1, 45]	✗ [1] ✗ [45] ∼	✓ [130]							✓ [131]			✓ [13]			
ESA-P (dupl. quer.) [13]	✗ [13]										✗ [13]	✓ [13]			
PFA [1]	✗ [1]														
RA [7, 64–66]															✓ [140] ∼
SMA-* (retraining) [1, 45]									✓ [131]		✓ [91]				
SMA-NN [2]				✓ [133]	✓ [9]	✓ [134]	↘ [135]								
SMA-CNN (PD, AL) [24]			↘ [24] ≀											✗ [24] ≀	
SMA-CNN (DS) [12]			✗ [12] ∼												
SMA-CNN (DS) [47]	✗ [47]														
SMA-CNN;RNN (Activethief) [25]			✗ [25] ∼												
SMA-CNN [8]	✗ [8]				✓ [9]	✓ [134]	↘ [135]								
SMA-CNN (Knockoff) [3]	↘ [3] ✓ [11]			✓ [133]		✓ [134]	✓ [135]			✓ [137]			✓ [11]		
SMA-CNN [30]	✓ [30] ∼		✓ [30] ∼	✓ [30] ∼	✓ [30] ∼										
SMA-RNN [5]	✗ [5]														
SMA-CNN (Black-box Dissector) [81]				✗ [81]	✗ [81]										
SMA-GAN [75]								✓ [75]							
SMA-Encoder (StolenEncoder) [79]	✗ [79]				✗ [79]										
SMA-Encoder [85]					↘ [85] ∼										

the current lineup is not complete – many combinations have simply not been considered, and their outcome would thus be unclear. Further, we want to mention that there is likely a difference in the exact outcome depending on who is performing the lineup, i.e. whether a novel attack tries to break a defence, or a defence tries to show that it is effective against an attack. Depending on who is in the driver's seat, it might be that the general knowledge of the technique (e.g. the importance of certain parameters) as well as the invested effort might favour the attacker or the defender. This again highlights the need for a systematic and "independent" assessment of the effectiveness of defence methods.

11 CONCLUSIONS

In this paper, we provided a comprehensive overview and systematisation of attacks and defences related to model stealing (model extraction). We first provide a common terminology, unifying the disparity of notions used across the literature. We explored the conditions, methodology, and goals of model stealing approaches and subsequently classified them, showing which attacks are possible in specific settings. This resulted in a comprehensive taxonomy and guideline. Moreover, we extensively analysed defence approaches and developed guidelines for choosing the most effective defence strategy. We then compared which defence is mitigating – resp. broken by – current attack strategies. Based on our survey and analysis, we observe several research issues, challenges, and future directions of research.

We observe a general *lack of standardised and systematic methodology in the research on model stealing*, and especially on reporting of both attacks and defences. For attacks, many works omit important details on the efficiency of the attacks. For example, for query-based attacks, it is often not stated how many queries are required, or how complex the model to be stolen is. Thus, it is difficult to compare efficiency between different methods.

We thus propose a methodology for conducting research in this field, based on the contributions in this paper: (1) Research papers should use a **common notation**; the terminology and taxonomies for attacks and defences proposed in this paper would be fitting candidates. (2) Using this terminology, research contributions should provide a **detailed threat model**, specifying the goal of the attacker (e.g. following our taxonomy) as well as their capabilities, e.g. the

knowledge of the model, the actions (e.g. querying or side-channel), and the resources (e.g. the query budget). (3) Having defined a concrete goal motivates which criteria should be used to **measure the attack effectiveness and -efficiency**. If no concrete goal can be defined, e.g. to not limit the attack to a specific application, we do encourage future works to be more comprehensive and measure all applicable criteria we have outlined in Section 5.2.2, such as fidelity, accuracy and transferability, to enable broad comparison with other works. We note that the methodology in evaluation and reporting results lags behind especially for side-channel attacks, which currently mostly consists of anecdotal evidence and qualitative claims, but does not offer much in terms of quantitative, dependable results.

In terms of methodology, a *unified evaluation framework* would enable more comparability and, thus, progress in techniques; this should include e.g. benchmark datasets, trained models, and open source- and unified implementations of attack- and defence approaches, as for instance in a recent initiative for adversarial examples [146].

Regarding defences and their performance against various attacks, we observe that the *current experimental lineup between attacks and defences is rather incomplete*. As many combinations were not studied, it is difficult to obtain a clear picture of the overall effectiveness of the proposed methods. A **large-scale, analytic and empirical evaluation** would help to significantly advance this aspect. As future research challenge, *adaptive attackers* – who are widely studied in evasion attacks against Machine Learning models [147] and constitute attackers who are aware of a certain defence and try to counter it – are not yet widely considered in model stealing. For example, against dynamic watermarking, an attacker could be utilising defences against data-poisoning-based backdoor attacks, such as [148].

Another future research challenge is the rather large *gap of dedicated defences against side-channel attacks* (SCA). The main strategy currently used is to rely on classical IT security measures, i.e. a stringent access control, both on the hardware- and software level, or the use of dedicated infrastructure (or effective isolation of processes) to avoid attacks that are exploiting e.g. shared memory access. As these strategies are likely limited in scope, there is, however, a need for pro-active methods that are model-inherent and can reduce the effectiveness of SCAs.

Overall, we note that there are both a large number of possible attacks against the investment made into the creation of trained ML models, but also defence options – but both are still requiring substantially more structured investigation. However, having a structured taxonomy allows to more systematically address the problem space and provides guidance not only for the attacker, but also the defender. Failing to consider and invest resources into these challenges may well lead to a shocking awakening if and when some attacker might subtly tell us they know (in much more detail than we wanted them to) what we invested in our GPU cycles in last summer, and that our model was gone in 60(k) queries – or to allow *us* to tell *them* that *we* know what *they* tried to do to our model last summer.

ACKNOWLEDGMENTS

This work received funding from the European Union's Horizon 2020 research and innovation programmender grant agreement No 826078 (FeatureCloud). This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains. SBA Research (SBA-K1) is a COMET Center within the COMET - Competence Centers for Excellent Technologies Programme and funded by BMK, BMAW, and the federal state of Vienna. COMET is managed by FFG.

REFERENCES

[1] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium*, Austin, TX, USA, 2016. USENIX Association.

[2] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *ACM Asia Conf. on Computer and Communications Security (ASIA CCS)*, Abu Dhabi, United Arab Emirates, 2017. ACM.

[3] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff Nets: Stealing Functionality of Black-Box Models. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019. IEEE.

[4] Yi Shi, Yalin Sagduyu, and Alexander Grushin. How to steal a machine learning classifier with deep learning. In *IEEE Int. Symposium on Technologies for Homeland Security (HST)*, Waltham, MA, USA, 2017. IEEE.

[5] Kalpesh Krishna, Gaurav Singh Tomar, Ankur Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves of Sesame Street: Model Extraction on BERT-based APIs. In *Int. Conf. on Learning Representations (ICLR)*, Virtual Event, 2020.

[6] Vahid Behzadan and William Hsu. Adversarial Exploitation of Policy Imitation, 2019. arXiv:1906.01121.

[7] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High Accuracy and High Fidelity Extraction of Neural Networks. In *USENIX Security Symposium*. USENIX Association, 2020.

[8] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. PRADA: Protecting Against DNN Model Stealing Attacks. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, Stockholm, Sweden, 2019. IEEE.

[9] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks. In *Int. Conf. on Learning Representations (ICLR)*, Virtual Event, 2020.

[10] Jacson Rodrigues Correia-Silva, Rodrigo F. Berriel, Claudine Badue, Alberto F. de Souza, and Thiago Oliveira-Santos. Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data. In *Int. Joint Conf. on Neural Networks (IJCNN)*, Rio de Janeiro, 2018. IEEE.

[11] Buse Gul Atli, Sebastian Szyller, Mika Juuti, Samuel Marchal, and N. Asokan. Extraction of Complex DNN Models: Real Threat or Boogeyman? In *Int. Workshop on Engineering Dependable and Secure Machine Learning Systems (EDSMLS)*, New York City, NY, USA, 2020. Springer International Publishing.

[12] Itay Mosafi, Eli Omid David, and Nathan S. Netanyahu. Stealing Knowledge from Protected Deep Neural Networks Using Composite Unlabeled Data. In *Int. Joint Conf. on Neural Networks (IJCNN)*, Budapest, Hungary, 2019. IEEE.

[13] Haonan Yan, Xiaoguang Li, Hui Li, Jiamin Li, Wenhai Sun, and Fenghua Li. Monitoring-based Differential Privacy Mechanism Against Query Flooding-based Model Extraction Attack. *IEEE Transactions on Dependable and Secure Computing*, 2021.

[14] Xueluan Gong, Qian Wang, Yanjiao Chen, Wang Yang, and Xinchang Jiang. Model Extraction Attacks and Defenses on Cloud-Based Machine Learning Models. *IEEE Communications Magazine*, 58(12), 2020.

[15] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. SoK: Security and Privacy in Machine Learning. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, London, 2018. IEEE.

[16] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. Review of Artificial Intelligence Adversarial Attack and Defense Technologies. *Applied Sciences*, 9(5), 2019.

[17] Muhammad Shafique, Mahum Naseer, Theocharis Theocharides, Christos Kyrkou, Onur Mutlu, Lois Orosa, and Jungwook Choi. Robust Machine Learning Systems: Challenges,Current Trends, Perspectives, and the Road Ahead. *IEEE Design & Test*, 37(2), 2020.

[18] Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. Towards Security Threats of Deep Learning Systems: A Survey. *IEEE Transactions on Software Engineering*, 2021.

[19] Inaam Ilahi, Muhammad Usama, Junaid Qadir, Muhammad Umar Janjua, Ala Al-Fuqaha, Dinh Thai Hoang, and Dusit Niyato. Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning, 2021. arXiv:2001.09684.

[20] Mihailo Isakov, Vijay Gadepally, Karen M. Gettings, and Michel A. Kinsy. Survey of Attacks and Defenses on Edge-Deployed Neural Networks (HPEC). In *IEEE High Performance Extreme Computing Conf.*, Waltham, MA, USA, 2019. IEEE.

[21] Daniel Lowd and Christopher Meek. Adversarial Learning. In *ACM SIGKDD Int. Conf. on Knowledge Discovery in Data Mining (KDD)*, Chicago, Illinois, USA, 2005. ACM Press.

[22] Burr Settles. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[23] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and Songbai Yan. Exploring Connections Between Active Learning and Model Extraction. In *USENIX Security Symposium*. USENIX Association, 2020.

[24] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. A framework for the extraction of Deep Neural Networks by leveraging public data, 2019. arXiv:1905.09165.

[25] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. ActiveThief: Model Extraction Using Active Learning and Unannotated Public Data. In *AAAI Conf. on Artificial Intelligence*, New York, NY, USA, 2020.

[26] Li Pengcheng, Jinfeng Yi, and Lijun Zhang. Query-Efficient Black-Box Attack by Active Learning. In *IEEE Int. Conf. on Data Mining (ICDM)*, Singapore, 2018. IEEE.

[27] Yi Shi, Yalin E. Sagduyu, Kemal Davaslioglu, and Jason H. Li. Active Deep Learning Attacks under Strict Rate Limitations for Online API Calls. In *IEEE Int. Symposium on Technologies for Homeland Security (HST)*, Woburn, MA, 2018. IEEE.

[28] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, Philadelphia, PA, USA, 2006. ACM Press.

[29] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges. *IEEE Signal Processing Magazine*, 35(1), 2018.

[30] Sanjay Kariyappa, Atul Prakash, and Moinuddin K Qureshi. MAZE: Data-Free Model Stealing Attack Using Zeroth-Order Gradient Estimation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, 2021. IEEE.

[31] Hui Xu, Yuxin Su, Zirui Zhao, Yangfan Zhou, Michael R. Lyu, and Irwin King. DeepObfuscation: Securing the Structure of Convolutional Neural Networks via Knowledge Distillation, 2018. arXiv:1806.10313.

[32] Binghui Wang and Neil Zhenqiang Gong. Stealing Hyperparameters in Machine Learning. In *IEEE Symposium on Security and Privacy SP*, San Francisco, CA, 2018. IEEE.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)*, Montréal, Canada, 2014. Curran Associates, Inc.

[35] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1), 2009.

[36] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *ACM Asia Conf. on Computer and Communications Security (ASIA CCS)*, Taipei, Taiwan, 2006. ACM Press.

[37] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84, 2018.

[38] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, et al. Exploiting Machine Learning to Subvert Your Spam Filter. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, San Francisco, CA, USA, 2008. USENIX Association.

[39] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Int. Conf. on Learning Representations (ICLR)*, Banff, AB, Canada, 2014.

[40] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *Conf. on Email and Anti-Spam*, Stanford, CA, USA, 2005.

[41] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, 2017. IEEE.

[42] Yi Shi, Yalin E. Sagduyu, Kemal Davaslioglu, and Jason H. Li. Generative Adversarial Networks for Black-Box API Attacks with Limited Training Data. In *IEEE Int. Symposium on Signal Processing and Information Technology (ISSPIT)*, Louisville, KY, USA, 2018. IEEE.

[43] Nikhil Joshi and Rewanth Tammana. GDALR: An Efficient Model Duplication Attack on Black Box Machine Learning Models. In *IEEE Int. Conf. on System, Computation, Automation and Networking (ICSCAN)*, Pondicherry, India, 2019. IEEE.

[44] Abdullah Ali and Birhanu Eshete. Best-Effort Adversarial Approximation of Black-Box Malware Classifiers. In *Int. Conf. on Security and Privacy in Communication Systems*, Cham, 2020. Springer International Publishing.

[45] Robert Nikolai Reith, Thomas Schneider, and Oleksandr Tkachenko. Efficiently Stealing Your Machine Learning Models. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, London, United Kingdom, 2019. ACM Press.

[46] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep Neural Network Fingerprinting by Conferrable Adversarial Examples. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

[47] Xiaoyong Yuan, Leah Ding, Lan Zhang, Xiaolin Li, and Dapeng Oliver Wu. ES Attack: Model Stealing Against Deep Neural Networks Without Data Hurdles. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[48] Kangjie Chen, Shangwei Guo, Tianwei Zhang, Xiaofei Xie, and Yang Liu. Stealing Deep Reinforcement Learning Models for Fun and Profit. In *Asia Conf. on Computer and Communications Security (ASIA CCS)*, Virtual Event Hong Kong, 2021. ACM.

[49] Takayuki Miura, Satoshi Hasegawa, and Toshiki Shibahara. MEGEX: Data-Free Model Extraction Attack against Gradient-Based Explainable AI, 2021. arXiv:2107.08909.

[50] Antonio Bărbălău, Adrian Cosma, Radu Tudor Ionescu, and Marius Popescu. Black-Box ripper: copying black-box models using generative evolutionary algorithms. In *Int. Conf. on Neural Information Processing Systems (NeurIPS)*, Red Hook, NY, USA, 2020. Curran Associates Inc.

[51] Jean-Baptiste Truong, Pratyush Maini, Robert J. Walls, and Nicolas Papernot. Data-Free Model Extraction. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, 2021. IEEE.

[52] Seong Joon Oh, M. Augustin, M. Fritz, and B. Schiele. Towards Reverse-Engineering Black-Box Neural Networks. In *Int. Conf. on Learning Representations (ICLR)*, Vancouver, B.C., Canada, 2018.

[53] Weizhe Hua, Zhiru Zhang, and G. Edward Suh. Reverse engineering convolutional neural networks through side-channel information leaks. In *Annual Design Automation Conf. (DAC)*, San Francisco, CA, USA, 2018. ACM.

[54] Xing Hu, Ling Liang, Shuangchen Li, Lei Deng, Pengfei Zuo, et al. DeepSniffer: A DNN Model Extraction Framework Based on Learning Architectural Hints. In *Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, Lausanne, Switzerland, 2020. ACM.

[55] Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. In *USENIX Security Symposium*. USENIX Association, 2020.

[56] Sanghyun Hong, Michael Davinroy, Yiğitcan Kaya, Stuart Nevans Locke, Ian Rackow, Kevin Kulda, Dana Dachman-Soled, and Tudor Dumitraş. Security Analysis of Deep Neural Networks Operating in the Presence of Cache Side-Channel Attacks, 2020. arXiv:1810.03487.

[57] Yun Xiang, Zhuangzhi Chen, Zuohui Chen, Zebin Fang, Haiyang Hao, Jinyin Chen, Yi Liu, Zhefu Wu, Qi Xuan, and Xiaoniu Yang. Open DNN Box by Power Side-Channel Attack. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(11), 2020.

[58] Yuankun Zhu, Yueqiang Cheng, Husheng Zhou, and Yantao Lu. Hermes Attack: Steal DNN Models with Lossless Inference Accuracy. In *USENIX Security Symposium*. USENIX Association, 2021.

[59] Daniel Teitelman, Itay Naeh, and Shie Mannor. Stealing Black-Box Functionality Using The Deep Neural Tree Architecture, 2020. arXiv:2002.09864.

[60] Yi Shi and Yalin E. Sagduyu. Evasion and causative attacks with adversarial deep learning. In *IEEE Military Communications Conf. (MILCOM)*, Baltimore, MD, 2017. IEEE.

Manuscript accepted in ACM Computing Surveys (CSUR): https://doi.org/10.1145/3595292

[61] Tegjyot Singh Sethi, Mehmed Kantardzic, and Joung Woo Ryu. 'Security Theater': On the Vulnerability of Classifiers to Exploratory Attacks. In *Intelligence and Security Informatics*, Cham, 2017. Springer International Publishing.

[62] Tegjyot Singh Sethi and Mehmed Kantardzic. Data driven exploratory attacks on black box classifiers in adversarial domains. *Neurocomputing*, 289, 2018.

[63] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples, 2016. arXiv:1605.07277.

[64] Smitha Milli, Ludwig Schmidt, Anca D. Dragan, and Moritz Hardt. Model Reconstruction from Model Explanations. In *Conf. on Fairness, Accountability, and Transparency (FAT)*, Atlanta GA USA, 2019. ACM.

[65] David Rolnick and Konrad Kording. Reverse-engineering deep ReLU networks. In *Int. Conf. on Machine Learning (ICML)*. PMLR, 2020.

[66] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic Extraction of Neural Network Models. In *Advances in Cryptology*, Cham, 2020. Springer International Publishing.

[67] Nicholas Roberts, Vinay Uday Prabhu, and Matthew McAteer. Model Weight Theft With Just Noise Inputs: The Curious Case of the Petulant Attacker. In *ICML Workshop on the Security and Privacy of Machine Learning*, Long Beach, CA, 2019.

[68] Ulrich Aïvodji, Alexandre Bolot, and Sébastien Gambs. Model extraction from counterfactual explanations, 2020. arXiv:2009.01884.

[69] Tatsuya Takemura, Naoto Yanai, and Toru Fujiwara. Model Extraction Attacks against Recurrent Neural Networks, 2020. arXiv:2002.00123.

[70] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realisation. In *ACM Asia Conf. on Computer and Communications Security (ASIA CCS)*, Nagasaki Japan, 2022. ACM.

[71] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing Links from Graph Neural Networks. In *USENIX Security Symposium*, Vancouver, B.C., Canada, 2021. USENIX Association.

[72] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. CloudLeak: Large-Scale Deep Learning Models Stealing Through Adversarial Examples. In *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, 2020. Internet Society.

[73] Sebastian Szyller, Vasisht Duddu, Tommi Gröndahl, and N. Asokan. Good Artists Copy, Great Artists Steal: Model Extraction Attacks Against Image Translation Generative Adversarial Networks, 2021. arXiv:2104.12623.

[74] Xuanli He, Lingjuan Lyu, Lichao Sun, and Qiongkai Xu. Model Extraction and Adversarial Transferability, Your BERT is Vulnerable! In *Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Virtual Event, 2021.

[75] Hailong Hu and Jun Pang. Stealing Machine Learning Models: Attacks and Countermeasures for Generative Adversarial Networks. In *Annual Computer Security Applications Conf. (ACSAC)*, Virtual Event USA, 2021. ACM.

[76] Xinyi Zhang, Chengfang Fang, and Jie Shi. Thief, Beware of What Get You There: Towards Understanding Model Extraction Attack, 2021. arXiv:2104.05921.

[77] Arne Aarts, Wil Michiels, and Peter Roelse. Leveraging Partial Model Extractions using Uncertainty Quantification. In *Int. Conf. on Cloud Networking*, Cookeville, TN, USA, 2021. IEEE.

[78] Yixu Wang and Xianming Lin. Enhance Model Stealing Attack via Label Refining. In *Int. Conf. on Intelligent Computing and Signal Processing (ICSP)*, Xi'an, China, 2022. IEEE.

[79] Yupei Liu, Jinyuan Jia, Hongbin Liu, and Neil Zhenqiang Gong. StolenEncoder: Stealing Pre-trained Encoders in Self-supervised Learning. In *ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, Los Angeles, CA, USA, 2022. ACM.

[80] Zeyang Sha, Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. Can't Steal? Cont-Steal! Contrastive Stealing Attacks Against Image Encoders, 2022. arXiv:2201.07513.

[81] Yixu Wang, Jie Li, Hong Liu, Yan Wang, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Black-Box Dissector: Towards Erasing-Based Hard-Label Model Stealing Attack. In *European Conf. on Computer Vision (ECCV)*, Cham, 2022. Springer Nature Switzerland.

[82] Anli Yan, Ruitao Hou, Xiaozhang Liu, Hongyang Yan, Teng Huang, and Xianmin Wang. Towards explainable model extraction attacks. *Int. Journal of Intelligent Systems*, 37(11), 2022.

[83] Sunandini Sanyal, Sravanti Addepalli, and R. Venkatesh Babu. Towards Data-Free Model Stealing in a Hard Label Setting. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA, 2022. IEEE.

[84] Yi Xie, Mengdie Huang, Xiaoyu Zhang, Changyu Dong, Willy Susilo, and Xiaofeng Chen. GAME: Generative-Based Adaptive Model Extraction Attack. In *European Symposium on Research in Computer Security (ESORICS)*, Cham, 2022. Springer International Publishing.

[85] Adam Dziedzic, Nikita Dhawan, Muhammad Ahmad Kaleem, Jonas Guan, and Nicolas Papernot. On the Difficulty of Defending Self-Supervised Learning against Model Extraction. In *Int. Conf. on Machine Learning (ICML)*. PMLR, 2022.

[86] Yun Shen, Xinlei He, Yufei Han, and Yang Zhang. Model Stealing Attacks Against Inductive Graph Neural Networks. In *IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2022. IEEE.

[87] David DeFazio and Arti Ramesh. Adversarial Model Extraction on Graph Neural Networks. In *Int. Workshop on Deep Learning on Graphs: Methodologies and Applications (DLGMA)*, New York, NY, USA, 2020.

[88] Xueluan Gong, Yanjiao Chen, Wenbin Yang, Guanghao Mei, and Qian Wang. InverseNet: Augmenting Model Extraction Attacks with Training Data Inversion. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, Montreal, Canada, 2021.

[89] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. Black-Box Attacks on Sequential Recommenders via Data-Free Model Extraction. In *ACM Conf. on Recommender Systems (RecSys)*, Amsterdam Netherlands, 2021. ACM.

[90] Yongjie Wang, Hangwei Qian, and Chunyan Miao. DualCF: Efficient Model Extraction Attack from Counterfactual Explanations. In *ACM Conf. on Fairness, Accountability, and Transparency (FAccT)*, Seoul Republic of Korea, 2022. ACM.
[91] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. BDPL: A Boundary Differentially Private Layer Against Machine Learning Model Extraction Attacks. In *European Symposium on Research in Computer Security (ESORICS)*, Cham, 2019. Springer International Publishing.
[92] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 2010.
[93] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *IEEE Int. Conf. on Computer Vision (ICCV)*, Venice, Italy, 2017. IEEE.
[94] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Int. Conf. on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
[95] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. 2019.
[96] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. arXiv:1810.04805.
[97] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. *ACM Computing Surveys*, 50(2), 2017.
[98] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014. Association for Computational Linguistics.
[99] Vasisht Duddu, Debasis Samanta, D. Vijay Rao, and Valentina E. Balas. Stealing Neural Networks via Timing Side Channels, 2019. arXiv:1812.11720.
[100] Yuntao Liu and Ankur Srivastava. GANRED: GAN-based Reverse Engineering of DNNs via Cache Side-Channel. In *ACM SIGSAC Conf. on Cloud Computing Security Workshop (CCSW)*, Virtual Event USA, 2020. ACM.
[101] Sanghyun Hong, Michael Davinroy, Yiğitcan Kaya, Dana Dachman-Soled, and Tudor Dumitraş. How to 0wn the NAS in Your Spare Time. In *Int. Conf. on Learning Representations*, 2020.
[102] Adnan Siraj Rakin, Md Hafizul Islam Chowdhuryy, Fan Yao, and Deliang Fan. DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories. In *IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2022. IEEE.
[103] Zhendong Wang, Xiaoming Zeng, Xulong Tang, Danfeng Zhang, Xing Hu, and Yang Hu. Demystifying Arch-hints for Model Extraction: An Attack in Unified Memory System, 2022. arXiv:2208.13720.
[104] Kota Yoshida, Takaya Kubota, Mitsuru Shiozaki, and Takeshi Fujino. Model-Extraction Attack Against FPGA-DNN Accelerator Utilizing Correlation Electromagnetic Analysis. In *Annual Int. Symposium on Field-Programmable Custom Computing Machines (FCCM)*, San Diego, CA, USA, 2019. IEEE.
[105] Jakub Breier, Dirmanto Jap, Xiaolu Hou, Shivam Bhasin, and Yang Liu. SNIFF: Reverse Engineering of Neural Networks With Fault Attacks. *IEEE Transactions on Reliability*, 2021.
[106] Francesco Regazzoni, Shivam Bhasin, Amir Ali Pour, Ihab Alshaer, Furkan Aydin, Aydin Aysu, et al. Machine Learning and Hardware security: Challenges and Opportunities -Invited Talk-. In *2020 IEEE/ACM Int. Conf. On Computer Aided Design (ICCAD)*, Virtual Event USA, 2020. ACM.
[107] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *USENIX Security Symposium*, Santa Clara, CA, 2019. USENIX Association.
[108] Dirmanto Jap, Ville Yli-Mäyry, Akira Ito, Rei Ueno, Shivam Bhasin, and Naofumi Homma. Practical Side-Channel Based Model Extraction Attack on Tree-Based Machine Learning Algorithm. In *Applied Cryptography and Network Security Workshops*, Cham, 2020. Springer International Publishing.
[109] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. DeepEM: Deep Neural Networks Model Recovery through EM Side-Channel Information Leakage. In *IEEE Int. Symposium on Hardware Oriented Security and Trust (HOST)*, San Jose, CA, USA, 2020. IEEE.
[110] Tommy Li and Cory Merkel. Model Extraction and Adversarial Attacks on Neural Networks Using Switching Power Information. In *Int. Conf. on Artificial Neural Networks (ICANN)*, Cham, 2021. Springer International Publishing.
[111] Anuj Dubey, Emre Karabulut, Amro Awad, and Aydin Aysu. High-Fidelity Model Extraction Attacks via Remote Power Monitors. In *Int. Conf. on Artificial Intelligence Circuits and Systems (AICAS)*, Incheon, Republic of Korea, 2022. IEEE.
[112] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and other Systems. In *Advances in Cryptology*. Springer, 1996.
[113] Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient Cache Attacks on AES, and Countermeasures. *Journal of Cryptology*, 23(1), 2010.
[114] Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *USENIX Security Symposium*, San Diego, CA, USA, 2014. USENIX Association.
[115] Tyler Hunt, Zhipeng Jia, Vance Miller, Ariel Szekely, Yige Hu, Christopher J. Rossbach, and Emmett Witchel. Telekine: Secure Computing with Cloud GPUs. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, Santa Clara, CA, 2020. USENIX Association.
[116] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K. Nicholas. Malware Detection by Eating a Whole EXE. In *Workshops AAAI Conf. on Artificial Intelligence*, New Orleans, LA, USA, 2018. AAAI Press.
[117] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *Int. Conf. on Learning Representations (ICLR)*, 2019.
[118] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset Inference: Ownership Resolution in Machine Learning. In *Int. Conf. on Learning Representations (ICLR)*, 2021.
[119] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled Watermarks as a Defense against Model Extraction. In *USENIX Security Symposium*. USENIX Association, 2021.

Manuscript accepted in ACM Computing Surveys (CSUR): https://doi.org/10.1145/3595292

[120] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N. Asokan. DAWN: Dynamic Adversarial Watermarking of Neural Networks. In *ACM Int. Conf. on Multimedia*, Virtual Event China, 2021. ACM.

[121] Abhishek Chakraborty, Daniel Xing, Yuntao Liu, and Ankur Srivastava. DynaMarks: Defending Against Deep Learning Model Extraction Using Dynamic Watermarking, 2022. arXiv:2207.13321.

[122] Yiming Li, Linghui Zhu, Xiaojun Jia, Yong Jiang, Shu-Tao Xia, and Xiaochun Cao. Defending against Model Stealing via Verifying Embedded External Features. In *AAAI Conf. on Artificial Intelligence*, Virtual Event, 2022.

[123] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. Model Extraction Warning in MLaaS Paradigm. In *Annual Computer Security Applications Conf. (ACSAC)*, San Juan, PR, USA, 2018. ACM.

[124] Zhanyuan Zhang, Yizheng Chen, and David Wagner. SEAT: Similarity Encoder by Adversarial Training for Detecting Model Extraction Attack Queries. In *ACM Workshop on Artificial Intelligence and Security*, Virtual Event Republic of Korea, 2021. ACM.

[125] Soham Pal, Yash Gupta, Aditya Kanade, and Shirish Shevade. Stateful Detection of Model Extraction Attacks, 2021. arXiv:2107.05166.

[126] Xinjing Liu, Zhuo Ma, Yang Liu, Zhan Qin, Junwei Zhang, and Zhuzhu Wang. SeInspect: Defending Model Stealing via Heterogeneous Semantic Inspection. In *European Symposium on Research in Computer Security (ESORICS)*, Cham, 2022. Springer International Publishing.

[127] Amir Mahdi Sadeghzadeh, Amir Mohammad Sobhanian, Faezeh Dehghan, and Rasool Jalili. HODA: Hardness-Oriented Detection of Model Extraction Attacks, 2022. arXiv:2106.11424.

[128] Adam Dziedzic, Muhammad Ahmad Kaleem, Yu Shen Lu, and Nicolas Papernot. Increasing the Cost of Model Extraction with Calibrated Proof of Work. In *Int. Conf. on Learning Representations (ICLR)*, 2022.

[129] Justin Grana. Perturbing Inputs to Prevent Model Stealing. In *IEEE Conf. on Communications and Network Security*, Avignon, France, 2020. IEEE.

[130] L. Guiga and A. W. Roscoe. Neural network security: Hiding CNN parameters with guided grad-CAM. In *Int. Conf. on Information Systems Security and Privacy (ICISSP)*, 2020.

[131] Xinran Wang, Yu Xiang, Jun Gao, and Jie Ding. Information Laundering for Model Privacy. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

[132] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. Defending Against Neural Network Model Stealing Attacks Using Deceptive Perturbations. In *IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 2019. IEEE.

[133] Sanjay Kariyappa and Moinuddin K. Qureshi. Defending Against Model Stealing Attacks With Adaptive Misinformation. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020. IEEE.

[134] Jinyin Chen, Changan Wu, Shijing Shen, Xuhong Zhang, and Jianhao Chen. DAS-AST: Defending Against Model Stealing Attacks Based on Adaptive Softmax Transformation. In *Int. Conf. on Information Security and Cryptology*, Cham, 2020. Springer International Publishing.

[135] Sanjay Kariyappa, Atul Prakash, and Moinuddin K. Qureshi. Protecting DNNs from Theft using an Ensemble of Diverse Models. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

[136] Jeonghyun Lee, Sungmin Han, and Sangkyun Lee. Model Stealing Defense against Exploiting Information Leak through the Interpretation of Deep Neural Nets. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, Vienna, Austria, 2022.

[137] Mantas Mazeika, Bo Li, and David Forsyth. How to Steer Your Adversary: Targeted and Efficient Model Stealing Defenses with Gradient Redirection. In *Int. Conf. on Machine Learning (ICML)*. PMLR, 2022.

[138] Ibrahim M. Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Adding Robustness to Support Vector Machines Against Adversarial Reverse Engineering. In *ACM Int. Conf. on Conference on Information and Knowledge Management (CIKM)*, Shanghai China, 2014. ACM.

[139] Hsiao-Ying Lin, Chengfang Fang, and Jie Shi. Bident Structure for Neural Network Model Protection:. In *Int. Conf. on Information Systems Security and Privacy (ICISSP)*, Valletta, Malta, 2020. SciTePress.

[140] Hervé Chabanne, Vincent Despiegel, and Linda Guiga. A Protection against the Extraction of Neural Network Models, 2020. arXiv:2005.12782.

[141] Kálmán Szentannai, Jalal Al-Afandi, and András Horváth. Preventing Neural Network Weight Stealing via Network Obfuscation. In *Computing Conf.*, Cham, 2020. Springer International Publishing.

[142] T. Kohno, A. Broido, and K.C. Claffy. Remote Physical Device Fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2), 2005.

[143] Isabell Lederer, Rudolf Mayer, and Andreas Rauber. Identifying Appropriate Intellectual Property Protection Mechanisms for Machine Learning Models: A Systematisation of Watermarking, Fingerprinting, Model Access, and Attacks. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[144] Cynthia Dwork. Differential Privacy. In *Int. Colloquium on Automata, Languages and Programming (ICALP)*, Venice, Italy, 2006. Springer.

[145] Vasisht Duddu and D. Vijay Rao. Quantifying (Hyper) Parameter Leakage in Machine Learning. In *Int. Conf. on Multimedia Big Data (BigMM)*, New Delhi, India, 2020. IEEE.

[146] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. In *Neur. Information Processing Systems, Datasets & Benchmarks Track*, 2021.

[147] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On Adaptive Attacks to Adversarial Example Defenses. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020.

[148] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In *Research in Attacks, Intrusions, and Defenses*, Cham, 2018. Springer International Publishing.