INVERSENET: Augmenting Model Extraction Attacks with Training Data Inversion

Xueluan Gong[1] , **Yanjiao Chen**[2*] , **Wenbin Yang**[1] , **Guanghao Mei**[1] and **Qian Wang**[1†]
[1]Wuhan University, China
[2]Zhejiang University, China
{xueluangong, yangwenbin, guanghao, qianwang}@whu.edu.cn, chenyanjiao@zju.edu.cn

Abstract

Cloud service providers, including Google, Amazon, and Alibaba, have now launched machine-learning-as-a-service (MLaaS) platforms, allowing clients to access sophisticated cloud-based machine learning models via APIs. Unfortunately, however, the commercial value of these models makes them alluring targets for theft, and their strategic position as part of the IT infrastructure of many companies makes them an enticing springboard for conducting further adversarial attacks. In this paper, we put forth a novel and effective attack strategy, dubbed INVERSENET, that steals the functionality of black-box cloud-based models with only a small number of queries. The crux of the innovation is that, unlike existing model extraction attacks that rely on public datasets or adversarial samples, INVERSENET constructs inversed training samples to increase the similarity between the extracted substitute model and the victim model. Further, only a small number of data samples with high confidence scores (rather than an entire dataset) are used to reconstruct the inversed dataset, which substantially reduces the attack cost. Extensive experiments conducted on three simulated victim models and Alibaba Cloud's commercially-available API demonstrate that INVERSENET yields a model with significantly greater functional similarity to the victim model than the current state-of-the-art attacks at a substantially lower query budget.

1 Introduction

Machine-learning-as-a-service (MLaaS) provides users with access to machine learning and data analytics via the cloud. Today, companies like IBM, Amazon, Google are offering highly sophisticated models, such as deep neural networks, that typically require enormous amounts of data and millions of stored parameters to train [Ribeiro *et al.*, 2015]. Clients retrieve their desired predictions by simply posting queries through an API, which means they need never be bothered with the highly technical details of preparing training sets, configuring model architectures, or specifying hyperparameters. In fact, these elements are generally completely unknown to the end-user, making most cloud-based models more of a black box than they already are.

However, this convenience is a double-edged sword as recent studies have revealed that attackers can launch model extraction attacks against black-box ML models. In a model extraction attack, the adversary queries a cloud-based model via an API and uses the results to reverse engineer a substitute model with similar functionality. There are several reasons an adversary might launch a model extraction attack. One is to sell what are typically enormously valuable models. Another is to pry open the model enough to be able to launch further attacks, e.g., adversarial examples. Thus, since Tramer et al. proposed the first model extraction method in 2016, numerous researchers have contributed their own insights, variations, and extensions to the literature (e.g., [Tramèr *et al.*, 2016; Shi *et al.*, 2017; Papernot *et al.*, 2017; Wang and Gong, 2018; Juuti *et al.*, 2019; Orekondy *et al.*, 2019a; Pal *et al.*, 2020; Yu *et al.*, 2020]). Tramer et al.'s [Tramèr *et al.*, 2016] method works on simple models such as logistics regression, SVM, decision trees, and shallow neural networks. Shi et al. [Shi *et al.*, 2017] proposed StealClassifier, which extracts functionality from Naive Bayes and SVM models designed for text classification. In [Papernot *et al.*, 2017], Papernot et al. discovered that adversarial examples generated by a substitute model can incur a high misclassification rate when fed into a black-box model. Duddu et al. [Duddu *et al.*, 2018] proposed the use of side channels to infer the depth of deep neural networks. Wang and Gong [Wang and Gong, 2018] proposed the first hyperparameter extraction method to work on a diverse range of machine learning models, including ridge regression, logistic regression, support vector machine, and neural networks. With the aim of extracting a black-box model's functionality, Orekondy et al. [Orekondy *et al.*, 2019a] presented an iterative model extraction strategy called KnockoffNet that constructs the query dataset using reinforcement learning. The most recent model extraction attacks are ACTIVETHIEF [Pal *et al.*, 2020] and CloudLeak [Yu *et al.*, 2020]. ACTIVETHIEF uses an active learning mechanism to select samples from a public dataset for query. CloudLeak proposed that adversarial examples are helpful for extracting the decision boundary of the black-box

*Co-corresponding Author
†Co-corresponding Author

models.

Although studies on model extraction attacks have made considerable progress, there are still challenges to tackle. First, building a substitute model with comparable performance to a victim model that has a complicated structure, like a DNN, generally requires querying the victim model inordinately many times. Such a vast number of queries not only incurs high query costs [Correia-Silva *et al.*, 2018] but also increases the risk of the attack being detected [Papernot *et al.*, 2017]. It is an easy task for cloud service providers to monitor user-server traffic and generate a warning when a suspicious stream of queries is detected. Second, useful information about the victim model can be learned from both natural data samples (from public datasets) [Orekondy *et al.*, 2019a; Pal *et al.*, 2020; Correia-Silva *et al.*, 2018] and synthetic samples (provided by the adversary [Juuti *et al.*, 2019; Papernot *et al.*, 2017; Tramèr *et al.*, 2016]) even when those data are different from the original training set. Admittedly, though, the substitute model is likely to have relatively low similarity to the victim model in these cases.

In this paper, we propose INVERSENET –a novel model extraction attack against black-box machine learning models by making use of model inversion techniques to greatly improve the success of model extraction. The method begins by initiating a primitive substitute model using query samples from public datasets. To reduce the number of queries, the coreset algorithm [Sener and Savarese, 2018], which is commonly used in active learning, is used to select the most representative samples instead of using the whole dataset. By developing an innovative model inversion approach that only selects data samples with high confidence scores to construct the inversion model, a reliable set of inversed data samples are constructed with far fewer queries. The initial substitute model is then retrained with the inversed data samples to obtain the final substitute model. This is the key to INVERSENET's effectiveness. As the inverse data samples are expected to have a similar functionality to the original training data samples, the established substitute model should be more analogous to the victim model.

Extensive experiments on both simulated victim models and a real-world API confirm superior performance by INVERSENET over the current state-of-the-art approaches [Pal *et al.*, 2020; Orekondy *et al.*, 2019a; Papernot *et al.*, 2017] in terms of achieving higher similarity between the substitute and victim models. Further analysis shows that both the sample selection algorithm based on high confidence scores and the data inversion are integral to boosting the efficiency and performance of model extraction attacks. Importantly, we also show that INVERSENET can evade PRADA [Juuti *et al.*, 2019], one of the state-of-the-art defense strategies against model extraction attacks.

In summary, this paper makes the following contributions.

- To the best of our knowledge, this is the first attempt to leverage model inversion techniques for a model extraction attack. The effectiveness of the approach lies in the fact that the inversed data samples resemble that of the original training data, which helps the substitute model to learn the functionalities of the victim model.

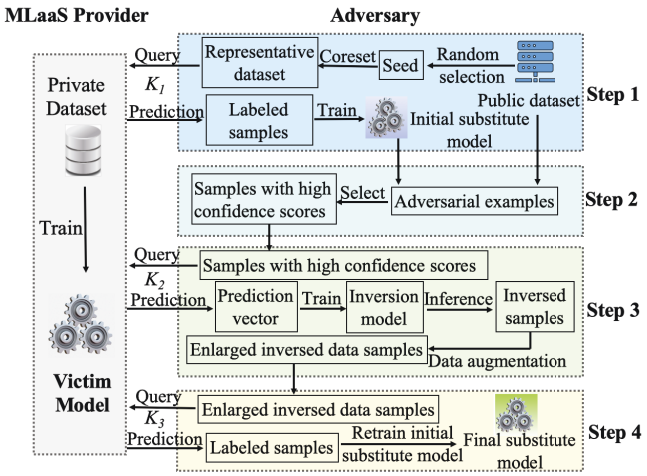


Figure 1: Overview of INVERSENET.

- To improve efficiency and significantly reduce the cost of queries when constructing an inversion model, we present a new sample selection method that chooses only the samples with high confidence scores.
- We validate the effectiveness of INVERSENET with extensive experiments on both simulated victim models and real-world APIs. INVERSENET performs well in strict black-box scenarios, where only top-1 results are returned. It is the most challenging scenario in model extraction attacks. We also show that INVERSENET is resistant to one of the most well-known defense strategies, PRADA.

2 Threat Model

Following existing works on model extraction attacks [Orekondy *et al.*, 2019a; Pal *et al.*, 2020], we consider a black-box scenario where the attacker's only access is through the API provided by the MLaaS. Given an input x, the API interface returns an M-dimensional prediction vector of confidence scores $\mathbf{y} = [y_1, y_2, ..., y_M] \in \mathbf{R}^M$, where M is the number of classes, and $\sum_{m=1}^{M} y_m = 1$. We consider the toughest case where the API only returns the class label (top-1 results) but not the confidence scores.

We consider the most realistic and challenging attack scenario, where the attacker has no prior knowledge of the victim model, including its parameters, hyperparameters, or architecture; has no access to the training or test data used to train or evaluate the victim model; has no information about the distribution of the training or test data. The attacker only has public dataset to query the victim model to glean useful information.

The adversary's goal is to steal the victim model from the cloud with a limited query budget, constructing a local substitute model with the same functionality as the victim model.

3 INVERSENET: Attack Strategy

As shown in Figure 1, INVERSENET proceeds through four major phases: initiating the substitute model, selecting sam-

ples for model inversion, inversing the training data samples, and retraining the substitute model.

- *Initiating substitute model*. To start with, we initiate a primitive substitute model by querying the victim model using carefully selected samples from the public dataset based on the coreset algorithm.
- *Selecting samples for inversion*. Given the primitive substitute model, we intend to improve the model with inversed training samples. We propose to inverse training samples based on samples with high confidence scores from the public dataset, which considerably reduces the query cost.
- *Inversing training samples*. For each class of the victim model, we are able to inverse a single representative average sample. To enrich the inversed training samples, we leverage data augmentation techniques.
- *Retraining substitute model*. We use the inversed training samples to query the victim model, based on which the substitute model is retrained to reach a high similarity with the victim model.

In model extraction attacks, it is ideal to use as few queries as possible to the victim model. This not only reduces query costs but also avoids being detected. On the one hand, for the adversary, probing the API should cost less than training a model with the same functionality himself. On the other hand, numerous queries might sound alarms with the cloud service provider, risking an account ban or other countermeasures. With INVERSENET, the adversary queries the victim model during three of the four phases, namely, substitute model initiation, training sample inversion, and model retraining, but with different carefully selected samples. In our experiments, we show that INVERSENET outperforms existing approaches with the same query budget and also conduct an ablation study on the effectiveness of the queries at each of the three phases.

3.1 Initiating Substitute Model

Instead of starting from scratch, model extraction attacks usually initialize a primitive substitute model. As far as we are concerned, almost all existing works use pre-trained models from model zoos, such as Caffe Model Zoo [Jia and Shelhamer, 2015]. In stark contrast, we establish our initial substitute model by using the query results of coreset data samples that are most representative of the entire data samples of the public dataset. This approach yields an ideal initial substitute model with a low query cost.

The key idea of the coreset algorithm is to select the most informative data samples, i.e., a "core set", that cover the entire dataset (e.g., ImageNet). In other words, the class label information of this coreset of samples is indicative of the class labels of the remaining samples.

To begin with, a set S_0 of k_0 seed samples is randomly selected from the public dataset. Each is marked as the center of a cluster. In the i-th iteration, the K samples that are most-distant from the current cluster centers S_{i-1} are selected. Each chosen sample is added to the set of data centers for selecting the next sample. Formally,

$$x_{k,i} = \arg_x \max_{x \in \widetilde{S}_{k,i}} \min_{x' \in S_i} ||x - x'||_1, k \in [1, K], \quad (1)$$

where $\widetilde{S}_{1,i} = S \setminus S_{i-1}$, and $\widetilde{S}_{k,i} = \widetilde{S}_{k-1,i} \setminus \{x_{k-1,i}\}, k \in [2, K]$. The cluster centers is updated as $S_i = S_{i-1} \cup \{x_{k,i}\}_{k=1}^{K}$.

After I iterations, we have a collective set of samples denoted as S_I. S_I is used to query the victim model to obtain the prediction labels. The labeled samples are then used to train an initial substitute model.

3.2 Selecting Samples for Inversion

Given the primitive substitute model, the next step is to improve the model with more query results from the victim model. There are two common practice for selecting query samples. The first is to select more samples from the public dataset using active learning or reinforcement learning techniques [Pal *et al.*, 2020; Orekondy *et al.*, 2019a]. However, since the public dataset may be quite different from the training dataset of the victim model, more samples from the public dataset may only be incrementally helpful. The second is to synthesize adversarial samples to learn the decision boundary of the victim model [Papernot *et al.*, 2017; Juuti *et al.*, 2019; Yu *et al.*, 2020]. Nonetheless, many defense strategies specifically target at detecting synthetic sample queries [Juuti *et al.*, 2019].

In INVERSENET, we seek for an unexplored territory of query sample selection. It is commonly agreed that, most ideally, the samples used to query the victim model should resemble the original training set as closely as possible. However, in the strict black-box setting, the original training set or a dataset in the same problem domain (PD) as the original training set are usually unavailable. Inspired by model inversion attacks that aim to recover the "average" images of the original training samples of a learning model [Fredrikson *et al.*, 2015; Yang *et al.*, 2019], we propose to leverage the recovered data samples to query the victim model, which should greatly enhance the performance of the substitute model.

The model inversion scheme we use is based on [Yang *et al.*, 2019], and in stark contrast to it, we have designed a more efficient method of constructing an inversion model that is specially tailored to model extraction. The method was inspired by a discovery that samples with high confidence scores are more useful for generating high-quality inversed data samples. Instead of querying the victim model to obtain confidence scores, the initial substitute model is utilized to judge if a sample is of high confidence score, in order to save query costs. This is reasonable since the initial substitute model has already learned some characteristics of the victim model. The advanced inversion method significantly reduces the number of queries required to construct the inversion model compared to [Yang *et al.*, 2019].

Samples with high confidence scores are those that are far away from the decision boundary of a model. For instance, consider a binary classifier f with $f(x) = 0$ as the decision boundary. The samples that satisfy $f(x) > 0$ are positive

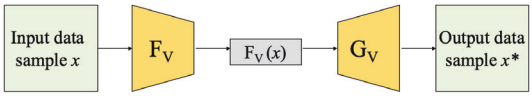


Figure 2: The model inversion process. The victim model F_V takes the data sample x as input and outputs a prediction result $F_V(x)$. The inversion model G_V takes the prediction vector as input and outputs the inversed data sample x^*.

samples and those that satisfy $f(x) < 0$ are negative samples. The samples with a large distance from the decision boundary are deemed to be the samples with high confidence scores since the classifier f is able to decide the class of these samples with high confidence. In contrast, the samples close to the decision boundary are considered to be samples with low confidence scores since they are easily misclassified.

To determine the distance between a sample and the decision boundary, we resort to the observation that it is relatively easy to push low-confidence samples across the decision boundary by adding a little noise, while it requires much more noise to change the label of high-confidence samples. Therefore, we generate an (untargeted) adversarial example [Ducoffe and Precioso, 2018] for each sample from the public dataset based on the initial substitute model. The required perturbation to generate the adversarial example is used to determine whether the sample is of a high confidence score or not.

Given a sample x, we seek the smallest noise Δx, such that $\tilde{x} = x + \Delta x$ will be misclassified to an incorrect label.

$$\min \ ||\Delta x||_2, \text{ s.t. } f(x + \Delta x) \neq f(x). \tag{2}$$

Noise is added to a sample iteratively until the sample is misclassified. To ensure that the adversarial sample crosses the decision boundary, Δx is multiplied with the constant $1 + \xi$ ($\xi \ll 1$) to yield the distance $D = (1 + \xi)\Delta x$. In our experiments, ξ was set to 0.02 [Moosavi-Dezfooli *et al.*, 2016]. The samples from the public dataset are sorted according to distance D in non-ascending order, and the first K samples are selected as having the highest confidence scores.

Notably, this set of selected samples may contain some of the same samples as in the coreset used for substitute model initiation. Hence, to further reduce costs, any overlapping samples are removed before querying the victim model F_V. The final set of selected samples with high confidence scores that need to query the victim model is denoted as S_J.

3.3 Inversing Training Samples

Given the selected high confidence score samples, the next step is to generate inversed training samples of the victim model. As shown in Figure 2, the inversion process can be viewed as an encoder-decoder architecture. The victim model F_V is the *encoder*; the inversion model G_V is the *decoder*; the prediction vector can be thought of as the latent space. The victim model F_V encodes an input sample x into a prediction vector $F_V(x)$ (i.e., the confidence score), and the decoder G_V decodes the prediction vector to obtain the data sample x^*.

The encoder-decoder architecture of the inversion process is similar but different from the *autoencoder* in [Baldi, 2012]. In [Baldi, 2012], the encoder is updated in an iterative manner. In contrast, in Figure 2, the victim model F_V is fixed but unknown in our black-box attack scenario. Moreover, the training dataset of F_V is unknown and cannot be used to train G_V, which is different from the case in [Baldi, 2012]. Both difficulties make it challenging to obtain the inversion model G_V.

The inversion process has three main steps. Firstly, the samples in S_J are fed into the victim model F_V to obtain their prediction vectors. Second, an inversion model G_V is trained using a truncation approach, i.e., truncate the F_V's predictions to the same dimension of the prediction vector on the training dataset, forcing G_V maximally recover the inversed samples. It can also help to prevent overfitting of G_V. Specifically, given a selected sample x with a high confidence score, the prediction vector $F_V(x)$ is truncated to m dimensions, which preserves the top m scores and sets the rest to zeros. We set m to 1 since in the toughest case, the commercial API only returns the top-1 results. The inversion model G_V is trained by minimizing the following objective function

$$C(G_V) = \mathrm{E}_{x \sim S_J}[L(G_V(trunc_1(F_V(x))), x)], \tag{3}$$

where L is the loss function, and $trunc_1(\cdot)$ is the truncate function that preserves the first score of the prediction vector. The last step is to input the truncated prediction vectors to the inversion model to obtain the inversed samples. As an example, assume that the victim model has five classes. To recover an inversed data sample of the second class, (0,1,0,0,0) would be fed to the inversion model G_V, which would return an inversed sample x^* representing the corresponding class.

The inversion model only produces a single inversed average sample for each class, which is not sufficient for retraining the substitute model. Thus, we augment inversed samples to enlarge the retraining set. We vary the value of truncated prediction vectors to the inversion model (e.g., use (0,0.9,0,0,0) to inverse the second class image) to obtain multiple inversed samples for the same class. We also use other augmentation techniques including cropping, scaling, rotating, linear augmenting, shearing, Gaussian blurring, and adding white Gaussian noise to copies of inversed samples.

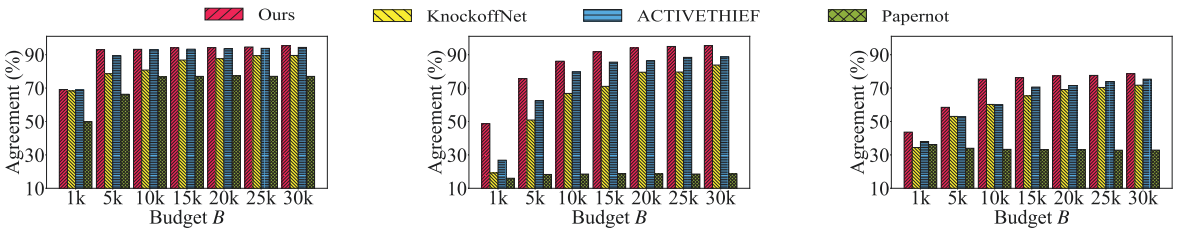
3.4 Retraining Substitute Model

With the initial substitute model and the augmented inversed dataset in hand, the victim model is then queried with the samples from the inversed dataset. The labeled responses returned from these queries are then used to retrain the initial substitute model, resulting in the final well-performed substitute model.

4 Implementation and Evaluation

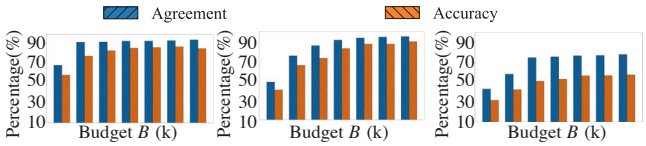
The simulated victim models are trained on three datasets: MNIST, GTSRB, and CIFAR10. The victim model structure is Classifier, and the substitute model structure is CNN32. For real-world commercial API, we use the pornography detection service provided by Alibaba Cloud[1], which outputs one of three labels, i.e., "porn", "sexy", or "normal". The

[1] ai.aliyun.com/lvwang/imgadult



(a) MNIST (b) GTSRB (c) CIFAR10

Figure 3: Comparison with KnockoffNet, ACTIVETHIEF, and Papernot.



(a) MNIST (b) GTSRB (c) CIFAR10

Figure 4: Impact of query budgets.

API's documentation does not provide any information about the training dataset or the model architecture. To extract the real-world API, we use ResNet18 [He *et al.*, 2016] as the substitute model architecture.

We use *agreement* to evaluate the similarity between the substitute model and the victim model. And *accuracy* denotes the prediction accuracy of the substitute model on the test set. We use PSNR and MSE to evaluate the quality of inversed samples. Details on the evaluation metrics are in the Appendix. All experiments were conducted on an Ubuntu 16.04 system with a 8-core Intel CPU and NVIDIA GPU.

4.1 Evaluation Results

We compare INVERSENET with state-of-the-art extraction attacks ACTIVETHIEF [Pal *et al.*, 2020], KnockoffNet [Orekondy *et al.*, 2019a], and Papernot [Papernot *et al.*, 2017]. For a fair comparison, the query budgets are the same for all algorithms. Figure 3 shows that INVERSENET has a higher agreement than the baselines for all datasets at all query budgets. Notably, the performance improvement is more pronounced when the query budget is limited. Both ACTIVETHIEF and KnockoffNet only use raw samples from public datasets to establish their substitute models. The performance advantage of INVERSENET is therefore attributed to using an inverse dataset to retrain the substitute model. Papernot also uses synthetic samples, but is ineffective when extracting the GTSRB and CIFAR10 models.

Impact of query budget. INVERSENET queries the victim model in three different phases with K_1, K_2 and K_3 samples respectively. The total query budget is $K = K_1 + K_2 + K_3$. In the experiments, the ratio between K_1, K_2, and K_3 is fixed at 0.45:0.45:0.1. Later, we will evaluate the effectiveness of each query phase with an ablation study. As shown in Figure 4, it is obvious that the performance of the substitute model gets better as the query budget increases. As the query budget grows from 1k to 20k, the agreement increases significantly, especially with the GTSRB and CIFAR10 models. However, with further budget increases, the rate of improvement slows down. At a query budget of 10k, the MNIST substitute model reaches an agreement of 93.2%, whereas the GTSRB model reaches an agreement of 86.1%, and the more complex CIFAR10 model yields an agreement of 75.4%. We also show the results of INVERSENET with full confidence score in the Appendix. For the real-world API, INVERSENET reaches an agreement of 76.87% with a budget of only 1k and 80.53% with 5k queries, which is very competitive for extracting a real-world API when the training set is unknown to the attacker. The attacker may exploit the stolen model for its own profit, thus the potential economic loss of such model extraction attacks may be fundamental.

Impact of substitute model structure. In this part, we investigate the impact of substitute model structure on the performance of INVERSENET. For comparison, we first use Classifer as the substitute model structure, which is the same as the victim model structure; then we use three different model structures for the substitute model, namely CNN32, CNN42, and ResNet18. The details of these model structures can be found in the Appendix. Table 1 shows that when substitute model structure is the same as or in the same family as the black-box victim model, the *agreement* is high. In the case of unknown victim model structure, a complex structure is effective in extracting the functionality of the victim model.

The effectiveness of high confidence score samples. We compare the inversed samples using randomly selected samples and using samples with high confidence scores (dubbed HCSS). For fair comparison, we use the same number of queries (10k) for HCSS and the random strategy. As shown in Figure 5, we can see that the PSNR of HCSS is higher than the random strategy for all datasets. Moreover, the experimental results of model inversion are comparable to the state-of-the-art data reconstruction attacks [Salem *et al.*, 2020]. The MSE of the inversed samples using HCSS is also lower than that using the random strategy. The results of MSE and more inversed samples are shown in the Appendix.

Ablation study on the effectiveness of queries. To verify the effectiveness of the query in each phase as we mentioned above, we conduct an ablation study by changing one of the parameters $K_i, i \in [1, 3]$ while fixing the other two parameters $K_j = 4,000, j \neq i$. As shown in Table 2, we can see that if either K_1 or K_2 increases, the agreement will increase.This indicates that selecting a larger coreset for substitute model initiation and selecting more high confidence score samples

	MNIST				GTSRB				CIFAR10			
	Classifier	CNN32	CNN42	ResNet18	**Classifier**	CNN32	CNN42	ResNet18	**Classifier**	CNN32	CNN42	ResNet18
10k	94.34%	93.2%	90.37%	90.52%	87.42%	86.1%	83.7%	83.93%	75.7%	75.4%	63.6%	64.3%
20k	95.5%	94.2%	93.37%	94.47%	94.71%	94%	88.6%	89.41%	77.48%	77.3%	67.42%	69.54%
30k	95.96%	95.4%	94.67%	94.78%	95.88%	95.4%	93.6%	92.52%	81.94%	78.6%	72.01%	71.04%

Table 1: Impact of substitute model structure

Figure 5: PSNR of inversed samples for MNIST, GTSRB, and CIFAR10. 1st row: the original training samples. 2nd row: inversed samples using random selection method. 3rd row: inversed samples using samples with high confidence scores.

Variable	Dataset	1,000	2,000	3,000	4,000
K_1	MNIST	80.2%	82.32%	85.48%	89.28%
	GTSRB	72.4%	72.92%	73.05%	75.18%
	CIFAR10	69.12%	70.96%	74.17%	75.55%
K_2	MNIST	88.23%	88.78%	89.26%	89.28%
	GTSRB	67.46%	71.04%	72.22%	75.18%
	CIFAR10	70.28%	72.05%	74.38%	75.55%
K_3	MNIST	89.37%	89.67%	88.59%	89.28%
	GTSRB	75.59%	75.65%	75.02%	75.18%
	CIFAR10	74.59%	75.39%	75.39%	75.55%

Table 2: Ablation study

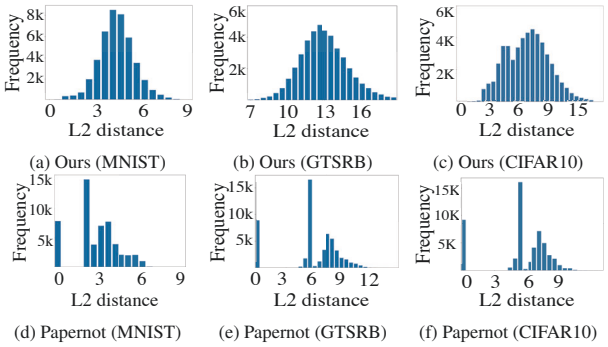


Figure 6: Distribution of distances between queries.

for training sample inversion both contribute to a better performance of INVERSENET. If K_3 increases, the agreement slightly fluctuates up and down, but basically there are little changes. However, if we aggressively set K_3 as 0, the agreement will decrease by approximately 2%. This shows that augmenting the inverse dataset is helpful, but the contribution is limited. The ablation study provides valuable guidance on the split of the query budget: more budget should be assigned to K_1 and K_2, and a small K_3 is enough.

Evading state-of-the-art defense. PRADA [Juuti *et al.*, 2019] is a defense strategy against model extraction attacks. Based on the assumption that the distribution of queries used by the attacker usually deviates from a normal (Gaussian) distribution, PRADA keeps track of the minimum distance between a new input sample and all previous samples in the same class to model the distribution of queries. The Shapiro-Wilk test statistic [Juuti *et al.*, 2019] is used to quantify whether the distribution of queries fits a normal distribution. PRADA is especially effective in detecting model extraction attacks using adversarial samples as queries. Both KnockoffNet and ACTIVETHIEF use solely natural samples from public dataset, thus will not be detected by PRADA. But the performance of KnockoffNet and ACTIVETHIEF is unsatisfactory. Papernot [Papernot *et al.*, 2017] leverages adversarial samples for query, thus it is easily detected by PRADA as the query distribution is far from a normal distribution, as shown in Figure 6. Our INVERSENET uses both natural data from public datasets but also an augmented inverse dataset. Figure 6 demonstrates that the query distribution of INVERSENET is still similar to a normal distribution. This indicates that INVERSENET can reach an ideal performance while evading defense strategies such as PRADA. More potential defenses are discussed in the Appendix.

5 Conclusion

This article presents the design, implementation, and evaluation of INVERSENET –a novel model extraction attack against black-box models using inversed datasets reconstructed by selecting only samples with high confidence scores, which greatly reduces the number of queries required to construct the inversion model. Extensive experiments confirm INVERSENET as a more successful method of model extraction attack than the current state-of-the-art approaches.

Acknowledgements

Qian Wang's work was partially supported by the National Key R&D Program of China under grant 2020AAA0107700 and the National Nature Science Foundation of China under grants U20B2049 and 61822207. Yanjiao Chen's work was partially supported by the National Nature Science Foundation of China under grant 61972296.

A Dataset of Victim Model

MNIST. We randomly choose 60,000 samples as the training set, and 10,000 samples as the test set.

GTSRB. The dataset is pre-divided into 39,209 training and 12,630 testing samples. Using annotated information, we cropped each image to its core area with size 32×32.

CIFAR10. We randomly select 50,000 and 10,000 samples as training and test set respectively.

B Query Dataset

For the MNIST victim model, we use EMNIST Letters [Cohen *et al.*, 2017] as the public dataset. For the GTSRB and CIFAR10 models, we draw samples from the ImageNet dataset. For the commercial API, we assembled a dataset that contains 135,325 public samples downloaded from the internet, divided into five categories, three of which are related to pornography. All samples are resized to 224×224. Note that there is no overlap between the query dataset and the private dataset (i.e., training dataset and test dataset) of both the simulated and real-world black-box victim models.

C Model Structure

Victim model. The structure of simulated victim models is Classifier, which contains four CNN blocks. Each block consists of a convolutional layer followed by a batch normalization layer, a max-pooling layer, and a ReLU activation layer. Two fully connected layers are added after the CNN blocks, and a softmax function is added to the last layer.

Substitute model. We use Classifier, CNN32, CNN42, and ResNet as the structure of the substitute model. CNN32 contains three CNN blocks, and each block comprises two repeated units of two convolutional layers and one pooling layer. Each convolutional layer is followed by a ReLU layer, a batchnorm layer, and a pooling layer followed by dropout. The output of the final pooling layer passes through a fully-connected layer and a softmax layer to obtain the final output. CNN42 contains 4 convolution blocks, each consisting of 2 repeated units of 2 convolutional layers and one pooling layer. Each convolutional layer is followed by ReLU and batchnorm layers, and a pooling layer followed by dropout. The output of the final pooling layer passes through a fully-connected layer and a softmax layer to obtain the final output. In ResNet18, the convolutional layers mostly have 3×3 filters and follow two simple design rules. First, for the same output feature map size, the layers have the same number of filters. Second, if the feature map size is halved, the number of filters is doubled to preserve the time complexity per layer. The total number of weighted layers is 18. A shortcut connection is added to each pair of 3×3 filters. We perform downsampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax.

Inversion model. Following [Yang *et al.*, 2019], the inversion model G_V comprises of five transposed CNN blocks. The first four are made up of a transposed convolutional layer followed by a batch normalization layer and a Tanh activation function. The last block contains a transposed convolutional layer followed by a Sigmoid activation function.

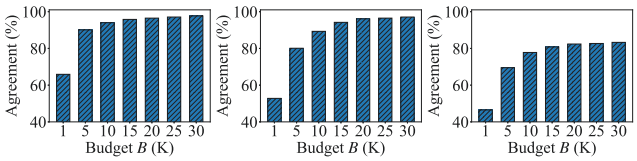


Figure 7: Performance with Full Confidence Score.

	MNIST		GTSRB		CIFAR-10	
	HCSS $\times10^{-5}$	Random $\times10^{-5}$	HCSS ($\times10^{-5}$)	Random $\times10^{-5}$	HCSS $\times10^{-5}$	Random $\times10^{-5}$
5k	12.5297	16.7109	16.8906	17.8962	12.5000	12.5725
10k	11.5170	16.2321	16.5144	17.3834	12.4622	12.5594
15k	11.4790	15.2694	15.5311	17.0791	12.4413	12.5209
20k	11.4557	14.6737	15.3711	16.9327	12.4269	12.4806
25k	11.3969	14.6215	15.3190	16.6913	12.4208	12.4629
30k	11.3780	14.5808	15.2458	16.5486	12.4062	12.4330

Table 3: MSE of inversed samples with high confidence score samples (HCSS) versus randomly selected samples.

D Evaluation Metrics

We use *agreement* to measure the similarity between the substitute model and the victim model in terms of their outputs,

$$Agreement(F_V, F_S, T) = \frac{1}{|T|} \sum_{x \in T} \mathbf{I}[F_V(x) = F_S(x)],$$

where F_V and F_S are the victim model and the substitute model respectively, T is the test dataset, and $\mathbf{I}(.)$ is the indicator function. *Agreement* calculates the fraction of test samples for which the substitute model F_S and the victim model F_V outputs the same predicted class.

We measure the quality of inversed training samples using mean squared error (MSE), which computes the difference between the inversed samples and the original training sample [Yang *et al.*, 2019]. Besides, we use Peak Signal-to-Noise Ratio (PSNR) as another metric to evaluate the pixel level recovery quality of the inversed samples. These metrics serve as indicators that samples with high confidence scores are indeed helpful to data inversion.

E Inversed Dataset

Figure 8∼11 show more representative inversed samples of MNIST, GTSRB, and CIFAR-10. The inversed samples using high confidence score samples are comparable to the state-of-the-art data reconstruction attacks [Salem *et al.*, 2020].

We present the MSE of inversed samples using our high confidence score samples (HCSS) and randomly selected samples in Table 3. For fair comparison, we use the same number of queries (10K) for HCSS and the random strategy. It is shown that the MSE of HCSS is lower than the random selection strategy, indicating that the inversed samples

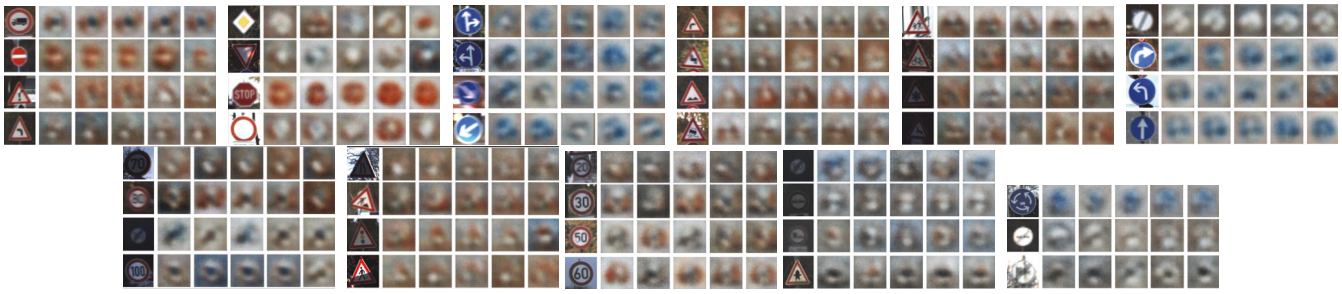


Figure 8: Performance of sample inversion on GTSRB. The first column is the original training samples. The other columns are the inversed data samples (including enlarged samples).

Figure 9: PSNR of more inversed images on GTSRB. The first row is the original training samples. The second row is the inversed images with model trained by randomly selected images. The third row is the inversed images with model trained by high confidence score samples.

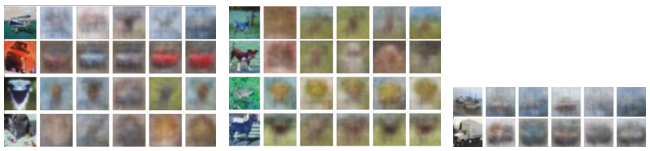


Figure 10: Performance of sample inversion on CIFAR-10. The first column is the original training samples. The other columns are the inversed data samples (including enlarged samples).

Figure 11: Performance of sample inversion on MNIST. The first column is the original training samples. The other columns are the inversed data samples (including enlarged samples).

of HCSS are more similar to the original samples than those of the random selection strategy.

Apart from MSE, we also quantitatively analyze the effectiveness of HCSS using PNSR. The results of the whole inversed GTSRB samples are shown in Figure 9. The results show that HCSS has higher PNSR than the random selection strategy for all classes.

F Performance with Full Confidence Score

In this appendix, we present the performance of INVERSENET when full confidence scores are available. Note that confidence scores only contribute to initiating the substitute model in INVERSENET. When inversing training samples, since we cannot access the training samples' confidence scores, we only use top-1 results to train the inversion model.

As shown in Figure 7, we can see that INVERSENET with full confidence score can achieve a higher agreement than INVERSENET with top-1 in all datasets. For example, INVERSENET with full confidence score yields an agreement of 94% with the relatively simple MNIST model at a query budget of 10k, while INVERSENET with top-1 reaches 93.2%. With the more complex GTSRB and CIFAR10 models at a query budget of 10k, INVERSENET with full confidence score returns an agreement of 89.3% (GTSRB) and 77.8% (CIFAR10), while INVERSENET with top-1 trailing at 86.1% (GTSRB) and 75.4% (CIFAR10).

G Potential Defense Strategy

To the best of our knowledge, there are two types of state-of-the-art defense strategies against model extraction attacks: 1) perturbing confidence scores (but ensuring correct prediction label), 2) detecting query sequence abnormalities. Since INVERSENET performs well with top-1 prediction results, type 1) defense strategies, e.g., [Lee *et al.*, 2019; Orekondy *et al.*, 2019b], are not effective against INVERSENET. The most recent work of 2) is PRADA, which is shown to be ineffective against INVERSENET.

A possible defense against INVERSENET is for the model owner to measure the knowledge gained by the adversary through query [Kesarwani *et al.*, 2018]. Once the gained knowledge hits a pre-designed threshold, the model owner may reject further queries to evade model extraction attacks. The information gain may be measured by training and updating a local proxy model for each client to estimate the information gain concerning a given validation set. However, such an approach is computationally expensive.

References

[Baldi, 2012] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *ICML Workshop on Unsupervised and Transfer Learning*, pages 37–49. JMLR.org, 2012.

[Cohen *et al.*, 2017] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: An extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017.

[Correia-Silva *et al.*, 2018] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. Copycat CNN: Stealing knowledge by persuading confession with random non-labeled data. In *International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2018.

[Ducoffe and Precioso, 2018] Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: A margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.

[Duddu *et al.*, 2018] Vasisht Duddu, Debasis Samanta, D Vijay Rao, and Valentina E Balas. Stealing neural networks via timing side channels. *arXiv preprint arXiv:1812.11720*, 2018.

[Fredrikson *et al.*, 2015] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[Jia and Shelhamer, 2015] Yangqing Jia and Evan Shelhamer. Caffe model zoo. *UC Berkeley*, 2015.

[Juuti *et al.*, 2019] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. PRADA: Protecting against DNN model stealing attacks. In *IEEE European Symposium on Security and Privacy*, pages 512–527, 2019.

[Kesarwani *et al.*, 2018] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. Model extraction warning in mlaas paradigm. In *34th Annual Computer Security Applications Conference*. ACM, 2018.

[Lee *et al.*, 2019] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. Defending against neural network model stealing attacks using deceptive perturbations. In *IEEE Security and Privacy Workshops*, 2019.

[Moosavi-Dezfooli *et al.*, 2016] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[Orekondy *et al.*, 2019a] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff Nets: Stealing functionality of black-box models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[Orekondy *et al.*, 2019b] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Utility-constrained defenses against model stealing attacks. *arXiv preprint arXiv:1906.10908*, 2019.

[Pal *et al.*, 2020] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. ACTIVETHIEF: Model extraction using active learning and unannotated public data. In *AAAI Conference on Artificial Intelligence*, pages 865–872. AAAI Press, 2020.

[Papernot *et al.*, 2017] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM Asia Conference on Computer and Communications Security*, pages 506–519, 2017.

[Ribeiro *et al.*, 2015] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. MlaaS: Machine learning as a service. In *IEEE International Conference on Machine Learning and Applications*, pages 896–902, 2015.

[Salem *et al.*, 2020] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In *USENIX Security Symposium*, pages 1291–1308. USENIX Association, 2020.

[Sener and Savarese, 2018] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*. OpenReview.net, 2018.

[Shi *et al.*, 2017] Yi Shi, Yalin Sagduyu, and Alexander Grushin. How to steal a machine learning classifier with deep learning. In *IEEE International Symposium on Technologies for Homeland Security*, pages 1–5, 2017.

[Tramèr *et al.*, 2016] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*. USENIX Association, 2016.

[Wang and Gong, 2018] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *IEEE Symposium on Security and Privacy*, 2018.

[Yang *et al.*, 2019] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 225–240, 2019.

[Yu *et al.*, 2020] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. CloudLeak: Large-scale deep learning models stealing through adversarial examples. In *Network and Distributed Systems Security Symposium*. The Internet Society, 2020.