Unlocking High-Fidelity Learning: Towards Neuron-Grained Model Extraction

Yaxin Xiao, *Student Member, IEEE,* Haibo Hu, *Senior Member, IEEE,* Qingqing Ye, *Member, IEEE,* Li Tang, Zi Liang, Huadi Zheng

***Abstract*—Model extraction (ME) attacks replicate valuable black-box machine learning (ML) models via malicious query interactions. Cutting-edge attacks focus on actively designing query samples to enhance model fidelity and imprudently adhere to the standard ML training approach. This causes a deviation from the true objective of learning a model over a task. In this paper, we innovatively shift our focus from query selection to training process optimization, aiming to boost the similarity of the copy model with the victim model from neuron to model level. We leverage neuron matching theory to attain this objective and develop a general training booster framework, MEBooster, to fully exploit this theory. MEBooster comprises an initial bootstrapping phase that furnishes initial parameters and an optimal model architecture, followed by a post-processing phase that employs fine-tuning for enhanced neuron matching. Notably, MEBooster can seamlessly integrate with all existing model extraction attacks, enhancing their overall performance. Performance evaluation shows up to 58.10% fidelity gain in image classification. From a defender's perspective, we introduce a novel defensive strategy called *Stochastic Norm Enlargement* (SNE) to mitigate the risk of such attacks by enlarging the model parameters' norm property in training. Performance evaluation shows up to 58.81% extractability (*i.e.*, fidelity) reduction.**

***Index Terms*—Machine Learning Privacy, Model Extraction Attack, Defense against Model Extraction**

I. Introduction

The *Machine Learning as a Service* (MLaaS) business has emerged to deploy machine learning (ML) models on cloud platforms, such as Microsoft Azure ML [1], Amazon AWS ML [2], and Google Cloud AI [3]. These models, through query interfaces, are now accessed by numerous applications in the fields of computer vision and natural language processing. However, recent model extraction (ME) attacks [4, 5, 6] have exposed vulnerabilities of MLaaS through these query interfaces, enabling adversaries to replicate a victim model without accessing its training data. Model privacy and data privacy are fundamentally interconnected [7, 8, 9, 10]. Such attacks not only compromise the privacy of model parameters but also threaten the privacy of the data underlying the model. For example, they facilitate various downstream attacks on data privacy and security, including membership inference [11], adversarial attacks [12, 13], and model inversion [14].

To extract deep and complex ML models to date, learning-based model extraction is the de-facto type of ME attacks. Existing works [5, 6, 15, 16] cast the attack as an active learning problem [15], where the goal is to acquire informative query samples from the victim model for training the copy model. These approaches follow a "**learning-a-task**" paradigm, which focuses on improving task accuracy through conventional training. However, this paradigm struggles to achieve high-fidelity extraction. The reason is that the copy model is trained under unsuitable conditions for replicating the victim model. As a result, it often converges to sub-optimal optima with low fidelity (*i.e.*, low similarity to the victim model). For example, ActiveThief [16], the state-of-the-art learning-based model extraction, can only achieve 94.25% fidelity even when we grant it full access to all training details of the victim model, such as architecture (ResNet [17]), initial parameters, training dataset (CIFAR [18]), and hyperparameters of the optimizer. That is, there is a discrepancy in the labels inferred by the original and the copy model for about 1 in every 20 test samples. This limitation arises from a fundamental mismatch between the goal of model extraction and the learning-a-task paradigm. However, this gap remains largely unaddressed in the existing literature.

In this paper, we reevaluate the role of learning in model extraction from a neuron-grained perspective and drive the learning process beyond previous expectations by introducing a generic training booster — MEBooster. The key idea is to **"learning-a-model"** instead of to "learning-a-task". Through learning a model, the copy model can even achieve neuron-level extraction to a certain extent [19, 20, 21]. To this end, MEBooster brings two new perspectives: bootstrapping the copy model's initialization states and post-processing fine-tuning. The former method assigns the copy model a superior initialization status by estimating the victim model's parameters [22] (see Section V-C). The initialization errors are further compensated by a width-expanded copy model that can accommodate multiple estimated results (see Section V-D). The latter extends neuron matching across more layers of the copy model [20] (see Section V-E). It is noteworthy that both perspectives of MEBooster can be directly adapted to existing learning-based model extraction attacks without changes or additional query costs.

Manuscript received October 22, 2024; revised June 09, 2025.

Yaxin Xiao, Haibo Hu, Qingqing Ye, and Zi Liang are with Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong SAR, China (e-mail: 20034165r@connect.polyu.hk, {haibo.hu, qqing.ye}@polyu.edu.hk, zi1415926.liang@connect.polyu.hk).

Li Tang is with the Department of Software Engineering, Yunnan University, Kunming 650091, China (e-mail: tangli@ynu.edu.cn).

Huadi Zheng is with Huawei Technologies Co., Ltd., Shenzhen 518129, China (e-mail: zhenghuadi@huawei.com).

MEBooster faces several challenges. First, existing parameter estimation methods are limited for two-layer linear neural networks [22]. In Section V-C3, we generalize such methods to complex models by encoding patch samples for middle convolutional layers. Second, these methods assume access to the probability distribution function of the input samples, which is typically unavailable in practice. To address this, Section V-C2 incorporates score matching [23] to model data distribution using ML models using implicit score loss. Third, existing optimal convergence theories focus only on the lowest layer of neural networks [20]. We extend this theory to facilitate upper layers (see Section V-E).

As our second goal, we explore the concept of "learning-a-model" from the defender's perspective. While MEBooster enhances factors that facilitate model extraction attacks, our defense focuses on identifying factors of the victim model that resist these attacks. Current defenses against model extraction primarily rely on prediction perturbation strategies [24, 25, 26, 27], and provide limited protection, especially against data-free MEs [5, 6, 28], where adversaries extensively explore the output space and thereby render these perturbations ineffective. In contrast, we investigate inherent properties that impact how easily a model can be learned without altering each prediction. Accordingly, in Section VII, we propose a defensive training strategy to adjust these properties to enhance the model's intrinsic resistance to extraction. In summary, we make the following contributions in this paper:

- To the best of our knowledge, this is the first study on learning-a-model instead of learning-a-task in the **training process** of model extraction attacks.
- We present a training booster framework, MEBooster, to exploit the potential advantage of the learning at the neuron-grained level.
- Extensive experiments are conducted to demonstrate the superiority and generality of MEBooster on various computer vision tasks. In the best case, MEBooster yields 58.10% fidelity gain over existing attacks without the booster.
- A defensive training strategy is proposed for the first time to exploit the hard-to-extract properties of the victim model. Experimental results show it can reduce the extractability (*i.e.*, fidelity) of the victim model by up to 58.81%.

The rest of the paper is organized as follows. Section II reviews related work. Section III defines the notations and the problem. Section IV provides an overview of the MEBooster framework. Section V elaborates on MEBooster's key components: initial bootstrapping and post-processing fine-tuning, both supported by neuron matching theory. Section VI discusses the experimental results of MEBooster. Section VII introduces a novel defensive training strategy. Finally, Section VIII concludes this work.

II. Related Works

A. Model Extraction Attacks

Recently, an increasing number of commercial ML models deployed with public query interfaces are shown to be highly replicable by model extraction attacks [4, 29, 30], which expose severe vulnerability in model confidentiality. Most attacks follow a learning-based approach, where the attacker approximates the target model using queried samples and off-the-shelf gradient descent (GD) training methods. Since the training data of a black-box victim model is usually private and inaccessible, early attackers have to construct and surrogate query datasets, which have been shown to be ineffective. For example, using random noise as query samples, model extraction is almost invalid [16]. To address this, Chandrasekaran *et al.* [15] propose a learning-based model extraction based on active learning, a type of semi-supervised learning that selectively chooses the training data to label so as to maximize extraction efficiency. From then on, acquiring informative query data becomes a key objective for almost all existing learning-based model extraction works [5, 6, 16, 29, 31]. Table I summarizes the features of existing learning-based ME methods. Several attackers, such as Knockoff [32], ActiveThief [16], and MExMI [33], utilize pool-based strategies to select samples from real-life public datasets, necessitating a large data pool. An alternative approach is query-synthesizing-based model extraction. One category employs synthetic active learning algorithms, exemplified by PRADA [31] and Papernot [29]. Another category of synthetic approaches is data-free model extraction, which has gained popularity, with methods such as MAZE [5], DFME [6], and DisGUIDE [34] emerging.

TABLE I
Main-stream Learning-based Model Extraction Attacks

Attacks	Data Acquisition		Training Technique
	Real-life	Synthesized	
Tramèr [4]		✓	–
Papernot [29]		✓	Structure Selection
Knockoff [32]	✓		–
PRADA [31]		✓	CV Search
ActiveThief [16]	✓		–
MAZE [5]		✓	–
DFME [6]		✓	–
MExMI [33]	✓		–
HODA [35]		✓	–
DisGUIDE [34]		✓	–
Bayes Attack [26]	✓		–

However, the focus of all these works on query data acquisition **overshadows other optimization opportunities for model extraction, especially in the training process**. As shown in Table I, there is a lack of studies on training techniques. So far, only Papernot *et al.* investigate the structure selection methods for copy models [29], and PRADA [31] uses cross-validation (CV) to search for training hyper-parameters.

B. Defenses Against Model Extraction

Current research addresses model extraction threats through three primary defense paradigms: (1) malicious query detection, (2) prediction perturbation, and (3) model watermarking. Detection-based methods [31, 36] aim to identify suspicious query patterns, while prediction perturbation [24, 25, 26, 27] systematically alters output predictions to degrade extraction performance. Model watermarking techniques [37, 38] embed identifiable signatures into models to trace unauthorized use. Among these, query detection remains vulnerable to coordinated or stealthy attacks, and model watermarking faces

growing threats from watermark removal techniques [39, 40]. In contrast, prediction perturbation remains the only approach capable of directly degrading the performance of extracted models.

However, prediction perturbation [24, 25, 26, 27] faces an inherent trade-off between privacy and utility, as excessive distortion degrades the experience of legitimate users. To complement these reactive defenses, we propose a proactive defense paradigm based on **model modification**. It reduces the extractability of victim models by strategically modifying their properties during training, offering a fundamentally different form of preemptive protection compared to post-deployment methods.

III. Preliminary and Problem Definition

A. Notations

The victim models are deep neural networks (DNN) trained for classification tasks with K classes in supervised learning. A DNN model $F(\cdot;\theta) : \mathcal{X} \Rightarrow \mathcal{Y}$ is defined over an input space $\mathcal{X} \in \mathbb{R}^d$ and an output space $\mathcal{Y}$, where d is the input dimension. For example, $\mathcal{X}$ can be images or texts, and $\mathcal{Y}$ are the image labels or text sentiments.

A typical L-layer DNN consists of layers such as linear, convolutional, activation, and pooling, with the Lth layer being the output layer. Common activation functions $\sigma(\cdot)$ include ReLU / Leaky ReLU [41, 42]. In layer l, the width/channel is n_l, with neuron weights and bias as $[w_{l,i}, b_{l,i}]$. The weight matrix $W_l = [w_{l,1}, ..., w_{l,n_l}]$ connects layers $l-1$ and l, where $w_{l,i} \in \mathbb{R}^{n_{l-1}}$ for linear layers, and $w_{l,i} \in \mathbb{R}^{n_{l-1}\times k_l \times k_l^*}$ for convolutional layers, with (k_l, k_l^*) being the kernel size. Given a batch of input samples $\{x_i\}_{i=1}^b$ of size b, the output of layer l is denoted as the matrix $\mathbf{f}_l\left(\{x_i\}_{i=1}^b\right) = \left[f_{l,1}\left(\{x_i\}_{i=1}^b\right), \ldots, f_{l,n_l}\left(\{x_i\}_{i=1}^b\right)\right] \in \mathbb{R}^{b\times n_l}$, where $f_{l,j}(\cdot)$ denotes the output of the j-th neuron in layer l across the batch. The activation function's diagonal matrix tensor for layer l is defined as $D_l\left(\{x_i\}_{i=1}^b\right) = \left[z_{l,1}\left(\{x_i\}_{i=1}^b\right), \ldots, z_{l,n_l}\left(\{x_i\}_{i=1}^b\right), 1\right] \in \mathbb{R}^{b\times(n_l+1)\times(n_l+1)}$. If the activation function $\sigma(\cdot)$ is ReLU, $z_{l,j}(x_i) \in \{0,1\}$ indicates whether the j-th neuron is activated for input x_i.

B. Problem Definition

The attacker has gained access to the query interface of the victim model and can obtain the inference results (with probabilities) of query samples chosen by her. As such, she performs ME attacks against F within a query budget, aiming to produce a **copy model** F' that closely matches the functionality of F'. This is mainly measured by **fidelity** [43], which is the proportion of label agreement of two models on an evaluation dataset D_t.

C. Threat Model

We assume that attackers are aware of a victim model's task and have access to its query APIs, enabling them to interpret the output data obtained. These attackers are expected to have good mastery of machine learning techniques, including standard initialization methods. For instance, in the context of image classification, they are familiar with widely adopted methods such as He initialization [44]. Besides, we assume the attacker possesses some preliminary knowledge about the victim model, such as the model family. We further assume that attackers have some prior knowledge of the victim model's family or architecture. This assumption is realistic because many MLaaS platforms publicly disclose architectural and hyperparameter details. Services such as AWS Marketplace [2] and Huawei AI Gallery [45] list model architectures and optimizer settings, and researchers from these providers often publish papers [46] that reveal additional implementation information.

IV. The ME Booster Framework

Framework Overview. Fig. 1 illustrates the general learning-based model extraction (ME) framework augmented by ME-Booster (in the green area). This framework is an abstraction of existing learning-based model extraction attacks, *e.g.*, DFME [6] and ActiveThief [16]. As shown in this figure, MEBooster focuses on training and consists of two parts: initial bootstrapping (steps ①-③), and the post-processing (step ⑥). We briefly introduce these two parts below, with detailed discussions in Section V.

Stage 1: Initial Bootstrapping. MEBooster first targets improving the copy model's initialization to counteract the performance limitations caused by random initialization in learning-based ME [43]. In the learning-a-model scenario, studies [47, 48, 49] indicate that a copy model initialized close to the victim model is more likely to converge to the ground-truth parameters via (stochastic) gradient descent, compared to a randomly initialized model which may settle at a local optimum. Hence, MEBooster **allocates some query budget to derive a reasonable estimation of the victim model's parameters**, thereby improving initialization.

Specifically, after collecting the victim model's architecture-related information by reconnaissance attacks, the attacker initializes the parameters in the copy model by a query-based method (Section V-C). MEBooster constructs the initial dataset on the victim model (step ①) with a small query budget B_{ini}, and then estimates lower-layer parameters based on the statistical value moment derived from this queried set (step ②). Since the initial norms of weights can affect the convergence of the copy model, they are re-scaled in step ③ [44] (Section V-D).

To further enhance the effectiveness of this parameter estimation, MEBooster over-widens the architecture of the copy model (step ③). In essence, it expands the width in each layer so that the copy model can fully exploit the outcome of the initialization with more neurons.

Stage 2: Learning-based Model Extraction. At this stage, a current learning-based ME [5, 6, 16] is executed. First, the query for the victim model is determined by an active learning (AL) process (step ④), either query-synthesizing-based or pool-based. Once the query budget is used, the attacker retrains the copy model with annotated samples (step ⑤), repeating the process until the query budget is exhausted.

Stage 3: Post Processing with Fine-tuning-boosted Neuron-grained Matching. In the outlined training process, two biases

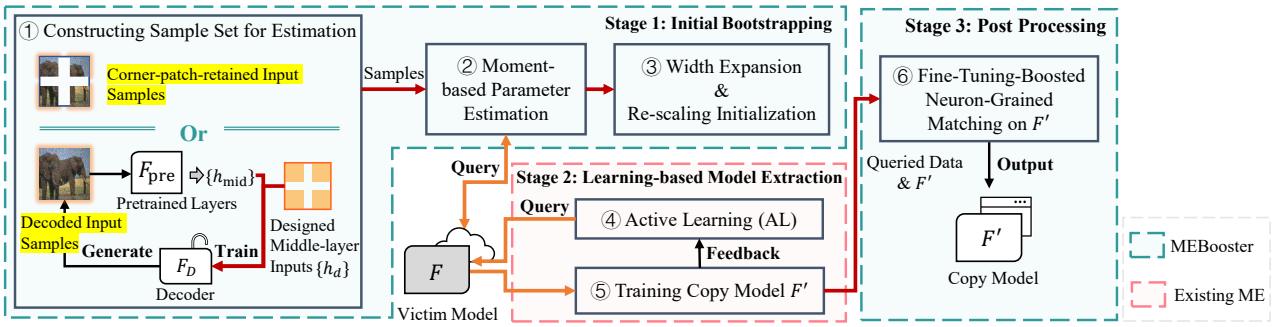


Fig. 1. The MEBooster framework for learning-based ME.

exist. First, earlier queried and trained samples have less impact on the copy model, as they are gradually excluded from further ME. Second, the copy model's training is inadequate during the query generation iterations. To address these issues, a theory on the neuron-grained matching achieved by learning is extended from the foundational lowest-layer neuron matching theory [20, 21]. Supported by it, we propose a post-processing step (step ⑥), fine-tuning (Section V-E), to match more neurons, particularly in upper layers. This step also mitigates the first bias by utilizing all queried samples equally.

V. Neuron-Grained Model Extraction

In this section, we introduce neuron matching theory to elucidate the mechanisms behind MEBooster. We then detail the three modules of MEBooster supported by this theory, which enhance the fidelity of the copy model through neuron-grained matching with the victim model. These modules include moment-based parameter estimation and width expansion during the initial bootstrapping phase, which aims to establish favorable initial parameters and an optimal architecture. Additionally, fine-tuning-boosted neuron-grained matching in the post-processing stage demonstrates how the learning-based method equips the copy model to match the victim's neurons across multiple layers of the neural network.

A. High-level Solution

Studies [20, 21] demonstrate that lower-layer neurons in a copy model can align with those in the victim model via gradient descent (Section V-B), *i.e.*, via "learning-a-model". This alignment indicates that such model extraction does more than just superficially learn the victim model's task; it fundamentally replicates its neurons, achieving neuron matching. However, they realize **neuron matching only at the lowest layer**, *i.e.*, the input layer, which makes them fall short of hi-fi extraction from a complex model. In this study, we aim to achieve closer alignment, *i.e.*, (1) a higher proportion of neuron matching, (2) greater similarity between the copy and victim neurons, and (3) deeper layers of neuron matching. Key to achieving the first two objectives is having closer initial states to the victim model and an over-width architecture. Consequently, we design optimization modules for moment-based parameter estimation (Section V-C) and width expansion

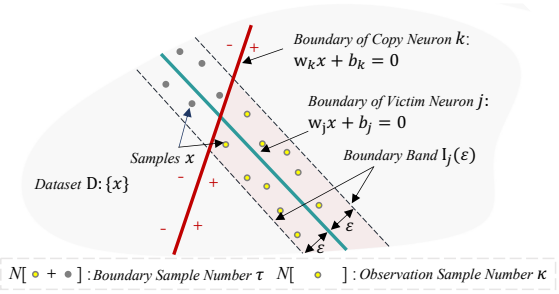


Fig. 2. Illustration of observation sample number and boundary sample number in model extraction.

(Section V-D) during the initial bootstrapping phase. We also introduce a re-scaling initialization approach to combine the advantages of both. Furthermore, we realize the last objective, *i.e.*, extending neuron matching to deeper layers, in the third module, fine-tuning-boosted neuron-grained matching. Combined with the initial bootstrapping, this module increases the neuron matching rate from lower to higher layers, even surpassing the lowest-layer neuron matching expectation outlined in existing theories.

B. Neuron Matching Theory

The phenomenon of neuron-grained matching (*i.e.*, convergence) in the lowest layer during "learning a model" [19, 20, 21] is due to gradient backpropagation. Theorem 5.3 proposes the theoretical conditions essential for achieving the lowest-layer neuron matching. By satisfying these conditions, the copy model could potentially enhance its ability to achieve better neuron matching during gradient descent, ultimately leading to improved fidelity (*i.e.*, similarity). Before delving into the theory of neuron matching, we first give the formal definition of neuron matching and observation sample number, whose symbols are illustrated in Fig. 2.

Definition 5.1: (Neuron Matching) On layer l, neuron j matches neuron k if they satisfy:

$$\langle f_{l,j}(\{x_i\}_{i=1}^b), f_{l,k}(\{x_i\}_{i=1}^b)\rangle \geq 1-\epsilon, \tag{1}$$

when ϵ is sufficiently small. $f_{l,j}(\{x_i\}_{i=1}^b)$ and $f_{l,k}(\{x_i\}_{i=1}^b)$ denote the output vectors of neurons j and k in layer l over the batch $\{x_i\}_{i=1}^b$, respectively.

Definition 5.2: (Observation Sample Number) For $0 < \varepsilon$, if a neuron j and a neuron k satisfies $N[I_j(\varepsilon) \cap E_k] \geq \kappa$

over a dataset D, the neuron j is observed by neuron k in the observation sample number κ, where $N[\cdot]$ calculates the number of samples, $I_j(\varepsilon)$ is the boundary band of neuron j, or formally $\{x \in D|((w_j^T x + b_j)/\|w_j\|) < \varepsilon\}$, and $E_k := \{x \in D|(w_k^T x + b_k) > 0\}$.

Theorem 5.3: (Lowest-layer Neuron Matching, Theorem 5 in [20]). For a victim neuron j, if the lowest-layer neurons in the copy model satisfy: (1) neuron j is observed by a copy neuron k with observation sample number:

$$N[I_j(C\epsilon/\kappa) \cap E_k] \geq \kappa = O(Qd^{3/2}), \tag{2}$$

and (2) the lowest-layer gradient on each sample is sufficiently small, then there will exist a copy model neuron k' matching neuron j.

In the above, Q is the number of the decision boundaries of neurons (tractable for 2-layer networks only [20]), d is the input dimension, and C is a constant.[1]

Theorem 5.3 states that the lowest-layer neuron matching is theoretically guaranteed if the observation sample numbers for all victim neurons satisfy Equation 2. To achieve this, the copy model should possess a sufficient number of observer neurons targeting the victim neurons, in addition to requiring a sufficient number of query samples. Therefore, we optimize the initial positioning of copy neurons through moment-based parameter estimation, bringing them closer to the victim neurons. Additionally, we increase the number of neurons in each layer of the copy model via width expansion. These two optimizations are combined in the re-scaling initialization to enhance the likelihood that copy neurons observe victim neurons. Next, we detail the three methods mentioned above: moment-based parameter estimation, width expansion, and re-scaling initialization.

C. Moment-based Parameter Estimation

To provide better initialization for copy neurons than standard Gaussian random vectors [44], we adopt moment-based parameter estimation to recover the basis vectors of the layer-wise span in deep neural networks (DNNs). However, existing moment-based estimation methods [50, 51] suffer from several critical limitations that hinder their practical applicability: (1) These methods are restricted to scenarios with parameterized input distributions, since they rely on computing the distribution's derivative; (2) They are limited to linear and input (lowest) layers.

In this work, we address each of these issues to enable practical use. First, we replace the derivative computation with score matching estimation (Section V-C2); then, we extend the theory to convolutional layers (Section V-C3) and intermediate layers (Section V-C4), respectively. Next, we begin by introducing the principles of moment-based weight estimation.

1) Moment-based Weight Estimation: The weight matrix of the lowest layer in the deep neural network can be inferred from the moment [22, 51], a statistical expected value of the output distribution's derivative relative to the input distribution.

[1]C is the angle ratio of the weights of two neurons and their output vectors, which is architecture-dependent [20].

We begin by introducing the concept of moments, followed by a derivation of how to estimate weights from them. Let the input distribution's probability density function be $p(x)$, and its score function $S(x)$ is defined as the gradient of the logarithm of $p(x)$ [51]

$$S(x) = \nabla_x log(p(x)) = \frac{\nabla_x p(x)}{p(x)} \in \mathbb{R}^d. \tag{3}$$

The first-order moment M_1 of an ML model F on the input distribution is

$$M_1 = \mathbb{E}(F(x) \otimes S(x)^T) \in \mathbb{R}^{K \times d}, \tag{4}$$

where $\otimes$ is the outer product of two vectors. The i-th row and j-th column element of M_1 is $F(x)_i \times S(x)_j$. According to Stein's Lemma [52], the moment M_1 can be expressed as:

$$M_1 = \mathbb{E}(F(x) \otimes (\nabla_x \log p(x))) = -\mathbb{E}(\nabla_x F(x)). \tag{5}$$

According to Stein's Lemma 5, moments can be expressed as linear mappings between the lowest-layer weights of F. Formally, it's described in Theorem 5.4 as follows.

Theorem 5.4: (Linear Mapping from The First Layer Weight to The Moment. Theorem 1 in [22]) For an MLP model F with first-layer weight matrix $W_1 = [w_1, \ldots, w_n]^T \in \mathbb{R}^{n \times d}$,

$$\begin{aligned} M_1 &= \mathbb{E}(F(x) \otimes \nabla_x log(p(x))) = -\mathbb{E}(\nabla_x F(x)) \\ &= AW_1 = \sum\nolimits_{i \in \{n\}} a_i w_i^T, \end{aligned} \tag{6}$$

where $S(x)$ is the score function of the distribution of x, M_1 is the first-order moment of F and $A = [a_1, \ldots, a_n]$ is a coefficient matrix, $A \in \mathbb{R}^{K \times n}$.

The chain rule indicates that the lowest-layer weight matrix is crucial in the derivative calculation of $F(x)$, making it an inherent factor of M_1.

Theorem 5.4 suggests that the bases of W_1 can be deducted from M_1 via sparse dictionary learning [53, 54] by treating W_1 as the sparse dictionary matrix of M_1, due to the inherent sparse constraint on weight matrices in supervised learning [55]. Then, the bases of W_1 can be utilized to initialize the neurons of the copy model.

2) Estimating Score Function with Score Matching: To estimate W_1, M_1 should be computed first. According to Equation 4, it begins with calculating the score function $S(x)$ w.r.t. the inputs, followed by obtaining M_1 through the expected outer product of $S(x)$ and $F(x)$. However, as defined in Equation 3, $S(x)$ is derived by taking the derivative of probability density $p(x)$, which is challenging since $p(x)$ of most datasets cannot be expressed. To address this issue, we adopt score matching algorithms [23, 56] to approximate the score function $S(x)$ without being aware of $p(x)$.

Specifically, score matching involves training a score model $\Psi(x)$ to best approximate the score of the actual distribution $p(x)$. Given the unknown nature of $p(x)$, an implicit form of score matching, known as the sliced score matching method [23], has been proposed to provide explicit regression targets. The loss for sliced score matching is designed to minimize the discrepancy between the modeled and actual distributions by utilizing efficiently computable sliced statistics, which do not require knowledge of $p(x)$, which is expressed

as follows:

$$\mathcal{L}_{\text{sm}} = \mathbb{E}_{v\in\{v\}}\mathbb{E}_{x\in D}[v\nabla_x^T\Psi(x)v^T + \frac{1}{2}(v^T\Psi(x))^2], \quad (7)$$

where $\{v\}$ is a set of random vectors, and D is the training dataset. The trained score model $\Psi(\cdot)$ learns the distribution characteristic of the training dataset. For a training sample x, its score is the model output $\Psi(x)$.

Algorithm 1 illustrates the overall algorithm of moment-based parameter estimation. First, an input sample set $\{x_i\}_{i=1}^{B_{\text{ini}}}$ (or $\{x\}$ for short) is constructed (Section V-C3 and Section V-C4) to estimate the distribution score of the inputs $\{h_{\text{mid-}i}\}_{i=1}^{B_{\text{ini}}}$ (or $\{h_{\text{mid}}\}$ for short) of the target layer, where B_{ini} is the budget assigned to initial bootstrapping. If the target layer is the first layer, $\{h_{\text{mid}}\}$ denotes the query samples $\{x\}$; otherwise, $\{h_{\text{mid}}\}$ denotes the output matrix of the previous layer. Second, the input samples are queried, and a collection of input-output pairs $D_{\text{ini}} = \{[h_{\text{mid-}i}, F(x_i)]\}_{i=1}^{B_{\text{ini}}}$ is constructed. Then the score function $S(h_{\text{mid}})$ of $\{h_{\text{mid}}\}$ is calculated, followed by the calculation of moment M_1 of D_{ini} by Equation 4. Finally, the bases $\{v_i\}_{i=1}^n$ of W_1 are estimated via sparse dictionary learning, like LISTA [53] and ER-SpUD [54].

Algorithm 1 Moment-Based Parameter Estimation

Input: Intial sub-budget B_{ini}, Victim model $F(\cdot)$
Output: Bases $\{v_i\}_{i=1}^n$
Construct the input sample set $\{x_i\}_{i=1}^{B_{\text{ini}}}$;
Get $D_{\text{ini}} = \{[h_{\text{mid-}i}, F(x_i)]\}_{i=1}^{B_{\text{ini}}}$ via querying $F(\cdot)$;
Train the score model $\Psi(x)$; ▷Equation 7
Compute the score function $S(h_{\text{mid}}) = \Psi(x)$;
Compute moment M_1 via $S(h_{\text{mid}})$ on D_{ini}; ▷Equation 4
/***Sparse Dictionary Learning (ER-SpUD)***/
for $j = 1, \ldots, d$ **do**
    Setup an ML model $F_j^{(s)}$;
    Setup a j-th basis vector e_j;
    Train $F_j^{(s)}$ with loss
    $\mathcal{L} = \|F_j^{(s)}(M_1)^T M_1\|$, s.t. $\|(M_1 e_j)^T F_j(M_1) - 1\| = 0$;
    $s_j = F_j^{(s)}(M_1)^T M_1$;
end for
$S = \{s_j\}_{j=1}^d$;
if $s_j \in S$ has elements smaller than a small number ξ, **then**
    Set those elements 0;
end if
Pick up n columns $\{v_i\}_{i=1}^n$ from S with minimum l_0 norm.

3) Corner-Patch-Retained Sample for Convolutional Layer Generalization: In this section, moment-based parameter estimation is extended from MLP to convolutional neural networks (CNNs). For CNNs with lowest-layer kernels $W_1 = [w_1, \ldots, w_n] \in \mathbb{R}^{n\times c\times k\times k^*}$ and input samples $x \in \mathbb{R}^{c\times m\times m^*}$, Theorem 5.4 no longer holds, where c is the number of input channels, m is sample height, and m^* is sample width. This is because the kernel interacts not with the entire sample but with its various overlapping patches.

To adapt Theorem 5.4 to convolutional layers, we propose a *corner-patch-retained* convolutional kernel estimation method. Specifically, we construct a tailored input distribution with the following properties: (1) only the top-left corner patch $u \in \mathbb{R}^{c\times k\times k^*}$, which aligns with the convolutional kernel size, contains non-zero values; all other elements in the input x are set to zero, as illustrated in Fig. 3; (2) the patch u is sampled from a known distribution (e.g., an independent multivariate Gaussian). Formally, the input distribution is defined as

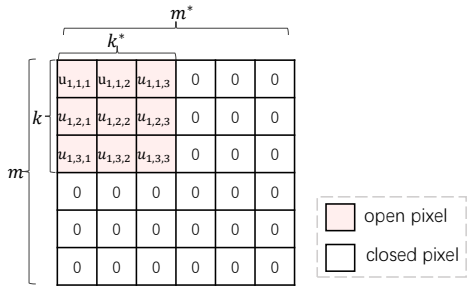


Fig. 3. Illustration of the *corner-patch-retained* input samples, where the channel dimension is set to $c = 1$. The colored region indicates the dimensions with non-zero values.

$$x_{j,s,t} = \begin{cases} u_{j,s,t} & \text{if } 0 \le j < c,\ 0 \le s < k,\ 0 \le t < k^* \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Let $\text{vec}(\cdot)$ denote the vectorization operator that flattens its input. By vectorizing both x and the first-layer kernel weights W_1, the mapping from W_1 to the moment M_1 becomes:

$$\begin{aligned} M_1 &= \mathbb{E}\big(F(x) \otimes \nabla_{\text{vec}(u)} \log p(\text{vec}(u))\big) = -\mathbb{E}\big(\nabla_{\text{vec}(u)} F(x)\big) \\ &= \sum\nolimits_{i=1}^n a_i\, \text{vec}(w_i)^\top + n, \end{aligned} \quad (9)$$

where n is a noise term introduced by the partial overlap (misalignment) between u and the kernels W_1. In the following, we demonstrate that the impact of n on estimating W_1 via Equation 9 is negligible.

The Impact of Kernel Overlap. Although variations in padding, stride, and kernel size may cause overlapping kernel steps on the corner patch, the moment-based weight estimation inherently minimizes the influence of such overlaps, *i.e.*, the noise term n. This is because kernel steps that only partially intersect with the corner patch u effectively contribute trivial elements to the dictionary. Consequently, sparse dictionary learning algorithms such as ER-SpUD [54] remain robust, as they are designed to select the most informative and independent elements for kernel recovery. Empirical evidence from overlapped kernel settings further validates this robustness, as detailed in Section VI-C.

Multiple Corner-patch-retained Samples. To save query budget, patches can be retained in multiple corners in a sample because they interfere with each other the least. For example, in image samples, this approach can quadruple query efficiency.

4) Generalization to Middle Layers by Decoding: Existing moment-based weight estimation methods cannot be applied to middle-layer weights, as the distribution of their input features $\{h_{\text{mid}}\}$ is typically too complex for accurate statistical modeling, resulting in excessive error in the moment score function $\Psi(\cdot)$. Therefore, it is crucial to construct input samples $\{x\}$ that induce suitable intermediate representations $\{h_{\text{mid}}\}$ for reliable moment-based weight estimation.

Decoding Method. We propose a decoder-based approach for

designing input samples, as illustrated in step ① of Fig. 1. This method assumes the lower layers F_{pre} (*e.g.*, embedding layers in text classification) are pretrained. To generate the input samples $\{x\}$ that induce a desired middle-layer distribution, a decoder F_D is trained to map from a target feature set $\{h_d\}$ to the input space. For linear layers, $\{h_d\}$ is drawn from an independent multivariate Gaussian, *i.e.*, $h_d \sim \mathcal{N}(0, \sigma^2 I)$. For convolutional layers, a *corner-patch-retained* structure is used, where only the four corners contain non-zero patches matching the kernel size and sampled from an independent multivariate Gaussian, with all other positions set to zero. This design helps reduce errors in score function approximation. The training process of F_D proceeds as follows. Given a target set $\{h_d\}$, the decoder F_D generates input samples $\{x\}$, which are then passed through the lower layers F_{pre} to produce intermediate features $\{h_{\text{mid}}\}$. F_D is optimized by minimizing the MSE loss between $\{h_d\}$ and $\{h_{\text{mid}}\}$, thereby learning to generate inputs $\{x\}$ that induce the desired middle-layer distribution.

D. Width Expansion and Re-scaling Initialization

Apart from using estimated weight for initialization, expanding the width of the copy model's architecture can also enhance its initial advantage. Over-width [20, 21], which increases the number of neurons per layer by a factor of μ (μ is called the over-width factor), ensures more neurons can be better initialized with a higher probability.

After estimating the initialization parameters and Performing Width Expansion, Re-scaling Initialization combines their benefits. It first re-scales the norms of estimated parameters for compatibility with off-the-shelf initialization algorithms [44], then distributes these parameters across the over-width architecture to initialize segments in parallel.

The norms of the neural network's parameters affect the convergence of the model; thus, in the neural network's default initialization, their neuron weight norms are usually re-scaled [44]. To retain the advantage from these norms, we re-scaled the bases $\{v_i\}_{i=1}^{n}$ of estimated W_1 when initializing the copy model. For example, for ReLU activated layers, the estimated parameters are re-scaled by the HE initialization method [44]:

$$w_i = \frac{v_i}{\|v_i\|}\sqrt{\frac{2}{d}}, i \in \{n\}, \tag{10}$$

where $d = c \times m \times m^*$ for the convolutional layer.

E. Fine-tuning-boosted Neuron-grained Matching

To match more neurons across multiple layers, we extend Theorem 5.3 to exploit neuron matching in upper layers achieved by the gradient descent in Theorem 5.5. It outlines the optimal states for any given layer, l.

Theorem 5.5: (Neuron Matching in Upper Layers) If all neurons in layer $l-1$ ($l > 1$) of the victim model are matched by at least one neuron in the copy model, then the input matrix to layer l over a batch of samples $\{x_i\}_{i=1}^{b}$ (abbreviated as $\{x\}$) in the copy model, denoted by $\mathbf{f}'_{l-1}(\{x\})$, is equivalent to the corresponding input matrix $\mathbf{f}_{l-1}(\{x\})$ in the victim model, *i.e.*, it satisfies:

$$\mathbf{f}_{l-1}(\{x\}) = A_{l-1}\mathbf{f}'_{l-1}(\{x\}), \forall\{x\},$$

where A_{l-1} is the transformation matrix and independent on $\{x\}$. For the victim's neuron j in layer l, if the neurons in copy model satisfy: (1) neuron j is observed by a copy neuron k with the observed sample number larger than $O((\exp(L-l))^{5/2} \times m_{l-1}^{3/2})$, and (2) the gradient in layer l on each sample is sufficiently small, there will exist a copy neuron k' matching neuron j by learning.

Proof. As Definition 5.1 describes, for two neurons at layer l with different architectures, we define the neuron matching between them as follows: a victim neuron j is matched by a copy neuron k, if for any batch of samples $\{x\}$, they satisfy:

$$\langle f_{l,j}(\{x\}), f'_{l,k}(\{x\})\rangle \leq (1-\epsilon), \tag{11}$$

where ε is sufficiently small. $\mathbf{f}_{l,j}(\{x\}) \in \mathbb{R}^{1\times b}$ and $\mathbf{f}'_{l,k}(\{x\}) \in \mathbb{R}^{1\times b}$, where b is the batch size of $\{x\}$.

Assume in layer $l-1$, all victim neurons are matched by one or more copy neurons, *i.e.*, for each victim neuron j, there exists at least one copy neuron k,

$$\langle f_{l-1,j}(\{x\}), f'_{l-1,k}(\{x\})\rangle \leq (1-\epsilon), \tag{12}$$

that is,

$$f'_{l-1,k}(\{x\}) \approx a_{kj} f_{l-1,j}(\{x\}), \tag{13}$$

where a_{kj} is a constant. As a result, the output matrix of layer $l-1$ on a batch of samples can be expressed as:

$$\begin{aligned}\mathbf{f}'_{l-1}(\{x\}) &= [f'_{l-1,1}(\{x\}), \ldots, f'_{l-1,n_{l-1}}(\{x\})]^T \\ &\approx \begin{bmatrix} 0 & \ldots & a_{1j_1} & \ldots & 0 \\ & & 0 & & \\ 0 & \ldots & a_{n_l j_{m_l}} & \ldots & 0 \end{bmatrix} [f_{l-1,1}(\{x\}), \ldots]^T \\ &= A_{l-1}\mathbf{f}_{l-1}(\{x\}),\end{aligned} \tag{14}$$

where A_{l-1} is constant for all input samples. As a result, the output matrix of layer l in the copy model and in the victim model would be:

$$\mathbf{f}'_l(\{x\}) = D'_l(\{x\})W_l'^T\mathbf{f}'_{l-1}(\{x\}) = D'_l(\{x\})W_l'^T A_{l-1}\mathbf{f}_{l-1}(\{x\}), \tag{15}$$

and

$$\mathbf{f}_l(\{x\}) = D_l(\{x\})W_l^T\mathbf{f}_{l-1}(\{x\}). \tag{16}$$

Thus, with a linear transformation, W'_l is mapped into the space of W_l. For $W_l'^T A_{l-1}$, the neuron matching situation is akin to the lowest layer discussed in Theorem 5.3. □

Theorem 5.5 implies that if a model's lower layers are well-matched neuron-wisely, and each of the victim's neurons in the upper layers is observed by the neurons of the copy model with a sufficient number of observation samples, fine-tuning until the backpropagated gradients are minimal enough will make the copy model gradually match the victim's upper layers as well. The overall query complexity can be explained as $O(\exp(\frac{5L}{2}) + \exp((\frac{5(L-1)}{2}) + \cdots + \exp(\frac{5}{2})) = \exp(O(L)))$. Therefore, in post processing, we supplement the overlooked learning process, especially for iterative query sample generation frameworks [5, 6], with fine-tuning.

VI. Experiments

We evaluate MEBooster and its variants on various state-of-the-art learning-based model extraction (ME) attacks. All

TABLE II
EXPERIMENTAL SETTINGS

Dataset / Model / Acc. (%)	Total Budget	ActiveThief Pool / Size	Initial Budget
MNIST / LeNet-5 / 99.17	1M	EMNIST/10K	1K
FMNIST / LeNet-5 / 89.88	10M	E, KMNIST /100K	1K
SVHN / ModelArt / 94.30	10M	ImageNet32 / 150K	3K
CIFAR10 / Resnet18 / 90.13	50M	ImageNet32 / 50K	20K
NEWS / DPCNN* / 84.80	20M	Dbpedia / 200K	40K
IMDB / DPCNN / 72.47	20M	Dbpedia / 100K	40K

* DPCNNs are constructed with pre-trained embeddings.

codes are open-source and available at https://github.com/mebooster/mebooster.

A. Setup

1) Baseline Attacks: We implemented three advanced learning-based ME attacks as baselines: two query-synthesizing-based (*i.e.*, data-free) MEs (DFME [6] and MAZE [5]) and one pool-based ME (ActiveThief [16]).

2) Query Budget & Datasets & Models: We evaluate the performance of MEBooster on six benchmark datasets encompassing both local and black-box MLaaS models, and ranging from images to texts: LeNet-5 [57] on MNIST [58], LeNet-5 on FMNIST [59], ModelArt [60] on SVHN [61], Resnet18 on CIFAR10 [18], DPCNN [62] on AG'S NEWS [63] (abbreviated as NEWS), and DPCNN [62] on IMDB [64].

The total query budgets are consistent for DFME and MAZE regardless of the initial bootstrapping phase, to ensure comparability of query budgets across different attacks. For ActiveThief, we utilize all available real-life data in its adversarial pool for all experiments and allocate additional sub-budgets for attacks involving initial bootstrapping. Table II summarizes experimental settings, where columns *Dataset* and *Model/Acc.* show the general information of victim models, and column *Initial Budget* shows the sub-budget for the initial bootstrapping.

Moreover, *Baseline* maintains the same architecture as the victim model, in line with the original implementations [6, 16, 25]. For black-box ModelArts [60], ResNet50 is used as per the official Codelabs documentation [65].

3) Attack Frameworks: To evaluate each component's impact in MEBooster, we build five training framework variants: *Baseline*, *WE only*, *RI only*, *MEBooster w/o FT*, and *MEBooster*. *WE only* uses the width expansion of MEBooster; *RI only* uses the rescaling initialization with estimated parameters, and *w/o FT* uses the entire MEBooster except for the post-processing.

B. Training Parameters

In our baseline, the copy model mirrors the victim model's structure, aligning with the prevailing view that this configuration is optimal. This setting is in line with the original implementations, where ActiveThief [16] employs a copy model identical to the victim's, and DFME [6] and MAZE [5] use models from the ResNet family. Other training parameters, like learning rate and optimizer, follow those specified in the original studies. For width expansion, the over-width factor is set to 5. All experimental results are the average measures of 5 trials.

1) Evaluation Metrics: We measure the effectiveness of MEBooster using fidelity and accuracy [43]. Query-based parameter estimation is evaluated by relative Initial Error Reduction (IER), which measures the error reduction of the estimated weight matrix $\{v_j\}_{j=1}^n$ compared to Gaussian random vectors. A higher IER means the estimated parameters are closer to the target parameters. Its formal definition is as follows. Their formal definitions are as follows.

a) Fidelity: Fidelity is measured by the proportion of similarity between the outputs of two models on the evaluation dataset D_t. Formally, fidelity $= \Pr_{x\in D_t}[\text{argmax}(F(x)) = \text{argmax}(F'(x))]$.

b) Accuracy: It refers to the test accuracy of the copy model F' on the evaluation dataset D_t. Formally, accuracy $= \Pr_{(x,y)\in D_t}[\text{argmax}(F'(x)) = y]$.

c) Initial Error Reduction (IER): IER measures the relative improvement in initial bootstrapping by comparing the estimated weights $\{v_j\}_{j=1}^n$ against Gaussian random weights $\{r_j\}_{j=1}^n$, with respect to their distance from the target victim layer's weights $W = [w_1, \ldots, w_n]$. The distance is computed using the ℓ_2 norm. A positive IER indicates that the estimated weights $\{v_j\}_{j=1}^n$ are closer to the ground truth W than the random initialization $\{r_j\}_{j=1}^n$. Formally,

$$\text{IER} = \frac{\sum_{i=1}^n \min_j \| \frac{w_i}{\|w_i\|} - \frac{r_j}{\|r_j\|} \| - \sum_{i=1}^n \min_j \| \frac{w_j}{\|w_j\|} - \frac{v_j}{\|v_j\|} \|}{\sum_{i=1}^n \min_j \| \frac{w_i}{\|w_i\|} - \frac{r_j}{\|r_j\|} \|}. \quad (17)$$

C. Overall Performance of MEBooster

Overall Results. Table III compares the performance of MEBooster's variants against baselines, and Table V displays query-based parameter estimation results. Table IV reports the computing costs. Bold highlights superior results and underlines signify major improvements. Overall, each of the key components of MEBooster significantly improves the fidelity of *Baseline* learning-based ME across different domains under the same query budget by up to 58.10%.

Effectiveness of Initial Bootstrapping. Initial bootstrapping leads to notable fidelity improvements across various ME attacks, driven by both width expansion and re-scaling initialization. To demonstrate the inherent advantage of the initial bootstrapping module, Table V reports the IER of its query-based parameter estimation, which achieves at least 1.06%. Since IER$>$ 0 implies improved similarity to the actual weights compared to existing initial methods, the results confirm its effectiveness. The benefit is more pronounced when the neuron dimensionality is smaller, likely because estimating parameters in high-dimensional space (*e.g.*, through score functions) introduces greater error due to distribution fitting challenges. This observation is consistent with the fidelity gains observed from re-scaling initialization (RI). Moreover, width expansion proves particularly beneficial in complex feature spaces. For example, while the *WE only* setting offers modest improvement over *Baseline* in MNIST, it achieves at least a 6.43% fidelity gain in FMNIST.

Effectiveness of Fine-tuning (FT). Comparing *MEBooster* with *MEBooster w/o FT*, we observe that in the post process,

TABLE III
RESULTS OF LEARNING-BASED MODEL EXTRACTION EXPERIMENTS

Dataset	ME Attacks	Fidelity % /(Accuracy / %)				
		Baseline	WE only	RI only	w/o FT	MEBooster
MNIST[57] / LeNet-5[58]	DFME	96.69 (96.36)	99.49 (99.05)	97.78 (97.47)	99.59 (99.12)	**99.60 (99.10)**
	MAZE	97.57 (97.37)	98.91 (98.61)	98.42 (98.21)	99.06 (98.67)	**99.33 (98.84)**
	ActiveThief	97.99 (97.78)	98.59 (98.13)	98.00 (97.75)	98.60 (98.21)	**98.71 (98.42)**
FMNIST[59] / LeNet-5	DFME	58.63 (57.68)	92.66 (87.75)	76.42 (73.70)	93.79 (88.33)	**94.32 (88.44)**
	MAZE	71.47 (70.21)	92.53 (87.69)	78.02 (75.65)	93.70 (88.21)	**96.50 (89.50)**
	ActiveThief	78.07 (75.37)	84.50 (81.12)	78.84 (76.08)	84.38 (80.91)	**86.28 (82.80)**
SVHN[61] / ModelArt[60]	DFME	90.14 (91.36)	92.32 (93.09)	91.82 (92.18)	94.05 (93.66)	**96.01 (94.27)**
	MAZE	90.32 (90.22)	92.89 (91.69)	92.44 (91.62)	93.46 (92.90)	**95.19 (94.16)**
	ActiveThief	90.23 (69.31)	91.43 (90.74)	92.11 (91.92)	92.74 (92.04)	**93.54 (93.15)**
CIFAR10[18] / Resnet18[17]	DFME	91.35 (87.18)	97.92 (90.18)	91.72 (87.34)	98.14 (90.18)	**98.99 (90.13)**
	MAZE	68.82 (67.54)	71.58 (70.56)	70.84 (69.35)	75.20 (73.49)	**82.54 (80.68)**
	ActiveThief	83.54 (82.13)	87.34 (86.29)	84.07 (82.52)	87.35 (85.94)	**87.86 (86.44)**
AG'S NEWS[63] / DPCNN[62]	DFME	83.04 (75.83)	94.60 (82.63)	90.30 (80.47)	98.13 (83.63)	**98.52 (83.63)**
	MAZE	34.83 (32.07)	88.58 (79.69)	85.62 (76.93)	92.88 (82.39)	**92.93 (82.44)**
	ActiveThief	68.82 (64.72)	74.46 (71.64)	69.41 (64.96)	74.70 (71.88)	**76.12 (73.35)**
IMDB[64] / DPCNN	DFME	92.50 (67.08)	93.06 (67.51)	93.53 (67.22)	95.09 (67.68)	**95.23 (67.62)**
	MAZE	78.72 (60.91)	87.25 (65.03)	81.82 (62.23)	91.90 (66.91)	**91.91 (67.00)**
	ActiveThief	84.80 (64.84)	85.85 (64.37)	85.60 (64.43)	86.30 (64.40)	**86.89 (64.47)**

the fine-tuning further improves up to 7.34% fidelity with additional 9.8% computing cost.

Evaluation of Neuron Matching. We evaluate layer-wise neuron matching under various settings on FMNIST and CIFAR10, focusing on the ratio of matched victim neurons. The neuron matching ratio ρ_{nm} at layer l is defined as the proportion of matched victim neurons:

$$\rho_{\text{nm}} = \frac{\sum_{j=1}^{n_l} \mathbf{1}(\max_{k\in\{m_l\}} \langle f_{l,j}(\{x\}), f'_{l,k}(\{x\})\rangle \geq 1-\epsilon)}{n_l} \times 100\%, \tag{18}$$

where n_l and m_l denote the number of neurons in layer l of the victim and copy models, respectively. $f_{l,j}$ denotes the output of the j-th neuron in layer l of the victim model, and $f'_{l,k}$ denotes the output of the k-th neuron in layer l of the copy model, and $\mathbf{1}$ is the indicator function. In the experiments, the batch size b is set to 512, and ϵ is set to 0.05 and 0.01, corresponding to matching scores of 0.95 and 0.99, respectively.

Results in Fig. 4 reveal that each MEBooster component improves neuron matching across layers, highlighting the ability of learning-based model extraction (ME) to closely approximate the victim model. Notably, DFME and MAZE, with their extensive synthetic queries, significantly exceed ActiveThief in neuron matching ratios, demonstrating they closely resemble the victim model in both the **target task** and **overall behavior**. This is in line with the near-perfect transferability in adversarial attacks using MEBooster-DFME/MAZE models, further discussed in Section VI-D. This progress indicates that combined with data-free ME, MEBooster advances learning-based ME into a new high-fidelity era, from neuron-level to models, overturning previous biases against the efficacy of learning in high-fidelity ME [43].

Computational Cost Analysis. Table IV reveals a clear correlation between computational cost investment and attack performance. MEBooster achieves the highest extraction quality under a fixed query budget, with the largest cost. To account for scenarios with limited computational resources, we provide a lighter variant, *w/o FT*, which significantly improves performance with only a moderate cost increase, typically $1.5\times$ to $5\times$. This option allows attackers to adjust their strategy based on available resources and desired attack results.

TABLE IV
COMPUTING COST OF LEARNING-BASED MODEL EXTRACTION EXPERIMENTS (IN SCIENTIFIC NOTATION OF SECONDS)

Dataset	ME Attacks	Computing Cost / Seconds (Sci Notation)				
		Baseline	WE only	RI only	w/o FT	MEBooster
MNIST	DFME	3.32e2	3.55e2	4.62e2	4.85e2	4.67e3
	MAZE	1.78e2	1.84e2	3.08e2	3.14e2	5.93e3
	ActiveThief	1.12e3	1.98e3	1.25e3	2.11e3	3.31e3
FMNIST	DFME	7.13e3	7.45e3	7.29e3	7.60e3	2.18e4
	MAZE	3.57e3	3.70e3	3.72e3	3.84e3	1.96e4
	ActiveThief	1.24e4	1.92e4	1.25e4	1.93e4	2.52e4
SVHN	DFME	3.39e5	3.50e5	3.40e5	3.51e5	4.09e5
	MAZE	3.38e5	3.43e5	3.38e5	3.44e5	4.22e5
	ActiveThief	7.24e4	7.64e4	7.26e4	7.67e4	8.96e4
CIFAR10	DFME	1.11e4	4.55e4	1.58e4	5.03e4	1.56e5
	MAZE	9.64e3	3.84e4	1.46e4	4.32e4	1.30e5
	ActiveThief	4.07e4	6.94e4	4.55e4	7.42e4	8.10e4
NEWS	DFME	3.67e2	1.12e3	6.72e3	7.47e3	3.73e4
	MAZE	3.70e2	8.80e2	6.72e3	7.24e3	4.23e4
	ActiveThief	2.69e3	2.69e3	9.05e3	9.05e3	1.48e4
IMDB	DFME	4.20e2	1.19e3	7.49e3	8.27e3	3.73e4
	MAZE	2.71e2	7.91e2	7.35e3	8.47e3	4.23e4
	ActiveThief	3.18e3	3.24e3	1.03e4	1.03e4	2.16e4

TABLE V
THE RESULTS OF PARAMETER ESTIMATION METHODS

Model	#Neuron	Dimension	Dataset	IER / %
LeNet5	6	25	MNIST	16.43
			FMNIST	16.28
ModelArt	–	–	SVHN	–
Resnet18	16	27	CIFAR10	6.03
DPCNN	100	750	AG'S NEWS	1.06
			IMDB	1.89

D. Impact of MEBooster on Follow-up Attacks

To explore the significance of the fidelity gain sustained by MEBooster, we conduct experiments on downstream attacks using copy models, including black-box adversarial attacks [12, 66] and membership inference (MI) attacks [67]. For adversarial attacks, we evaluated the transferability of the

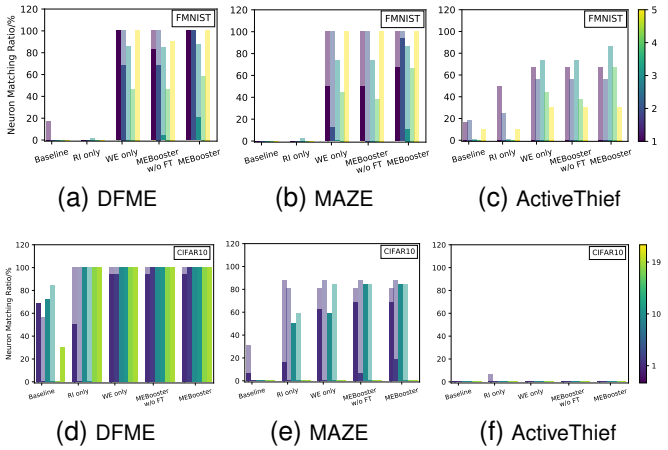


Fig. 4. Neuron matching ratio. The color bar integer is the index of layers. Low-opacity bars reflect matching scores above 0.95, while high-opacity bars reflect scores over 0.99.

TABLE VI
THE RESULTS OF FOLLOW-UP ADVERSARIAL ATTACKS

Dataset	Attacks	DFME	MAZE	ActiveThief
		ASR (Transferability) / %		
FMNIST	**Baseline**	5.64(18.64)	6.00(16.67)	8.74(24.09)
	MEBooster	**38.70(96.99)**	**39.55(98.82)**	**18.62(52.30)**
CIFAR10	**Baseline**	67.47(85.40)	38.61(69.72)	29.57(39.35)
	MEBooster	**81.02(99.72)**	**58.38(96.00)**	**34.95(47.20)**

adversarial samples created on copy models using FGSM [12] (with ϵ values of 0.1 for FMNIST and 0.03 for CIFAR-10) to the victim model. For MI attacks, we attack the copy models via unsupervised MI attack [67]. Tables VI and VII report the follow-up attacking performance, showing that a higher fidelity of copy models leads to higher downstream attack performance. This indicates that besides replicating the victim model's functionality, copy models further leak the membership privacy of the victim model's training data and its decision boundaries, making it more vulnerable to adversarial attacks.

E. Impact of Width Expansion

In the initial bootstrapping, we introduced an over-width factor for the architecture design. To explore the effect of this parameter on MEBooster, we report the impact of the over-width factor on MEBooster in FMNIST and CIFAR10 experiments in Fig. 5. We observe that in various model extraction attacks, moderate width expansion can exhibit distinctive advantages.

TABLE VII
THE RESULTS OF FOLLOW-UP MEMBERSHIP INFERENCE ATTACKS

Dataset	Attacks	DFME	MAZE	ActiveThief
		MI Accuracy, F1 Score / %		
FMNIST	**Baseline**	50.41, 51.50	51.42, 53.45	50.24, 51.19
	MEBooster	**50.67, 52.03**	**51.66, 53.81**	**50.66, 52.36**
CIFAR10	**Baseline**	71.75, 72.02	61.87, 48.14	68.00, 59.49
	MEBooster	**81.73, 79.84**	**69.70, 61.97**	**78.70, 63.65**

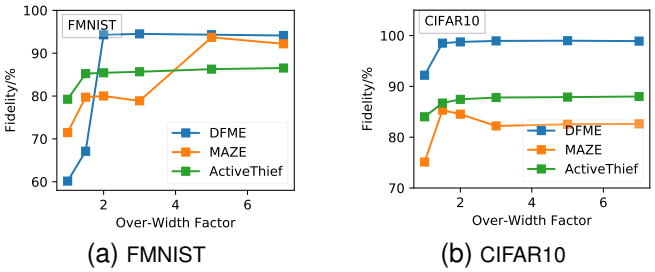


Fig. 5. The impact of the over-width factor of width expansion on MEBooster.

TABLE VIII
RESULTS OF OPTIMIZATION METHODS WITH SIMILAR MEMORIES

Method	Fidelity (Accuracy) / %			Memory
	DFME	MAZE	ActiveThief	
FMNIST				
Width Expansion	**92.66 (87.75)**	**92.53 (87.69)**	**84.50 (81.12)**	0.06MB
Deep-LeNet	67.61 (58.80)	72.71 (70.73)	78.61 (76.14)	0.08MB
Ensemble	60.12 (59.18)	72.99 (71.69)	79.61 (76.87)	0.06MB
CIFAR10				
Width Expansion	**97.83 (90.12)**	**71.49 (70.51)**	**87.28 (86.21)**	5.77MB
Resnet50	82.80 (80.52)	69.18 (67.53)	80.80 (80.80)	7.94MB
Ensemble	92.15 (83.32)	69.93 (68.82)	85.72 (84.13)	5.77MB

F. Comparison of Width Expansion and Other Optimization Methods

We further investigate the superiority of width expansion by comparing it with two other standard optimization methods using similar memories. The first adopts a more complex architecture for the copy model, *e.g.*, Resnet50 [17] to steal a Resnet18 victim model. For the LeNet5 victim model, the copy model utilizes a deeper CNN with two additional convolutional layers appended to LeNet-5, which have 5×5 kernels and widths of 32 and 16, respectively. The second ensembles a set of models trained separately [68].

The implementation details are as follows. In step ③ of MEBooster (see Fig. 1), width expansion is replaced with the above two methods while other steps are retained. To ensure they all consume similar memory, for the width expansion experiment, the over-width factor is set to 3; for the ensemble experiment, three models are employed. Table VIII shows the attack performance of these methods against width expansion for FMNIST and CIFAR10. We observe that width expansion consistently performs the best, thanks to its architectural advantage rather than its high memory usage.

G. The Impact of Architecture Knowledge

We investigate how MEBooster behaves when the attacker adopts a mismatched architecture, which usually happens in proprietary ML systems [69]. Specifically, for CIFAR10 and FMNIST victim models in Table II, we set copy models as Resnet-24 [17] and 5-layer Pytorch CNN [70] respectively. Table IX reports the effectiveness of MEBooster, where it still brings significant gains to all attacks by up to 30.06% fidelity improvement.

VII. DEFENSE AGAINST LEARNING-BASED ME

Turning our attention to defense, we explore tuning the properties of the victim model's parameters to defend against

TABLE IX
ARCHITECTURE-AGNOSTIC MODEL EXTRACTION ATTACKS

Dataset	ME Attacks	Fidelity (Accuracy) / %				
		Baseline	WE only	RI only	w/o FT	MEBooster
FMNIST	DFME	51.32 (50.22)	76.77 (74.33)	53.46 (52.13)	82.08 (78.97)	**82.38 (79.54)**
	MAZE	68.99 (67.26)	85.43 (82.00)	73.25 (70.84)	86.09 (82.44)	**87.41 (83.81)**
	ActiveThief	61.04 (58.81)	79.20 (76.41)	65.16 (63.07)	80.54 (77.83)	**81.31 (78.34)**
CIFAR10	DFME	91.81 (87.42)	98.03 (90.23)	92.26 (87.88)	98.31 (90.21)	**98.82 (90.93)**
	MAZE	69.22 (67.43)	75.12 (73.68)	76.45 (74.66)	76.72 (75.25)	**77.62 (76.74)**
	ActiveThief	83.71 (81.88)	87.45 (85.78)	83.67 (82.26)	87.67 (86.27)	**88.32 (86.91)**

learning-based model extraction. Previous methods such as BDPL [27], Adaptive Misinformation [25], and GRAD^2 [24] protect victim models by perturbing prediction outputs. However, their effectiveness diminishes against data-free model extraction [6, 34] attacks, where the adversary can explore an output space much larger than that induced by real-life data. To address the limitations of this line of work, we propose a novel defense paradigm, namely **model modification**. This method steers model properties [20, 48] during training to enhance its resistance to learning-based model extraction.

A. Defense Strategy: Stochastic Norm Enlargement

We use the L2 norm of the victim model's weight matrices as the critical property. Zhang *et al*. [48] showed that the complexity of learning to recover a neural network is polynomially related to λ, corresponding to the maximum singular value, *i.e.*, l2 norm, of each layer's weight matrix.

We introduce the Stochastic Norm Enlargement (SNE) defense, which guides weight matrices in each layer towards larger L2 norms during training by adding a regularization term to the loss. To prevent training crashes, z layers are stochastically chosen to be incorporated into the loss at each epoch, as described in Equation 19.

$$loss = L(\mathbf{F}(x), y) + \frac{\varphi}{\sum_i^z \|W_i\|}, \tag{19}$$

where $L(\cdot)$ denotes the original loss function (*e.g.*, cross-entropy loss), and φ is the norm regularization factor.

B. Empirical Evaluation

We compare SNE with two state-of-the-art defensive strategies, namely GRAD^2 [24] and adaptive misinformation [25], both with a perturbation l_1 distance of 0.5. In SNE, we set the factor φ to 5 and z to 5. Table X reports the defense performance against learning-based ME frameworks *Baseline* and *MEBooster*, with the lowest attack fidelity bolded.

We observe that victim models with SNE defense exhibit remarkably low extractability at a price of slightly lower model accuracy. Against these SNE-defended models, all model extraction attacks, particularly DFME and MAZE, show marked degradation. We speculate that the reason is that the sample complexity theory about model recovery arises from synthetic training data [48]. On the other hand, as observed in Section VI-C, ActiveThief, utilizing real-life data, learns tasks rather than models, offering better resistance to SNE defense. Additionally, compared to strategies like GRAD^2 and adaptive misinformation with higher perturbation distance ($\epsilon = 0.5$), SNE is more effective in most attacks, reducing fidelity by up to 64.81% in the DFME attack on FMNIST models, versus a maximum of 32.29% for its counterparts.

TABLE X
THE RESULTS OF DEFENDING METHODS AGAINST LEARNING-BASED MODEL EXTRACTION

Attacks	Fidelity / % (Accuracy / %)			
	No Defence	SNE (Ours)	GRAD^2	Adap Mis*
FMNIST (Baseline)				
ΔAccuracy/%	–	-1.48	-1.71	-1.68
DFME	58.63 (57.68)	**16.10 (14.94)**	48.71 (47.62)	51.52 (51.02)
MAZE	71.47 (70.21)	**58.36 (57.83)**	68.50 (67.93)	70.62 (69.53)
ActiveThief	78.07 (75.37)	**65.92 (64.37)**	78.13 (75.42)	78.09 (75.39)
FMNIST (MEBooster)				
DFME	94.32 (88.44)	**29.51 (28.31)**	62.03 (61.27)	88.55 (85.26)
MAZE	96.50 (89.50)	**72.83 (71.74)**	89.96 (84.31)	90.18 (85.57)
ActiveThief	86.28 (82.80)	**72.89 (71.76)**	79.63 (76.15)	84.46 (82.75)
CIFAR10 (Baseline)				
ΔAccuracy/%	–	-1.78	-1.03	-1.13
DFME	91.35 (87.18)	**79.52 (78.37)**	87.13 (96.17)	89.68 (87.92)
MAZE	68.82 (67.54)	**61.02 (60.01)**	62.51 (61.42)	64.00 (62.95)
ActiveThief	83.54 (82.13)	83.39 (82.06)	81.21 (80.63)	**79.16 (78.04)**
CIFAR10 (MEBooster)				
DFME	98.99 (90.13)	**88.05 (87.73)**	95.55 (89.46)	94.00 (88.41)
MAZE	82.54 (80.68)	**65.04 (63.83)**	77.51 (76.47)	78.19 (77.03)
ActiveThief	87.86 (86.44)	85.95 (83.61)	**84.37 (82.64)**	85.96 (83.63)

* "Adap Mis" is short for "Adaptive Misinformation".

VIII. CONCLUSION

This paper pioneers the exploration of neuron-grained model extraction by boosting the **training** process. Through initial bootstrapping (including width expansion and rescaling initialization) and post-processing fine-tuning, the proposed MEBooster framework can achieve a fidelity gain of up to 58.10%. Notably, MEBooster crossover data-free ME reaches remarkable similarity from the neuron to the model level, ushering learning-based ME into a new era for challenging high-fidelity ME. Furthermore, we introduce a novel defense training strategy, Stochastic Norm Enlargement (SNE), leveraging the hard-to-extract properties of the victim model. Both MEBooster and SNE are extensively evaluated in real-life datasets and state-of-the-art models under various ME attacks. Future research will delve into various learning strategies to enhance high-fidelity model extraction.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No: 92270123 and 62372122), the Research Grants Council, Hong Kong SAR, China (Grant No: 15226221, 15225921, and 15208923), and the Innovation and Technology Fund (Grant No: ITS-140-23FP).

REFERENCES

[1] "Microsoft azure - machine learning," https://azure.microsoft.com/services/machine-learning, accessed: 2025-06-02.

[2] "Amazon aws marketplace - pre-trained machine learning models," https://aws.amazon.com/marketplace/solutions/machine-learning/pre-trained-models, accessed: 2025-06-02.

[3] "Vertex ai - google cloud," https://cloud.google.com/vertex-ai, accessed: 2025-06-02.

[4] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, 2016.

[5] S. Kariyappa, A. Prakash, and M. K. Qureshi, "Maze: Data-free model stealing attack using zeroth-order gradient estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[6] J. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[7] S. Vinterbo, "Privacy: a machine learning view," *IEEE Transactions on Knowledge and Data Engineering*, 2004.

[8] P. Zhang, X. Fang, Z. Zhang, X. Fang, Y. Liu, and J. Zhang, "Horizontal multi-party data publishing via discriminator regularization and adaptive noise under differential privacy," *Inf. Fusion*, 2025. [Online]. Available: https://doi.org/10.1016/j.inffus.2025.103046

[9] J. Feng, Y. Wu, H. Sun, S. Zhang, and D. Liu, "Panther: Practical secure two-party neural network inference," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, 2025. [Online]. Available: https://doi.org/10.1109/TIFS.2025.3526063

[10] G. Xu, S. Xu, J. Ning, T. Zhang, X. Huang, H. Li, and R. Lu, "New secure sparse inner product with applications to machine learning," *CoRR*, vol. abs/2210.08421, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.08421

[11] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[13] J. Wu and J. He, "A unified framework for adversarial attacks on multi-source domain adaptation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, 2022.

[14] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[15] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Exploring connections between active learning and model extraction," in *29th USENIX Security Symposium*, 2020.

[16] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy, "Activethief: Model extraction using active learning and unannotated public data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[19] D. Saad and S. A. Solla, "Dynamics of on-line gradient descent learning for multilayer neural networks," in *Advances in Neural Information Processing Systems 8 (NeurIPS 1995)*, 1995.

[20] Y. Tian, "Student specialization in deep rectified networks with finite width and input dimension," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020.

[21] M. Zhou, R. Ge, and C. Jin, "A local convergence theory for mildly over-parameterized two-layer neural network," in *Conference on Learning Theory*. PMLR, 2021.

[22] H. Sedghi and A. Anandkumar, "Provable methods for training neural networks with sparse connectivity," *arXiv preprint arXiv:1412.2693*, 2015.

[23] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced score matching: A scalable approach to density and score estimation," in *Proceedings of The 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, ser. Proceedings of Machine Learning Research, vol. 115. PMLR, 2020.

[24] M. Mazeika, B. Li, and D. Forsyth, "How to steer your adversary: Targeted and efficient model stealing defenses with gradient redirection," in *International Conference on Machine Learning*. PMLR, 2022.

[25] S. Kariyappa and M. K. Qureshi, "Defending against model stealing attacks with adaptive misinformation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[26] M. Tang, A. Dai, L. DiValentin, A. Ding, A. Hass, N. Z. Gong, and Y. Chen, "Modelguard: Information-theoretic defense against model extraction attacks," in *33rd USENIX Security Symposium*, 2023.

[27] H. Zheng, Q. Ye, H. Hu, C. Fang, and J. Shi, "Bdpl: A boundary differentially private layer against machine learning model extraction attacks," in *European Symposium on Research in Computer Security*. Springer, 2019.

[28] J. Beetham, N. Kardan, A. S. Mian, and M. Shah, "Dual student networks for data-free model stealing," 2023.

[29] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM*

on Asia Conference on Computer and Communications Security, 2017.

[30] J. Liang, R. Pang, C. Li, and T. Wang, "Model extraction attacks revisited," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. ACM, 2024.

[31] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "Prada: Protecting against dnn model stealing attacks," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019.

[32] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[33] Y. Xiao, Q. Ye, H. Hu, H. Zheng, C. Fang, and J. Shi, "Mexmi: Pool-based active model extraction crossover membership inference," in *Advances in Neural Information Processing Systems*, 2022.

[34] J. Rosenthal, E. Enouen, H. V. Pham, and L. Tan, "Disguide: Disagreement-guided data-free model extraction," in *Thirty-Seventh AAAI Conference on Artificial Intelligence*, B. Williams, Y. Chen, and J. Neville, Eds. AAAI Press, 2023.

[35] A. M. Sadeghzadeh, A. M. Sobhanian, F. Dehghan, and R. Jalili, "HODA: hardness-oriented detection of model extraction attacks," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, 2024.

[36] S. Pal, Y. Gupta, A. Kanade, and S. K. Shevade, "Stateful detection of model extraction attacks," *CoRR*, vol. abs/2107.05166, 2021.

[37] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[38] P. Lv, H. Ma, K. Chen, J. Zhou, S. Zhang, R. Liang, S. Zhu, P. Li, and Y. Zhang, "Mea-defender: A robust watermark against model extraction attack," in *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024.

[39] R. Min, Z. Qin, L. Shen, and M. Cheng, "Towards stable backdoor purification through feature shift tuning," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[40] R. Zhu, D. Tang, S. Tang, X. Wang, and H. Tang, "Selective amnesia: On efficient, high-fidelity and blind suppression of backdoor effects in trojaned machine learning models," in *44th IEEE Symposium on Security and Privacy, SP*. IEEE, 2023, pp. 1–19.

[41] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[42] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

[43] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[45] "Huawei cloud ai gallery," https://developer.huaweicloud.com/develop/aigallery/usecase/list, accessed: 2025-06-02.

[46] S. Rongali, K. Arkoudas, M. Rubino, and W. Hamza, "Training naturalized semantic parsers with very little data," *arXiv preprint arXiv:2204.14243*, 2022.

[47] H. Fu, Y. Chi, and Y. Liang, "Guaranteed recovery of one-hidden-layer neural networks via cross entropy," *IEEE Transactions on Signal Processing*, 2020.

[48] K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon, "Recovery guarantees for one-hidden-layer neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*. PMLR, 2017.

[49] X. Zhang, Y. Yu, L. Wang, and Q. Gu, "Learning one-hidden-layer relu networks via gradient descent," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019.

[50] A. Anandkumar, R. Ge, and M. Janzamin, "Learning overcomplete latent variable models through tensor methods," in *Proceedings of The 28th Conference on Learning Theory (COLT 2015)*, ser. Proceedings of Machine Learning Research, P. Grünwald, E. Hazan, and S. Kale, Eds., vol. 40. Paris, France: PMLR, 3–6 Jul 2015.

[51] M. Janzamin, H. Sedghi, and A. Anandkumar, "Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods," *arXiv preprint arXiv:1506.08473*, 2015.

[52] C. Stein, P. Diaconis, S. Holmes, and G. Reinert, "Use of exchangeable pairs in the analysis of simulations," in *Stein's Method: Expository Lectures and Applications*. Institute of Mathematical Statistics, 2004.

[53] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

[54] D. A. Spielman, H. Wang, and J. Wright, "Exact recovery of sparsely-used dictionaries," in *Proceedings of the 25th Annual Conference on Learning Theory (COLT)*, ser. Proceedings of Machine Learning Research, S. Mannor, N. Srebro, and R. C. Williamson, Eds., vol. 23. Edinburgh, Scotland: PMLR, 25–27 Jun 2012.

[55] M. Thom and G. Palm, "Sparse activity and sparse connectivity in supervised learning," *Journal of Machine Learning Research*, vol. 14, 2013.

[56] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching," *Journal of Machine Learning Research*, vol. 6, 2005.

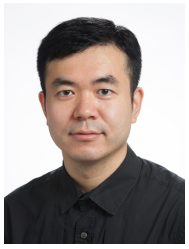
[57] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 1998.

[58] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[59] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[60] "Huawei cloud modelarts," https://www.huaweicloud.com/intl/en-us/product/modelarts.html, accessed: 2025-06-02.
[61] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
[62] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.
[63] G. M. D. Corso, A. Gulli, and F. Romani, "Ranking a stream of news," in *Proceedings of the 14th International Conference on World Wide Web*, 2005.
[64] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.
[65] "Huawei codelab," https://codelabs.developer.huaweicloud.com, accessed: 2025-06-02.
[66] F. Yang, Z. Chen, and A. Gangopadhyay, "Using randomness to improve robustness of tree-based models against evasion attacks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 2, 2020.
[67] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *26th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2019.
[68] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.
[69] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn dnn architectures," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.
[70] "5-layer cnn model in pytorch examples," https://github.com/pytorch/examples/blob/main/mnist_hogwild/main.py, accessed: 2025-06-02.

IX. Biography Section

Yaxin Xiao received the M.Eng. and B.Eng. degree from Mechatronics Engineering, Zhejiang University, Hangzhou, China in 2020 and 2017, respectively. She is currently working towards a PhD degree in the Department of Electrical and Electronic Engineering at The Hong Kong Polytechnic University, Hong Kong. Her research interests include adversarial machine learning.



Haibo Hu is a professor in the Department of Electrical and Electronic Engineering, Hong Kong Polytechnic University and is affiliated with the Research Centre for Privacy and Security Technologies in Future Smart Systems. He serves as the Programme Leader for the BSc (Hons) in Information Security. His research interests include cybersecurity, data privacy, the Internet of Things, and adversarial machine learning. He has published over 110 research papers in refereed journals, international conferences, and book chapters. As principal investigator, he has received over 20 million HK dollars of external research grants from Hong Kong and mainland China. He is an associate editor of ACM Transactions on Privacy and Security (TOPS). He is the recipient of a number of titles and awards, including IEEE MDM 2019 Best Paper Award, WAIM Distinguished Young Lecturer, ICDE 2020 Outstanding Reviewer, VLDB 2018 Distinguished Reviewer, ACM-HK Best PhD Paper, Microsoft Imagine Cup, and GS1 Internet of Things Award. He is a senior member of ACM, IEEE and CCF, and a certified Cisco CCNA Security Trainer.

Qingqing Ye is an assistant professor in the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University. She received her PhD degree from Renmin University of China in 2020. She has received several prestigious awards, including Hong Kong RGC Early Career Award, IEEE S&P Travel Award, and National Scholarship. Her research interests include data privacy and security, and adversarial machine learning.

Li Tang is an assistant professor in the Department of Software Engineering, Yunnan University. She received her PhD degree from the Department of Electronic and Electrical Engineering, Hong Kong Polytechnic University, in 2023. Her research interests include data integrity and privacy, and multimedia authentication.

Zi Liang received the M.Eng. and B.Eng. degree from Xi'an Jiaotong University and Northeastern University in 2023 and 2020, respectively. He is currently working towards the PhD degree in The Hong Kong Polytechnic University. He is interested in revealing privacy and security risks related to large language models.

Huadi Zheng received BEng degree from Sun Yat-sen University and PhD degree from The Hong Kong Polytechnic University. In 2021, he was selected by Hong Kong Science Park as a talent for the Technology Leaders of Tomorrow Program. He joined Shield Lab at Huawei as a senior engineer. His research interests include trustworthy AI/ML, IoT security and speech technology.