ES Attack: Model Stealing against Deep Neural Networks without Data Hurdles

Xiaoyong Yuan[1], **Lei Ding**[2], **Lan Zhang**[1], **Xiaolin Li**[3], **Dapeng Wu**[4]
[1]Michigan Technological University [2]American University [3]Cognization Lab [4]University of Florida

Abstract

Deep neural networks (DNNs) have become the essential components for various commercialized machine learning services, such as Machine Learning as a Service (MLaaS). Recent studies show that machine learning services face severe privacy threats - well-trained DNNs owned by MLaaS providers can be stolen through public APIs, namely model stealing attacks. However, most existing works undervalued the impact of such attacks, where a successful attack has to acquire confidential training data or auxiliary data regarding the victim DNN. In this paper, we propose *ES Attack*, a novel model stealing attack without any data hurdles. By using heuristically generated synthetic data, *ES Attack* iteratively trains a substitute model and eventually achieves a functionally equivalent copy of the victim DNN. The experimental results reveal the severity of *ES Attack*: i) *ES Attack* successfully steals the victim model without data hurdles, and *ES Attack* even outperforms most existing model stealing attacks using auxiliary data in terms of model accuracy; ii) most countermeasures are ineffective in defending *ES Attack*; iii) *ES Attack* facilitates further attacks relying on the stolen model.

1 Introduction

As one of the typical business models, Machine-Learning-as-a-Service (MLaaS) provides a platform to facilitate users to use machine learning models (Hunt et al. 2018). Users can access well-trained machine learning models via public APIs provided by MLaaS providers, without building a model from scratch. MLaaS allows users to query machine learning models in the form of pay-per-query and get responses of the model's predictions. Recent studies show that machine learning services face severe privacy threats: model stealing attacks steal functionally equivalent copies from MLaaS providers through multiple queries (Tramèr et al. 2016; Orekondy, Schiele, and Fritz 2019; Correia-Silva et al. 2018; Pal et al. 2019; Yu et al. 2020). Model stealing attacks exploit the tensions between queries and their corresponding feedback, *i.e.*, the output predictions. Tramer *et al.* extract machine learning model's parameters by solving equations derived from the model architecture (Tramèr et al. 2016). However, it requires the exact knowledge of ML architectures and becomes difficult to scale up to steal deep neural networks (DNNs) (Papernot et al. 2017). Existing model stealing attacks against DNNs require specific knowledge of the model's training data: the exact training data (Wang and Gong 2018), seed samples from the training data(Juuti et al. 2019), and auxiliary data that shares similar attributes as the training data or within the same task domain (Orekondy, Schiele, and Fritz 2019; Correia-Silva et al. 2018; Pal et al. 2019).

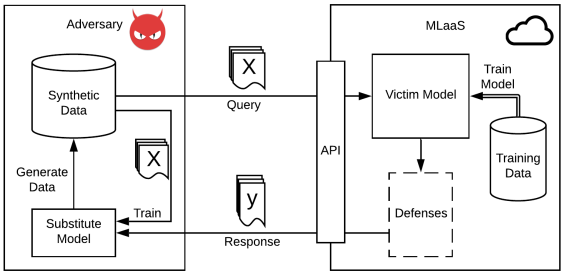


Figure 1: **Diagram of *ES Attack***.

Most existing model stealing attacks require the knowledge of training data or auxiliary data regarding the victim deep neural networks, which undervalued the impact of model stealing attacks. In practice, these data are not always accessible. Due to recent regulations on data protection (*e.g.*, GDPR and CCPA), many types of personal data are hard to acquire, such as health data and biometric data. In many domains, companies collect data for their own business and are reluctant to share their data. Government and other organizations usually lack resources and financial supports to create open datasets. Often, the quality of public data is out-of-date and questionable without updates and maintenance. The availability of appropriate data protects the victim models from being stolen by the existing model stealing attacks.

In this paper, we introduce *ES Attack*, a new class of model stealing attacks against DNNs without data hurdles. *ES Attack* heuristically generates synthetic data to overcome the limitations of existing approaches. Figure 1 illustrates the diagram of *ES Attack*. The adversary queries the victim model with the synthetic data $\boldsymbol{x}$ and labels the data using responses $\boldsymbol{y}$ via the MLaaS provider's APIs. The MLaaS provider may deploy defenses to prevent model leakage. A substitute model is iteratively trained using the synthetic data of $\boldsymbol{x}$ with corresponding labels $\boldsymbol{y}$ and eventually approximates the functionality of the victim model. There are two key steps in *ES Attack*: E-Step to Estimate the parameters in a substitute model and S-Step to Synthesize the

dataset for attacking.

Compared to existing model stealing attacks, *ES Attack* does not require i) information about the internals of the victim's DNN (*e.g.*, its architecture, hyper-parameters, and parameters), and ii) priori knowledge of the victim model's training data. The adversary only observes responses given by the victim DNN.

In summary, our contributions are fourfold:

1) We propose a novel model stealing framework *ES Attack* that does not require any knowledge of the victim model's training data. Compared to model stealing using auxiliary datasets, our proposed *ES Attack* improves the model accuracy by 44.57%.
2) *ES Attack* generates better synthetic datasets compared with the auxiliary datasets in terms of quality and diversity.
3) We demonstrate that the stolen model successfully facilitates black-box adversarial attacks against the victim model.
4) Three investigated countermeasures are not effective in preventing *ES Attack*.

To our best knowledge, this is the first work to steal deep neural networks without priori knowledge of training data.

2 Problem Statement

In this paper, we consider an image classification task. We define a task domain $\mathcal{T}$ with a training dataset $\mathcal{D}_{\text{train}}$ and a test dataset $\mathcal{D}_{\text{test}}$. $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ consist of a set of images and their corresponding labels $\{(\boldsymbol{x}, \boldsymbol{l})\}$. The victim model f_v is well-trained by an MLaaS provider on private training dataset $\mathcal{D}_{\text{train}}$. Users can access the trained DNN model f_v by querying data sample $\boldsymbol{x}$. MLaaS then responds with the predicted probabilities of K classes $\boldsymbol{y} = f_v(\boldsymbol{x})$ provided by the DNN model.

The goal of model stealing adversaries is to build a model f_s that is functionally equivalent to the victim model f_v by pretending themselves as normal users. We evaluate the performance of model stealing on the test dataset $\mathcal{D}_{\text{test}}$. Note that $\mathcal{D}_{\text{test}}$ is unknown to the adversary, which is used for evaluation only. By mimicking the behavior of the victim model, the stolen model can achieve good performance on the unknown test dataset $\mathcal{D}_{\text{test}}$. We assume that the adversary does not know 1) the victim model f_v, such as model architecture, model parameters, and hyper-parameters; 2) the victim's training data $\mathcal{D}_{\text{train}}$ or auxiliary data close to $\mathcal{D}_{\text{train}}$.

3 *ES Attack*

In this section, we present the design of *ES Attack*, and propose two heuristic approaches for data synthesis.

3.1 Design of ES Attack

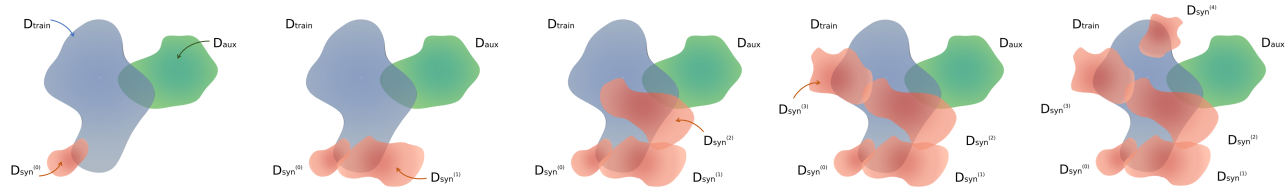
Model stealing attacks aim to build a model f_s that is functionally equivalent to the victim model f_v. Theoretically, if the adversary can train the substitute model on all the samples in the input space of f_v, the substitute model can achieve the same performance as the victim model. However, it is infeasible to query all the samples in the input space. Therefore, a critical step for model stealing attacks is to explore the input sub-space that represents the task domain $\mathcal{T}$. Adversaries will mimic the behavior of victim models in the input sub-space. (Orekondy, Schiele, and Fritz 2019; Correia-Silva et al. 2018; Pal et al. 2019) leverage public datasets as an auxiliary dataset to train the substitute model to approximate the output of victim model. The auxiliary data share common attributes with $\mathcal{D}_{\text{train}}$, which can be used to train the substitute model. However, these approaches are not practical due to many reasons: i) Data with shared attributes is not always available. Confidential data such as medical images, financial records are not publicly available. The scarcity of data is still a critical problem in many domains. ii) The relationship between the available auxiliary data on the public domain and the task domain $\mathcal{T}$ is unclear, which brings a challenge to select a proper auxiliary dataset. The rationale for selecting a specific auxiliary dataset is missing in most of the existing approaches. In the experiments, we show that using a randomly generated dataset, a special case of an auxiliary dataset, fails to steal the victim model. iii) The quality of data used for training the substitute model cannot be improved during model stealing. The data samples are fixed in the auxiliary dataset.

Therefore, we propose an *ES Attack* to heuristically explore the potential input space related to task domain $\mathcal{T}$ by learning from the victim model. We outline *ES Attack* in Algorithm 1. First, *ES Attack* initializes a randomly synthetic dataset $\mathcal{D}_{\text{syn}}^{(0)}$, which may share few attributes with $\mathcal{D}_{\text{train}}$, most likely fewer than $\mathcal{D}_{\text{aux}}$. Second, *ES Attack* trains a substitute model f_s based on the samples from the synthetic dataset and their predictions from the victim model. Then, *ES Attack* can generate a better synthetic dataset using the improved substitute model; in the meanwhile, the better synthetic dataset can help improve the substitute model. In this way, *ES Attack* iteratively synthesizes the datasets and trains the substitute model to improve the quality of the synthetic dataset and the performance of the substitute model. Eventually, the synthetic datasets will approximate the private training dataset, and the substitute model will approximate the victim model or steal the victim model.

Figure 2 illustrates the progress of data synthesis during *ES Attack*. In Figure 2, we compare the synthetic datasets $\mathcal{D}_{\text{syn}}^{(t)}$ (in red) with the victim's training dataset $\mathcal{D}_{\text{train}}$ (in blue) and the auxiliary dataset $\mathcal{D}_{\text{aux}}$ (in green). $\mathcal{D}_{\text{aux}}$ may share similar input space with $\mathcal{D}_{\text{train}}$, but in most cases, adversaries may not know the distance between the distribution of $\mathcal{D}_{\text{aux}}$ and the distribution of $\mathcal{D}_{\text{train}}$. Hence, $\mathcal{D}_{\text{train}}$ may not be fully covered by $\mathcal{D}_{\text{aux}}$. However, after initializing the synthetic dataset $\mathcal{D}_{\text{syn}}^{(0)}$, *ES Attack* will iteratively improve the synthetic datasets $\mathcal{D}_{\text{syn}}^{(t)}(t = 1, 2, 3, 4, \ldots)$, and explore more space in the training dataset $\mathcal{D}_{\text{train}}$.

Two key steps in *ES Attack*, E-Step and S-Step, are described as follows.

E-Step: Estimate parameters in the substitute model on the synthetic dataset using knowledge distillation. The knowledge distillation approach transfers the knowledge from f_v



(a) Initial Steal ($t = 0$) (b) The 1st Steal ($t = 1$) (c) The 2st Steal ($t = 2$) (d) The 3rd Steal ($t = 3$) (e) The 4th Steal ($t = 4$)

Figure 2: **Progress of Data Synthesis During *ES Attack***.

Algorithm 1 *ES Attack*

INPUT: the victim model f_v, the number of stealing epochs N.

1: Initialize a synthetic dataset $\mathcal{D}_{\text{syn}}^{(0)}$ by randomly sampling $\boldsymbol{x}$ from a Gaussian distribution.
2: Construct an initial substitute model $f_s^{(0)}$ by initializing the parameters in the model using Kaiming Initialization (He et al. 2015).
3: **for** $t \leftarrow 1$ to N **do**
4: **E-Step:** Estimate the parameters in the substitute model $f_s^{(t)}$ using knowledge distillation on the synthetic dataset $\mathcal{D}_{\text{syn}}^{(t-1)}$.
5: **S-Step:** Synthesize a new dataset $\mathcal{D}_{\text{syn}}^{(t)}$ based on the knowledge of the substitute model $f_s^{(t)}$.
6: **end for**

OUTPUT: the substitute model $f_s^{(N)}$.

to f_s without much performance degradation:

$$f_s^{(t)} \leftarrow \arg\min_{f_s} \mathrm{L}_{\text{KD}}(f_s, f_v; \mathcal{D}_{\text{syn}}^{(t-1)}), \tag{1}$$

where $f_s^{(t)}$ denotes the substitute model at iteration t and $\mathcal{D}_{\text{syn}}^{(t-1)}$ denotes the synthetic dataset at the previous iteration $t-1$. The objective function L_{KD} is defined as knowledge distillation loss to make $f_s^{(t)}$ approximate the victim model f_v:

$$\mathrm{L}_{\text{KD}}(f_s, f_v; \mathcal{D}_{\text{syn}}) = \frac{1}{|\mathcal{D}_{\text{syn}}|} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{syn}}} \mathrm{L}_{\text{CE}}(f_s(\boldsymbol{x}), f_v(\boldsymbol{x})), \tag{2}$$

where L_{CE} denotes the cross-entropy loss. We train the substitute model by minimizing the objective function (Equation 1) for M epochs using Adam (Kingma and Ba 2014).

S-Step: Synthesize the dataset $\mathcal{D}_{\text{syn}}^{(t)} = \{\boldsymbol{x}\}$ consisted of the synthetic input samples. We will describe two data synthesis approaches in Section 3.2.

3.2 Two approaches for Data Synthesis (S-Step)

We introduce two approaches to generate the synthetic data: namely, *DNN-SYN* and *OPT-SYN*. Both approaches start with generating a set of random labels and then generate data samples based on the labels and the substitute model. The data synthesis approaches aim to synthesize samples that can be classified as pre-chosen labels by the substitute model with high confidence.

DNN-SYN. We design a DNN-based generator to synthesize images that can be classified by our substitute model with high confidence. The design of image generator G follows the major architecture of Auxiliary Classifier GAN (ACGAN) (Odena, Olah, and Shlens 2017), a variant of Generative Adversarial Network (GAN), which can generate images with label conditioning. We refer to the data generation approach using DNN as *DNN-SYN*.

We describe the procedure of *DNN-SYN* as follows:

Step 1: Randomly assign a set of labels $\boldsymbol{l} = \{\boldsymbol{l_1}, \boldsymbol{l_2}, \ldots, \boldsymbol{l_n}\}$, where $\boldsymbol{l_i}$ denotes a K-dimensional one-hot vector.

Step 2: Train a DNN generator G with parameter w_G to generate data from a random latent vector $\boldsymbol{z_i}$. G is optimized that the generated data can be classified by f_s as assigned labels L with high confidence:

$$\min_{w_G} \quad \mathrm{L}_{\text{img}}(G, \boldsymbol{l}) \stackrel{\text{def}}{=} \sum_{i}^{n} \mathrm{L}_{\text{CE}}(f_s(G(\boldsymbol{z_i}, \boldsymbol{l_i})), \boldsymbol{l_i}). \tag{3}$$

Step 3: Generate a synthetic dataset using the generator trained in Step 2: $\mathcal{D}_{\text{syn}} = \{G(\boldsymbol{z_i}, \boldsymbol{l_i})\}$.

In addition, mode collapse is one of the critical problems for training GANs. To avoid mode collapse in *DNN-SYN*, we use a mode seeking approach to increase the diversity of data samples (Mao et al. 2019). Mode seeking has been shown simple yet effective in mitigating mode collapse. We generate two images $G(\boldsymbol{z_i^1})$ and $G(\boldsymbol{z_i^2})$ using latent vectors $\boldsymbol{z_i^1}$ and $\boldsymbol{z_i^2}$ and maximize the ratio of the distance of images to the distance of latent vectors. In other words, we minimize the mode seeking loss L_{ms}:

$$\mathrm{L}_{\text{ms}}(G, \boldsymbol{l}) \stackrel{\text{def}}{=} \frac{d(\boldsymbol{z_i^1}, \boldsymbol{z_i^2})}{d(G(\boldsymbol{z_i^1}, L), G(\boldsymbol{z_i^2}, L))}, \tag{4}$$

where $d(\cdot, \cdot)$ denotes a distance metric. In this paper, we use ℓ_2 norm distance. We sum up the original objective function L_{img} and the regularization term L_{ms} and minimize the new objective function $\mathrm{L}_{\text{DNN}} = \mathrm{L}_{\text{img}} + \lambda \mathrm{L}_{\text{ms}}$. λ denotes the hyper-parameter to adjust the value of regularization. In the experiment, we set λ as 1.

OPT-SYN. Instead of training a generator to synthesize the dataset, we propose an optimization-based data synthesis approach, *OPT-SYN*, which operates on the input space directly and does not suffer the problem of mode collapse. In addition, *OPT-SYN* explores a more diverse label space compared to the one-hot labels used in *DNN-SYN*. *OPT-SYN* first explores the possible prediction vectors $\{\boldsymbol{y}\}$ in the task domain $\mathcal{T}$ and then minimizes the cross-entropy loss between $\{\boldsymbol{y}\}$ and the substitute model's prediction on the synthetic data:

$$\min_{\boldsymbol{x}} \mathrm{L}_{\text{CE}}(f_s^{(t)}(\boldsymbol{x}), \boldsymbol{y}), \tag{5}$$

where $f_s^{(t)}$ denotes the substitute model at the tth stealing epoch. In the experiments, we find that *OPT-SYN* performs better than *DNN-SYN* does in general.

The proposed *OPT-SYN* approach is detailed as follows.

Step 1: To explore the possible prediction vectors, we sample each random vector $\boldsymbol{y} = \{y_1, y_2, \ldots, y_K\}$ from a K-dimensional Dirichlet distribution with parameter $\boldsymbol{\alpha}$. Dirichlet distribution is commonly used as conjugate prior distribution of categorical distribution. From the Dirichlet distribution, we can sample prior probabilities $\{y_1, y_2, \ldots, y_K\}$, where $y_i \in (0, 1)$ and $\sum_{i=1}^{K} y_i = 1$. $\boldsymbol{\alpha}$ is referred to as the concentration parameter, which controls the distribution. The probability density function of Dirichlet distribution $Dir(K, \boldsymbol{\alpha})$ can be calculated by:

$$f(y_1, y_2, \ldots, y_K, \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} y_i^{\alpha_i - 1}, \qquad (6)$$

where $B(\boldsymbol{\alpha})$ denotes the gamma function and $\sum y_1, y_2, \ldots, y_K = 1$. In the experiment, we randomly sample the parameter $\boldsymbol{\alpha}$ from a Gaussian distribution: $\boldsymbol{\alpha} \sim \mathcal{N}(0, 1)$ to explore the possible Dirichlet distribution.

Step 2: Given the prediction vector $\boldsymbol{y}$, we synthesize data $\boldsymbol{x}$ by iteratively minimizing the objective function 5. The goal is to generate a data sample $\boldsymbol{x}^*$ that $f_s^{(t)}$ predicts $\boldsymbol{x}^*$ close to $\boldsymbol{y}$. An adaptive gradient-based optimization algorithm, Adam (Kingma and Ba 2014), is applied to optimize the objective function iteratively.

4 Evaluation of *ES Attack*

In this section, we evaluate our proposed *ES Attack* on three different neural networks and four image classification datasets. We compare our results with two baseline attacks. Moreover, we investigate the data synthesized during the attacks in terms of data quality and diversity.

4.1 Experiment Setup

Victim Models and Datasets. We train three types of DNN models on four datasets and use them as the victim models in our experiments. We train LeNet5 (LeCun et al. 1998) on the MNIST (LeCun et al. 1998) and KMNIST (Clanuwat et al. 2018) dataset. We train ResNet18 (He et al. 2016) and ResNet34 (He et al. 2016) on the SVHN (Netzer et al. 2011) and CIFAR10 (Krizhevsky et al. 2009) dataset. LeNet5 is trained for 30 epochs using an SGD optimizer with a learning rate of 0.1 on the MNIST and KMNIST dataset. We train ResNet18 and ResNet34 for 200 epochs with an initial learning rate of 0.1 on the SVHN and CIFAR10 dataset. We reduce the learning rate by 10 after 80 and 120 epochs. We select the models with the highest test accuracies as the victim models.

Settings of *ES Attack*. For *DNN-SYN*, we input a 100-dimensional random latent vector and a one-hot label vector into DNN-based generator G. The substitute model f_s and DNN-based generator G is trained by an Adam optimizer with a learning rate of 0.001. f_s and G are trained alternatively for 2,000 epochs each on the MNIST, KMNIST, and SVHN dataset ($N = 2000, M = 1$), and 15,000 epochs each on the CIFAR10 dataset ($N = 15000$, $M = 1$).

For *OPT-SYN*, we synthesize data for 30 iterations ($M = 30$) in each stealing epoch using an Adam optimizer with a learning rate of 0.01. We train the adversary model for 10 epochs on the synthetic dataset ($M = 10$). We repeat the stealing for 200 epochs on the MNIST, KMNIST, and SVHN dataset ($N = 200$), and 1,500 epochs on the CIFAR10 dataset ($N = 1500$). To speed up the stealing process, we augment the synthetic dataset by random horizontal flip, horizontal shift and adding Gaussian noise.

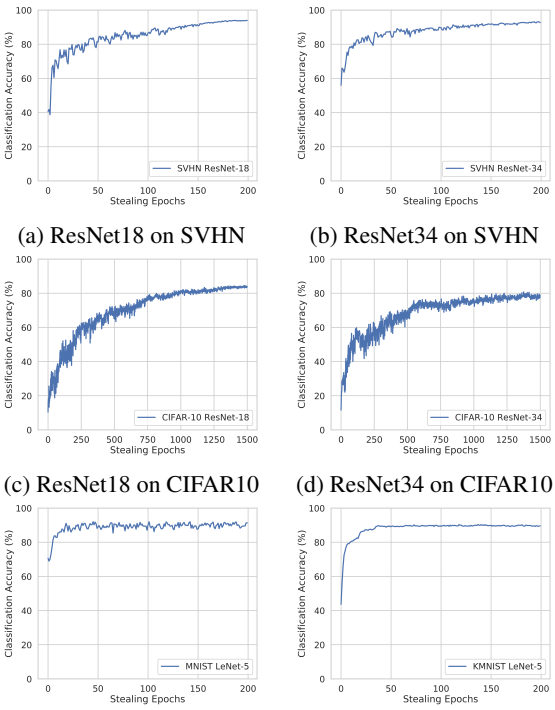
Table 1: Performance comparison of model stealing attacks.

Dataset	Model	Victim accuracy (%)	Attacks	Substitute accuracy (%)
SVHN	ResNet18	95.40	Random	50.71
			Auxiliary	74.84
			DNN-SYN	93.95
			OPT-SYN	**93.97**
SVHN	ResNet34	95.94	Random	60.95
			Auxiliary	82.00
			DNN-SYN	**93.34**
			OPT-SYN	93.19
CIFAR10	ResNet18	91.12	Random	11.72
			Auxiliary	48.73
			DNN-SYN	33.44
			OPT-SYN	**84.60**
CIFAR10	ResNet34	91.93	Random	14.45
			Auxiliary	38.55
			DNN-SYN	12.69
			OPT-SYN	**80.79**
MNIST	LeNet5	99.10	Random	72.18
			Auxiliary	**98.96**
			DNN-SYN	91.02
			OPT-SYN	92.03
KMNIST	LeNet5	95.67	Random	56.39
			Auxiliary	59.43
			DNN-SYN	90.37
			OPT-SYN	**90.37**

Baseline Model Stealing Attacks. We compare *ES Attack* with two baseline attacks - model stealing using randomly generated data and auxiliary data. First, if the adversary has no knowledge of the victim's training data, randomly generated data could be the only dataset the adversary can leverage. We form a random dataset with the random data sampled from a Gaussian Distribution $\mathcal{N}(0, 1)$ and their prediction from the victim model. We train our substitute model using the random dataset iteratively. Second, we consider public data as an auxiliary dataset. We use data samples from other public datasets and query the victim model with them. We construct an auxiliary dataset and train the substitute model on it. To make a fair comparison, we make sure that all the model stealing attacks, including two baseline attacks and two *ES Attacks* (*DNN-SYN* and *OPT-SYN*), train their substitute models for the same epochs.

4.2 Performance Evaluation

We evaluate the performance of *ES Attacks* using two data synthesis approaches and compare them with two baseline attacks. We report the accuracy of model stealing attacks in



(a) ResNet18 on SVHN (b) ResNet34 on SVHN

(c) ResNet18 on CIFAR10 (d) ResNet34 on CIFAR10

(e) LeNet5 on MNIST (f) LeNet5 on KMNIST

Figure 3: Substitute model accuracy during attacks.

Table 1. We compare the results with two baseline attacks that use randomly generated data (Random) and auxiliary data (Auxiliary) to steal the victim model.

From the evaluation, we observe that *OPT-SYN* can successfully steal the victim models over all the datasets and model architectures. Our proposed attacks achieve better performance compared with two baseline attacks. **On average, *OPT-SYN* improves the best accuracy by 44.57% compared to the best results of two baseline attacks.** *OPT-SYN* performs as well as *DNN-SYN* on the SVHN, MNIST, and KMNIST dataset. However, *DNN-SYN* cannot achieve a good performance on the CIFAR10 dataset, which is a more complex dataset and the generator G in *DNN-SYN* may still cause the mode collapse problem. Both our proposed attacks perform worse than the attacks using auxiliary data (KMNIST) on the MNIST dataset, which suggests that the auxiliary data can be used in the model stealing if the auxiliary data well-represent the target task and the data are available to the adversary.

Note that we assume that the adversary has no knowledge of any victim's data, which means the adversary cannot evaluate the substitute model on a validation dataset and select the best substitute model during the attack. If the performance of the stealing attacks fluctuates, then the adversary cannot guarantee the best performance of the substitute model. The convergence of the substitute model is essential for the stealing attacks without a validation dataset. Our experiments show that the performance of the substitute model converges after a few stealing epochs (Figure 3). If the adversary has the knowledge of a validation dataset or the victim's test dataset $\mathcal{D}_{\text{test}}$, the adversary will achieve the best accuracy. Otherwise, the adversary will use the substitute model in the last stealing epoch ($t = N$). We observe the subtle difference between the best accuracy and the last accuracy achieved by the substitute model (0.79% difference on average for *OPT-SYN* and 1.53% for *DNN-SYN*). The stable convergence suggests that our proposed attacks do not rely on a validation dataset.

Table 2: *ES Attack* using Different DNN Architectures.

Dataset	Victim model	Substitute model	Victim accuracy (%)	Substitute accuracy (%)
SVHN	ResNet18	ResNet34	95.40	94.64
	ResNet34	ResNet18	95.94	94.03
CIFAR10	ResNet18	ResNet34	91.12	82.54
	ResNet34	ResNet18	91.93	62.73

We find that model stealing attacks are economically practical in real-world settings. To steal the victim model for the MNIST, KMNIST, and SVHN dataset, it costs $48K according to the pricing of Amazon AWS (Amazon 2020). It costs $360K to steal the victim model for the CIFAR10 dataset. The expenses are much less than hiring ML experts and collecting data from scratch and can be further reduced via crowdsourcing.

4.3 Sensitivity Analysis of DNN Architectures

We assume the adversary has no knowledge of the victim model's architecture. The adversary may choose a different architecture of the substitute model from that of the victim model. Thus, we investigate the stealing performance with different DNN architectures in our experiments. For the SVHN and CIFAR10 dataset, we use ResNet18 for the victim model and ResNet34 for the substitute model and vice versa.

For the experimental results, we do not observe the performance loss of model stealing with different DNN architectures for the SVHN dataset, compared with using the same architecture. For the CIFAR10 dataset, we observe subtle performance loss using ResNet34 to steal ResNet18 models. The only degradation of performance occurs when the adversary uses a small model ResNet18 to steal a large model ResNet34. We believe the degradation is due to the gap between the size of the victim model and that of the substitute model. We find similar performance degradation in many other tasks using knowledge distillation. The performance of the student model (substitute model in our paper) will be degraded if there is a gap between student and teacher (victim) (Mirzadeh et al. 2019). The adversary can easily avoid performance degradation by selecting a large model. From our experiments, if the adversary chooses a DNN model with the same size or the large size compared with the victim model, the adversary will be able to steal a substitute model with high accuracy.

4.4 Quality Analysis of Synthetic Data

In this section, we investigate the quality of synthetic data. Borrowing measurements for GANs, we analyze the quality and diversity of synthetic data using Inception Score (IS) (Salimans et al. 2016) and Fréchet Inception Distance

Table 3: Synthetic Data Analysis using IS and FID.

Dataset	Model	Victim $\mathcal{D}_{\text{train}}$	Auxiliary $\mathcal{D}_{\text{aux}}$		Random $\mathcal{D}_{syn}^{(0)}$		Synthetic $\mathcal{D}_{syn}^{(N)}$	
		IS	IS	FID	IS	FID	IS	FID
MNIST	LeNet5	9.86	4.22	274.80	1.22	500.16	4.63	257.55
KMNIST	LeNet5	9.96	4.75	1073.92	1.78	1498.75	4.70	962.47
CIFAR10	ResNet18	6.89	2.58	5.34	3.45	5.82	4.32	3.31
CIFAR10	ResNet34	7.64	4.16	18.16	6.75	14.88	6.65	18.29
SVHN	ResNet18	6.89	2.24	7.62	3.45	5.82	4.32	3.31
SVHN	ResNet34	7.64	4.16	18.16	6.75	14.88	6.65	18.29

(FID) (Heusel et al. 2017). To calculate FID, we used the features from the layer before the last linear layer and compared our synthesis data with the training data. We find that the synthetic data achieves better quality and higher diversity compared to the auxiliary data.

We compare the four types of data and report the average values of IS and FID in Table 3: 1) victim's training dataset, 2) auxiliary dataset used in the baseline attack, 3) random data generated in the first initialization epoch ($t = 0$), 4) the synthetic data generated in the last stealing epoch ($t = N$). The value of FID is evaluated by comparing the data with the victim's training data. From our analysis, we find that synthetic data usually achieves better quality and high diversity than the auxiliary data (higher IS value and lower FID value). On average over six settings, synthetic data $\mathcal{D}_{syn}^{(1)}$ achieves 60.58% higher IS values and 27.64% lower FID values than the auxiliary data $\mathcal{D}_{\text{aux}}$, which suggests better quality and higher diversity of synthetic images. The victim's training data always achieve the highest IS value: the training data is the best representation of the input space among data we investigate. The Random Data are always the worst data due to the low IS values and high FID values.

We illustrate the examples of generated images in Appendix (Figures 5, 6, 7, 8). Humans cannot recognize these images, yet these synthetic data can be used to train a substitute model with high accuracy.

4.5 Further Attacks: A Case Study on Black-Box Adversarial Attacks

DNNs are vulnerable to adversarial examples, a slight modification on the original data sample that can easily fool DNNs (Szegedy et al. 2014; Carlini and Wagner 2017; Madry et al. 2017; Yuan et al. 2019). Black-box adversarial attacks assume that the adversary can only access the output of the model victim instead of its internal information, which is an emerging topic in adversarial attacks. After *ES Attack*, the adversary has full knowledge of the stolen substitute model. Can the adversary leverage the knowledge to conduct adversarial attacks against the victim model?

We evaluate the black-box adversarial attack against the victim model using the substitute model. We implement a white-box ℓ_∞-PGD attacks (Madry et al. 2017) and leverage the transferability of adversarial examples to conduct the attack. PGD attack is an iterative gradient-based attack in a white-box setting and has been proven effective in many machine learning models and tasks.

We report the success rate of adversarial attacks against our substitute model and the victim model (transferring attack) in Table 4. We compare the success rate of three adversarial attacks: 1) white-box attacks against the victim model, 2) white-box attacks against the substitute model, and 3) black-box attacks against the victim model via transferring. For the third attack, we evaluate the adversarial examples generated against substitute model (white-box attacks) on the victim model. We show the performance of the white-box ℓ_∞-PGD attack against the victim model as well. From the experimental results, the black-box adversarial attacks using the substitute model can achieve the same success rate as the white-box, which suggests that the substitute models can transfer the adversarial examples to the victim model successfully. Almost all black-box adversarial attacks achieve high accuracy rates (over 90%). We observe that the success rates of the black-box attacks against the victim models are less than that of the white-box attacks against the substitute models, but the change is subtle. Hence, most adversarial examples can be transferred from the substitute model to the victim model.

Table 4: Success rate of adversarial attacks.

Dataset	Model	Attack success rate (%)		
		white-box victim model	white-box substitute model	black-box victim odel
SVHN	ResNet18	99.95	99.94	98.71
SVHN	ResNet34	99.93	99.90	98.21
CIFAR10	ResNet18	100.00	100.00	93.60
CIFAR10	ResNet34	100.00	100.00	100.00
MNIST	LeNet5	86.07	99.57	92.14
KMNIST	LeNet5	66.44	99.72	98.99

Table 5: Evaluation of defenses against model stealing.

Dataset	Model	Accuracy w/o defense (%)	Accuracy w/ rounding (%)	Accuracy w/ top-k (%)
SVHN	ResNet34	93.19	92.58	88.76
CIFAR10	ResNet34	80.79	80.31	69.64
MNIST	LeNet5	92.03	96.69	95.43
KMNIST	LeNet5	90.37	90.03	91.11

5 Countermeasures of Model Stealing

In this section, we discuss the defense strategies of MLaaS providers and evaluate their effectiveness. Given the good performance of *OPT-SYN* on all the datasets, we evaluate three countermeasures against *OPT-SYN*. We find that the countermeasures are ineffective in defending or detecting proposed *OPT-SYN*.

5.1 Rounding Prediction

The MLaaS providers fix the decimals of the output prediction and zero-out the rest to provide only the necessary information. For example, if we round the prediction with 2 decimals, then $round(0.2474, r = 2) = 0.25$. We deploy rounding predictions with 2 decimals as a defensive strategy. Our experiments show that none of the model stealing attacks are affected by rounding to two decimals (Table 5). On average, the best accuracy of the substitute model even increases by 0.55% and the last accuracy only decreases by 0.30%.

5.2 Top-K Prediction

MLaaS providers only provide predictions of K classes with the highest probabilities. Since the adversary is aware of the Top-K defense used by MLaaS provider, we slightly change the adversary's attack by making up the missing probabilities of the rest classes. The adversary remains the probabilities of the Top-K classes and set the rest classes with the same probabilities. From the experiments, we observe that Top-1 prediction will not affect much on most datasets (Table 5). For the MNIST and KMNIST dataset, we find that the accuracy of the substitute model even gets improved. Top-1 prediction is only effective in preventing model stealing on the CIFAR10 dataset. However, we believe Top-1 prediction is a very strong defense, which will also affect normal users by providing very limited information. On average, the best accuracy of the substitute model with Top-1 prediction is only decreased by 2.86%. We find our attacks are minimally impacted by reducing the informativeness of black-box predictions in the response.

5.3 Anomaly Detection

Anomaly detection identifies the abnormal queries sent from users and detects the abnormal behavior that deviates from normal ones. For example, PRADA assumes that the distance between normal queries follows Gaussian distribution and detects the abnormal queries (Juuti et al. 2019). We evaluate the effectiveness of anomaly detection using PRADA against *OPT-SYN* We analyzed 300,000 image samples from the first five stealing epochs on the MNIST dataset. None of the synthetic images can be detected by PRADA. We believe that because the images are generated starting from the Gaussian distribution, the distances between queried images are too small to be detected by PRADA. Moreover, we find it is not practical for MLaaS providers to deploy a PRADA detector due to its high response time. In our experiments, it takes about 33 hours to process 300,000 images (2.46 images per second on average). With more images to be detected, the average response time will be further increased. Therefore, PRADA is ineffective and infeasible to detect the proposed *ES Attack*.

6 Related Work

6.1 Model Stealing Attacks and Defenses

Several studies have been proposed for model stealing attacks. Tramer *et al*. investigated stealing model parameters using equation solving (Tramèr et al. 2016). However, this approach is hard to extend to DNNs, which contains a larger number of than conventional machine learning models do. Papernot *et al*. proposed a similar framework to steal DNNs by training a substitute model (Papernot et al. 2017). Their goal is to approximate the victim model's decision boundaries to facilitate the adversarial attacks rather than to maximize the substitute model's accuracy. Thus their substitute model achieves a much lower classification accuracy compared to our work. In addition, to generate adversarial examples, their approach requires a small set of inputs that represents the input domain. In our work, we do not need this strong assumption. From the experimental results, the stolen model from *ES Attack* achieves a higher accuracy compared to that from (Papernot et al. 2017). Existing model stealing attacks against DNNs require an auxiliary dataset. Orekondy *et al*. proposed stealing attacks that assume access to a large dataset and use active learning to select the best samples to query (Orekondy, Schiele, and Fritz 2019). Correia-Silva *et al*. leveraged public datasets from the same task domain but with different distribution to steal DNNs. In our work, we assume the adversary does not have any auxiliary data related to the task domain.

Several detection approaches have been proposed for model stealing attacks. Juuti *et al*. detected the deviated distribution of queries from normal behavior (Juuti et al. 2019). Similarly, Kesarwani *et al*. proposed a detection tool that uses information gain to measure the model learning rate by users with the increasing number of queries (Kesarwani et al. 2018). The learning rate is measured to the coverage of the input feature space in the presence of collusion. We evaluate (Juuti et al. 2019) in our experiments and find that the detection approach is ineffective for our model stealing attacks.

6.2 Knowledge Distillation

In our model stealing attacks, we use distillation to transfer knowledge from the victim model to the substitute model. Knowledge distillation is widely used in model compression by transferring the knowledge from one model (teacher model) to another (student model) (Hinton, Vinyals, and Dean 2015; Buciluǎ, Caruana, and Niculescu-Mizil 2006). Most knowledge distillation approaches require the knowledge of training data. Recently, knowledge distillation without training data has recently been investigated (Lopes, Fenu, and Starner 2017; Nayak et al. 2019; Yin et al. 2020; Haroush et al. 2020) when the training data is infeasible due to large data size or privacy concerns. These approaches leverage inaccessible information of victim models (*e.g*., intermediate weights, model gradients, and activation statistics). However, model stealing adversaries cannot access such information. Hence, these data-free knowledge distillation approaches can not be applied in model stealing attacks.

7 Conclusion

We demonstrated that our attacks successfully stole various DNNs from the MLaaS providers without any data hurdles. Even without the knowledge on the victim's dataset, *ES Attack* (*OPT-SYN*) outperforms the two baseline attacks by 44.57% on average of four datasets in terms of best accuracy. Our experimental results illustrated the better quality and higher diversity of the generated synthetic data comparing with the auxiliary data, which benefits *ES Attack*. In addition, most existing defenses are ineffective to prevent *ES Attack*, where new countermeasures should be provided. Moreover, the stolen model can be used to conduct black-box adversarial attacks against the victim model, and sometimes the black-box attack achieves higher success rates compared with the white-box attack.

8 Ethics Impact

This paper address a critical challenge in the existing machine learning services: machine learning models can be stolen through queries. Most existing model stealing attacks require the knowledge of the private training data, which undervalued the impact of model stealing attacks. In this work, we show that even without any training data, the adversary can successfully steal the machine learning model. On one side, the adversary can provide the same machine learning service with a much lower price compared to MLaaS provider due to the low cost of model stealing attacks. On the other side, more importantly, model stealing attacks can facilitate further serious attacks. We hope the severity of model stealing attacks can attract attention of the community and encourages the researchers in academia and industry to investigate effective countermeasures.

References

Amazon. 2020. Amazon Rekognition pricing. URL https://aws.amazon.com/rekognition/pricing/.

Buciluǎ, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model compression. In *Proceedings of the 12th ACM international conference on Knowledge discovery and data mining (SIGKDD)*, 535–541. ACM.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 39–57. IEEE.

Clanuwat, T.; Bober-Irizar, M.; Kitamoto, A.; Lamb, A.; Yamamoto, K.; and Ha, D. 2018. Deep Learning for Classical Japanese Literature .

Correia-Silva, J. R.; Berriel, R. F.; Badue, C.; de Souza, A. F.; and Oliveira-Santos, T. 2018. Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data. In *International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Haroush, M.; Hubara, I.; Hoffer, E.; and Soudry, D. 2020. The Knowledge Within: Methods for Data-Free Model Compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 6626–6637.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Hunt, T.; Song, C.; Shokri, R.; Shmatikov, V.; and Witchel, E. 2018. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961* .

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

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

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Krizhevsky, A.; et al. 2009. Learning multiple layers of features from tiny images.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.; et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

Liu, Y.; Chen, X.; Liu, C.; and Song, D. 2017. Delving into transferable adversarial examples and black-box attacks. *International Conference on Learning Representations (ICLR)* .

Lopes, R. G.; Fenu, S.; and Starner, T. 2017. Data-free knowledge distillation for deep neural networks. *Neural Information Processing Systems (NIPS) Workshop* .

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* .

Mao, Q.; Lee, H.-Y.; Tseng, H.-Y.; Ma, S.; and Yang, M.-H. 2019. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1429–1437.

Mirzadeh, S.-I.; Farajtabar, M.; Li, A.; and Ghasemzadeh, H. 2019. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *arXiv preprint arXiv:1902.03393* .

Nayak, G. K.; Mopuri, K. R.; Shaj, V.; Radhakrishnan, V. B.; and Chakraborty, A. 2019. Zero-Shot Knowledge Distillation in Deep Networks. In *International Conference on Machine Learning (ICML)*, 4743–4751.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning* .

Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 427–436.

Odena, A.; Olah, C.; and Shlens, J. 2017. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Orekondy, T.; Schiele, B.; and Fritz, M. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4954–4963.

Pal, S.; Gupta, Y.; Shukla, A.; Kanade, A.; Shevade, S.; and Ganapathy, V. 2019. A framework for the extraction of Deep Neural Networks by leveraging public data. *arXiv preprint arXiv:1905.09165* .

Papernot, N.; McDaniel, P.; and Goodfellow, I. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* .

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the ACM on Asia conference on computer and communications security (AsiaCCS)*, 506–519. ACM.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, 2234–2242.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. *International Conference on Learning Representations (ICLR)* .

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

Wang, B.; and Gong, N. Z. 2018. Stealing hyperparameters in machine learning. In *IEEE Symposium on Security and Privacy (S&P)*, 36–52. IEEE.

Yin, H.; Molchanov, P.; Li, Z.; Alvarez, J. M.; Mallya, A.; Hoiem, D.; Jha, N. K.; and Kautz, J. 2020. Dreaming to Distill: Data-free Knowledge Transfer via DeepInversion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu, H.; Yang, K.; Zhang, T.; Tsai, Y.-Y.; Ho, T.-Y.; and Jin, Y. 2020. Cloudleak: Large-scale deep learning models stealing through adversarial examples. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*.

Yuan, X.; He, P.; Zhu, Q.; and Li, X. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems* 30(9): 2805–2824.

A Architecture of Generator G in *DNN-SYN*

Figure 4 illustrates the architecture of DNN-based generator in *DNN-SYN*. We follow the major architecture of ACGAN. We feed a random latent vector $\boldsymbol{z_i}$ and a generated one-hot label l_i into generator G. We concatenate these two vectors and up-sample them using several transposed convolution layers. Transposed convolution layers are parameterized by learnable weights. Each transposed convolution layer is followed by a batch normalization layer and a ReLU activation function except the last transposed convolution layer. 4 transposed convolution layers are used in the model. In the final layer, we use a Tanh function to output a synthesis image within $(-1, 1)$.

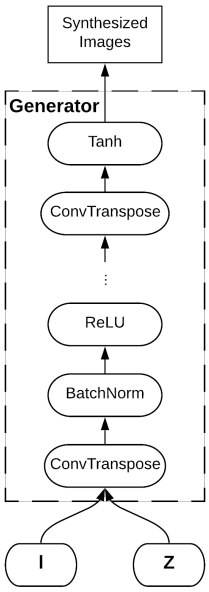


Figure 4: The DNN generator G used in *DNN-SYN*.

B Synthetic Example Images by *OPT-SYN*

We synthesize images using *OPT-SYN* with the best substitute model. We compare them with the victim's training data in the SVHN, CIFAR-10, MNIST and KMNIST datasets (Figure 5, 6, 7, 8). First row: images in the victim's training dataset. Second row: images in the synthetic dataset. Third Row: corresponding labels of images.

In our data-agnostic attack, the generated synthetic data is used to train a substitute model. Although the synthetic data, *i.e.*, a synthetic image, is unrecognizable to humans, it can be used to train a substitute DNN model with very good stealing performance as it helps capture the decision boundaries of the DNNs. Based on this observation, we have the first finding: the human-unrecognizable synthetic images can be used by DNN models to classify real images. Similarly, Nguyen *et al.* found the fact that the generated images, *i.e.*, imperceptible to humans, can be classified by a well-trained DNN with high confidence (Nguyen, Yosinski, and Clune 2015). Therefore, our analysis sheds light on the differences between human vision and DNNs.

Furthermore, we find that the synthetic dataset in the first stealing epoch $\mathcal{D}_{syn}^{(1)}$ shows higher diversity and better quality than the synthetic dataset generated in the last stealing epoch $\mathcal{D}_{syn}^{(N)}$ based on IS and FID. We believe it is due to either 1) the first synthetic dataset contributes more to the model training, or 2) IS and FID are not good metrics to measure the difference between synthetic data.

C Experimental Settings of Black-box Adversarial Attacks

Black-box adversarial attacks assume that the adversary can only access the output of the model by querying the machine learning service, but the adversary does not know the internal information of the model (*e.g.*, gradients). Transferability is commonly used to conduct black-box adversarial attacks, by training a surrogate model to transfer the adversarial attacks (Papernot, McDaniel, and Goodfellow 2016; Liu et al. 2017). In the experiments, we demonstrate how model stealing facilitates black-box adversarial attacks through transferability.

The evaluation of the black-box adversarial attack is outlined as follows: 1) Steal the victim model using *ES Attack* and get a substitute model; 2) Perform a white-box ℓ_∞-PGD attack against the substitute model and generate adversarial examples; 3) Evaluate the generated adversarial examples on the victim model. We evaluate the transferability of substitute model to perform a black-box adversarial attack. In the experiment, we use the test dataset $\mathcal{D}_{test}$ as our evaluation dataset. We follow the adversarial settings in (Madry et al. 2017) and consider the untargeted adversarial attacks, where adversarial examples can be classified as any classes other than the ground-truth class. For the MNIST and KMNIST dataset, we run 40 iterations of ℓ_∞-PGD attack with a step size of 0.01. We set the maximal perturbation size as 0.3. For the SVHN and CIFAR10 dataset, we run 20 iterations with a step size of 2/255. The maximal perturbation size is set as $8/255$.

D Performance of Countermeasures

D.1 Rounding Prediction

We investigate the impact of rounding decimals on the *ES Attack*. Figure 9 and 10 show the results of experiments with class probabilities rounded to 0-5 decimals. We compare the after-rounding classification accuracy of the substitute model and the victim model. Class probabilities rounded to 2 to 5 decimals have no effect on the adversary's success. When rounding further to 1 decimal, the attack is weakened, but still successful. When we round the precision to 0 decimal - the victim model only outputs 0 or 1 - the attack is further weakened, but still can predict in most cases (over 80% accuracy). We observe that rounded information brings the uncertainty of the prediction, while this uncertainty sometimes will slightly improve the training.

D.2 Top-K Prediction

In top-k prediction, the adversary remains the probabilities of the Top-K classes and set the rest classes with the same probabilities. For example, given an prediction output of $[0.5, 0.02, 0.3, 0.02, 0.15, 0.01]$, MLaaS provider

Figure 5: **The SVHN Dataset**

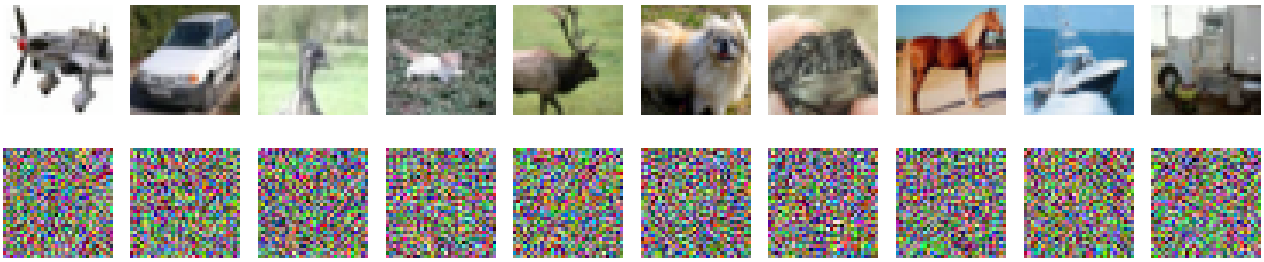


Figure 6: **The CIFAR-10 Dataset**

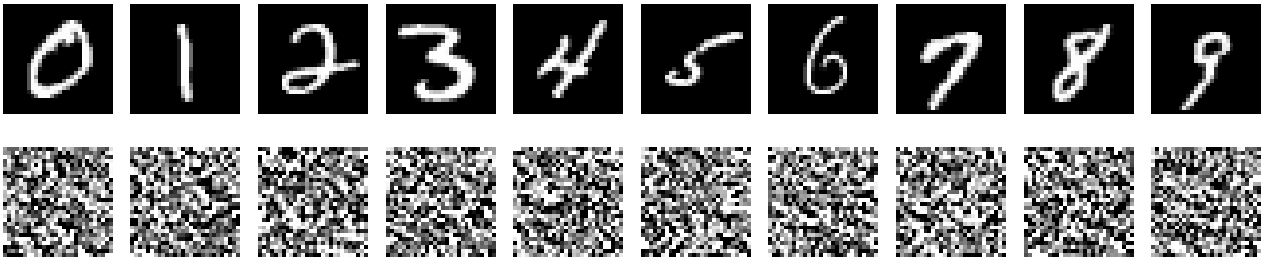


Figure 7: **The MNIST Dataset**

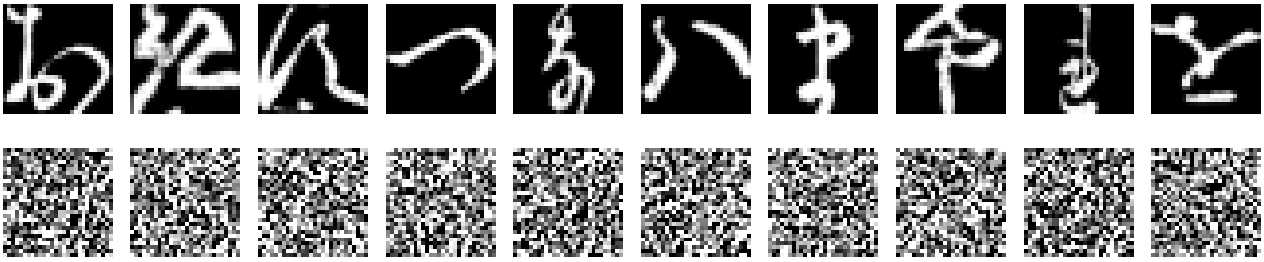


Figure 8: **The KMNIST Dataset**

will convert it to $[0.5, 0.0, 0.3, 0.0, 0.0, 0.0]$ using Top-2 defense and the adversary will then make it up to $[0.5, 0.05, 0.3, 0.05, 0.05, 0.05]$. We compare the impact of probabilities with different numbers (K) of classes on our model stealing attacks (Figure 11 and 12). The performance of model stealing attacks is not decreased with fewer classes providing probabilities (small K).

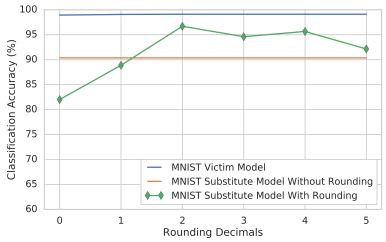


Figure 9: **Evaluation of Rounding Prediction on the MNIST dataset**.

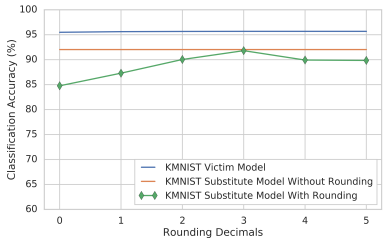


Figure 10: **Evaluation of Rounding Prediction on the KMNIST dataset**.

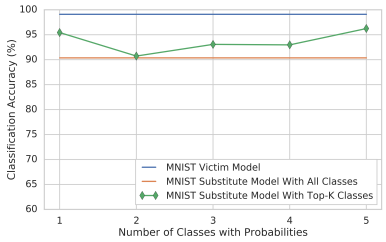


Figure 11: **Evaluation of Top-K Prediction on the MNIST dataset**.

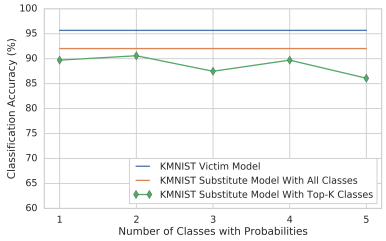


Figure 12: **Evaluation of Top-K Prediction on the KMNIST dataset**.