MAZE: Data-Free Model Stealing Attack Using Zeroth-Order Gradient Estimation

Sanjay Kariyappa
Georgia Institute of Technology
Atlanta GA, USA
sanjaykariyappa@gatech.edu

Atul Prakash
University of Michigan
Ann Arbor MI, USA
aprakash@umich.edu

Moinuddin K Qureshi
Georgia Institute of Technology
Atlanta GA, USA
moin@gatech.edu

Abstract

High quality Machine Learning (ML) models are often considered valuable intellectual property by companies. Model Stealing (MS) attacks allow an adversary with black-box access to a ML model to replicate its functionality by training a clone model using the predictions of the target model for different inputs. However, best available existing MS attacks fail to produce a high-accuracy clone without access to the target dataset or a representative dataset necessary to query the target model. In this paper, we show that preventing access to the target dataset is not an adequate defense to protect a model. We propose MAZE *– a data-free model stealing attack using zeroth-order gradient estimation that produces high-accuracy clones. In contrast to prior works, MAZE uses only synthetic data created using a generative model to perform MS.*

Our evaluation with four image classification models shows that MAZE provides a normalized clone accuracy in the range of $0.90\times$ *to* $0.99\times$*, and outperforms even the recent attacks that rely on partial data (JBDA, clone accuracy* $0.13\times$ *to* $0.69\times$*) and on surrogate data (KnockoffNets, clone accuracy* $0.52\times$ *to* $0.97\times$*). We also study an extension of MAZE in the partial-data setting, and develop* MAZE-PD*, which generates synthetic data closer to the target distribution. MAZE-PD further improves the clone accuracy (*$0.97\times$ *to* $1.0\times$*) and reduces the query budget required for the attack by* $2\times$*-*$24\times$*.*

1. Introduction

The ability of Deep Neural Networks (DNNs) to achieve state of the art performances in a wide variety of challenging computer-vision tasks has spurred the wide-spread adoption of these models by companies to enable various products and services such as self-driving cars, license plate reading, disease diagnosis from medical images, activity classification from images and video, and smart cameras. As the performance of ML models scales with the training data [11], companies invest significantly in collecting vast amounts of data to train high-performance ML models. Protecting the confidentiality of these models is vital for companies to maintain a competitive advantage and to prevent the stolen model from being misused by an adversary to compromise security and privacy. For example, an adversary can use the stolen model to craft adversarial examples [9, 30, 32], compromise user membership privacy through membership inference attacks [29, 34, 21], and leak sensitive user data used to train the model through model inversion attacks [6, 35, 38]. Thus, ML models are considered valuable intellectual properties of the owner and are closely guarded against theft and data leaks.

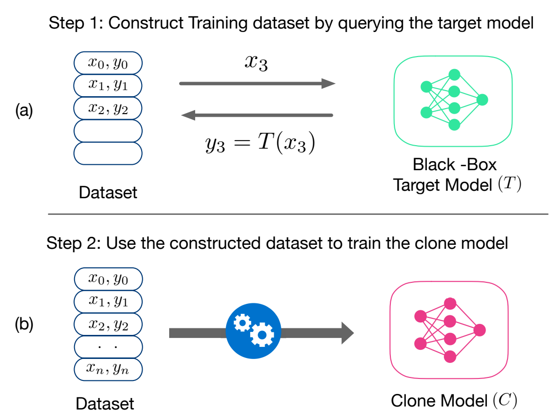

Figure 1. Model stealing attacks: The target model is queried using a set of inputs $\{x_i\}_{i=1}^n$ to obtain a labeled training dataset $\{x_i, y_i\}_{i=1}^n$, which is used to train the clone model.

Model functionality stealing attacks compromise the confidentiality of ML models by allowing an adversary to train a *clone model* that closely mimics the predictions of the target model, effectively copying its functionality. These attacks only require black-box access to the target model where the adversary can access the predictions of the model for any given input. Fig. 1 illustrates the steps involved in carrying out a MS attack. The adversary first queries the target model T with various inputs $\{x_i\}_{i=1}^{n}$ and uses the predictions of the target model $y_i = T(x_i)$ to construct a labeled dataset $\mathcal{D} = \{x_i, y_i\}$. This dataset is then used to train a clone model C to match the predictions of T.

In the current state of the art methods (e.g., [26, 24]), the availability of in-distribution or similar surrogate data to query the target model plays a key role in the ability of the attacker to train high accuracy clone models. However, in most real-world scenarios, the training data is not readily available to the attacker as companies typically train their models using proprietary datasets. To carry out MS in such a data-limited setting, existing attacks either assume partial availability of the target dataset or the availability of a *surrogate* dataset that is semantically similar to the target dataset (e.g., using CIFAR-100 to attack a CIFAR-10 model). For example, *Jacobian-Based Dataset Augmentation (JBDA)* [26] is an attack that uses a subset of the training data to create additional synthetic data, which is used to query the target model. *KnockoffNets* [24] is another MS attack that uses a *surrogate* dataset to query the target model. These attacks become ineffective without access to the target dataset or a representative surrogate dataset. [1]

This paper is the first to show that a highly accurate MS attack is feasible without relying on any access to the target dataset or even a surrogate dataset – our method only relies on synthetically-generated out-of-distribution data – but results in high-accuracy clones on in-distribution data. We make the following key contributions in our paper:

Contribution 1: We propose *MAZE*– the first data-free model stealing attack capable of training high-accuracy clone models across multiple image classification datasets and complex DNN target models. In contrast to existing attacks that require some form of data to query the target, MAZE uses synthetic data created using a generative model to carry out MS attack. Our evaluations across DNNs trained on various image classification tasks show that MAZE provides a normalized clone accuracy of $0.90\times$ to $0.99\times$ (normalized clone accuracy is the accuracy of the clone model expressed as a fraction of the target-model accuracy). Despite not using any data, MAZE outperforms recent attacks that rely on partial data (JBDA, clone accuracy of $0.13\times$ to $0.69\times$) or surrogate data (KnockoffNets, clone accuracy of $0.52\times$ to $0.97\times$).

Contribution 2: Our key insight is to draw inspiration from data-free knowledge distillation (KD) and zeroth-order gradient estimation to train the generative model used to produce synthetic data in MAZE. Similar to data-free KD, the generator is trained on a disagreement objective, which encourages it to produce synthetic inputs that maximize the disagreement between the predictions of the target (teacher) and the clone (student) models. By training the clone model on such synthetic examples we can improve the alignment of the clone model's decision boundary with that of the target, resulting in a high-accuracy clone model.

In data-free KD, training the generator on the disagreement objective is possible since white-box access to the teacher model is available. But, unlike in data-free KD, MAZE operates in a black-box setting. We therefore leverage *zeroth-order gradient estimation (ZO)* [22, 7] to approximate the gradient of the black-box target model and use this to train the generator. Unfortunately, we found a direct application of ZO gradient estimation to be impractical on real-world image classification models since the dimensionality of the generator's parameters can be in the order of millions. We propose a way to overcome the dimensionality problem by estimating gradients with respect to the significantly lower-dimensional synthetic input and show that our method can be successfully used to train a generator in a query-efficient manner.

Contribution 3: In some cases, partial datasets may be available. Recognizing that, we propose an extension of MAZE, called *MAZE-PD*, for scenarios where a small partial dataset (e.g., 100 examples) is available to the attacker. MAZE-PD leverages the available data to produce queries that are closer to the training distribution than in MAZE by using generative adversarial training. Our evaluations show that MAZE-PD provides near-perfect clone accuracy ($0.97\times$ to $1.0\times$), while reducing the number of queries by $2\times$-$24\times$ compared to MAZE.

In summary, our key finding is that an attacker only requires black-box access to the target model and no in-distribution data to create high-accuracy clone models in the image classification domain. If even a very limited amount of in-distribution data is available, near-perfect clone accuracy is feasible. This raises questions on how machine learning models can be better protected from competitors and bad actors in this domain.

2. Related Work

Several types of MS attacks have been proposed in recent literature. Depending on the goal of the attack, MS attacks can be categorized into: (1) parameter stealing (2) hyper-parameter stealing (3) functionality stealing attacks. Parameter stealing attacks [31, 18] focus on stealing the exact model parameters, while hyper-parameter stealing attacks [33, 23] aim to determine the hyper-parameters used in the model architecture or the training algorithm of the target model. Our work, MAZE and MAZE-PD, are designed to carry out a *functionality stealing* attack, where the goal is to replicate the functionality of a blackbox target model by training the clone model on the predictions of the target. As the attacker typically does not have access to the dataset used to train the target model, attacks need alternate

[1] We refer the interested readers to Section 6.1 of the KnockoffNets paper [24] for a discussion on the importance of using semantically similar datasets to carry out the attack.

forms of data to query the target model and perform model stealing. Depending on the availability of data, functionality stealing attacks can be classified as using (1) partial-data, (2) surrogate-data, or (3) data-free, i.e., synthetic data. We discuss prior works in each of these three settings and also briefly discuss relationship between model stealing and knowledge distillation.

2.1. Model Stealing with Partial Data

In the partial-data setting, the attacker has access to a subset of the data used to train the target model. While this in itself may be insufficient to carry out model stealing, it allows the attacker to craft synthetic examples using the available data. *Jacobian Based Dataset Augmentation (JBDA)* [26] is an example of one such attack that assumes that the adversary has access to a small set of *seed* examples from the target data distribution. The attack works by first training a clone model C using the seed examples and then progressively adding synthetic examples to the training dataset. JBDA uses a perturbation based heuristic to generate new synthetic inputs from existing labeled inputs. E.g., from an input-label pair (x, y), a synthetic input x' is generated by using the jacobian of the clone model's loss function $\nabla_x \mathcal{L}\left(C\left(x;\theta_c\right), y\right)$ as shown in Eqn. 1.

$$x' = x + \lambda sign\left(\nabla_x \mathcal{L}\left(C\left(x;\theta_c\right), y\right)\right) \tag{1}$$

The dataset of synthetic examples $\{x'_i\}$ generated this way are labeled by using the predictions of the target model $y'_i = T(x'_i)$ and the labeled examples $\{x'_i, y'_i\}$ are added to the pool of labeled examples that can be used to train the clone model C. In addition to requiring a set of seed examples from the target distribution, a key limitation of JBDA is that, while it works well for simpler datasets like MNIST, it tends to produce clone models with lower classification accuracy for more complex datasets. For example, our evaluations in Section 5 show that JBDA provides a normalized clone accuracy of only $0.13\times$ (GTSRB dataset) and $0.18\times$ (SVHN dataset).

2.2. Model Stealing with Surrogate Data

In the surrogate data setting, the attacker has access to alternate datasets that can be used to query the target model. KnockoffNets [24] is an example of a MS attack that is designed to operate in such a setting. With a suitable surrogate dataset, KnockoffNets can produce clone models with up to $0.97\times$ the accuracy of the target model. However, the efficacy of such attacks is dictated by the availability of a suitable surrogate dataset. For instance, if we use the MNIST dataset to perform MS on a FashionMNIST model, it only produces a clone model with $0.41\times$ the accuracy of the target model (See Table 1 for full results). This is because the surrogate dataset is not representative of the target dataset, which reduces the effectiveness of the attack.

2.3. Data-Free Model Stealing

In the data-free setting, the adversary does not have access to any data. This represents the hardest setting to carry out MS as the attacker has no knowledge of the data distribution used to train the target model. A recent work by Roberts et al. [28] studies the use of inputs derived from various noise distributions to carry out MS attack in the data-free setting. While this attack works well for simple datasets like MNIST, our evaluations show that such attacks do not scale to more complex datasets such as CIFAR-10 (we obtained relative clone accuracy of only $0.11\times$), limiting their applicability (See Table 1 for full results).

2.4. Knowledge distillation

Model stealing is related to knowledge-distillation (KD) [12], but in KD, unlike in model stealing, the target model is available to the attacker and is simply being summarized into a simpler architecture. Appendix E further discusses works in data-free KD and explain why the these works are not directly applicable for MS attacks.

3. Preliminaries

The goal of this paper is to develop a model functionality stealing attack in the data-free setting, which can be used to train a high-accuracy clone model only using black-box access to the target model. We formally state the objective and constraints of our proposed model stealing attack.

Attack Objective: Consider a target model T that performs a classification task with high accuracy. Our goal is to train a clone model C that replicates the functionality of the target model by maximizing the accuracy on a test set $\mathcal{D}_{test}$ as shown in Eqn. 2.

$$\max_{\theta_C} \mathop{\mathbb{E}}_{x,y\sim\mathcal{D}_{test}} [Acc(C(x;\theta_C), y)] \tag{2}$$

Attack Constraints: We assume that the adversary does not know any details about the Target model's architecture or the model parameters θ_T. The adversary is only allowed black-box access to the target model. We assume the *soft-label* setting where the adversary can query the target model with any input x and observe its output probabilities $\vec{y} = T(x;\theta_T)$. We consider model stealing attacks under two settings based on the availability of data:

1. *Data-free setting (Primary goal):* The adversary does not have access to the dataset $\mathcal{D}_T$ used to train the target model or a good way to sample from the target data distribution $\mathbb{P}_T$. (Section 4)

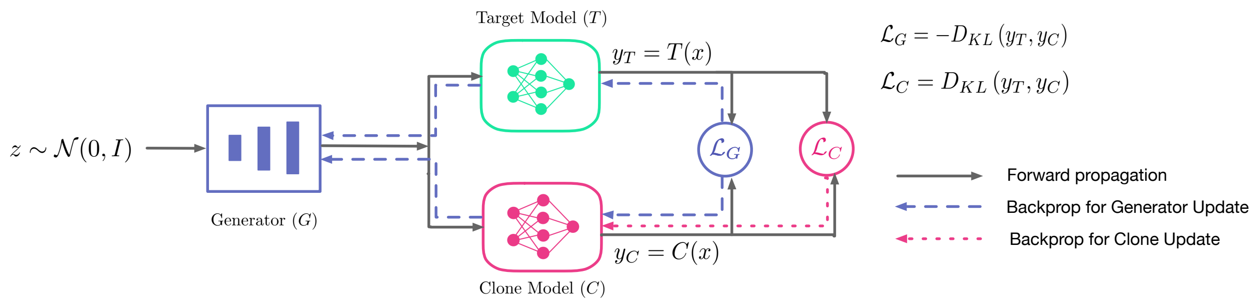

Figure 2. MAZE Attack Setup: MAZE uses a generative model G to produce the synthetic input queries $\{x\}$ to perform Model Stealing. The clone model C is trained to match the predictions of the target model T. G is trained to produce queries that maximize the dissimilarity between y_T and y_C. Optimizing $\mathcal{L}_G$ requires backpropagation through T to update G. However, we only have black-box access to T, therefore we use zeroth-order gradient estimation to perform gradient descent on $\mathcal{L}_G$.

2. *Partial-data setting (Secondary goal):* The adversary has access to a small subset (e.g., 100) of training examples randomly sampled from the training dataset of the target model. (Section 6)

For both of these settings we assume the availability of a test set $\mathcal{D}_{test}$, which is used to report the test accuracies of the clone models produced by our attack.

4. MAZE: Data-Free Model Stealing

We propose *MAZE*, a data-free model stealing attack using zeroth order gradient estimation. Unlike existing attacks, MAZE does not require access to the target or a surrogate dataset and instead uses a generative model to produce the synthetic queries for launching the attack. Fig. 2 shows an overview of MAZE. In this section, we first describe the training objectives of the clone and the generator model. We then motivate the need for gradient estimation to update G in the black-box setting of MS attack and show how zeroth-order gradient estimation can be used to optimize the parameters of G. Finally, we discuss our algorithm to carry out model stealing with MAZE.

4.1. Training the Clone Model

The clone model is trained using the input queries produced by the generator. The generator G takes in a low dimensional latent vector z, sampled from a random normal distribution, and produces an input query $x \in \mathbb{R}^d$ that matches the input dimension of the target classifier (Eqn. 3). We use x to obtain the output probabilities of the target model $\vec{y_T}$ and clone model $\vec{y_C}$ on x as shown in Eqn. 4.

$$x = G(z; \theta_G); \;\; z \sim \mathcal{N}(0, \boldsymbol{I}) \tag{3}$$

$$\vec{y_T} = T(x; \theta_T); \; \vec{y_C} = C(x; \theta_C) \tag{4}$$

Where θ_T, θ_C and θ_G represent the parameters of the target, clone, and generator models, respectively. The clone model is trained using the loss function in Eqn. 5 to minimize the KL divergence between $\vec{y_C}$ and $\vec{y_T}$.

$$\mathcal{L}_C = D_{KL}(\vec{y_T} \| \vec{y_C}) \tag{5}$$

4.2. Training the Generator Model

The generator model G synthesises the queries necessary to perform model stealing. Similar to recent works in data-free KD [20, 35, 5], MAZE trains the generator to produce queries that maximize the disagreement between the predictions of the teacher and the student by maximizing the KL-divergence between $\vec{y_T}$ and $\vec{y_C}$. The loss function used to train the generator model is described by Eqn. 6, which we refer to as the *disagreement objective*.

$$\mathcal{L}_G = -D_{KL}(\vec{y_T} \| \vec{y_C}) \tag{6}$$

Training G on this loss function maximizes the disagreement between the predictions of the target and the clone model. Since C and G have opposing objectives, training both models together results in a two-player game, similar to *Generative Adversarial Networks* [8], resulting in the generation of inputs that maximize the learning of the clone model. By training C to match the predictions of T on the queries generated by G, we can perform knowledge distillation and obtain a highly accurate clone model.

Training G using the loss function in Eqn. 6 requires backpropagating through the predictions of the target model T, as shown by the dashed lines in Fig. 2. Unfortunately, as we only have black-box access to T, we cannot perform back-propagation directly, preventing us from training G and carrying out the attack. To solve this problem, our insight is to use *zeroth-order gradient estimation* to approximate the gradient of the loss function $\mathcal{L}_G$. The number of black-box queries necessary for ZO gradient estimation scales with the dimensionality of the parameters being optimized. Estimating the gradients of $\mathcal{L}_G$ with respect to the generator parameters θ_G directly is expensive as the generator has on the order of millions of parameters. Instead, we choose to estimate the gradients with respect to the synthetic input x produced by the generator, which has a much lower dimensionality (3072 for CIFAR-10), and use this estimate to back propagate through G. This modification allows us to compute a gradient estimates in a query efficient manner to update the generator model. The following section describes how we efficiently apply zeroth-order gradient estimation to train the generator model.

4.3. Train via Zeroth-Order Gradient Estimate

Zeroth-order gradient estimation [22, 7] is a popular technique to perform optimization in the black-box setting. We use this technique to train our generator model G. Recall that our objective is to update the generator model parameters θ_G using gradient descent to minimize the loss function $\mathcal{L}_G$ as shown in Eqn. 7.

$$\theta_G^{t+1} = \theta_G^t - \eta \nabla_{\theta_G} \mathcal{L}_G \tag{7}$$

Updating θ_G in this way requires us to compute the derivative of the loss function $\nabla_{\theta_G}\mathcal{L}_G$. By the use of chain-rule, $\nabla_{\theta_G}\mathcal{L}_G$ can be decomposed into two components as shown in Eqn. 8.

$$\nabla_{\theta_G}\mathcal{L}_G = \frac{\partial \mathcal{L}_G}{\partial \theta_G} = \frac{\partial \mathcal{L}_G}{\partial x} \times \frac{\partial x}{\partial \theta_G} \tag{8}$$

We can compute the second term $\frac{\partial x}{\partial \theta_G}$ in Eqn. 8 by performing backpropagation through G. Computing the first term $\frac{\partial \mathcal{L}_G}{\partial x}$ however requires access to the model parameters of the target model (θ_T). Since T is a black-box model from the perspective of the attacker, we do not have access to θ_T, which prevents us from computing $\frac{\partial \mathcal{L}_G}{\partial x}$ through backpropagation. Instead, we propose to use an approximation of the gradient by leveraging zeroth-order gradient estimation. To explain how the gradient estimate is computed, consider an input vector $x \in \mathbb{R}^d$ generated by G that is used to query T. We can estimate $\frac{\partial \mathcal{L}_G}{\partial x}$ by using the method of forward differences [27] as shown in Eqn. 9.

$$\hat{\nabla}_x \mathcal{L}_G(x; u_i) = \frac{d \cdot (\mathcal{L}_G(x + \epsilon u_i) - \mathcal{L}_G(x))}{\epsilon} u_i \tag{9}$$

Where u_i is a random variable drawn from a d dimensional unit sphere with uniform probability and ϵ is a small positive constant called the *smoothing factor*. The random gradient estimate, shown in Eqn. 9, tends to have a high variance. To reduce the variance, we use an averaged version of the random gradient estimate [4, 17] by computing the forward difference using m random directions $\{u_1, u_2, ..u_m\}$, as shown in Eqn. 10.

$$\hat{\nabla}_x \mathcal{L}_G(x) = \frac{1}{m}\sum_{i=1}^{m} \hat{\nabla}_x \mathcal{L}_G(x; u_i) \tag{10}$$

Where $\hat{\nabla}_x \mathcal{L}_G$ is an estimate of the true gradient $\nabla_x \mathcal{L}_G$. By substituting $\hat{\nabla}_x \mathcal{L}_G$ into Eqn. 8, we can compute an approximation for the gradient of the loss function of the generator: $\hat{\nabla}_{\theta_G}\mathcal{L}_G$. The gradient estimate $\hat{\nabla}_{\theta_G}\mathcal{L}_G$ computed this way can be used to perform gradient descent by updating the parameters of the generator model θ_G according to Eqn. 7. By updating θ_G, we can train G to produce the synthetic examples required to perform model stealing.

4.4. MAZE Algorithm for Model Stealing Attack

We outline the algorithm of MAZE in Algorithm 1 by putting together the individual training algorithms of the generator and clone models. We start by fixing a query budget Q, which dictates the maximum number of queries we are allowed to make to the target model T. ϵ is the smoothing parameter and m is the number of random directions used to estimate the gradient. We set the value of ϵ to 0.001 in our experiments. N_G, N_C represent the number of training iterations and η_G, η_C represent the learning rates of the generator and clone model, respectively. N_R denotes the number of iterations for experience replay.

Algorithm 1: MAZE Algorithm for Model Stealing Attack

```
Input: T, Q, ε, m, N_G, N_C, N_G, η_G, η_C
Output: Clone model C(·; θ_C)
Initialize G(·; θ_G), C(·; θ_C), q ← 0, 𝒟 ← {}
while q < Q do
    // Generator Training
    for i ← 0 to N_G do
        x = G(z) : z ~ 𝒩(0, I)
        ℒ_G = −D_KL (T(x)‖C(x))
        ∇̂_θG ℒ_G ← ZO_grad_est(G, T, C, x, ε, m)
        θ_G ← θ_G − η_G ∇̂_θG ℒ_G
    end
    // Clone Training
    for i ← 0 to N_C do
        x = G(z) : z ~ 𝒩(0, I)
        ℒ_C = D_KL (T(x)‖C(x))
        θ_C ← θ_C − η_C ∇_θC ℒ_C
        𝒟 ← 𝒟 ∪ {(x, T(x))}
    end
    // Experience Replay
    for i ← 0 to N_T do
        (x, y_T) ~ 𝒟
        ℒ_C = D_KL (y_T‖C(x))
        θ_C ← θ_C − η_C ∇_θC ℒ_C
    end
    q ← update(q)
end
```

The outermost loop of the attack repeats till we exhaust our query budget Q. The attack algorithm involves three phases: 1. *Generator Training* 2. *Clone Training* and 3. *Experience Replay*. In the *Generator Training* phase, we perform N_G rounds of gradient descent for G, which is trained to produce inputs x that maximize the KL-divergence between the predictions of the target and clone model. θ_G is updated by using zeroth-order gradient estimates as described in Section 4.3. This is followed by the *Clone Training* phase, where we perform N_C rounds of gradient descent for C. In each round, we generate a batch of inputs $x = G(z)$ and use these inputs to query the target model. The clone model is trained to match the predictions of the target model by minimizing $D_{KL}(T(x) \| C(x))$. The input, prediction pair: $(x, T(x))$ generated in each round is stored in dataset $\mathcal{D}$. Finally, we perform *Experience Replay*, where we train the clone on previously seen inputs that are stored in $\mathcal{D}$. Retraining on previously seen queries reduces *catastrophic forgetting* [19] and ensures that the clone model continues to classify old examples seen during the earlier part of the training process correctly.

4.5. Computing the Query Cost

The target model needs to be queried in order to update both the generator and the clone models. Considering a batch size of 1, one training iteration of G requires $m + 1$ queries to T for the zeroth-order gradient estimation and each training loop of C requires 1 query. Experience replay, on the other hand, does not require any additional queries to T. Thus, with a batch size of B, the query cost of each iteration is described by Eqn. 11

$$\text{Query cost per iteration} = B(N_G(m+1) + N_C) \quad (11)$$

We use $B = 128$, $N_G = 1, N_C = 5, N_R = 10$ and $m = 10$ in our experiments, unless stated otherwise. Thus, each iteration of the attack requires 2048 queries. We use a query budget of $5M$ for FashionMNIST and SVHN and a query budget of $30M$ for GTSRB and CIFAR-10 datasets to report our results.

5. Experimental Evaluation

We validate our attack by performing model stealing on various target models and provide experimental evidence to show that our attack can produce high accuracy clone models without using any data. We compare our results against two prior works– *KnockoffNets* and *Jacobian Based Dataset Augmentation* (JBDA)– and show that the clone models produced by our attack have comparable or better accuracy than the ones produced by these prior works, despite not using any data. In addition, we also perform sensitivity studies to understand the impact of various attack parameters including query cost and number of gradient estimation directions in Appendix B.

5.1. Setup: Dataset and Architecture

We perform our evaluations by attacking DNN models that are trained on various image-classification tasks. The datasets and target model accuracies used in our experiments are mentioned in Table 1. We use a LeNet for the FashionMNIST and ResNet-20 for the other datasets as the target model. Our attack assumes no knowledge of the target model and uses a randomly initialized 22-layer WideResNet [37] as the clone model for all the datasets. In general, any sufficiently complex DNN can be used as the clone model. We use an SGD optimizer with an initial learning rate of 0.1 to train our clone model. For G, we use a generative model with 3 convolutional layers. Each convolutional layer in G is followed by a batchnorm layer and the activations are upsampled to ensure that the outputs generated by G are of the correct dimensionality corresponding to the dataset being attacked. We use an SGD optimizer with an initial learning rate of 0.0001 to train G. The learning rates for both the clone and generator models are decayed using cosine annealing.

5.2. Configuration of Existing Attacks

Existing MS attacks either use surrogate data or synthetic datasets derived from partial access to the target dataset. We compare MAZE with the following attacks:

1. KnockoffNets [24] attack uses a surrogate dataset to query the target model to construct a labeled dataset using the predictions of the target model. This labeled dataset is used to train the clone model. We use MNIST, CIFAR10, CIFAR100, and CIFAR10 as the surrogate datasets for FashionMNIST, SVHN, CIFAR10, and GTSRB models, respectively. In each case, we query the target model with the training examples of the surrogate dataset. We then use the dataset constructed from these queries to train the clone model for 100 epochs using an SGD optimizer with a learning rate of 0.1 with cosine annealing scheduler.

2. JBDA [26]: attack performs MS by using synthetic examples to query the target model. These synthetic examples are generated by adding perturbations to a set of *seed* examples, which are obtained from the data distribution of the target model. The perturbations are computed using the Jacobian of the clone model's loss function (Eqn. 1). We start with an initial dataset of 100 seed examples and perform 6 rounds[2] of synthetic data augmentation with the clone model being trained for 10 epochs between each round. λ in Eqn. 1 dictates the magnitude of the perturbation. We set this to a value of 0.1. We use Adam optimizer with a learning rate of 0.001 to train the clone model.

3. Noise: To test if inputs sampled from noise can be used to carry out MS attack, we design a *Noise* attack. We follow the proposal by Roberts et al. [28] and use random samples from an Ising prior model to query the target model. This attack serves as a baseline data-free MS attack to compare with our proposal.

[2] We found that the accuracy of the JBDA attack stagnates beyond 6 augmentation rounds. This is in line with the observations made by Juuti et al. [14].

Table 1. Comparison of clone accuracies obtained from various attacks. Numbers in the bracket express the accuracy as a multiple of the target model accuracy. MAZE obtains high accuracy ($0.90\times$ to $0.99\times$), despite not using any data.

Dataset	Target Accuracy (%)	MAZE (data-free)	KnockoffNets (surrogate data)	JBDA (partial-data)	Noise (data-free)
FashionMNIST	91.04	**81.9 (0.90×)**	47.26 (0.52×)	62.65 (0.69×)	62.91 (0.69×)
SVHN	95.25	**93.85(0.99×)**	92.77 (0.97×)	17.16(0.18×)	51.86 (0.54×)
GTSRB	97.43	88.31 (0.91×)	**89.86 (0.92×)**	12.80(0.13×)	38.38 (0.39×)
CIFAR-10	92.26	**89.85 (0.97×)**	82.56 (0.89×)	25.11 (0.27×)	10.17 (0.11×)

5.3. Key Result: Normalized Clone Accuracy

Table 1 shows the clone-accuracy obtained by attacking various target models using MAZE. The numbers in brackets express the clone accuracy normalized to the accuracy of the target model being attacked. We also compare MAZE with existing MS attacks and highlight the best clone accuracy for each dataset in bold. Our results show that MAZE produces high accuracy clone models with a normalized accuracy greater than $0.90\times$ for all the target models under attack. In contrast, the baseline *Noise* attack fails to produce high accuracy clone models for most of the datasets.

Furthermore, the results from our attack also compare favorably against *KnockoffNets* and *JBDA*, both of which require access to some data. We find that the effectiveness of KnockoffNets is highly dependent on the surrogate data being used to query the target model. For example, using MNIST to attack FashionMNIST dataset results in a low accuracy clone model ($0.52\times$ target accuracy) as these datasets as visually dissimilar. However, using CIFAR-100 to query CIFAR-10 results in a high accuracy clone model ($0.89\times$ target accuracy) due to the similarities in the two datasets. JBDA seems to be effective for attacking simpler datasets like $FashionMNIST$, but the accuracy reduces when attacking more complex datasets. This is in part because JBDA produces queries that are highly correlated to the initial set of "seed" examples, which sometimes results in worse performance even compared to noise (e.g. SVHN). By using the disagreement objective to train the generator, MAZE can generate queries that are more useful in training the clone model and result in higher accuracy of clones ($0.91\times$-$0.99\times$) compared to other attacks like JBDA ($0.13\times$-$0.69\times$) that use synthetic data.

6. MAZE-PD: MAZE with Partial-Data

The accuracy and speed of our attack can be improved if a few examples from the training-data distribution of the target model are available to the adversary. In this section we develop *MAZE-PD*, an extension of MAZE to the partial-data setting. In the data-free setting of MAZE, G is trained on a *disagreement objective* to produce inputs that maximize the disagreement between the target and the clone model. In the presence of a limited amount of data, we can additionally train the generator to produce inputs that are closer to the target distribution by using the *Waserstein Generative Adversarial Networks (WGANs)* [1] training objective. We observe that even a small amount of data from the target distribution (100 examples) can enable the generator to produce synthetic inputs that are closer to the target distribution (see Appendix D for example images). By improving the quality of the generated queries, MAZE-PD not only improves the effectiveness of the attack but also allows the attack to succeed with far fewer queries compared to MAZE. In this section, we describe how WGANs can be incorporated into the training of the generator model to develop MAZE-PD. We also provide empirical evidence to show that MAZE-PD improves clone accuracy and reduces query cost significantly compared to MAZE.

6.1. Incorporating WGAN in MAZE

We describe the modifications to the training algorithm of MAZE (Algorithm 1) to incorporate WGAN training in the partial-data setting. In addition to the generator (G) and clone (C) models, we define a *critic model* D, which estimates the Wasserstein distance between the target data distribution $\mathbb{P}_T$ and the synthetic data distribution of the generator $\mathbb{P}_G$ using the function described in Eqn. 12.

$$W\left(\mathbb{P}_T, \mathbb{P}_G\right) = \max_{\theta_D} \mathbb{E}_{x\sim\mathbb{P}_T}\left[D(x)\right] - \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[D\left(G(z)\right)\right] \tag{12}$$

The generator model aims to produce examples closer to the target distribution by minimizing the Wasserstein distance estimated by the critic model. To incorporate the WGAN objective in the training of the generator, we modify the original loss function of G (Eqn. 6) with an additional term as shown in Eqn. 13.

$$\begin{aligned} &x = G(z); z \sim \mathcal{N}(0, I) \\ &\mathcal{L}_G = -D_{KL}(T(x)\|C(x)) - \lambda D(x) \end{aligned} \tag{13}$$

The first term in Eqn. 13 represents the disagreement loss from MAZE and the second term is the WGAN loss.

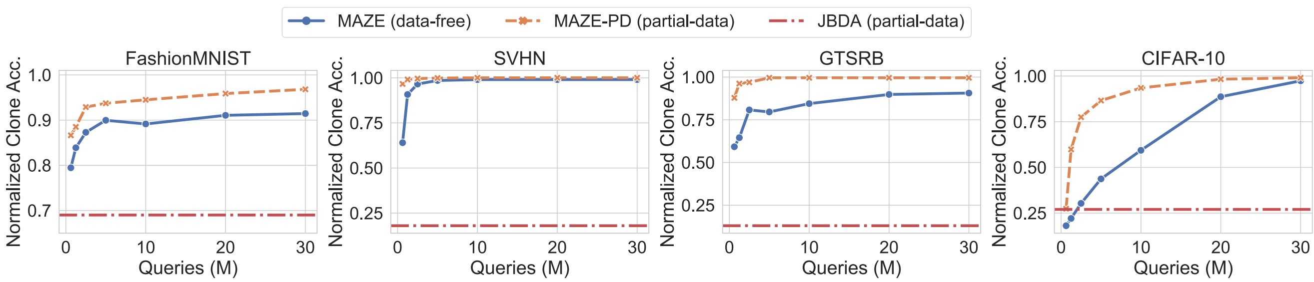

Figure 3. Normalized clone accuracy of MAZE (data-free), MAZE-PD (partial-data), and JBDA (partial-data) as the query budget is varied. Our results show that for a given query budget, MAZE-PD can train a clone model with higher accuracy than MAZE. The accuracy of MAZE-PD is also significantly better than the JBDA attack.

The hyper-parameter λ balances the relative importance between these two losses. To train the *critic model* D, we add an extra training phase (described by Algorithm 2) to our original training algorithm. We also include a gradient penalty term $GP = (\|\nabla_x D(x)\|_2 - 1)^2$ in $\mathcal{L}_D$ to ensure that D is 1-Lipschitz continuous. We refer the reader to Appendix A for a more detailed explanation of WGANs.

Algorithm 2: Critic Training

// Critic Training
for $i \leftarrow 0$ **to** N_d **do**
  $z \sim \mathcal{N}(0, I); x \sim \{x_i\}_{i=1}^{n}$
  $\mathcal{L}_D = D(G(z)) - D(x) + GP$
  $\theta_D \leftarrow \theta_D - \eta_D \nabla_{\theta_D} \mathcal{L}_D$
end

The training loops for the *clone training* and *experience replay* in Algorithm 1 remain unchanged. Using these modifications we can train a generator model that produces inputs closer to the target distribution $\mathbb{P}_T$.

6.2. Results: Clone Accuracy with Partial Data

We repeat the MS attack using MAZE-PD in the partial data setting. We assume that the attacker now has access to 100 random examples from the training data of the target model, which is roughly 0.2% of the total training data used to train the target model. We use $\lambda = 10$ in Eqn. 13 and $N_d = 10$ in Algorithm 2. Note that critic training does not require extra queries to the target model. The rest of the parameters are kept the same as before. Fig 3 shows our results comparing the normalized clone accuracy obtained with MAZE-PD and MAZE (data-free) for various query budgets. For a given query budget, MAZE-PD obtains a higher clone accuracy compared to MAZE and achieves near-perfect clone accuracy ($0.97\times$-$1.0\times$) for all the datasets. Additionally, MAZE-PD offers a reduction of $2\times$ to $24\times$ in the query budget compared to MAZE for a given clone accuracy (see Appendix C).

Comparison with JBDA: We compare the performance of MAZE-PD with JBDA, which also operates in the partial-data setting. JBDA produces low clone accuracies for most datasets (less than $0.30\times$ for SVHN, GTSRB, and CIFAR-10). In contrast, MAZE-PD obtains highly accurate clone models ($0.97\times$-$1.0\times$) across all four datasets.

7. Conclusion

This paper proposes *MAZE*, a high-accuracy MS attack that requires no input data. To the best of our knowledge, MAZE is the first data-free MS attack that works effectively for complex DNN models trained across multiple image-classification tasks. MAZE uses a generator trained with zeroth-order optimization to craft synthetic inputs, which are then used to copy the functionality of the target model to the clone model. Our evaluations show that MAZE produces clone models with high classification accuracy ($0.90\times$ to $0.99\times$). Despite not using any data, MAZE outperforms recent attacks that rely on partial-data or surrogate-data. Our work presents an important step towards developing highly accurate data-free MS attacks.

In addition, we propose *MAZE-PD* to extend MAZE to the partial-data setting, where the adversary has access to a small number of examples from the target distribution. MAZE-PD uses generative adversarial training to produce inputs that are closer to the target distribution. This further improves accuracy ($0.97\times$ to $1.0\times$) and yields a significant reduction in the number of queries ($2\times$ to $24\times$) necessary to carry out the attack compared to MAZE.

8. Acknowledgements

We thank M. Emre Gursoy, Ryan Feng, Poulami Das and Gururaj Saileshwar for their feedback. This work was partially supported by a gift from Facebook and is partially based upon work supported by the National Science Foundation under grant numbers 1646392 and 2939445 and under DARPA award 885000. We thank NVIDIA for the donation of the Titan V GPU that was used for this research.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 7, 11

[2] Jianbo Chen and Michael I Jordan. Boundary attack++: Query-efficient decision-based adversarial attack. *arXiv preprint arXiv:1904.02144*, 2(7), 2019. 14

[3] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018. 14

[4] John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015. 5

[5] Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019. 4

[6] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015. 1

[7] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. 2, 5

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 4

[9] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 1

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017. 11

[11] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009. 1

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[13] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018. 14

[14] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019. 6

[15] Sanjay Kariyappa and Moinuddin K Qureshi. Defending against model stealing attacks with adaptive misinformation, 2019. 15

[16] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. Defending against model stealing attacks using deceptive perturbations. *arXiv preprint arXiv:1806.00054*, 2018. 15

[17] Sijia Liu, Jie Chen, Pin-Yu Chen, and Alfred O Hero. Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications. *arXiv preprint arXiv:1710.07804*, 2017. 5

[18] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, 2005. 2

[19] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 6

[20] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. In *Advances in Neural Information Processing Systems*, pages 9547–9557, 2019. 4, 13

[21] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910*, 2018. 1

[22] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017. 2, 5

[23] Seong Joon Oh, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 121–144. Springer, 2019. 2

[24] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019. 2, 3, 6

[25] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. In *ICLR*, 2020. 15

[26] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017. 2, 3, 6

[27] Boris T Polyak. Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1987. 5

[28] Nicholas Roberts, Vinay Uday Prabhu, and Matthew McAteer. Model weight theft with just noise inputs: The curious case of the petulant attacker. *arXiv preprint arXiv:1912.08987*, 2019. 3, 6

[29] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017. 1

[30] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. 1

[31] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016. 2

[32] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv*, 2017. 1

[33] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 36–52. IEEE, 2018. 2

[34] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018. 1

[35] Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M Alvarez, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. *arXiv preprint arXiv:1912.08795*, 2019. 1, 4, 14

[36] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016. 14

[37] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 6

[38] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. *arXiv preprint arXiv:1911.07135*, 2019. 1