GAME: Generative-Based Adaptive Model Extraction Attack

Yi Xie[1], Mengdie Huang[1], Xiaoyu Zhang[1], Changyu Dong[2], Willy Susilo[3], and Xiaofeng Chen[1](✉)

[1] State Key Laboratory of Integrated Service Networks (ISN)
Xidian University, Xi'an 710071, P.R. China
xfchen@xidian.edu.cn

[2] School of Computing, Newcastle University Newcastle NE1 7RU, UK
Changyu.dong@gmail.com

[3] School of Computing and IT, University of Wollongong, Wollongong 2522, NSW, Australia
wsusilo@uow.edu.au

Abstract. The outstanding performance of deep learning has prompted the rise of Machine Learning as a Service (MLaaS), which significantly reduces the difficulty for users to train and deploy models. For privacy and security considerations, most models in the MLaaS scenario only provide users with black-box access. However, previous works have shown that this defense mechanism still faces potential threats, such as model extraction attacks, which aim at stealing the function or parameters of a black-box victim model. To further study the vulnerability of publicly deployed models, we propose a novel model extraction attack named **G**enerative-Based **A**daptive **M**odel **E**xtraction (GAME), which augments query data adaptively in a sample limited scenario using auxiliary classifier GANs (AC-GAN). Compared with the previous work, our attack has the following advantages: adaptive data generation without original datasets, high fidelity, high accuracy, and high stability under different data distributions. According to extensive experiments, we observe that: (1) GAME poses a threat to victim models despite the model architectures and the training sets; (2) synthetic samples closed to decision boundary without deviating from the center of the target distribution can accelerate the extraction process; (3) compared to state-of-the-art work, GAME improves relative accuracy by 12% at much lower data and query costs without the reliance on domain relevance of proxy datasets.

Keywords: Model extraction attack · Data augmentation · Adaptive strategy · Auxiliary classifier GANs.

1 Introduction

Deep learning models have been deployed in more and more fields, such as computer vision, natural language processing, and speech recognition, for their amazing ability to solve various challenging classification problems. Due to the high market demand for deep learning technology, the concept of Machine Learning as a Service (MLaaS) has been rapidly promoted. More and more companies train and deploy machine learning models in this way, which greatly lower the software and hardware thresholds for individual developers and small businesses. In the MLaaS scenario, the open deployment of the model can not only facilitate users but also enterprise. However, due to data breaches and laws, such as the European Union's General Data Protection Regulation (GDPR), this practice faces many restrictions. In recent years, much work has been done to study the privacy and security issues of the MLaaS models, which includes: membership inference attacks [11,23,24], model inversion attacks [5,22,34], and model extraction attacks [1,10,18,20,21,27]. They respectively aim to infer whether a particular sample is present in the training set, reconstruct the private training data, and steal the function or parameters of the victim model.

Among these attacks, the first two attacks on sample privacy have been widely studied, while the attacks on the model privacy is still in its infancy. In model extraction attack, the attacker attempts to steal the function/parameters of the victim black-box model, which will compromise the model owner's interests. In addition, it can also be a stepping stone to other attacks, e.g., model evasion attacks [20]. At present, most model extraction attacks use learning-based methods: the prediction vector output by the model is used as the soft label of the input sample to train the substitute model, which is similar to the knowledge distillation technology [8] in the field of model compression [6]. To evaluate the performance of the attack, two metrics, accuracy and fidelity [10], are introduced into the field. The former focuses on classifying samples correctly, while the latter focuses on improving the similarity between the predictions.

Based on our observations, three challenges prevent the implementation of a model extraction attack: limited data, irrelevant proxy distribution, and expensive query cost. Firstly, for business and security considerations, the internal information (including training data) of MLaaS models is often not publicly released, making it hard for attackers to construct a query set. To this end, many works have adopted the method of data-free knowledge distillation [4,15] to attack. However, due to the black-box access to the victim model in the model extraction scenario, the attacker needs to use gradient approximation [3,31] for backpropagation. Unfortunately, this query-based gradient approximation method often increases the query budget exponentially. Secondly, it is difficult to find a suitable distribution of proxy datasets. This cannot be ignored because it directly affects the attacker's query efficiency. To this end, some studies [30,36] have found that *adversarial samples* near the boundary can greatly improve the attack performance. However, such works often only use attack success rates (ASRs) of adversarial examples generated by the substitute model to evaluate the stealing performance, which ignores the performance of substitute models on the original task. Finally, to reduce the attack cost and the risk of being intercepted by the defense mechanism, the attacker needs to design an efficient query strategy to reduce the total number of queries without affecting the extraction performance significantly. With regard to this, some works [18,19] try to use active learning methods to build a query queue from public datasets. Unfortunately, the upper limit of its attack performance depends on the mutual information between the chosen public dataset and the original training set.

1.1 Our Contribution

To address the issues faced by model extraction attacks, we first propose a data augmentation algorithm that combines active learning and dynamic updating mechanisms. On the basis of this method, we design a novel **G**enerative-Based **A**daptive **M**odel **E**xtraction (GAME) in a practical scenario, where the attacker has no access to original training sets. Compared with previous works, GAME has the following advantages: (i) The AC-GAN based data augmentation algorithm can provide the attacker with enough query samples in the limited samples scenario. Besides, the class control mechanisms of AC-GAN can reduce the granularity of synthesized samples. (ii) The category selection strategy based on active learning works well with AC-GAN. Specifically, this strategy could help the attacker generate query samples which have higher distillation efficiency. (iii) The output distribution of the generator is adaptively fine-tuned according to multiple feedback indicators. Theoretical analysis and extensive experiments show that setting a reasonable feedback indicator can improve the GAME attack performance. In summary, we make the following contributions:

- We first propose to use *boundary samples without deviating from the target distribution* to conduct model extraction attacks in the limited samples scenario where it is difficult to obtain original sets or even public datasets.
- We propose an AC-GAN based data augmentation method for model extraction attack, which combines two strategies, active learning and generator dynamic updating, to increase the efficiency of stealing.
- We conducted extensive experiments to show that GAME can achieve higher fidelity and accuracy compared with state-of-the-art methods, especially in limited proxy samples scenarios. Furthermore, the effectiveness of the active learning and generator dynamic updating strategy is also demonstrated.

1.2 Related Work

Model Extraction Attacks. The emergence of model extraction attacks stems from the tension between the public access and model confidentiality of MLaaS platforms. In 2016, Tramèr et al. [27] proposed equation-solving attack and path-finding attack against simple models, demonstrating the feasibility of extraction attacks. Following this, many works also studied model extraction attacks against complex deep neural networks [1,10,12,18]. Most of the early works tried to find methods that can exactly recover the weight of the victim model, such as [10] and [21]. However, these methods are often inefficient and difficult to implement. To this end, several works have turned their attention to model equivalent extraction attacks [1,12,18], which only attempt to steal a functionally approximate model without the requirements on model structure and parameters.

The core idea of most model-equivalent extraction attacks originates from knowledge distillation techniques [8], that is, query the victim model to obtain soft labels for training substitute models. However, two problems faced by knowledge distillation are also exacerbated in the field of model stealing: limited samples problem and query efficiency problem. In most MLaaS scenarios, it is difficult for attackers to improve the performance of substitute models with limited query samples. Jacobian-based data augmentation [20] is an earlier work that attempted to solve this problem. In the work, original training samples are perturbed with the Jacobian matrix to enlarge the query set. Another work Black-Box Ripper [1] uses a generative evolution algorithm to generate high-response samples. Differing from these methods, some works focus on data-free model extraction attacks, which is close to the field of data-free knowledge distillation, such as [12,28,30]. This kind of work often uses gradient approximation methods, which leads to excessive query costs.

On the other hand, considering the benefits and defense mechanisms, the attacker needs to minimize the query cost without compromising stealing performance. Knockoff [18] formalizes sample selection in model extraction attack as a Multi-armed Bandit Problem and uses the Gradient Bandit Algorithm to choose the most informative samples. The work ActiveThief [19] uses active learning methods to reduce the query cost when using public data sets for model stealing. We propose a high-fidelity model extraction attack GAME, which considers the above two challenges at the same time.

Knowledge Distillation. Although the goal of knowledge distillation, which is compressing large machine learning models, is different from that of model extraction attacks, their execution processes are similar: using the teacher (victim) model to label a series of samples for student (substitute) model training.

Knowledge distillation faces the same problems as model extraction attacks.

One problem is insufficient query samples. Specifically, the model owner may be unable to collect enough query samples for knowledge distillation. To deal with this problem, the work [2] proposes a GAN-based Data-Free Learning (DAFL) method. In [16], "Data impression", was crafted from random noise to train a substitute model. The work [35] uses a conditional generator to generate high-confidence samples of specific classes for the teacher model.

Another problem is the low query efficiency. The work [29] proposes an active-learning-based method combined with mix-up technology. Similarly, the work [33] uses an uncertainty-based mix-up method to reduce the computation costs of both the teacher and the student models.

Although there are some similarities between model extraction and knowledge distillation, most solutions are hard to transfer to the model extraction scenario due to the different access rights.

1.3 Organization

The rest of this paper is organized as follows: In Section 2, we introduce the necessary preliminaries. In Section 3, we present the specific design of the GAME attack. In Section 4, GAME is evaluated under different dataset and model architecture settings, and compared with SOTA works. The ablation study about GAME is presented in Section 5. We conclude this paper in Section 6.

2 Preliminaries

2.1 Problem Formulation

In this subsection, we formalize the model extraction attack problem. Suppose a victim model N_V is deployed on an MLaaS platform, which only grants black-box access rights to users (including malicious users). Any samples uploaded by users will be fed into the victim model N_V to get the prediction vector y_{pred}:

$$y_{pred} = N_V(x). \tag{1}$$

The attacker's goal is to find a substitute model N_S that is functionally equivalent to N_V in the victim's domain $\mathcal{D}_\mathcal{V}$. Formally, the objective of model extraction attacks is:

$$\arg\max_{N_S} \mathcal{P}_{x\sim\mathcal{D}_\mathcal{V}} \left[\arg\max_i N_V^i(x) = \arg\max_i N_S^i(x)\right], \tag{2}$$

where i represents the component of the ith class of the prediction vector output by the model, and x is the sample randomly sampled from the sample domain D_V. However, due to the confidentiality of the victim's training set, it's difficult for the attacker to obtain the precise distribution of D_V. A practical approach for the attacker is to minimize the difference between the two models in the proxy sample domain D_P, that is:

$$\arg\min_{N_S} \mathbb{E}_{x\sim\mathcal{D}_P} \left[\mathcal{L}\left(N_V(x), N_S(x)\right)\right], \tag{3}$$

where $\mathcal{L}$ represents the loss function, which is used to measure the distance between the predictions of two models on the input x. It can be inferred that when the correlation between D_P and D_V is higher, the substitute model trained by the attacker has higher fidelity, according to the Eq. (2) and Eq. (3).

2.2 Adversary Capability

Attack surface. The attacker can only use the prediction API N_V^* provided by the MLaaS platform to fit the decision boundary of the victim model, which means that the internal information of the victim model is agnostic to the attacker, including model structure, parameters, and training datasets. Therefore, the attacker cannot perform back-propagation through the victim model. Besides, due to the pay-per-use manner, the attacker has to pay for each query. After the attacker pays for the query and sends x to N_V^*, the MLaaS platform may have two different responses: (i) the whole prediction vector y_{pred} of N_V^*; (ii) the top-1 class $t_{pred} = \arg\max_i y_{pred}^i$. The attacker can get better performance when the MLaaS platform returns the whole prediction vector y_{pred} as it contains more information.

Proxy datasets. Depending on the adversary's capability, the query datasets used by the attacker can be divided into the following four categories: (i) The whole original datasets, which means the attacker has the same training samples as the victim model N_V. This situation only occurs in scenarios where N_V is trained on public datasets. (ii) Part of original datasets. In this case, the attacker has a low-density sampling of the original distribution (or original domain D_V). The limited number of samples may reduce the generalization performance of the substitute model in D_V. (iii) Domain-related datasets. Due to the commercial characteristics of the MLaaS platform, the service provider will not hide the task information of the victim model, so the attacker can easily collect the relevant public data sets as the proxy set. (iv) Domain-unrelated datasets. When no task information about the victim model is provided, the attacker can only randomly select some public dataset. In Section 5.3, we discuss in detail the effect of proxy dataset distribution on GAME.

Model architecture. The architecture and hyper-parameters of the victim model are also important private information for model extraction attacks. However, this information is often unavailable for MLaaS users (including attackers) due to privacy concerns or business strategies. The model structure used by the attacker can be divided into the following two categories: (i) Same architecture. If the attacker knows the specific architecture of the victim model (although it is challenging), he will adopt it to the substitute model. (ii) Task-related architecture. If the attacker has no information about the architecture of the victim model, the substitute model could be designed by the task information.

2.3 Auxiliary Classifier GANs

In this subsection we detail the basics aboutauxiliary classifier GANs (AC-GAN) [17] as it is the core of our proposed data augmentation algorithm. AC-GAN employs label conditioning in GAN training to improve the quality of the generated samples and control the specific categories of samples.

Like traditional GAN, AC-GAN is also composed of two parts: a generator and a discriminator. The difference is that the generator of AC-GAN needs to input a label y_g in addition to the latent noise z. Besides, the output of the discriminator includes both the probability distribution over source $P(S|X)$ and the probability distribution over class labels $P(C|X)$. The objective function of AC-GAN can be divided into two parts, (i) the log-likelihood of the correct source, L_S, and (ii) the log-likelihood of the correct class, L_C, which are as follows:

$$L_S = E\left[\log P\left(S = \text{ real } \mid X_{\text{real}}\right)\right] + E\left[\log P\left(S = \text{ fake } \mid X_{\text{fake}}\right)\right], \qquad (4)$$

$$L_C = E\left[\log P\left(C = c \mid X_{\text{real}}\right)\right] + E\left[\log P\left(C = c \mid X_{\text{fake}}\right)\right]. \qquad (5)$$

Then the goal of the discriminator and generator can be defined by the following two formulas:

$$G = \arg\max_{G} L_C - L_S, \tag{6}$$

$$D = \arg\max_{D} L_C + L_S. \tag{7}$$

Solving the formula above through an iterative training method allows the generator G to simulate the original dataset distribution.

3 Design of GAME

3.1 Core Idea

The goal of the model extraction attack is to obtain a replica of the victim model. Specifically, it attempts to ensure that the two models can output similar prediction vectors of any input, as denoted in the Eq. (2) above. From the perspective of the model decision boundary, the attacker needs to fit the boundary of the victim model to improve the performance (the similarity to the victim model) of the substitute model on seen and unseen samples. As shown in Fig. 1(b), it is difficult to achieve this due to the lack of original data. Data augmentation is a way to handle the problem but still faces two challenges: **similarity** and **efficiency**. Similarity means that the synthesized samples should have a similar distribution to the real samples in the victim domain. Efficiency means that the attacker attempts to achieve higher fidelity with the same query costs.

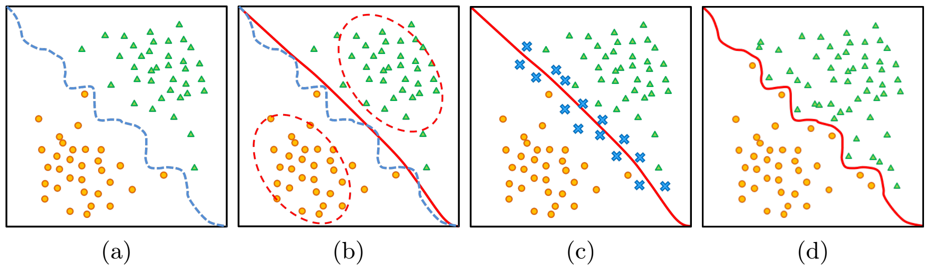

Fig. 1: The strategy of Data Augmentation: (a) Original decision boundary (blue dotted line) of victim model. (b) Using samples at the center of the distribution (yellow dots & green triangles) can only approximate a rough boundary (red solid line). (c) GAME uses augmented samples (blue cross) to obtain boundary information more efficiently and precisely. (d) The stolen decision boundary is very close to the original decision boundary.

To address the challenges mentioned above, this work proposed a model extraction algorithm based on an adaptive data augmentation strategy. The core

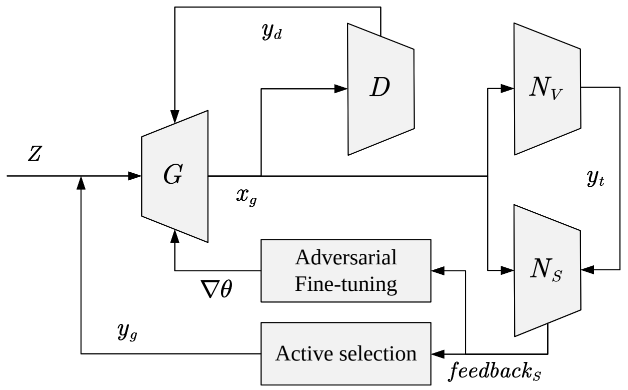

Fig. 2: Framework overview of GAME.

idea of the strategy is to generate samples close to the **key part** of the decision boundary without deviating from the center of the target distribution, as shown in Fig. 1(c) - (d). To achieve this goal, the attacker takes four steps: (i) Learning the distribution of the proxy dataset. Due to the confidentiality of the training sets used by the victim model, attackers can only obtain a proxy dataset. Then the distribution of proxy dataset is learned with the AC-GAN model, part of which cloud be regarded as the initial version of the generator for subsequent steps. (ii) Selecting categories based on active learning strategy. Due to the difference in the distribution of the original training set and the proxy dataset, it is a wise choice to adjust the class distribution of the synthetic samples. GAME adopts an active learning-based approach, which is described in detail in Section 3.2. (iii) Generating boundary samples with an adaptive strategy. The category selected in the previous step is used as the condition of AC-GAN for sample generation. Meanwhile, we proposed an adversarial fine-tuning algorithm to generate samples near the boundary. (iv) Distilling victim model with synthetic samples. Finally, using the samples generated in the previous step, the knowledge of the victim model can be transferred to the local substitute model.

3.2 Framework

As shown in Fig. 2, the GAME framework consists of 6 modules: generator G, discriminator D, victim model N_V, substitute model N_S, adversarial fine-tuning module, and active selection module. The arrow in the figure shows the data flow of the framework, and it can be divided into four phases: Target Distribution Learning (TDL), Active Categories Selecting (ACS), Adaptive Generator Updating (AGU), and Generative Model Distillation (GMD). The whole process of GAME attack is shown in Algorithm 1.

Algorithm 1 GAME attack

```
Input: Proxy data set $D_p = (X_p, Y_p)$, victim model API $N_V^*$.
Output: Substitute model $N_S$
 1: Initialize generator $G$ and discriminator $D$
 2: Target Distribution Learning
 3: $G, D \leftarrow Train_{G,D}(X_p, Y_p)$ // Train ACGAN with $D_p$
 4: Active Categories Selecting
 5: Updating $P_\pi$
 6: $Y_g \sim P_\pi$ // Generating with active learning strategy
 7: $X_0 \leftarrow \emptyset$
 8: for $i \leftarrow 1$ to $n\_epochs$ do
 9:     Substitute Model Distillation
10:     $X_g \leftarrow G(z, Y_g)$
11:     $Y_V \leftarrow N_V^*(X_g)$
12:     $X_{all} \leftarrow X_{all} \cup X_g$ , $Y_{all} \leftarrow Y_{all} \cup Y_V$
13:     $N_S \leftarrow Train_{N_S}(X_g, Y_V)$
14:     AC-GAN Dynamically Updating
15:     $X_g \leftarrow G(z, Y_g)$
16:     $Loss_{Total} \leftarrow \beta_1 \times L_{res} + \beta_2 \times L_{bou} + \beta_3 \times L_{adv} + \beta_4 \times L_{dif}$
17:     $\theta_G \leftarrow \theta_G - \frac{\partial Loss_{Total}}{\partial \theta_G}$
18:     Update generator $G$ with $\theta_G$
19: end for
20: $N_S \leftarrow Train_{N_S}(X_{all}, Y_{all}))$ // Final train
21: return $N_S$
```

Target Distribution Learning. In the TDL phase, the AC-GAN is used to learn the distribution of the proxy dataset $D_p = \{(x_i, y_i)|i = 1, ..., N\}$ owned by the attacker. In this phase, the generator G and discriminator D are trained normally, as denoted in Eq. (4) - Eq. (7). Specifically, in every iteration, the generator G needs to get a latent noise z and a target generation label y_g as input to generate sample x_g. Then the discriminator D will output the prediction vector y_d, which consists of two parts: $y_d = (y_s, y_c)$. The y_s represents the validity probability, and y_c represents the prediction vector. Hence G could be updated to improve the validity probability y_s and decrease the distance between y_g and y_c. Then the discriminator needs to be updated to learn the difference between the synthetic sample and the initial dataset D_p. After enough rounds of iterative training, the generator G can generate samples that fit the distribution of D_p.

Active Categories Selecting. Before AC-GAN generates query samples, the category y_g of the synthetic samples needs to be determined by the active learning strategy. Two active learning strategies are proposed: (i) Prediction uncertainty, which expects to select the class with the highest uncertainty to the substitute model N_S. The probability that the ith category is selected can be calculated with:

$$P_{unc}^i = \frac{c_i}{\sum_{t=1}^N c_t}, c_i = 1 - \max\{softmax[N_S(G(z,i))]\}, \tag{8}$$

where c_i represents the unconfidence of N_S for the sample of the ith class. (ii) Deviation distance, which represents the size of the predicted distance of N_S and N_V for the same sample. It could be calculated by:

$$P^i_{dev} = \frac{d_i}{\sum_{t=1}^{N} d_t}, d_i = KL[N_S(G(z,i)), N^*_V(G(z,i))]. \tag{9}$$

Additionally, KL-divergence, denoted as KL, is used to measure the distance between the two outputs.

Generative Model Distillation. After the categories of synthetic samples are determined, generative knowledge distillation is implemented using AC-GAN, which includes three steps: (i) generating query samples; (ii) querying the victim model to get the prediction; (iii) training substitute model on the query sample set. Some tricks are introduced to improve the performance of distillation: in the first step, the target label y_g is randomly assigned to ensure the balance between classes; during the second step, the whole prediction vector y_c (instead of the top-1 class of y_c) output by the discriminator D is used as a soft label of input sample x_g; during the third step, we use the Kullback-Leibler divergence loss to obtain more information contained in soft labels.

Adaptive Generator Updating. As stealing progresses, the substitute model N_S exhibits increasingly similar behavior to the victim model N_V. Therefore N_S could be regarded as the shadow model of N_V for fine-tuning the generator G, instead of requiring white-box access to the victim model for backpropagation. Thus, to improve the knowledge transfer efficiency, GAME introduces the following indicators to fine-tuning the generator G.

(i) Model responsivity L_{res}. Synthetic samples should be as close as possible to some key features of the victim domain. GAME achieves this by increasing the non-negative logits layer output of N_S, which is denoted as f_S:

$$L_{res} = -\sum_{i=1}^{N} \max(0, f^i_S). \tag{10}$$

(ii) Boundary distance L_{bou}. As described in Section 3.1, GAME uses boundary samples to improve extraction efficiency. It is achieved by minimizing the distance between the top-2 prediction vector components of substitute model N_S, which could be denoted as:

$$L_{bou} = N_S(x)^{top1} - N_S(x)^{top2}. \tag{11}$$

(iii) Adversarial correction L_{adv}. As a complement to L_{res}, we also introduce an indicator L_{adv} based on the idea of adversarial examples, which aims to help the generator generate samples that cross the decision boundary. Formally, the adversarial correction indicator can be defined as follows:

$$L_{adv} = -CE(N_S(x), \arg\max_i N_S(x)^i). \tag{12}$$

(iv) Prediction difference L_{dif}. We also introduce the indicator L_{dif}, which accelerates the stealing process by increasing the disagreement of synthetic samples for both the victim model and the substitute model. Concretely, it can be defined as:

$$L_{dif} = -KL(N_S(x), N_V(x)). \tag{13}$$

Therefore, the total loss function of the generator in the iterative update process is denoted as:

$$Loss_{Total} = \beta_1 \times L_{res} + \beta_2 \times L_{bou} + \beta_3 \times L_{adv} + \beta_4 \times L_{dif}. \tag{14}$$

Where β_i in the formula is a manually adjusted weighting factor to balance the effect of each loss term on the total loss. The attacker needs to minimize this loss function to get better performance of the generator G. Since the parameters of the substitute model N_S will change at each iteration, the generator also needs to recalculated the loss function to update its parameters.

4 Performance Evaluation

4.1 Experiments Settings

Datasets and victim model. In this experiment, we use two pairs of datasets for experiments: (i) MNIST [14] as the original dataset and Fashion-MNIST [32] as the proxy dataset; (ii) BelgiumTSC [26] as the original dataset and GTSRB [9] as the proxy dataset. For each dataset, we resize all images to 32×32 and shuffle the order. We use LeNet [14] and AlexNet [13] as the architecture of the victim model in the MNIST and BelgiumTSC experiments, respectively. These victim models were trained for 15 epochs on MNIST and 20 epochs on BelgiumTSC with ADAM at an initial learning rate of 0.001.

Attacker model. We use 4 different architectures for the attacker model: half-LeNet and VGG-16 [25] for Fashion-MNIST, half-AlexNet, and ResNet-18 [7] for GTSRB. These models were trained for 40 epochs with ADAM. The initial learning rate is 0.1 for half-LeNet and 0.01 for VGG-16. The query budget is 8k for Fashion-MNIST and 6k for GTSRB. For the generator, we adopt a structure consisting of 3 convolutional layers, interleaved with up-sampling layers, batch normalization layers, and ReLU activations. The discriminator had five convolutional layers, interleaved with ReLU activations, dropout layers (with 0.25 probability), and batch normalization layers (except for the last layer).

Evaluation metric. We evaluate the performance by two metrics: (i) Fidelity: the similarity of the top-1 class of the output vector between the substitute model and the victim model. (ii) Accuracy: computed on the top-1 class of the prediction with true labels. Formally, these two metrics could be defined as:

$$Fidelity(N_S) = \frac{1}{|D_{test}|} \sum_{(x,y)\in D_{test}} \mathbf{I}(t_{N_S}(x) = t_{N_V^*}(x)), \tag{15}$$

$$Accuracy(N_S) = \frac{1}{|D_{test}|} \sum_{(x,y)\in D_{test}} \mathbf{I}(t_{N_S}(x) = y), \tag{16}$$

$$t_f(x) = \arg\max_i f(x)^i. \tag{17}$$

Where $\mathbf{I}(\cdot)$ represents the indicator function, and $t_f(x)$ represents the largest component of the output of the function f.

4.2 Results

We compare the attack performance of GAME with the other three attacks: (i) Baseline, which randomly selects samples from the proxy dataset for querying, and then uses the output as soft labels to train the substitute model. (ii) Knockoff [18], which uses an active sample selection method based on the gradient bandit algorithm to improve the query efficiency. (iii) JBDA [20], which utilizes Jacobian-based dataset augmentation to augment the query sets. All comparisons were performed under the same settings (including learning rate, model architecture, query budget, number of training epochs, etc.). Each experiment was run at least three times. In addition to the two metrics mentioned in Section 4.1, we also introduce relative accuracy, relative($\times$), to show the difference in accuracy between the substitute model and the victim model. The detailed results are shown in Table 1 and Table 2.

Table 1: Comparison of fidelity, accuracy and relative accuracy obtained from various attacks on the MNIST & Fashion-MNIST datasets.

Dataset	Architecture	Method	Fidelity(%)	Accuracy(%)	Relative($\times$)
MNIST	LeNet	Victim model	100	98.74	1.00
Fashion-MNIST (Proxy)	half-LeNet	Baseline	75.42 ± 1.93	75.12 ± 1.97	0.76
		Knockoff[18]	83.30 ± 3.90	82.93 ± 3.84	0.84
		JBDA[20]	79.85 ± 3.74	79.44 ± 3.73	0.80
		GAME (**Ours**)	90.93 ± 1.61	90.36 ± 1.67	0.92
	VGG-16	Baseline	81.25 ± 1.83	80.96 ± 1.85	0.82
		Knockoff[18]	81.02 ± 2.23	80.70 ± 2.22	0.82
		JBDA[20]	69.76 ± 0.83	69.42 ± 0.91	0.70
		GAME (**Ours**)	86.41 ± 1.44	85.97 ± 1.47	0.87

MNIST & Fashion-MNIST. We present the fidelity and accuracy of extracted models across various attacks for Fashion-MNIST in Table 1. In this experiment, the victim model achieved an accuracy of 98.74% after training for 15 epochs on the MNIST dataset. We find that each method showed advantages over the baseline in the half-LeNet experiment. Among these attacks, GAME gets the highest fidelity (90.93%) and accuracy (90.36%), which achieves a relative accuracy of 0.92. When the substitute model architecture is changed to VGG-16, the performance of all method is reduced, but GAME still leads other attacks with the fidelity of 86.41%.

BelgiumTSC & GTSRB. GAME shows similar advantages in the second pair of dataset experiments, as shown in Table 2. In this experiment, the victim model achieved an accuracy of 98.29% after training for 20 epochs on the BelgiumTSC dataset. When the structure of the substitute model is set to half-AlexNet, all methods have more than 10% improvement compared to the baseline, and GAME achieves the best attack performance. Besides, regardless of the substitute model architecture, the GAME attack exhibits certain stability: the standard deviation of fidelity and accuracy are relatively small, especially in the experiments of ResNet-18.

Table 2: Comparison of fidelity, accuracy and relative accuracy obtained from various attacks on the BelgiumTSC & GTSRB datasets.

Dataset	Architecture	Method	Fidelity(%)	Accuracy(%)	Relative(×)
BelgiumTSC	AlexNet	Victim model	100	98.29	1.00
GTSRB (Proxy)	half-AlexNet	Baseline	63.79 ± 0.56	63.20 ± 0.53	0.64
		Knockoff[18]	74.33 ± 1.83	73.39 ± 1.86	0.75
		JBDA[20]	74.09 ± 1.93	73.39 ± 1.99	0.75
		GAME (**Ours**)	76.74 ± 1.06	75.88 ± 1.07	0.77
	ResNet-18	Baseline	66.79 ± 0.98	65.92 ± 1.18	0.67
		Knockoff[18]	73.26 ± 1.99	72.16 ± 2.04	0.73
		JBDA[20]	69.34 ± 0.78	68.49 ± 0.72	0.70
		GAME (**Ours**)	74.52 ± 0.37	73.77 ± 0.46	0.75

5 Ablation Study

5.1 Impact of Category Selection Strategies

To evaluate the impact of different category selection strategies on attack efficiency, we evaluated the performance on two pairs of datasets while keeping the experiment settings consistent with Section 4.1. To improve the reliability of the results, each experiment was run for 6 rounds and the average value is shown in Fig. 3. Due to the high accuracy of the two victim models (98.74% for the MNIST model and 98.29% for the BelgiumTSC model), the fidelity and accuracy of each substitute model are close.

According to the first row (experiments on MNIST&Fashion-MNIST) of Fig. 3, the uncertainty-based active learning strategy outperforms others with the query budget increasing. When the query budget is increased to 5k, the deviation-based strategy achieves better performance than that of the random strategy.

As for the second row (experiments on BelgiumTSC&GTSRB) of Fig. 3, the uncertainty-based strategy also gets a good performance. However, the performance of the deviation strategy is weaker than that of the random strategy when the query budget is greater than 3k. We believe that it may be related to the distribution of the substitute dataset compared to the original dataset, which is further discussed in Section 5.3. In conclusion, choosing the uncertainty-based strategy is the best practice for launching a GAME attack.

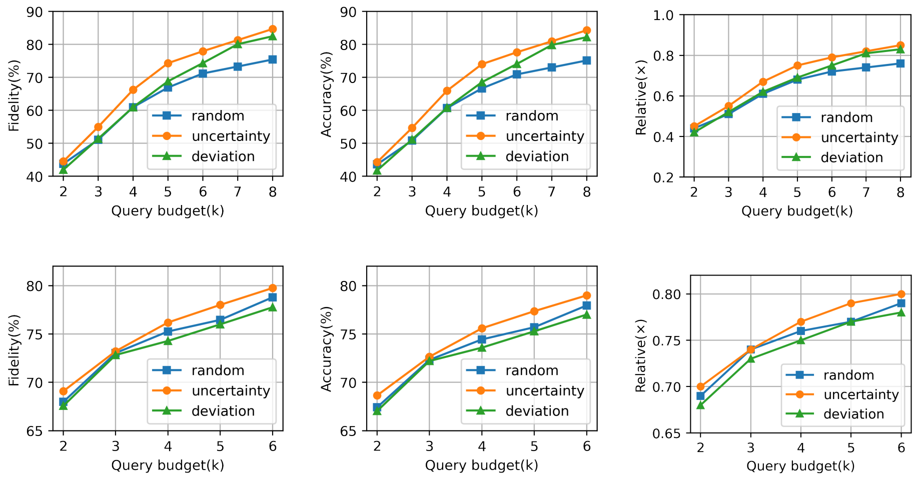

Fig. 3: Comparison of different active learning strategies. The first row shows the attack performance on the MNIST&Fashion-MNIST dataset, and the second row shows the performance on the BelgiumTSC&GTSRB dataset, which are evaluated from three indicators: fidelity, accuracy, and relative accuracy.

5.2 Impact of Fine-tuning Indicators

We compare the performance of the fine-tuning indicators and their combination strategies. For a fair comparison, we keep all configurations unchanged except the fine-tuning indicators. Furthermore, to eliminate the impact of generator performance on experiments, we use the same pre-trained AC-GAN model for all experiments, i.e., updating on the same initial generator. For multiple indicators experiments, we use coefficients β_i to control the proportion of each item to balance each indicator, as shown in Eq. (14). Each experiment was run multiple

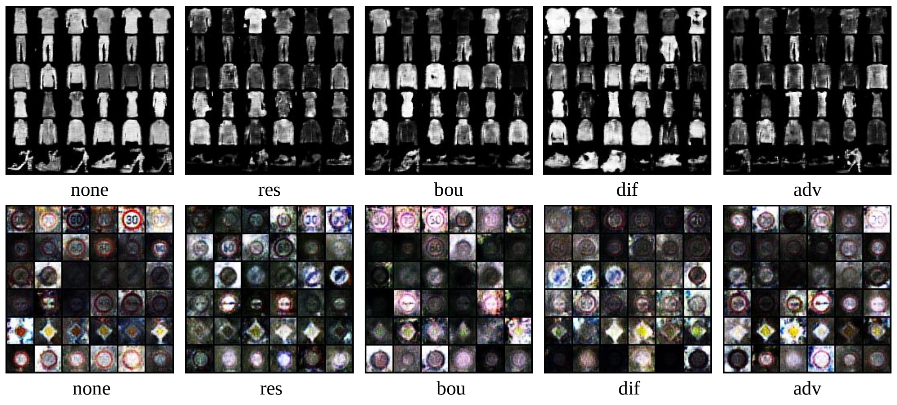

Fig. 4: Synthetic samples of different indicators. The first row shows the synthetic Fashion-MNIST images. The second row show the synthetic GTSRB images.

times to ensure correctness. The results are shown in Fig. 5, where n represents the number of indices to be combined, and *none* represents the attack with pre-trained AC-GAN without fine-tuning. We also show samples generated by different indicators, as shown in Fig. 4. The displayed samples are randomly generated without manual screening.

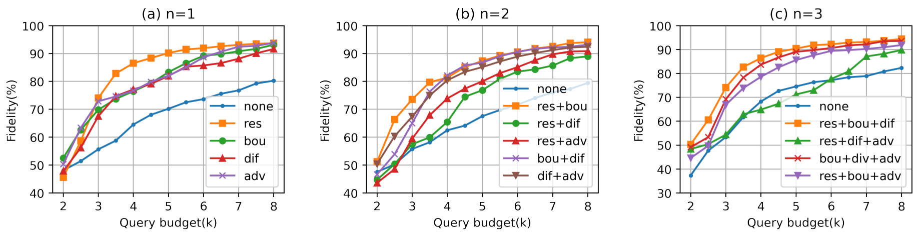

Fig. 5: The performance of fine-tuning indicators combination strategies.

According to Fig. 5, all feedback items show better performance than *none*, which illustrates the positive effect of fine-tuning. Among them, the *res* item achieved the best performance, which leads the other items by a large margin. In Fig. 5(b), the combination of *res* and *bou* achieves the best results, and they also achieved good performance in Fig. 5(a), respectively. However, the combination of *res* and *dif* does not show a corresponding advantage, although they perform well in Fig. 5(a). In the experiment of three indicators, $res + bou + dif$ achieves the best performance, in which the substitute model fidelity reaches 94.42%.

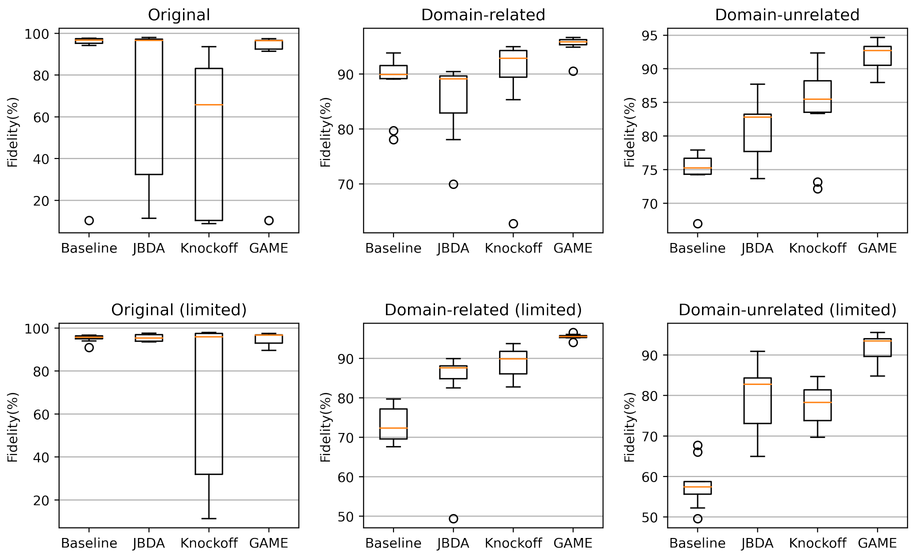

Fig. 6: Performance of different attacks with different proxy distribution and proxy size. The first row shows the performance with size-unlimited proxy sets. The second row shows the performance with size-limited proxy sets.

5.3 Impact of the Proxy Dataset Distribution & Size

We measure the fidelity and stability of four attacks under different proxy dataset assumptions about two aspects: distribution and size. First, we selected three datasets (MNIST, EMNIST-digits, and Fashion-MNIST) to simulate different proxy set distribution corresponding to different attacker capabilities. Secondly, we test the performance of all attacks on size-unlimited and size-limited proxy sets, the latter of which makes the attack more difficult.

In the size-unlimited proxy set experiments (the first row of Fig. 6), all settings keep the same as that in Section 4.1, except the pre-training epochs of AC-GAN is modified according to the dataset complexity. In the size-limited proxy set experiments (the second row of Fig. 6), all attackers can only get a proxy set of size 4k, which is half of the query budget. For both experiments, every sub-experiment was run 10 times. We use two metrics to evaluate these four attacks, fidelity (median of the box) and stability (length of the box).

From the fidelity view, all attacks are sensitive to proxy set distribution, except for GAME, as shown in the first row of Fig. 6. With the proxy dataset becomes less relevant to the task domain, the average fidelity achieved by the other three attacks tends to decrease. A similar phenomenon occurs when the size of proxy set is limited, as shown in the second row of Fig. 6. From the view of stability, both of the GAME and baseline attacks get a good performance regardless of how the distribution and size of proxy sets change, as their boxes are shorter in length. The other two attack methods are less stable, especially when the proxy data distribution is set to the original distribution. In summary, the GAME attack can achieve the best fidelity among these attacks and is more stable to different distribution and size of the proxy sets.

6 Conclusion

In this paper, we introduce GAME, a novel model extraction attack, which allows an attacker to use only a small amount of public proxy data for adaptive data augmentation, even if the attacker does not know the architecture of the victim model. To cope with the limited samples problem often faced by model extraction attackers, we design a data augmentation algorithm based on AC-GAN, which can efficiently generate query samples of specified categories. To address the issue of query efficiency, we propose an active learning-based category selection module and a feedback indicator-based adaptive generator updating strategy, respectively. According to extensive experiments, the GAME scheme exhibits excellent stealing ability with higher stability.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (Nos. 61960206014 and 62121001), the Fundamental Research Funds for the Central Universities (No. ZDRC2204), and China 111 Project (No. B16037).

References

1. Barbalau, A., Cosma, A., Ionescu, R.T., Popescu, M.: Black-box ripper: Copying black-box models using generative evolutionary algorithms. In: Advances in Neural Information Processing Systems (NIPS). vol. 33, pp. 20120–20129 (2020)
2. Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., Tian, Q.: Data-free learning of student networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3514–3522 (2019). https://doi.org/10.1109/iccv.2019.00361
3. Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM workshop on artificial intelligence and security (AISec@CCS). pp. 15–26 (2017). https://doi.org/10.1145/3128572.3140448
4. Fang, G., Song, J., Shen, C., Wang, X., Chen, D., Song, M.: Data-free adversarial distillation. CoRR **abs/1912.11006** (2019)
5. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (CCS). pp. 1322–1333 (2015). https://doi.org/10.1145/2810103.2813677
6. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In: International Conference on Learning Representations (ICLR) (2016)

7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/cvpr.2016.90
8. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR **abs/1503.02531** (2015)
9. Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., Igel, C.: Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In: The 2013 international joint conference on neural networks (IJCNN). pp. 1–8 (2013). https://doi.org/10.1109/ijcnn.2013.6706807
10. Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., Papernot, N.: High accuracy and high fidelity extraction of neural networks. In: 29th USENIX Security Symposium (USENIX). pp. 1345–1362 (2020)
11. Jia, J., Salem, A., Backes, M., Zhang, Y., Gong, N.Z.: Memguard: Defending against black-box membership inference attacks via adversarial examples. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security (CCS). pp. 259–274 (2019). https://doi.org/10.1145/3319535.3363201
12. Kariyappa, S., Prakash, A., Qureshi, M.K.: Maze: Data-free model stealing attack using zeroth-order gradient estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13814–13823 (2021). https://doi.org/10.1109/cvpr46437.2021.01360
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS). vol. 25 (2012)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
15. Micaelli, P., Storkey, A.J.: Zero-shot knowledge transfer via adversarial belief matching. In: Advances in Neural Information Processing Systems (NIPS). vol. 32 (2019)
16. Nayak, G.K., Mopuri, K.R., Shaj, V., Radhakrishnan, V.B., Chakraborty, A.: Zero-shot knowledge distillation in deep networks. In: International Conference on Machine Learning (ICML). pp. 4743–4751 (2019)
17. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. In: International conference on machine learning (ICML). pp. 2642–2651 (2017)
18. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: Stealing functionality of black-box models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4954–4963 (2019). https://doi.org/10.1109/cvpr.2019.00509
19. Pal, S., Gupta, Y., Shukla, A., Kanade, A., Shevade, S., Ganapathy, V.: Activethief: Model extraction using active learning and unannotated public data. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). vol. 34, pp. 865–872 (2020). https://doi.org/10.1609/aaai.v34i01.5432
20. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security (AsiaCCS). pp. 506–519 (2017). https://doi.org/10.1145/3052973.3053009
21. Rolnick, D., Kording, K.: Reverse-engineering deep relu networks. In: International Conference on Machine Learning (ICML). pp. 8178–8187 (2020)

22. Salem, A., Bhattacharya, A., Backes, M., Fritz, M., Zhang, Y.: {Updates-Leak}: Data set inference and reconstruction attacks in online learning. In: 29th USENIX Security Symposium (USENIX). pp. 1291–1308 (2020)
23. Salem, A., Zhang, Y., Humbert, M., Fritz, M., Backes, M.: Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In: Network and Distributed Systems Security Symposium (NDSS) (2019). https://doi.org/10.14722/ndss.2019.23119
24. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy (SP). pp. 3–18 (2017). https://doi.org/10.1109/sp.2017.41
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR) (2015)
26. Timofte, R., Zimmermann, K., Van Gool, L.: Multi-view traffic sign detection, recognition, and 3d localisation. Machine vision and applications **25**(3), 633–647 (2014). https://doi.org/10.1109/wacv.2009.5403121
27. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: 25th USENIX security symposium (USENIX). pp. 601–618 (2016)
28. Truong, J.B., Maini, P., Walls, R.J., Papernot, N.: Data-free model extraction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4771–4780 (2021). https://doi.org/10.1109/cvpr46437.2021.00474
29. Wang, D., Li, Y., Wang, L., Gong, B.: Neural networks are more productive teachers than human raters: Active mixup for data-efficient knowledge distillation from a blackbox model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1498–1507 (2020). https://doi.org/10.1109/cvpr42600.2020.00157
30. Wang, W., Yin, B., Yao, T., Zhang, L., Fu, Y., Ding, S., Li, J., Huang, F., Xue, X.: Delving into data: Effectively substitute training for black-box attack. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4761–4770 (2021). https://doi.org/10.1109/cvpr46437.2021.00473
31. Wang, Y., Du, S., Balakrishnan, S., Singh, A.: Stochastic zeroth-order optimization in high dimensions. In: International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 1356–1365 (2018)
32. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR **abs/1708.07747** (2017)
33. Xu, G., Liu, Z., Loy, C.C.: Computation-efficient knowledge distillation via uncertainty-aware mixup. CoRR **abs/2012.09413** (2020)
34. Yang, Z., Zhang, J., Chang, E.C., Liang, Z.: Neural network inversion in adversarial setting via background knowledge alignment. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 225–240 (2019). https://doi.org/10.1145/3319535.3354261
35. Yoo, J., Cho, M., Kim, T., Kang, U.: Knowledge extraction with no observable data. In: Advances in Neural Information Processing Systems (NIPS). vol. 32 (2019)
36. Yu, H., Yang, K., Zhang, T., Tsai, Y., Ho, T., Jin, Y.: Cloudleak: Large-scale deep learning models stealing through adversarial examples. In: Network and Distributed Systems Security Symposium (NDSS) (2020). https://doi.org/10.14722/ndss.2020.24178