A STUDY ON LEARNING-BASED MODEL EXTRACTION ATTACKS AND DEFENSE METHODS

YAXIN XIAO

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University
Department of Electrical and Electronic Engineering

A Study on Learning-based Model Extraction Attacks and Defense Methods

Yaxin Xiao

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
April 2025

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Yaxin Xiao

Abstract

The widespread adoption of Machine Learning as a Service (MLaaS) has exposed cloud-deployed black-box models to growing security risks, particularly from model extraction attacks (MEAs). In these attacks, adversaries exploit prediction interfaces to replicate proprietary models, subsequently enabling secondary privacy breaches or adversarial attacks through extracted model insights. Driven by the intellectual property (IP) theft crisis, this thesis first systematically investigates MEA risks and then explores defense strategies from multiple perspectives.

While existing MEA research focuses on query optimization to maximize attack success, two critical attack amplifiers remain underexplored: (1) the mutual reinforcement between model theft and training data privacy leakage and (2) the impact of initial bootstrapping on extraction performance. This work reveals that model extraction and membership inference attacks, which aim to identify training data, can strengthen each other through an iterative process. Furthermore, we reveal that optimized initial parameters and more compatible model architectures enable MEAs to replicate models at the neuron level. This strategy not only boosts the performance of model extraction attacks but also redefines their severity because it provides substitute models with neuron-level precision for downstream attacks.

To counter the threats of MEAs, we propose two defense strategies. The first is a proactive method, which leverages the model's hard-to-replicate properties to reduce its extractability, preventing MEAs from producing high-fidelity extracted models.

The second is a passive forensic approach using black-box model watermarks, which embeds ownership signals into the extracted models. Compared to existing watermarking methods, CFW not only transfers more effectively to extracted models but also resist adaptive removal attacks. By uncovering key mechanisms that amplify model extraction attacks and introducing effective countermeasures, this study offers strong protection for MLaaS platforms against intellectual property theft.

Publications Arising from the Thesis

1. Y. Xiao, Q. Ye, H. Hu, H. Zheng, C. Fang, and J. Shi. "MExMI: Pool-based Active Model Extraction Crossover Membership Inference." *36th Conference on Neural Information Processing Systems* (NeurIPS). 2022.

2. Y. Xiao, Q. Ye, L. Hu, H. Zheng, H. Hu, Z. Liang, Y. Jiao, "Privacy Attacks Exploiting Residuals: How Approximate Machine Unlearning Fails to Protect Privacy". *The IEEE/CVF International Conference on Computer Vision* (ICCV), 2025.

3. Y. Xiao, H. Hu, Q. Ye, L. Tang, Z. Liang, H. Zheng. "Unlocking High- Fidelity Learning: Towards Neuron-Grained Model Extraction". *IEEE Transactions on Dependable and Secure Computing* (TDSC), 2025.

4. T. Li, Q. Ye, H. Hu, Q. Xue, Y. Xiao, and J. Li. "DeepMark: A Scalable and Robust Framework for DeepFake Video Detection". *ACM Transactions on Privacy and Security* 27, no. 1 (2024): 1-26. https://doi.org/10.1145/3629976.

5. H. LI, L. Bai, Q. Ye, H. Hu, Y. Xiao, H. Zheng, J. Xu. "A Sample-Level Evaluation and Generative Framework for Model Inversion Attacks". *The Annual AAAI Conference on Artificial Intelligence.* https://doi.org/10.1609/aaai.v39i17.34012. 2025

6. Z. Liang, P. Wang, R. Zhang, H. Hu, S. Zhang, Q. Ye, N. Xu, Y. Xiao, C. Zhang, L. Cui. "Exploring Intrinsic Alignments within Text Corpus". *The Annual AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v39i26.34957. 2025

7. Z. Liang, H. Hu, Q. Ye, Y. Xiao, R. Li. "Does Low Rank Adaptation Lead to Lower Robustness against Training-Time Attacks?". *Forty-second International Conference on Machine Learning* (ICML), 2025.

8. Z. Liang, Q. Ye, Y. Wang, S. Zhang, Y. Xiao, R. Li, J. Xu, H. Hu. "Yes, My LoRD. Guiding Language Model Extraction with Locality Reinforced Distillation". *Annual Meeting of the Association for Computational Linguistics* (ACL), 2025.

9. R. Du, Q. Ye, Y. Xiao, L. Yu, Y. Fu, H. Hu. "Dual Utilization of Perturbation for Stream Data Publication under Local Differential Privacy". *The IEEE 41rd International Conference on Data Engineering*. https://doi.org/10.48550/arXiv.2504.14993. 2025.

10. Y. Xiao, Q. Ye, Z. Liang, H. Li, R. Li, H. Zheng, H. Hu, "Class-feature Watermark: A Resilient Black-box Watermark Against Model Extraction Attacks". Submitted to *The Annual AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

11. Y. Xiao, H. Hu, Qi. Ye, J. Xu, L. Tang, H. LI, H. Zheng. "Privacy-Preserving Proof-of-Learning via Watermark Trajectory". Submitted to *IEEE Transactions on Dependable and Secure Computing* (TDSC), 2025.

12. Z. Liang, H. Hu, Q. Ye, Y. Xiao, H. Li. "Why Are My Prompts Leaked? Unraveling Prompt Extraction Threats in Customized Large Language Models". arXiv preprint arXiv:2408.02416, 2024.

13. H. Li, R. Sun, Q. Ye, B. Zhao, H. Hu, Y. Xiao, Z. Liang, X. Zhang, J. Xue, H. Hu. "FeatMark: Feature-level Watermark Protection against Mimicry Attacks with Diffusion Models". Submitted to *Network and Distributed System Security* (NDSS), 2026.

14. R. Li, H. Hu, Z. Liang, Y. Xiao, Q. Ye. "Incorporating User Preferences into TAP Conflict Handling". Submitted to *IEEE Internet of Things Journal*, 2025

Acknowledgments

My PhD journey has been a deeply meaningful chapter of my life. When I first began this path, I struggled for years to understand the difference between absorbing knowledge (as I did in engineering) and doing real research. Now I realize that research requires critical, systematic thinking – this is how we build strong arguments. I also finally grasped why I needed to learn those mathematics and science concepts. At first, knowledge felt disconnected from the real purpose, and studying for exams seemed like a waste of mental energy. But through research, I discovered how mathematics forms the foundation of scientific analysis. This rekindled the curiosity I felt when I first learned about modern science. Beyond these lessons, I owe sincere thanks to those who supported me along the way.

First and foremost, I thank my supervisor, Prof. Hu, for teaching me how to conduct research from scratch. His positive attitude and the supportive research environment he created allowed me to focus entirely on my work without stress. Unlike many PhD students, I never faced mental health challenges, and this is largely thanks to his guidance.

I am also grateful to Prof. Ye and the entire Astaple group. Their kindness, brilliant insights, and sincere dedication to research taught me invaluable lessons. The friendships we formed during this time are treasures I will carry forever.

Additionally, I thank my family, my husband, Mr Han Rui, and my cat for their steady support. They never pressured me and gave me the peace of mind to fully

dedicate myself to this journey.

Finally, I thank PolyU and its administrative staff for their efficient organization. Through their dedicated work, I was able to finish my PhD degree smoothly without unnecessary distractions.

This PhD journey has not only answered academic questions but also helped me grow as a person. I've learned to better understand myself, embrace my values, and strive to live with authenticity.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in machine learning (ML) have significantly shifted the paradigm in all walks of life. To stay competitive, industries are accelerating the deployment of ML-driven solutions, which drives upgrades in areas such as intelligent manufacturing [46] and autonomous driving [22]. This growing demand fuels the growth of Machine Learning as a Service (MLaaS) [79], a service pattern where providers such as Microsoft Azure ML [11], Amazon AWS ML [2], and Google Cloud AI [7] grant users query access to proprietary models through pay-per-query APIs. Users can obtain predictions by uploading data, without incurring the high cost of model development.

Within this paradigm, the confidentiality of deployed ML models is a critical asset. Terms of service clearly state that providers retain ownership of their models (*e.g.*, Google models [39]), and recognize them as legally protected intellectual property (IP). However, this IP is increasingly threatened by model extraction attacks (MEAs) [100], which aim to replicate target models by exploiting query access, even when the models are deployed in black-box form.

Model extraction attacks pose escalating threats to machine learning ecosystems, which are driven by three compelling adversarial incentives. First, adversaries can replicate proprietary models to bypass years of R&D investment. For instance, a

fine-tuned large language model (LLM) developed by a financial startup to summarize investment strategies was recently leaked on the dark web [14]. Second, model extraction provides a path to economic efficiency. It allows adversaries to avoid the prohibitive costs of API usage fees while evading rate-limited access controls. Third, the extracted model can be exploited for downstream privacy attacks, including membership inference [45] and model inversion [34, 116], which exposes sensitive training data.

These cascading risks range from IP theft to data breaches, which underscore the critical need for robust defenses against model extraction in real-world ML deployments. Therefore, this thesis conducts a thorough investigation of model extraction attacks. It first investigates their potential impact, then explores effective defense strategies from different perspectives. Accordingly, Section 1.1 outlines the motivations of our design to enhance the attack performance of model extraction, and Section 1.2 introduces our proposed defense approaches.

1.1 Enhanced Frameworks for Model Extraction Attacks

1.1.1 Existing Works

Existing model extraction attacks can be categorized into direct recovery [48] and active-learning-based model extraction attacks [21, 76]. Active learning (AL) refers to those semi-supervised training methods that aim to find the most informative training dataset with a limited query budget [36]. Since both active learning and model extraction aim to train a model with as few queries as possible, active learning has become increasingly popular in model extraction attacks. Depending on the availability of real-life samples for querying, learning-based model extraction can be further di-

vided into pool-based active model extraction (PAME) and query-synthesizing-based active model extraction (SAME). The former assumes the presence of samples, from which query samples are iteratively selected using a pool-based or stream-based active learning. Classic pool-based model extraction include ActiveThief [76] and Knockoff [74]. The latter does not assume the presence of such samples and obtains them by generative methods [21, 92]. Classic query-synthesizing-based model active extraction include PRADA [52], MAZE [53] and DFME [101].

1.1.2 Model Extraction Crossover Membership Inference

While learning-based model extraction leverages techniques from active learning, a **critical difference between active learning and model extraction** has long been overlooked. Since active learning is essentially a training method, the data pool is where training samples are drawn, which means this pool must have the same feature distribution as the training dataset. However, in an MEA the adversary has no access to the victim model's training dataset or even samples in the same problem domain [27]. Unfortunately, **all previous pool-based model extraction works ignore this difference and assume the samples in the pool are homogeneous** to those in the training dataset.

We address this issue through **identifying homogeneous samples in the data pool** and making full use of them for a "guided" model extraction. We exploit membership inference (MI), an attack that infers the training samples from a given dataset [90], to select them and train a copy model. In turn, the extracted model can enhance the accuracy of membership inference. As such, we propose an iterative extraction framework MExMI where ME and MI reinforce each other through iterations (See Chapter 4). Within limited query budget, the final outcome consists of both a high-fidelity copy model and an accurate set of training samples.

1.1.3 Towards Neuron-grained Model Extraction

On the other hand, while learning-based model extraction is the de-facto type of ME attacks [53, 76, 101], it is widely believed that learning-based strategies are not well-suited for achieving high-fidelity extraction due to their non-determinism [48]. Consequently, the copy model tends to converge to sub-optima, resulting in low fidelity (*i.e.*, low similarity to the victim model). For example, ActiveThief [76], 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 [44]), initial parameters, training dataset (CIFAR [57]), and hyperparameters of the optimizer. That is, there is a discrepancy in the labels inferred by the original model and the copy model for about 1 in every 20 test samples. Essentially, such limitations arise from their "learning-a-task" work mode, in which they search for the best copy model in terms of task accuracy from a space of copy models with different initialization states. **"Learning-a-task"** deviates from the objective of model extraction and yet is neglected in the literature.

To boost the full potential effectiveness of model extraction, we re-evaluate 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 (See Chapter 5). Its key idea is to **"learn-a-model"** instead of to "learn-a-task". Through learning a model, the copy model can even achieve neuron-level extraction to certain extent [81, 99, 124].

1.2 Defending Methods against Model Extraction Attacks

In the second part of this thesis, we address the threat that model extraction poses to a model's intellectual property (IP) by proposing two defense strategies. The first is

a proactive approach which rethinks model training from the defender's perspective and develops a new strategy to reduce the extractability of the victim model. This is based on the theory that the difficulty of recovering a model by learning is subject to specific critical properties of the model parameters [122]. In Chapter 6.1, we develop a defensive training strategy for the victim model, which adjusts such properties to make the model hard to extract.

The other defense proposed in this thesis is a passive approach based on black-box model watermarks [50], a promising forensic method for verifying model ownership in the context of MEAs, as detailed in Chapter 6.2. These watermarks embed a specific task as a marker, enabling ownership verification of surrogate models obtained through extraction, since such tasks may be partially transferred during the process. Our key observation is that prior studies [50, 69] overestimated watermark resilience against removal threats due to the use of ill-suited removal methods. In this work, we explore how black-box watermarks can both verify ownership of extracted models and remain robust against dedicated removal attacks tailored to the model watermarks effective under MEA.

1.3 Contributions

In summary, this thesis makes the following contributions:

- We propose an iterative framework, MExMI, where model extraction attacks (ME) and membership inference attacks (MI) reinforce each other. To support this, we propose the few-shot versions for both shadow-model and unsupervised MI. To boot-strap shadow-model MI, we develop an indicative quality metric of shadow models and design a metric-based shadow-model training algorithm. Extensive experimental results confirm that MExMI achieves fidelity improvements ranging from 11.14% to 94.07% and reaches 84.13% MI precision,

comparable to the state-of-the-art MI attack [84], which assumes an unlimited query budget.

- We propose MEBooster, a training booster framework to exploit the potential advantage of model extraction attacks (MEAs) at the neuron-grained level. MEBooster is validated on various image tasks and text tasks and demonstrate its effectiveness and generalizability. In the best case, it yields a 58.10% fidelity improvement over extraction methods without the booster.

- A proactive defensive training strategy is proposed, which exploits the hard-to-extract properties of the victim model for the first time. Experimental results show it can reduce the extractability (*i.e.*, fidelity) of the victim model by up to 58.81%.

- A resilient watermarking approach named Class-Feature Watermarks (CFW) is proposed to detect the infringement of MEA. CFW leverages class-level artifacts to resist WRK. To maintain its effectiveness and resilience during MEA, we optimize its transferability and stability in MEA. Comprehensive evaluations across diverse domain tasks show that CFW excels across multiple aspects, offering robust resilience against removal attacks, high MEA transferability, and minimal impact on model utility.

1.4 Roadmap

The rest of the thesis is organized as follows. Chapter 2 reviews related work. Chapter 3 presents the necessary background and problem definition. Chapter 4 introduces the MExMI framework. Chapter 5 elaborates on the MEBooster framework. One mitigates potential MEA threats, while the other detects MEA infringements using model watermarking techniques. Finally, Chapter 7 concludes the thesis.

Chapter 2

Related Works

2.1 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 (ME, MEA) [48, 77, 100], which expose severe vulnerabilities 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 largely ineffective [76]. To address this, Chandrasekaran *et al.* [21] propose 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 studies [52, 53, 76, 77, 101]. In terms of query data availability, ME can be categorized into query-synthesizing ME and query-acquiring ME. The former is used when the

Table 2.1: Main-stream Learning-based Model Extraction Attacks

Attacks	Year	Data Acquisition		Data Pool Requirements	Training Technique
		Pool	Synthesized		
Tramèr [100]	2016		✓	None	–
Papernot [77]	2017		✓	Domain subset	Structure Selection
Knockoff [74]	2019	✓		Public pool	–
PRADA [52]	2019		✓	Domain subset	CV Search
ActiveThief [76]	2020	✓		Public pool	–
MAZE [53]	2021		✓	None	–
DFME [101]	2021		✓	None	–
HODA [82]	2023		✓	None	–
DisGUIDE [80]	2023		✓	None	–
Bayes Attack [97]	2024	✓		Public pool	–

adversary does not possess enough real data for query, which includes iterative active frameworks [52, 77] and minimax-game frameworks [53, 101]. Despite saving the data collection cost, it suffers from huge query budget demands. The latter is used when the adversary spares extra cost to actively collect real data for query [27, 48, 76].

Table 2.1 summarizes the features of existing learning-based ME methods. Several attackers, such as Knockoff [74], ActiveThief [76], and MExMI [108], 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 [52] and Papernot [77]. Another category of synthetic approaches is data-free model extraction, which has gained popularity, with methods such as MAZE [53], DFME [101] and DisGUIDE [80] emerging.

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 2.1, there is a lack of studies on training techniques. So far, only Papernot *et al.* investigate the structure selection methods for copy models [77], and PRADA [52] uses cross-validation (CV) to search for the training hyper-parameters.

Table 2.2: Comparison of defense methods against different attacks

Attack	Query Detection		Mitigation Defense				Model Watermarking		
	PRADA[52]	VarDetect[75]	GRAD^2[72]	Adaptive[54]	DBLP[119]	ModelGuard[97]	EWE[50]	MBW[55]	MEA-D[69]
Tramèr[100]	✓	✓	△	△	△	△	✓	✗	✓
Papernot[77]	✓	✓	△	△	△	△	✓	✗	✓
Knockoff[74]	✗	✗	△	△	△	△	✓	✗	✓
PRADA[52]	✓	✓	△	△	△	△	✓	✗	✓
ActiveThief[76]	✗	✗	△	△	△	△	✓	✗	✓
MAZE[53]	✓	✓	△	△	△	△	✓	✗	✓
DFME[101]	✓	✓	△	△	△	△	✓	✗	✓
HODA[82]	✓	✓	△	△	△	△	✓	✗	✓
DisGUIDE[80]	✓	✓	△	△	△	△	✓	✗	✓
Bayes Attack[97]	✗	✗	△	△	△	△	✓	✗	✓

Adaptive: Adaptive Misinformation [54]
✓: Defense is effective against the attack
✗: Defense is ineffective against the attack
△: Defense provides partial mitigation

2.2 Defending Against Model Extraction Attacks

Current research addresses model security through three principal defense paradigms: (1) malicious query detection, (2) mitigation defense, and (3) model watermarking techniques. Detection-based approaches [52, 75] identify suspicious query patterns, while existing mitigation defense [54, 72] systematically alter prediction outputs to degrade extraction quality. Model watermarking solutions [50, 69] embed identifiable signatures to trace stolen models. While query detection remains vulnerable to coordinated attacks, prediction perturbation and model watermarking demonstrate better operational viability, warranting focused analysis. Table 2.2 compares representative defenses across these three categories against various known extraction attacks. A check mark indicates effective defense, a cross denotes ineffectiveness, and a triangle represents partial mitigation.

2.2.1 Mitigation Defense through Prediction Perturbation

Output perturbation [54, 72, 96] protects victim models by deliberately distorting API responses without altering core model parameters. However, practical deployment faces a fundamental privacy-utility trade-off, as excessive distortion compromises legitimate user experience. Further, the emergence of adaptive model extraction attacks [25] capable of neutralizing conventional perturbation defenses has spurred

renewed defensive innovation, which is exemplified by ModelGuard [97].

Complementing existing reactive defenses, this study proposes **model modification** as a novel proactive paradigm. It reduces the extractability of victim models by strategically modifying their properties during training., establishing a preemptive defense mechanism fundamentally distinct from post-deployment protections.

2.2.2 Ownership Verification in Model Extraction Attacks

2.2.2.0.1 Black-box Model Watermarks Black-box watermarks are a prominent approach to safeguarding machine learning (ML) models from Model Extraction Attacks (MEA) [50, 55, 69] infringement. State-of-the-art approaches typically embed backdoor-based tasks by modifying inputs and relabeling them to form distinct watermark patterns. Since these watermarks are task-based, they can be transferred through MEA. Recent studies focus on enhancing their MEA transferability to ensure extracted models retain the embedded markers, making extraction without watermark retention difficult. Techniques to improve transferability include representation entanglement via Soft Nearest Neighbor Loss (SNNL) [50] and composite sample generation for input-space entanglement [69]. However, existing methods fall short of systematically evaluating their resilience against removal attacks. Table 2.3 summarizes state-of-the-art black-box watermarking methods which can defend MEAs, highlighting their data types, impact on model utility, transferability through MEA, and resilience against various attacks, which are further detailed below.

2.2.2.0.2 Removal Attacks Threatening Black-box Watermarks Black-box watermark tasks can be removed if they are decoupled from the domain task. This principle underpins existing threats to black-box watermarks, including watermark-targeted removal methods [13, 24] and applicable backdoor removal techniques [63, 64, 67, 68, 103, 112, 121, 126], which do not require watermark samples. Among these

Table 2.3: Black-box Model Watermarking Defending against Model Extraction Attacks (MEA)

Watermark	**Resilience Against Removal Attacks**			
	Reversion-type [103, 112]	Learning Induced [63, 68]	Neuron Pruning [67, 121]	WRK (Ours)
EWE [50]	✓	✓	✓	✗
RS [17]	✗	✗	✓?	✗
MBW [55]	✗	✗	✓?	✗
MEA-Defender [69]	✓?	✓	✓	✗
CFW (Ours)	✓	✓	✓	✓

✗: No. ✓: Yes. ✓?: Yes but largely degraded.

removal methods, three decoupling perspectives are exploited: input space, features, and neurons/channels. Input space decoupling aims to reverse watermark samples first and then unlearn them, with techniques including NC [103], I-BAU [112], and Aiken [13]. Feature decoupling removes watermark tasks through learning-induced forgetting, such as ABS [68], NAD [63], SEAM [126], and REFIT [24]. The third type, neuron/channel decoupling, targets watermark tasks at the neuron level through pruning methods, including Fine Pruning [67], CLP [121], and RNP [64]. However, these methods struggle against black-box watermarks that are deeply entangled with domain tasks [50, 69]. For example, adversarial samples [50] and composite samples [69] form non-linear decision boundaries that are hard to reverse [113]. Existing methods fail to account for this entanglement, leading to an overestimation of watermark resilience.

Chapter 3

Background Knowledge and Definitions

3.1 Notations of Machine Learning Models

The victim models are deep neural networks (DNN) trained for classification tasks with K classes in supervised learning. A DNN model $F_\theta(\cdot) : \mathcal{X} \Rightarrow \mathcal{Y}$ (F for short) is defined over 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. To train such a model, we assume a supervised learning process on the training dataset $D = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n \subseteq \mathcal{X} \times \mathcal{Y}$. D contains n labelled samples, where $\mathbf{x}_i$ is the i-th training sample, and $\mathbf{y}_i$ is its one-hot format label vector. That is, if the sample label is k, $\mathbf{y}_i[k] = 1$, and $\mathbf{y}_i[j] = 0$ for $\forall j \neq k$. [1]

A typical L-layer DNN consists of layers like linear, convolutional, activation, and pooling, with L-th layer being the output layer. Common activation functions $\sigma(\cdot)$ include ReLU / Leaky ReLU [12, 71]. In layer l, the width/channel is n_l, with

[1] Throughout this paper, we exclude special training algorithms, e.g., co-training, mutual mean-teaching, and sharpness-aware minimization [33, 106, 117]. Neither the victim nor the adversary uses these algorithms.

neuron weights and bias as $\theta_{l,i} \leftarrow [w_{l,i}, b_{l,i}]$. 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. For input samples $\{\mathbf{x}_i\}_{i=1}^b$ ($\{\mathbf{x}\}$ for short) with size b, $\mathbf{F}_{\theta^l}(\{\mathbf{x}\}) = [f_{l,1}(\{\mathbf{x}\}), \ldots, f_{l,n_l}(\{\mathbf{x}\})] \in \mathbb{R}^{b \times n_l}$ denotes layer l's output matrix. Here, θ^l represents the subset from the input layer up to layer l. $D_l(\mathbf{x}) = [z_{l,1}(\{\mathbf{x}\}), ..., z_{l,n_l}(\{\mathbf{x}\}), 1] \in \mathbb{R}^{b \times (n_l+1) \times (n_l+1)}$ denote the diagonal matrix of the activation function. For example, if the activation function is ReLU, $z_{l,i}(\{\mathbf{x}\})$ is either 0 or 1.

For the DNN model $F(\cdot;\theta)$, the output of each layer is referred to as its representation, a matrix (or a vector) that captures intermediate features. For an input $\mathbf{x}$, the representation at layer l is denoted as $F_{\theta^l}(\mathbf{x})$. The output at layer L is a length-K vector, called the logits, *i.e.*, logits=$F_{\theta^L}(\mathbf{x})$ ($F(\mathbf{x})$ for short). Normalized logits yield the predicted probability for each class. The class probabilities for $\mathbf{x}$ can be described as: probabilities = $[\Pr(Y=0|\mathbf{x}), \ldots, \Pr(Y=K|\mathbf{x})]$. F's predicted label is the indices with the highest probability, denoted by $\hat{y} = \arg\max F(\mathbf{x})$.

3.1.1 Problem Definition: Model Extraction Attacks (MEAs)

In *Machine Learning as a Service* (MLaaS), ML models are deployed on cloud platforms to provide services via query APIs [4, 5, 6]. Given the resource-intensive training and the high privacy of their training data, these ML models are considered valuable assets and are deployed in black-box format.

However, model extraction attacks (MEA) [100] threaten the intellectual property of these black-box models by exploiting queried input-output pairs. In an MEA, the adversary aims to replicate the victim model F locally, obtaining a **copy model** F' without access to additional information, *e.g.*, model structures or training samples. MEA performance is typically evaluated using two metrics: the copy model's **accuracy** (ACC) on the evaluation dataset D_t, and **fidelity** (FID) [48]. Fidelity

measures the similarity between the victim and copy models by their label agreement rate on D_t. To date, learning-based model extraction is the de-facto approach of MEA [53, 76, 101], where the adversary first queries the victim model and then trains the copy model using the queried results [76, 108].

3.1.2 Threat Model

We assume a well-trained black-box **victim model** F is deployed in a machine-learning-as-a-service (MLaaS) with a chargeable query interface. The threat model is a game between an adversary aiming to steal the model by model extraction and a defender implementing proactive defending strategies or ownership protection.

We assume that the adversary either only has black-box access to the MLaaS model, or knows its architecture (e.g. in AWS Marketplace [3] and Huawei AI Gallery [8]). In addition, the adversary can collect a large number of unannotated public samples to construct an adversary data pool P. These attackers are expected to have good mastery of machine learning techniques, including common initialization methods. For instance, in the context of image classification, they are familiar with widely adopted methods such as He initialization [43].

To protect the victim model's ownership against model extraction, the defender may implement two defending strategies. He either employs **anti-model-extraction (anti-MEA) defenses**, such as perturbation on the output probability vector [119, 120]. or embeds **black-box model watermarks** which enable ownership verification through queries. Given the prevalence of removal research [113], this game with model watermarks becomes more complex: the adversary may attempt to **remove the watermark** after theft to avoid ownership detection, which threatens ownership verification. Thus, for defenders, the challenge extends beyond watermarking the copy model in indirect model theft attacks, *i.e.*, MEA; a more critical issue is ensuring the watermark resilience against removal attacks.

We formalize the threat model in Game 3.1.1.

Game 3.1.1. *The game proceeds between a defender (the model owner)* **D** *and an adversary* **A**.

1) ***(Optional) Anti-MEA Training***. **D** *trains the victim model F using anti-MEA strategies to reduce extractability while preserving high model utility.*

2) ***(Optional) Watermark Embedding***. **D** *embeds black-box watermarks $\mathcal{W}$ into the victim model F. The watermark-related objects (*e.g.*, watermark samples) and training samples are securely stored on a trusted platform, ideally with a timestamp.*

3) ***Model Stealing***. **A** *obtains the local copy model F' through model extraction attacks.*

4) ***Watermark Removal and Copy Model Deployment***. **A** *attempts to remove the watermark from F' using limited domain data D_d. Afterward, F' is deployed and provides query access.*

5) ***(Optional) Watermark Verification***. *To verify the ownership of the suspected model F',* **D** *queries F' with watermark samples, obtaining the watermark evidence $\mathcal{W}$.*

6) ***Settlement 1***. *If* **A** *fails to replicate a high-fidelity copy model F' from F, then* **D** *wins and* **A** *loses.*

7) ***Settlement 2***. *When* **A** *successfully replicates a high-fidelity copy model F', if the watermark evidence $\mathcal{W}$ indicates that F' is stolen from F,* **D** *wins, and* **A** *loses. If the watermark evidence $\mathcal{W}$ fails or does not exist,* **A** *wins and* **D** *loses.*

8) ***Settlement 3***. *If $\mathcal{W}$ suggests that an innocent model is misidentified as stolen from F, the defender* **D** *loses.*

3.1.3 Problem Formulation of Model Watermarking

For the defender to succeed in the ownership game (Game 3.1.1) in model extraction attacks, the model watermarks must satisfy the following criteria:

Prop. 1. *Utility Preservation.* The watermark should not harm the target model's functionality for benign users.

Prop. 2. *High MEA transferability.* The watermark should be retained in copy models obtained through MEAs.

Prop. 3. *Correctness.* The watermark should reliably identify stolen models without false ownership claims.

Prop. 4. *Resilience.* The watermark remains valid after being attacked by watermark removal.

Prop. 5. *Stability.* A derivative property of resilience, stability ensures that a watermark's resilience in the copy model remains consistent with that in the victim model.

Prop. 6. *Stealthiness.* The watermark cannot be detected by adversaries.

Chapter 4

MExMI: Model Extraction Crossover Membership Inference

This chapter presents MExMI, a unified framework in which model extraction attacks (MEA) and membership inference attacks (MIA) reinforce each other.

As noted in Section 1.1.2, the query set in MEA is typically non-homogeneous [27], as adversaries lack access to the victim's training data or domain-specific samples. However, **prior MEA methods overlook this gap and directly apply active learning algorithms [36] designed for homogeneous settings** to pursue query efficiency. As a result, they underestimate the true potential of MEA. To demonstrate how non-homogeneous datasets can affect the ME performance, we build a Wide-ResNet-based victim model using the first 25k training image samples of CIFAR10. Then we extract this model using ActiveThief [76], an existing pool-based active model extraction (PAME) benchmark, from three adversary data pools: (1) pool A consists of the second 25k training samples of CIFAR10, (2) pool B consists of the middle 25k training samples, and (3) pool C consists of the first 25k training samples. In other words, the victim's training set does not overlap with A (non-homogeneous) but overlaps with half of B (quasi-homogeneous), and completely with

Table 4.1: Homogeneous v.s. Non-Homogeneous Data Pool for Model Extraction

Train Copy Model from Pool	**Pool A**	**Pool B**	**Pool C**
Training Data Homogeneity	**None**	**Partial**	**Full**
Fidelity / %	90.32	91.29	92.30

C (fully homogeneous). Table 4.1 shows the fidelity (*i.e.*, similarity to the victim model) of the extracted copy models from A, B and C. We observe that homogeneous samples contributes more to the success of model extraction than non-homogeneous samples.

To address this gap, MExMI leverages MIA to identify training-homogeneous samples from the candidate query set, which benefits the effectiveness of MEA. In return, MIA benefits from the high-fidelity surrogate model produced by MEA, which allows unrestricted querying for membership inference.

When designing MExMI, three critical challenges emerge. First, existing MIA methods [84, 90] do not account for query cost, which is a crucial constraint in MEA. To address this, we leverage the training data of the extracted (copy) model to perform MI without incurring additional query overhead. Second, state-of-the-art MI attacks often rely on assumptions that are incompatible with MExMI or typical pool-based model extraction settings. For instance, shadow-model MI assumes access to a labeled dataset drawn from the same distribution and of the same size as the target model's training data [90]. To remove this barrier, we propose a quality metric to evaluate and optimize shadow models without relying on a large labeled dataset. Using this approach, we adapt both shadow-model MI and unsupervised MI [84, 111] to fit within the MExMI framework. Third, current PAME attacks overlook the potential of utilizing purified training samples selected from the query pool. In response, we designed three modules to facilitate PAME to make the most of it.

Roadmap. The rest of this chapter details the proposed methodology and experimental validation. Section 4.1 provides an overview of the MExMI framework and

its key components. Section 4.1.5 introduces the adaptive MI algorithms tailored for MExMI. Section 4.2 presents extensive experimental results evaluating the effectiveness of the proposed framework.

4.1 MExMI Framework

In this section, we present our iterative model extraction framework MExMI where ME and MI reinforce each other. An MI attack aims to distinguish those individuals $\hat{D}$ from a population P that exist in the victim model F's **training dataset** [90]. As illustrated in Fig. 4.1 and pseudocode in Algorithm 1, the input of MExMI is an adversary data pool P and the access to a black-box victim model F, and its outputs are the copy model F' and the inferred training dataset $\hat{D}$. In the first iteration, the adversary chooses k initial seed samples $[\mathbf{x}_1, \ldots, \mathbf{x}_k]_0$ from P without putting them back, where k is the query budget per iteration. These samples are fed to the victim model F, which outputs a probability vector $F(\mathbf{x})$. Then an MI attack model F_{MIA} is constructed using the queried dataset $[(\mathbf{x}_1, F(\mathbf{x}_1)), \ldots]_0$ or the copy model (see Section 4.1.5). F_{MIA} is used in both MI Post-Filter and MI Pre-Filter modules. The queried dataset is then passed through the MI Post-Filter, which weighs them according to their probability of being a training sample of the victim model. Then the weighted queried dataset is used to train the copy model F' , which is then fed to the MI Pre-Filter to refine the adversary data pool for AL sample selection of k queries in the next iteration. The process is repeated until the total query budget b is depleted. Thereafter, without consuming any query budget, F_{MIA} is used to launch MI attacks on F' and the data pool to obtain the final inferred training dataset $\hat{D}$. This dataset will be used in a semi-supervised learning on F' to release the final copy model F'.

From Fig. 4.1, MExMI has three key modules on top of the basic PAME iterative framework [76], namely, MI Pre-Filter, MI Post-Filter and semi-supervised boosting

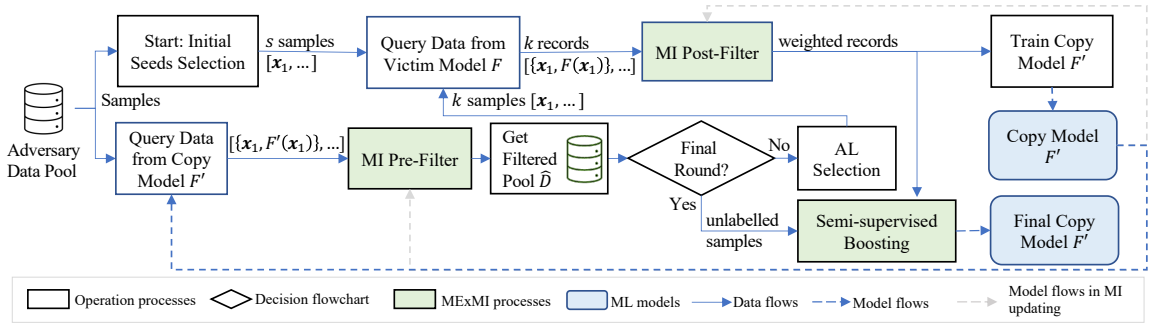


Figure 4.1: MExMI iterative framework.

Algorithm 1 MExMI attack

Input: F, P; **Parameters:** k, b, ω;
Output: F', $\hat{D}$;
$[\mathbf{x}_1, \ldots, \mathbf{x}_k]_0 \leftarrow \texttt{RandomPick}(P)$;
$P_r \leftarrow P \setminus [\mathbf{x}_1, \ldots, \mathbf{x}_k]_0$, $d \leftarrow d - k$, $i = 0$;
$D_q \leftarrow [\{\mathbf{x}_1, F(\mathbf{x}_1)\}, \ldots]_0$;
$F_{MIA} \leftarrow \texttt{MIUpdate}(D_q, P)$; ▷ Section 4.1.5
while $b >= 0$ **do**
    $D_{q+}, D_{q-} \leftarrow F_{MIA}(D_q)$;
    $\texttt{AssignWeight}(D_{q+}, D_{q-}, \omega)$;
    $F'_i \leftarrow \texttt{WeightLossTrain}(D_{q+}, D_{q-})$;
    $P_t \leftarrow F_{MIA}(F'_i, P_r)$;
    $[\mathbf{x}_1, \ldots, \mathbf{x}_k]_{i+1} \leftarrow \texttt{ActiveLearning}(P_t, F'_i)$;
    $P_r \leftarrow P_r \setminus [\mathbf{x}_1, \ldots, \mathbf{x}_k]_{i+1}$,
    $D_q \leftarrow [\{\mathbf{x}_1, F(\mathbf{x}_1)\}, \ldots]_{i+1} \cup D_q$, $d \leftarrow d - k$, $i \leftarrow i + 1$;
end while
$\hat{D} \leftarrow F_{MIA}(F', P)$;
$F' \leftarrow \texttt{SemiSupervisedTrain}(D_q, \hat{D})$;

that will be elaborated in rest of this section. As will become clear, they are orthogonal to each other, so they can be turned on or off independently. For ease of presentation, the construction of the MI attack model F_{MIA}, a key issue in the MExMI framework, will be introduced later in Section 4.1.5.

4.1.1 MI Pre-Filter

MI Pre-Filter works before AL sample selection. The core idea is to use an MI attack model F_{MIA} to choose from the adversary data pool P only those samples that are highly homogeneous to the victim's training data D. Training the copy model with them, both models can thus exhibit strong resemblance. Ideally, F_{MIA} should perform membership inference attack directly on the victim model F, which is truly trained from D. However, since such training causes extra query budget on F, F_{MIA} attacks the copy model F' instead as the latter exhibit similar training data property.

4.1.2 MI Post-Filter

MI Post-Filter works after querying a sample $\mathbf{x}$ from the victim model. The rationale of this filter is that since the victim model returns the probability vectors $F(\mathbf{x})$, the adversary can infer if $\mathbf{x}$ belongs to the training dataset D by an MI attack model F_{MIA}. Obviously a negative membership result means this sample may not lead to a high-fidelity copy model F', so its contribution to the training process should be lowered by reducing its weight in the training loss calculation. We use a parameter ω ($\omega > 1$) to denote the weighted loss ratio of a positive membership sample to a negative one.

4.1.3 Semi-Supervised Boosting

The main idea of this module is that MExMI gains the results of its MI attack— an inferred training dataset $\hat{D}$ of the victim model. Although this dataset is not labeled and there is no more query budget to label them at the end of MExMI, we can still train the copy model F' on this dataset together with the queried set using *semi-supervised learning* algorithms. Note that this module is unique in MExMI as other PAME methods cannot distinguish training samples from others in the data pool.

Nonetheless, this module is intended to improve model accuracy only, not fidelity, because semi-supervised learning can divert the copy model's ability to follow the same label distribution as the victim model's training data. Therefore, **MExMI only applies semi-supervised boosting after the final iteration and when higher accuracy is needed**.

4.1.4 Pool-based Active Learning Algorithms

MExMI can use various pool-based AL algorithms available in the literature. In this paper, we focus on three AL algorithms based on different metrics of samples: uncertainty, diversity, and vulnerability to deep-fool perturbation. As AL algorithms often calculate the distance between samples [118], MExMI also provides an encoding process to reduce dimension for high-dimensional feature space. In essence, it uses F'_{i-1}, the copy model trained in the previous iteration, as an encoder. When inter-sample distances are required, we use inter-vector distances between the outputs of the encoder for calculation. The distance is calculated using the output probability vectors of F' whenever the calculation of the distance is needed.

4.1.4.0.1 Entropy Uncertainty One of the most common ways to measure uncertainty is entropy [62]. The larger the entropy, the higher the uncertainty level is. For a sample $\{\mathbf{x}, \hat{\mathbf{y}}\}$ in the data pool, where $\hat{\mathbf{y}} = F'(\mathbf{x}) = [\Pr(1|\mathbf{x}), \Pr(2|\mathbf{x}), \ldots, \Pr(K|\mathbf{x})]^T$, its entropy is defined as

$$\Phi_{ent}(\mathbf{x}) = -\sum_{k=1}^{K} \Pr(k|\mathbf{x}) \log(\Pr(k|\mathbf{x})), \tag{4.1}$$

where K is the number of class labels.

4.1.4.0.2 Greedy K-center One classic diversity-based AL is the greedy k-center algorithm [88]. Let $D_q := [\{\mathbf{x}_q, F(\mathbf{x}_q)\}, \ldots]$ denote the set of samples se-

lected previously, and $P := [\mathbf{x}_p, \ldots]$ the data pool. Greedy k-center algorithm sets $[F'(\mathbf{x}_q), \ldots]$ as cluster centers and selects the sample $\mathbf{x}_s$ that has the largest Euclidean distance from all existing centers. Formally,

$$\mathbf{x}_s = argmax_{\mathbf{x}_p \in P}\{min_{(\mathbf{x}_q, \mathbf{y}_q) \in D_q} \|F'(\mathbf{x}_p) - F'(\mathbf{x}_q)\|_2^2\}. \tag{4.2}$$

Then we query the selected sample $\mathbf{x}_s$ and update D_q by $D_q = D_q \cup \{\mathbf{x}_s, \mathbf{y}_s\}$, where $\mathbf{y}_s = F(\mathbf{x}_s)$.

4.1.4.0.3 Adversarial Deep-Fool Deep-Fool based AL (DFAL) uses sample perturbation attack in Deep Fool to calculate the distance between samples and decision boundaries, a.k.a., margin, and then selects those with the smallest perturbation distance to query [32]. The perturbation algorithm iteratively perturbs samples by adding linear noise until the samples are misclassified by the copy model F'.

4.1.5 Adaptive Membership Inference

In MExMI, an MI attack model F_{MIA} is trained after the initial seeds query. For MI to play a role in MExMI framework in early stage iterations, F_{MIA} must be accurate enough even when there are only few samples. In this section, we renovate existing MI algorithms to be adaptive to their training sample sizes and thus suitable for the MExMI framework. We focus on two state-of-the-art black-box MI attack paradigms: (1) shadow-model MI [84, 90], and (2) unsupervised MI [84].

4.1.6 Shadow-Model Membership Inference

The rationale of shadow-model MI is to obtain a shadow model similar to the victim model, so that their output probability vectors for training and non-training samples are also distinguishable in a similar manner. As such, the adversary can build a

binary MI classifier from these samples instead of from those of the victim model. In order to approximate the victim model, a shadow model should (1) draw its training dataset from the same distribution as in the victim model, (2) have roughly the same size of training set as the victim model, and (3) have the same training algorithm. However, in MExMI, neither (1) nor (2) holds because:

a) The queried samples may not be drawn from the same distribution as in the victim's training samples.

b) The number of queried samples is significantly smaller than that of the victim's training set size, especially in the beginning phase.

As such, the design principle of our adaptive shadow-model MI attack is to work under **a limited number of labeled training samples in a different distribution** from the victim model. To start with, we need to know how to estimate the quality of a shadow model so that we can tell when it is good enough for MExMI. Intuitively, the fidelity of the shadow model against the victim model can serve as the performance indicator, but it is inaccessible from the adversary's side, especially when MExMI just starts. As such, we need an easy-to-access performance metric.

4.1.6.0.1 Measuring Quality of Shadow-model MI An MI attack works by distinguishing the output probability vector distribution $\mathbb{Y}$ of the victim model F on training samples $\mathbb{X}$ from non-training samples. A shadow-model MI estimates the above on a shadow model — it distinguishes the output probability vector distribution $\mathbb{Y}^{(s)}$ of shadow models F_s on training samples $\mathbb{X}^{(s)}$ from non-training samples. As such, to improve the attack accuracy, $\mathbb{Y}^{(s)}$ should be as similar as possible to the victim's $\mathbb{Y}$. We measure the similarity by the bias of the expectation values between $\mathbb{Y}^{(s)}$ and $\mathbb{Y}$, denoted by $\mathbf{b}$. For each shadow model that targets at class $j \in [1, \ldots, K]$, $\mathbf{b}_j$ is formally defined as:

$$\mathbf{b}_j = \frac{1}{n_j^{(s)}} \sum_{\mathbf{y}^{(s)} \in \mathbb{Y}_j^{(s)}} \mathbf{y}^{(s)} - \frac{1}{n_j} \sum_{\mathbf{y} \in \mathbb{Y}_j} \mathbf{y}, \tag{4.3}$$

where n denotes the size of training set, and the superscript (s) denotes the shadow model. The following theorem shows that the gap between the training loss Δl is positively correlated to the bias of expectations between $\mathbb{Y}^{(s)}$ and $\mathbb{Y}$. Therefore, it can serve as a quality measurement for F_s. We can minimize it to enhance the quality of $\mathbb{Y}^{(s)}$ approximating $\mathbb{Y}$.

Theorem 1 (**Quality measurement for shadow-model MI**). Given a shadow model F_s which has the same model structure and hyper-parameters as the victim model F, the gap Δl between the training loss of F_s and F is positively correlated to the bias of expectations between $\mathbb{Y}^{(s)}$ and $\mathbb{Y}$, i.e., $\Delta l \propto \mathbf{b}$.

Proof. For a multi-classifier with K labels and n training samples, the training loss l is measured using cross-entropy:

$$l = -\frac{1}{n} \sum_{\mathbf{x} \in \mathbb{X}} \sum_{y=1}^{K} \Pr(Y = y|\mathbf{x}) \log(\Pr(\hat{Y} = y|\mathbf{x})), \tag{4.4}$$

where Y is the ground truth label variable, and $\hat{Y}$ is the predicted label variable. Since the ground truth probability vectors are in one-hot format, the training loss in the j-th class, l_j, can be rewritten as:

$$l_j = -\frac{1}{n} \sum_{\mathbf{x} \in \mathbb{X}_j} \log(\Pr(\hat{Y} = j|\mathbf{x})) \tag{4.5}$$

Therefore, the gap of loss in the j-th class between the distribution of the shadow model $\mathbb{Y}^{(s)}$ and that of the victim model $\mathbb{Y}$ is:

$$\Delta l_j = \frac{1}{n^{(s)}} \sum_{\mathbf{y}^{(s)} \in \mathbb{Y}_j^{(s)}} \log(y_j^{(s)}) - \frac{1}{n} \sum_{\mathbf{y} \in \mathbb{Y}_j} \log(y_j), \tag{4.6}$$

where y_j represents the j-th element of $\mathbf{y}$. On the other hand, we define b'_{jj} as the log-bias between the two logarithm distributions for class j:

$$b'_{jj} \equiv \frac{1}{n_j^{(s)}} \sum_{\mathbf{y}^{(s)} \in \mathbb{Y}_j^{(s)}} \log(y_j^{(s)}) - \frac{1}{n_j} \sum_{\mathbf{y} \in \mathbb{Y}_j} \log(y_j). \tag{4.7}$$

From Eqn. 4.6 and 4.7, we get:

$$b'_{jj} = \frac{\Delta l_j}{a_j}, \tag{4.8}$$

where a_j is the proportion of class j in both shadow and victim model training sets. $\mathbf{b}_j \in \mathbb{R}^K$ is a vector, and its element in dimension j is denoted by b_{jj}.

To complete the proof, in Eqn.4.8 we need to replace b'_{jj} with b_{jj}, the j-th element in vector $\mathbf{b}_j$. This replacement is correct because of the following two assumptions, whose validity will be verified experimentally in Section 4.2.

Assumption 1. The correlation coefficient ρ between a distribution and its logarithm distribution is positive. Formally,

$$\rho_{|b'_{jj}|,|b_{jj}|} > 0, \tag{4.9}$$

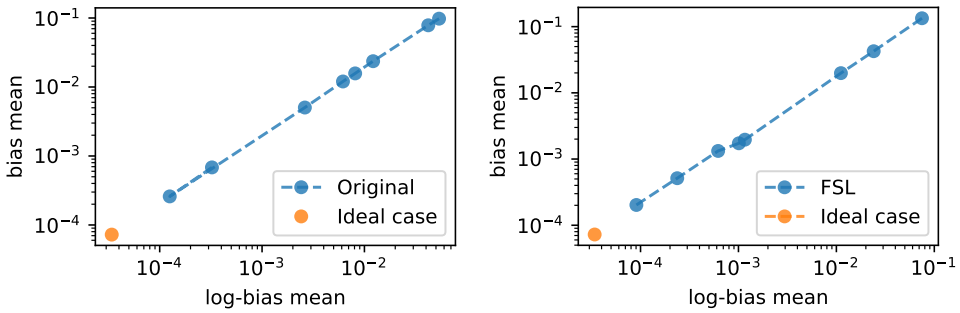
where $j \in [1, \ldots, K]$.

Assumption 2. For two models with the same structure and hyper-parameters, for any class j, the correlation coefficient ρ between the output vector distribution and its j-th element is positive. Formally,

$$\rho_{|b_{jj}|,|\mathbf{b}_j|} > 0, \tag{4.10}$$

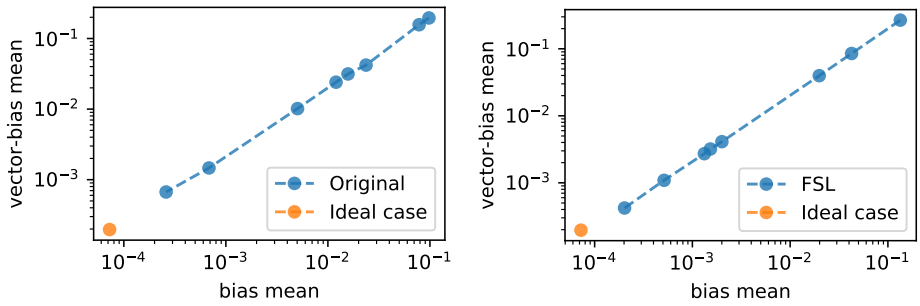
where $j \in [1, \ldots, K]$.

Based on the above two assumptions, we replace b'_{jj} with b_{jj} in Eqn.4.8 and obtain $\Delta l_j \propto \mathbf{b}_j$. □



(a) original shadow-model MI (b) FSL shadow-model MI

Figure 4.2: The results of log-bias mean vs. bias mean.



(a) original shadow-model MI (b) FSL shadow-model MI

Figure 4.3: The results of bias mean vs. vector-bias mean.

To verify Assumptions 1 and 2 of Theorem 1, we calculated the *bias mean* as the mean of bias of all classes, i.e., $\sum_{j=1}^{K} |b_{jj}|$, the *log-bias mean* as the mean of log-bias of all classes, i.e., $\sum_{j=1}^{K} |b'_{jj}|$, and the *vector-bias mean* as the mean of scalar bias $\sum_{dim} \sum_{j=1}^{K} |\mathbf{b}_j|$ of all classes. See Section 4.2.1 for experimental setups. The results are shown in Figure 4.2 and Figure 4.3 respectively. We observe that:

1 The bias mean is positively correlated with the log-bias mean, which justifies Assumption 1.

2 The vector-bias mean is positively correlated with the bias mean, which justifies Assumption 2.

4.1.6.0.2 Metric-based Shadow-model MI To obtain Δl, the gap between the training loss of both models, we need them both. However, the victim's training loss

is not available to the adversary. Nonetheless, in practice a victim model is valuable for extraction mainly because this model accurately predicts the training data, or equivalently, its training loss is smaller than other models of the same training set size. As such, we can replace Δl with the loss of the shadow model, denoted by l_s, compensated by its training dataset size $n^{(s)}$. Furthermore, for the sake of comparing various shadow models, only the relative rather than the absolute value of the gap between shadow models and the victim model ΔF is needed. So we propose the following metric Q as a negative relative value of ΔF. The larger the Q, the better the quality of a shadow model. Formally,

$$\frac{1}{\Delta F} \sim Q(l_s, n^{(s)}) = f(l_s) \times (n^{(s)})^a, \tag{4.11}$$

where $f(\cdot)$ is a non-negative non-linear decreasing function, and $a \in [0, 1]$ scales down the impact of $n^{(s)}$. Note that Q does not require that shadows' training data come from the same distribution as the victim's.

From the above equation, there are two ways to increase the quality metric Q: (1) reducing l_s using good training algorithms on shadow models, and (2) increasing $n^{(s)}$ by augmenting the training set. Few-shot learning (FSL), a training paradigm to improve models' accuracy with a limited number of examples [104], can serve both purposes. For example, we can use FSL approaches, namely data augmentation [58] and transfer learning [45], to train shadow models. In MExMI, we use the following two FSL approaches to train shadow models, leading to two metric-based shadow-model MI.

Data-augmented shadow-model MI. Data augmentation expands the training dataset to accelerate convergence by adding synthetic samples transformed from existing samples. In image classification, an image can be flipped, panned or rotated to enrich the training set [58].

Transfer shadow-model MI. Transfer learning shares the knowledge from a source

domain to reduce the training complexity in a target domain. In image classification, NN models are suitable for transfer learning, because their shallow layers learn task-independent abstract features, and deep layers are more task-related [45]. As such, an adversary can transfer the shallow layers of a pre-trained NN model to initialize a shadow model's parameters and to accelerate its convergence. This technique is valid even if the pre-trained model's problem domain is different from that of the shadow models.

4.1.7 Unsupervised Membership Inference

Recent works [84, 111] have shown the effectiveness of unsupervised learning on MI attack models. In such models, the feature values are usually the top-m score, loss or entropy of the output probabilities, and the output value serves as the confidence of membership inference — if the value is higher than an adversary-specified threshold c, the sample is inferred as in the training dataset and vice versa. To set this threshold, the adversary first gets a set of non-member samples and then query them to get corresponding top-m scores. The top t percentile of these scores can serve as a threshold [84]. To save the query cost, the copy model instead of the victim model should be queried.

4.2 Experiment

In this section, we first conduct experiments to validate the shadow model quality metric Q. Then we evaluate the attack performance of four variants of MExMI — Pre-Filter only, Post-Filter only, MExMI without semi-supervised boosting, and MExMI — against state-of-the-art ME and MI attacks.

4.2.1 Experiment Setup

Datasets. We perform PAME attacks on two image datasets, namely CIFAR10 [57] and Street View House Numbers (SVHN) [73], and a text dataset AG'S NEWS which contains corpus of AG's news articles [28]

- **CIFAR10**. CIFAR10 is an image dataset in color (with 3 channels) with 10 class labels, 50k training samples, and 10k test samples. The image samples have a resolution of 32 and are evenly distributed into 10 classes. It is a well-known benchmark dataset to evaluate image classifiers.
- **SVHN**. SVHN is another 32-resolution benchmark for color image classification. It consists of street-view images of door numbers, which are labeled with digits from "0" to "9". The dataset contains 73,257 images for training, and 26,032 images for testing.
- **AG'S NEWS**. AG'S NEWS is a benchmark for text classification. It consists of titles and descriptions of articles from 4 news classes, namely "World", "Sports", "Business" and "Sci/Tech". This dataset contains 120k training samples and 7.6k test samples.

Victim Model. For the image classification task, we use Wide-ResNet-28-10 [117] trained on CIFAR10 as the victim model with an accuracy of 96.10%. We also use a cloud MLaaS, ModelArts [9], to train an online victim model on SVHN that leads to 94.30% accuracy. In the text classification task, we use DPCNN [51] as the victim model which achieves an accuracy of 89.88%.

Running Environment. Experiments are implemented with Python 3.7 on a desktop computer running Windows 10 with AMD Ryzen 7 2700X CPU and 64 GB RAM. All experimental results are the average measures of 5 trials.

Adversarial data pool. In the default CIFAR10 experiments, the pool consists

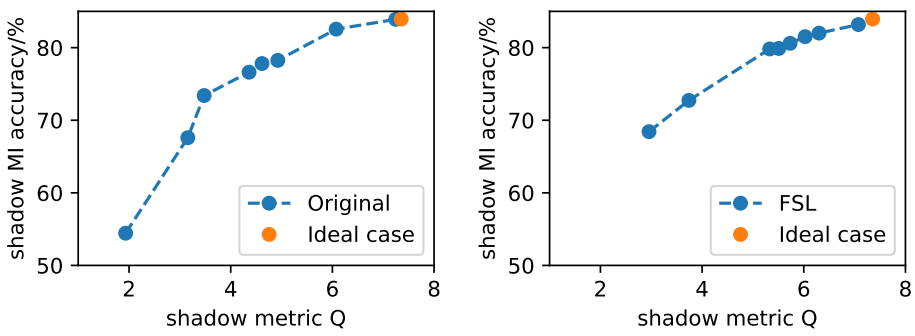
of 50k training samples and 100k from the ImageNet32 [26]. In the default AG'S NEWS experiments, the pool has 50k training samples and 100k from Dbpedia [16]. Note that as with existing pool-based ME [76], **MExMI does not require the pool to contain training samples**. The main reason for such a mixed dataset composition is for us to evaluate the performance of MI [90] and show how much it can be enhanced by ME. In Section 4.2.4, we evaluate the performance of MExMI when the pool has no training sample at all.

Implementation Details for ML-leaks [84] Membership Inference Attacks. For transfer shadow-model MI, it needs prior knowledge of the source model. In the experiment, we use a 5-block model with the same shallow structure as the victim model as the source model and train it on CIFAR100. We then transfer the parameters of the three shallow blocks to initialize the shadow models. For data-augmented shadow-model MI, we add augmented samples to the training dataset to double its size. We adopt the same augmentation policy as in [117], which includes inverting, rotating, sharpening etc., but excludes those methods used to train models for a fair comparison. The test dataset consists of 10k victim's training images as positive samples and another 10k non-training images as negative samples.

4.2.2 Metric-based Shadow-Model MI

Implementation Details. Recall that the hyper-parameters (such as epoch, initial learning rate, and optimizer) of shadow models F_s can be adjusted to maximize the metric Q. Parameter a in Q is set as 0.05 and $f(\cdot)$ is set as $-log_{10}(\cdot)$.[1] The training dataset of F_s, denoted by $D^{(s)}$, is constructed by random sampling from P, and its size varies from the set $\{2000, 5000\}$. A non-training dataset, denoted by $D_n^{(s)}$, is also randomly sampled from $P \setminus D^{(s)}$. We implement two metric-based shadow-model MI methods in Section 4.1.5: *original shadow-model MI* [90] and *FSL shadow-model MI*

[1] In our experiments $l_s \in (0, 1)$ where $-log_{10}(\cdot)$ is non-negative and ever decreasing.



(a) Original (b) FSL

Figure 4.4: Experiment results of metric-based shadow-model MI under different settings.

that utilizes transfer learning [45] and data augmentation respectively. For transfer shadow-model MI, it needs prior knowledge of the source model. In the experiment, we use a 5-block model with the same shallow structure as the victim model as the source model and train it on CIFAR100. We then transfer the parameters of the three shallow blocks to initialize the shadow models. For data-augmented shadow-model MI, we add augmented samples to the training dataset to double its size. We adopt the same augmentation policy as in [117], which includes inverting, rotating, sharpening etc., but excludes those methods used to train models for a fair comparison. For each MI method, we vary their hyper-parameters and thus Q in various settings. An ideal shadow-model MI is built as a reference by using the same training dataset as the victim model.

Results. We plot the MI attack models' accuracy with respect to shadow metric Q in Fig. 4.4. We observe that Q is positively correlated with the attack accuracy of shadow-model MI and therefore can guide the training process of the shadow models, whether using the original or FSL training algorithms. The recall rate of all MI attacks is almost 100%, so it is omitted from the figure. In addition, once the metric Q is large enough (≥ 6), metric-based shadow-model MI attack models can achieve almost the same accuracy as the ideal shadow-model MI [84] even for 2k samples.

4.2.3 Overall Performance of MExMI

4.2.3.1 Implementation Details

We compare six PAME attacks, including four MExMI variants, namely, baseline ME without MI, Pre-Filter only, Post-Filter only, MExMI without semi-supervised boosting, regular MExMI (which adopts Mix-Match semi-supervised methods [18] for image classifiers, and consistency regularization [83] semi-supervised methods for text classifiers), and the ideal ME attack. The baseline ME is ActiveThief [76], which is the state-of-the-art PAME attack.[2] The ideal ME uses the real training samples as its pool. Moreover, to comprehensively evaluate MExMI, we added two additional state-of-the-art query-synthesizing-based ME baselines, including PRADA [52] and DFME [101] which share the same query budget with PAME attacks. Each MExMI variant has a MI result. In our experiments, we compare our MI attacks that don't cost additional query budget with the existing MI attacks [84] that assume infinite query budget.

The AL algorithms used are entropy uncertainty [62], greedy k-center [88] and adversarial deep-fool [32] (see Section 4.1.4 for a brief explanation). The last algorithm is not evaluated on AG'S NEWS since there is no trivial way to adapt it to text classification. The hyper-parameters are set as follows: *initial learning rate* (lr)=0.03 for image classifiers, lr=0.01 for text classifiers, *momentum*=0.5, *weight decay*=$10e-4$, *epochs*=150. The optimization method is SGD accelerated by Nesterov Gradient Method [30]. The copy models in MExMI share the same architecture as the victim models. The testing samples in CIFAR10, AG'S NEWS, and SVHN are used as the test dataset D_t for calculating the fidelity and accuracy of the copy models, respectively. The MI attack model F_{MIA} is trained on initial seed samples. The preset

[2]INVERSENET [37] is another state-of-the-art work that adopts model inversion to assist ME. We do not include it in the experiments for two reasons. First, its performance is similar to ActiveThief under our 16k-query budget setting. Second, it augments query selection from the data pool with query synthesis from the model, which can be considered orthogonal to our method.

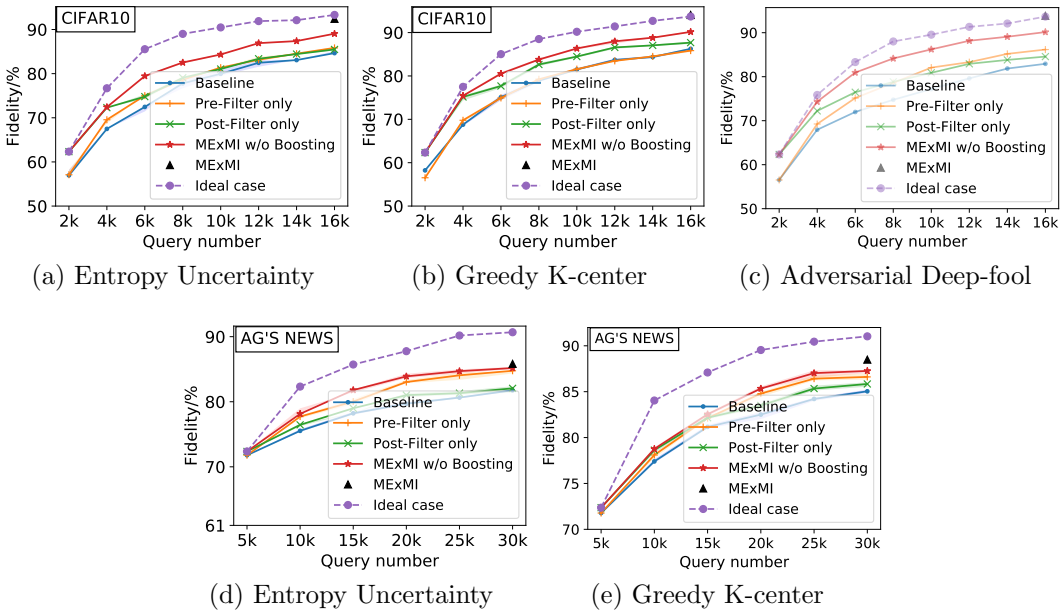
weights ratio ω in MI Post-Filter is 5 : 1. For CIFAR10 experiments, MExMI queries 2k samples in each round with a total of 8 rounds. For AG'S NEWS experiments, there are 6 rounds, each with 5k samples. All experimental results are the average measures of 5 trials. For CIFAR10 experiments, MExMI queries 2k samples in each round with a total of 8 rounds. For AG'S NEWS experiments, there are 6 rounds, each with 5k samples. To be fair, all attacks, including the ideal one, use the same initial seed samples. The adaptive shadow-model MI used in MI Pre-Filter and MI Post-Filter is trained on initial seed samples for 150 epochs. The preset weights ratio ω in MI Post-Filter is 5 : 1.

4.2.3.2 Overall Results of MExMI

We use fidelity and model's accuracy obtained from the test datasets D_t (CIFAR10 and AG'S NEWS test sets) to evaluate copy models against the victim model (see Section 3.1.2), and use accuracy, precision and recall of the inferred training datasets against ground truth to evaluate the MI accuracy.

Fig. 4.5 plots the fidelity of various PAME attacks with respect to iterations on CIFAR10 and AG'S NEWS, respectively [3], and Table 4.2 shows the final results. Fig. 4.6 plots the accuracy, precision and recall of the MI attack of each MExMI variant. The results indicate that MExMI greatly boosts the potential of AL algorithms in PAME, and breaks the curse of query budget of existing MI. Overall, MExMI performs the best and achieves a fidelity gain of 7.76%, 7.8%, and 11.14% on CIFAR10 over the baseline ME attack in all three AL methods. A similar gain of 4.46% and 3.46% is observed on AG'S NEWS over the baseline PAME attacks in both two AL methods. Without additional queries, the MI attacks of MExMI yield up to 83.20% accuracy, 84.13% precision and 75.93% recall on CIFAR10, and 68.77% accuracy, 71.73% precision and 82.53% recall on AG'S NEWS respectively, both on par with existing MI

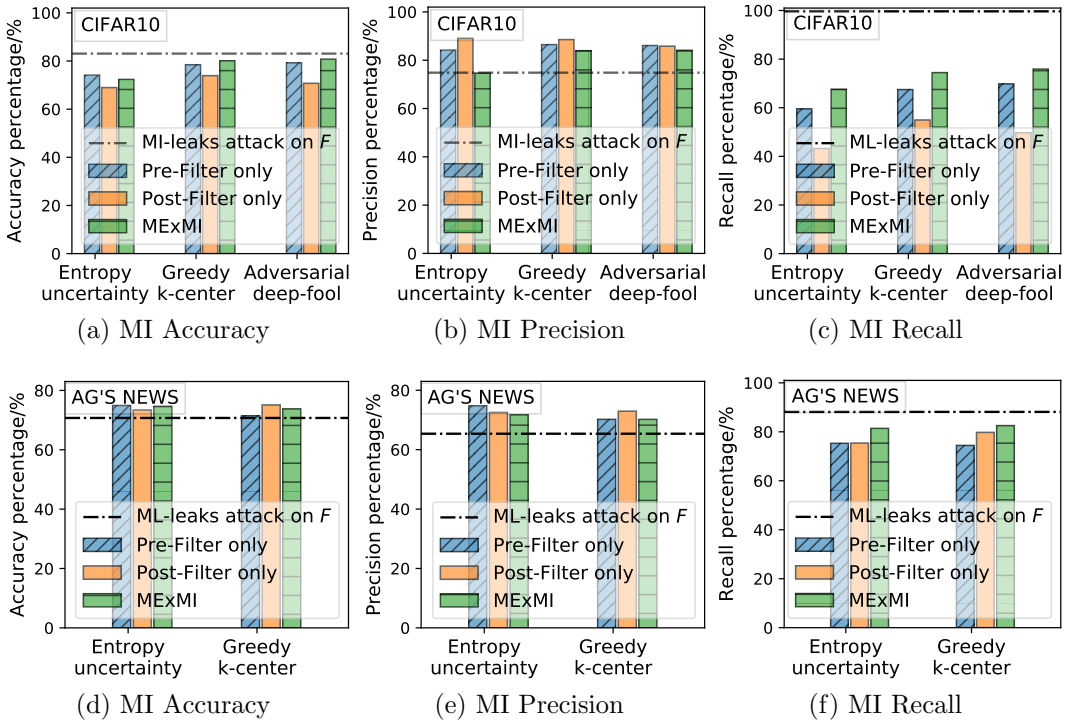
[3]The label 'MExMI w/o Boosting' in figures is an abbreviation for MExMI without semi-supervised boosting.



(a) Entropy Uncertainty (b) Greedy K-center (c) Adversarial Deep-fool

(d) Entropy Uncertainty (e) Greedy K-center

Figure 4.5: PAME results on CIFAR10 (16k budget) and AG'S NEWS (30k budget). Shadows represent error bars.



(a) MI Accuracy (b) MI Precision (c) MI Recall

(d) MI Accuracy (e) MI Precision (f) MI Recall

Figure 4.6: MI attack results of MExMI. "ML-leaks" refers to the shadow-model MI attack in [84].

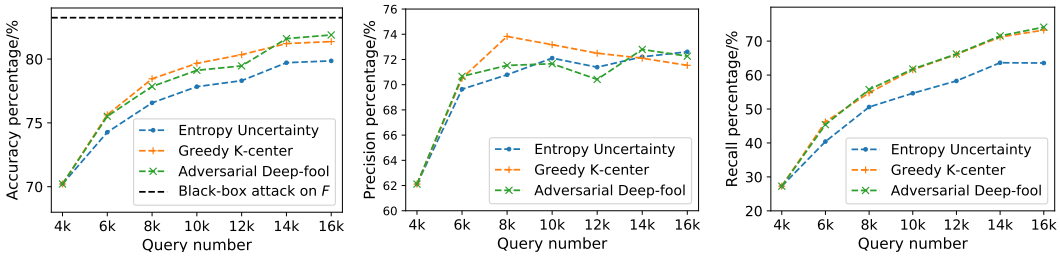
Table 4.2: Results of Default PAME Experiments

Fidelity (Accuracy)/%	**CIFAR10**			**AG'S NEWS**	
	Entropy Uncertainty	**Greedy K-center**	**Adversarial Deep-fool**	**Entropy Uncertainty**	**Greedy K-center**
PRADA [52]/DFME [101]	61.32 (60.12) / 11.20 (10.32)			- / 30.23 (25.00)	
Baseline(ActiveThief)	84.65 (83.78)	86.26 (85.69)	82.93 (82.27)	81.36 (78.66)	85.03 (81.92)
Pre-Filter only	85.38 (85.38)	85.84 (85.22)	86.17 (85.48)	84.76 (81.57)	86.61 (83.36)
Post-Filter only	85.48 (84.71)	87.68 (86.86)	84.57 (84.00)	82.06 (79.29)	85.84 (82.38)
MExMI w/o Boosting	89.10 (88.69)	90.16 (89.21)	90.14 (89.58)	85.18 (81.98)	87.26 (84.18)
MExMI	**92.41(91.80)**	**94.06(93.43)**	**94.07(93.47)**	**85.82(82.54)**	**88.49(85.36)**
Ideal case	93.31 (93.03)	93.71 (93.38)	93.66 (93.25)	90.68 (87.51)	91.03 (87.68)

attacks [84] that assume infinite query budgets.

Impact of MI Post-Filter and MI Pre-Filter. In Fig. 4.5, on CIFAR10 attacks with MI Post-Filter always outperform those without it, by up to a 1.64% increase on fidelity in the final results. The gain is more eminent in the beginning iterations, and then gradually decreases. On AG'S NEWS, MI Post-Filter also performs effectively, with a maximum boost of 1.57% on fidelity over those without it. As for MI Pre-Filter, except for the greedy k-center one in CIFAR10 experiments, attacks with Pre-Filter always outperform those without it. In addition, the gain does not decrease with more iterations, because the adversary data pool is much larger than the total query budget. To understand the underlying mechanism why MI Pre-Filter works, we track the filtering results of Pre-Filter in each iteration of MExMI attack on CIFAR10. The results are shown in Fig. 4.7. We observe that MI Pre-Filter can accurately find victim's training samples in the remaining pool, so the training set for copy model is gradually restored through iterations. Interestingly, we find that when MI Pre-filter and MI Post-filter work together, they can achieve a greater gain on fidelity than the sum of individual gains when they work separately. This suggests that **the two filters truly reinforce each other in our MExMI framework.**

Impact of Semi-Supervised Boosting. In Fig. 4.5 and Table 4.2, MExMI outperforms MExMI without semi-supervised boosting by at least 3.11% on CIFAR10 and 0.56% on AG'S NEWS in terms of accuracy, which indicates that MExMI does benefit from effective MI attacks. A fidelity gain is observed in MExMI since the copy



(a) MI Accuracy (b) MI Precision (c) MI Recall

Figure 4.7: Performance of MI Pre-Filter in MExMI on CIFAR10.

models' accuracy is closer to that of the victim model.

MI Performance in MExMI. The precision and recall of the adaptive shadow-model MI attacks of three MExMI variants are shown in Fig. 4.6. MExMI always performs the best and can achieve up to 83.20% accuracy and 84.13% precision on CIFAR10, and 68.77% and 71.73% precision on AG'S NEWS. This precision is even better than the state-of-the-art MI — ML-leaks [84] (75.25% on CIFAR10, 65.37% on AG'S NEWS) which assumes unlimited query budget. Furthermore, our adaptive MI attacks have no additional cost when inferring training samples.

Discussion About Potential Defenses. There are two potential defenses against MExMI. First, MExMI is subject to MI-related defensive strategies that can reduce the accuracy of MI, such as using differential privacy [31], which in turn lowers the fidelity of MExMI. Second, as with other model extraction attacks, MExMI can also be defended by model provenance techniques, such as watermark embedding [50], to detect copyright infringement from an extracted model.

4.2.4 Impact of Adversary Data Pool on PAME

In this experiment, we study how the quality of adversary data pool affects the outcome of PAME attacks. Since there are cases where the data pool does not contain any training data, the results of MI attacks are not evaluated. To be fair, we fix the

Table 4.3: Impact of the Adversary Data Pool on PAME Attacks with 16k Query Budget

The Proportion of P_v in P		Baseline	Pre-Filter only	Post-Filter only	MExMI w/o Boosting	MExMI
Fidelity (Accuracy)/%	**0%**	76.30(75.62)	74.23(73.93)	76.29(75.71)	77.40(76.80)	**79.11(78.51)**
	25%	81.07(80.48)	83.91(83.50)	83.55(83.05)	89.99(88.78)	**92.91(91.96)**
	33.33%	82.93(82.27)	86.17(85.48)	84.57(84.00)	90.14(89.58)	**94.07(93.47)**

Table 4.4: Impact of Output Access on PAME Attacks

Fidelity (Accuracy)/%		Baseline	Pre-Filter only	Post-Filter only	MExMI w/o Boosting	MExMI
Output Access	**Probabilities**	82.93(82.27)	86.17(85.48)	84.57(84.00)	90.14(89.58)	**94.07(93.47)**
	Top-1 Scores	79.53(79.00)	81.89(81.33)	81.94(81.18)	85.60(84.89)	**91.00(90.40)**

size of the pool to 150k samples and change the proportion of training data in it to vary its quality. Since the total training samples are 50k, this proportion is capped at 1/3.

The results on CIFAR10 are shown in Table 4.3. We can see that the quality of the data pool greatly affects each PAME attack. MExMI consistently outperforms the baseline irrespective of the quality, even in the complete absence of victim's training data, i.e., when the adversary has no access to the true training samples. In such extreme cases, we also observe that Pre-Filter only is outperformed by the baseline. This is due to the fact that the Pre-Filter cannot find any training data in the pool and thus excludes most of them for training. Since the filtered data pool is too small, active learning might not be effective.

4.2.5 Impact of Output Access

We investigate the impact of output access granted to our PAME attacks. We limit the output access to top-1 score and show the results on CIFAR10 in Table 4.4. It indicates that even with limited output access, the three modules of MExMI still perform consistently well. Among various PAME attacks, MExMI always performs the best and can outperform the baseline by 11.47% on fidelity.

Table 4.5: Transferability of FGSM attacks

Active Learning	**Transferability / %**		
	Entropy Uncertainty	**Greedy K-center**	**Adversarial Deep-fool**
Baseline	51.76	57.47	57.59
Pre-Filter only	55.96	59.72	**60.99**
Post-Filter only	59.17	62.53	57.90
MExMI	**63.87**	**62.66**	58.57

4.2.6 Transferability of Adversarial Attacks

We measured the transferability of adversarial samples obtained from the FGSM [38] adversarial attacks (at a rate of $\epsilon = 0.1$) on the PAME copy models. The transferability rate is the fraction of these samples misclassified by the victim model. The results on CIFAR10 are shown in Table 4.5. MExMI consistently has higher transferability rate than the baseline, indicating that our MExMI algorithms are also superior from this perspective.

4.2.7 Impact of Weight Ratio in MI Post-Filter

In MI Post-Filter, we introduce a weight ratio ω between loss weights of the inferred training data and non-training data, which has an effect on both Post-Filter only and MExMI without semi-supervised boosting variants. In this experiment, we vary ω in CIFAR10 experiments and show the results in Table. 4.6. We observe that ω has a very limited impact on the fidelity and therefore our MExMI framework is robust to this parameter.

Table 4.6: Impact of Post-Filter Weight Ratio

Weight Ratio	**Fidelity (Accuracy) / %**		
	3:1	**5:1**	**7:1**
Post-Filter only	85.12 (84.20)	84.57 (84.00)	84.59 (83.93)
MExMI w/o Boosting	89.61 (99.84)	90.14 (89.58)	90.21 (89.61)

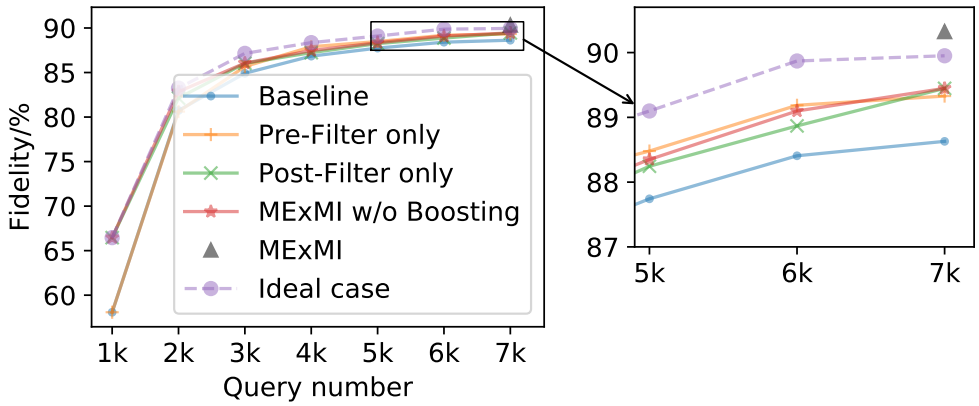


Figure 4.8: PAME attacks' results in ModelArts on SVHN (7k budget).

Table 4.7: MI attacks' results in ModelArts

MI Attacks	**Precision**/%	**Recall**/%
Pre-Filter only	86.94	91.69
Post-Filter only	87.17	90.50
MExMI	87.78	91.77
ML-leaks on F	87.83	92.38

4.2.8 Case Study: Blackbox Attacks on MLaaS ModelArts

We use a real case study to show the feasibility and real-life impact of MExMI. We target *ModelArts*, the MLaaS provided by Huawei Cloud [9] where developers can train and deploy their ML models in the cloud, and then access them via Web API (e.g., CURL).

We train and deploy a classification model on ModelArts without knowledge of its architecture using SVHN and then perform various PAME attacks on it. Our adversarial data pool consists of 2.5k SVHN and 5k ImageNet32 images as we have to measure both ME results and MI results of MExMI. Since ModelArts returns top-5 probabilities with three decimal places, we choose unsupervised MI as the MI module in MExMI. The parameter tolerant percentage t is set to 0.06 in the threshold decision method [84]. The architecture of the copy model is VGG16 with batch normalization. The query budget is 7k, and the size of initial seeds as well as the step are both 1k.

The experimental results of the adversarial deep-fool PAME are shown in Fig. 4.8. All

four MExMI variants outperform the baseline. MExMI achieves the highest 90.32% fidelity and MExMI without semi-supervised boosting comes the second with 89.45% fidelity. The final MI results of MExMI framework are shown in Table 4.7. We observe similar precision and recall of the three variants, all on par with ML-leaks [84], the state-of-the-art unsupervised MI attack that exhausts all pool data, which costs ten times higher.

4.2.9 Impact of ML Optimizations

As there are many emerging optimization methods in ML, in this subsection we investigate what impact they have on PAME attacks. In particular, we focus on the following two methods:

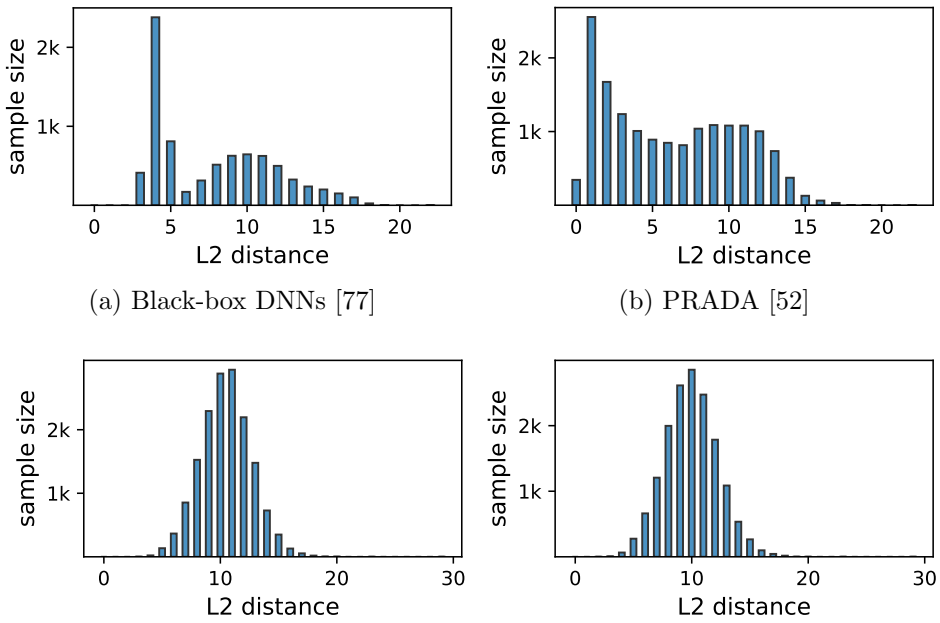
- **Data augmentation.** It is used in the training process to prevent overfitting. This method has become increasingly popular in the domain of image classification [78]. As shown in the experiments below, applying data augmentation in PAME can significantly improve the fidelity.

- **Ensembles for neural networks.** Ensembling a set of models trained separately is well known for effectively reducing generalization error [42]. As shown in the experiments below, applying ensembles in PAME can improve the fidelity.

We use the performance results in Section 4.2.3 on CIFAR10, especially MExMI without semi-supervised boosting, as the baseline in this experiment. We then use the transforming policy in [117] to perform a richer data augmentation and model averaging ensemble method. The results are shown in Table 4.8, where "Data-Aug" denotes data augmentation. Richer data augmentation improves baselines' fidelity by 1.18%, and the ensemble method further improves fidelity by another 1.52% to 92.84%. These results warn us that in a never-ending battle between model owners

Table 4.8: Performance Boosting Using Different ML Optimization Methods

Methods	**Fidelity (Accuracy) / %**		
	Baseline	**Baseline + Data-Aug**	**Baseline + Data-Aug + Ensemble**
Entropy Uncertainty	89.10 (88.69)	90.10 89.97)	**91.57 91.36)**
Greedy K-center	90.16 (89.21)	91.11 (90.98)	**92.86 (92.60**)
Adversarial Deep-fool	90.14 (89.58)	91.32 (90.80)	**92.84 (92.32)**



(a) Black-box DNNs [77]

(b) PRADA [52]

(c) ActiveThief [76]

(d) MExMI (this work)

Figure 4.9: Distribution of L2 distance required in PRADA defence.

and model extractors, emerging technologies in ML may favor the latter rather than the former.

4.2.10 The Ability of Evading PRADA Defence

For image classification, PRADA[52] is the state-of-the-art method to detect ME attacks. The detection is based on the distribution of consecutive query data, as PRADA believes that the adversary tends to issue query samples across an exceptional feature space. To evaluate the ability of MExMI evading such detection, we measure the minimal L2 distance between query samples of MExMI and several benchmark

ME attacks on CIFAR10. The results are shown in Fig. 4.9. We can see that MExMI and the baseline attack are not different from benign queries (subject to Gaussian distribution) and both cannot be detected by PRADA. In contrast, the distributions of PRADA attack[52] and Black-box DNNs[77] have distinct traits, which are thus easier to be detected.

Chapter 5

MEBooster: Towards Neuron-Grained Model Extraction

This chapter introduces MEBooster, a novel training framework that advances model extraction attacks. Grounded in neuron matching theory [99], MEBooster enables a neuron-level investigation into the replicability and impact of model extraction attacks.

To enhance extraction fidelity, MEBooster introduces two complementary training-phase strategies: bootstrapping the initialization and post-training fine-tuning. The first approach provides the copy model with a stronger initialization by estimating parameters of the victim model [86], and further mitigates estimation errors using a width-expanded architecture capable of accommodating multiple hypotheses. The second approach extends neuron alignment to deeper layers of the copy model [99], refining representation consistency. Notably, both techniques can be seamlessly integrated into existing learning-based model extraction attacks without modification or additional query cost.

MEBooster faces several challenges. First, existing parameter estimation methods are limited to two-layer linear neural networks [86]. In Section 5.2.3.2, 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 input samples, typically unavailable in practice. To address this, Section 5.2.3.1 incorporates score matching [93] to model data distribution using ML models with implicit score loss. Third, existing optimal convergence theories focus only on the lowest layer of neural networks [99]. We extend this theory to facilitate upper layers (see Section 5.2.5).

Roadmap. The rest of this chapter presents the details of MEBooster framework and its experimental evaluations. Section 5.1 provides an overview of the MEBooster framework. Section 5.2 elaborates on MEBooster's key components: initial bootstrapping and post-processing fine-tuning, both supported by neuron matching theory. Section 5.3 discusses the experimental results of MEBooster.

5.1 The ME Booster Framework

Framework Overview. Fig. 5.1 illustrates the general learning-based model extraction (ME) framework augmented by MEBooster (in the green area). This framework is an abstraction of existing learning-based model extraction attacks, *e.g.*, DFME [101] and ActiveThief [76]. 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 5.2.

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 [48]. In the learning-a-model scenario, studies [35, 115, 122] 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,

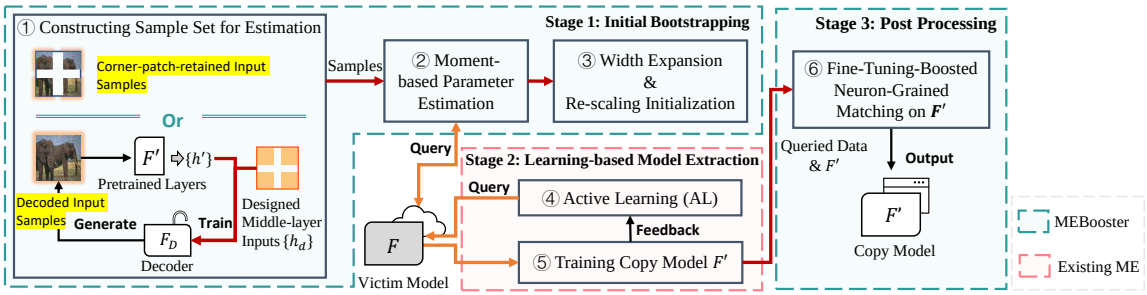


Figure 5.1: The MEBooster framework for learning-based ME.

compared to a randomly initialized model which may settle at a local optimum. Hence, MEBooster **allocates some query budget to derive a good 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 5.2.3). 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 [43] (Section 5.2.4).

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. In this stage, a current learning-based ME [53, 76, 101] 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 exist. First, earlier queried and trained samples impact the copy model less, 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 [99, 124]. Supported by it, we propose a post-processing step (step ⑥), fine-tuning (Section 5.2.5), to match more neurons, particularly in upper layers. This step also mitigates the first bias by utilizing all queried samples equally.

5.2 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, aimed at establishing 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 neural network layers.

5.2.1 High-level Solution

Studies [99, 124] demonstrate that lower-layer neurons in a copy model can align with those in the victim model via gradient descent (Section 5.2.2), *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 high-fidelity 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 the first two objectives are closer initial states to the victim model and an over-width architecture. Consequently, we design optimization modules for moment-based parameter estimation (Section 5.2.3) as well as width expansion (Section 5.2.4) during the initial bootstrapping phase, and 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.

5.2.2 Neuron Matching Theory

The phenomenon of neuron-grained matching (*i.e.*, convergence) in the lowest layer during "learning a model" [81, 99, 124] is due to gradient backpropagation. Theorem 1 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. 5.2.

Definition 1. *(Neuron Matching) On layer l, neuron j matches neuron k if they satisfy:*

$$\langle f_{l,j}(\{\boldsymbol{x}\}), f_{l,k}(\{\boldsymbol{x}\})\rangle \leq \epsilon, \tag{5.1}$$

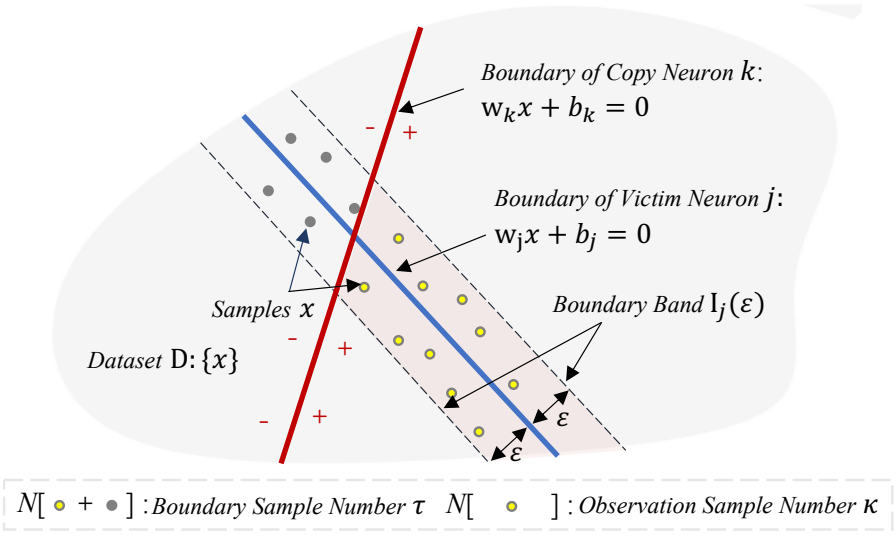


Figure 5.2: Illustration of observation sample number and boundary sample number in model extraction.

when ϵ is sufficiently small. $f_j(\boldsymbol{x})$ is the output vector of the l-layer neuron j over a batch of samples $\{\boldsymbol{x}_i\}_{i\in\{b\}}$.

Definition 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 1. *(Lowest-layer Neuron Matching, Theorem 5 in [99]). For a victim neuron j, if the lowest-layer neurons in copy model satisfy: (1) neuron j is observed by a copy neuron k with observation sample number:*

$$N[I_j(C\epsilon/\kappa) \cap E_k] \geq \kappa = O(Qd^{3/2}), \tag{5.2}$$

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 [99]), d is the input dimension, and C is a constant.[1]

Theorem 1 states that the lowest-layer neuron matching is theoretically guaranteed if the observation sample numbers for all victim neurons satisfy Equation 5.2. 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 re-scaling initialization to enhance the likelihood of copy neurons observing victim neurons. Next, we detail the three methods mentioned above: moment-based parameter estimation, width expansion, and re-scaling initialization.

5.2.3 Moment-based Parameter Estimation

The moment-based parameter estimation provides the copy model with much better initial parameters than Gaussian random vectors [43]. It is achieved by estimating the bases of a layer-wise span in DNNs [15, 49], starting from the lowest (linear) layer to the middle (convolutional) layers.

5.2.3.1 Moment-based Weight Estimation

The weight matrix of the lowest layer in the deep neural network can be inferred from the moment [49, 86], a statistical expected value of the output distribution's derivative relative to the input distribution. We begin by introducing the concept of moments, followed by a derivation of how to estimate weights from them. Let the

[1] C is the angle ratio of the weights of two neurons and their output vectors, which is architecture-dependent [99].

input distribution's probability density function be $p(\mathbf{x})$, and its score function $S(\mathbf{x})$ is defined as the gradient of the logarithm of $p(\mathbf{x})$ [49]

$$S(\mathbf{x}) = \nabla_{\mathbf{x}} log(p(\mathbf{x})) = \frac{\nabla_{\mathbf{x}} p(\mathbf{x})}{p(\mathbf{x})} \in \mathbb{R}^d. \tag{5.3}$$

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

$$M_1 = \mathbb{E}(F(\mathbf{x}) \otimes S(\mathbf{x})^T) \in \mathbb{R}^{K \times d}, \tag{5.4}$$

where $\otimes$ is the outer product of two vectors. The i-th row and j-th column element of M_1 is $F(\mathbf{x})_i \times S(\mathbf{x})_j$. According to Stein's Lemma [95], the moment M_1 can be expressed as:

$$M_1 = \mathbb{E}(F(\mathbf{x}) \otimes (\nabla_{\mathbf{x}} \log p(\mathbf{x}))) = -\mathbb{E}(\nabla_x F(\mathbf{x})). \tag{5.5}$$

According to Stein's Lemma 5.5, moments can be expressed as linear mappings between the lowest-layer weights of F. Formally, it's expressed in Theorem 2 as follows.

Theorem 2. *(Linear Mapping from The First Layer Weight to The Moment. Theorem 1 in [86]) For an MLP model F with first-layer weight matrix $W_1 = [w_1, \ldots, w_n]^T \in \mathbb{R}^{n \times d}$,*

$$\begin{aligned} M_1 &= \mathbb{E}(F(\boldsymbol{x}) \otimes \nabla_{\boldsymbol{x}} log(p(\boldsymbol{x}))) = -\mathbb{E}(\nabla_{\boldsymbol{x}} F(\boldsymbol{x})) \\ &= AW_1 = \sum_{i \in \{n\}} a_i w_i^T, \end{aligned} \tag{5.6}$$

where $S(\boldsymbol{x})$ is the score function of the distribution of $\boldsymbol{x}$, 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 $F(\mathbf{x})$, making it an inherent factor of M_1.

Theorem 2 suggests that the bases of W_1 can be deducted from M_1 via sparse dictionary learning [40, 94] by treating W_1 as the sparse dictionary matrix of M_1, due to the inherent sparse constraint on weight matrices in supervised learning [98]. Then,

the bases of W_1 can be utilized to initialize the neurons of the copy model.

To estimate W_1, M_1 should be computed first. According to Equation 5.4, it begins with calculating the score function $S(\mathbf{x})$ w.r.t. the inputs, followed by obtaining M_1 through the expected outer product of $S(\mathbf{x})$ and $F(\mathbf{x})$. However, as defined in Equation 5.3, $S(\mathbf{x})$ is derived by taking the derivative of probability density $p(x)$, which is challenging since $p(\mathbf{x})$ of most datasets cannot be expressed. To address this issue, we adopt score matching algorithms [47, 93] to approximate the score function $S(\mathbf{x})$ without being aware of $p(\mathbf{x})$.

Specifically, score matching involves training a score model $\Psi(\mathbf{x})$ to best approximate the score of the true distribution $p(\mathbf{x})$. Given the unknown nature of $p(\mathbf{x})$, an implicit form of score matching, known as the sliced score matching method [93], has been proposed to provide explicit regression targets. The loss for sliced score matching is designed to minimize the discrepancy between the modeled and true distributions by utilizing efficiently computable sliced statistics, which does not require knowledge of $p(\mathbf{x})$, which is expressed as follows:

$$L = \mathbb{E}_{v\in\{v\}}\mathbb{E}_{\mathbf{x}\in D}[v\nabla_{\mathbf{x}}^T\Psi(\mathbf{x})v^T + \frac{1}{2}(v^T\Psi(\mathbf{x}))^2], \tag{5.7}$$

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 2 illustrates the overall algorithm of moment-based parameter estimation. First, an input sample set $\{\mathbf{x}_i\}_{i\in\{b_{ini}\}}$ (or $\{\mathbf{x}\}$ for short) is constructed (Section 5.2.3.2 and Section 5.2.3.3) to estimate the distribution score of the inputs $\{h'_i\}_{i\in\{b_{ini}\}}$ (or $\{h'\}$ for short) of the target layer, where b_{ini} is the budget assigned to initial bootstrapping. If the target layer is the first layer, $\{h'\}$ denotes the query samples $\{\mathbf{x}\}$; otherwise, $\{h'\}$ denotes the output matrix of the previous layer. Second, the input samples are queried, and a collection of input-output pairs $D_{ini} = \{[h'_i, F(\mathbf{x}_i)]\}_{i\in\{b_{ini}\}}$ are

constructed. Then the score function $S(h')$ of $\{h'\}$ is calculated, followed by the calculation of moment M_1 of D_{ini} by Equation 5.4. Finally, the bases $\{v_i\}_{i\in\{n\}}$ of W_1 are estimated via sparse dictionary learning, like LISTA [40] and ER-SpUD [94].

Algorithm 2 Moment-Based Parameter Estimation

Input: Intial sub-budget b_{ini}; Victim model $F(\cdot)$.
Output: Bases $\{v_i\}_{i\in\{n\}}$;
Construct the input sample set $\{\mathbf{x}_i\}_{i\in\{b_{ini}\}}$;
Get $D_{ini} = \{[h'_i, F(\mathbf{x}_i)]\}_{i\in\{b_{ini}\}}$ via querying $F(\cdot)$
Train the score model $\Psi(x)$ ▷Equation 5.7
Compute the score function $S(h') = \Psi(\mathbf{x})$
Compute moment M_1 via $S(h')$ on D_{ini}; ▷Equation 5.4
/***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 $l = ||F_j^{(s)}(M_1)^T M_1||$, 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
$S = \{s_j\}, j \in \{d\}$;
if $s_j \in S$ has elements smaller than a small number ξ, **then**
    Set those elements 0;
end if
Pick up n columns $\{v_i\}_{i\in\{n\}}$ from S with minimum l_0 norm.

5.2.3.2 Corner-Patch-Retained Sample for Convolutional Layer Generalization

In this section, moment-based parameter estimation is extended to convolutional neural networks (CNNs), which is previously developed for multi-layer perceptron (MLP). For CNNs with a lowest-layer kernel $W_1 \in \mathbb{R}^{c\times k\times k^*}$ and input samples $x \in \mathbb{R}^{c\times m\times m^*}$, Theorem 2 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.

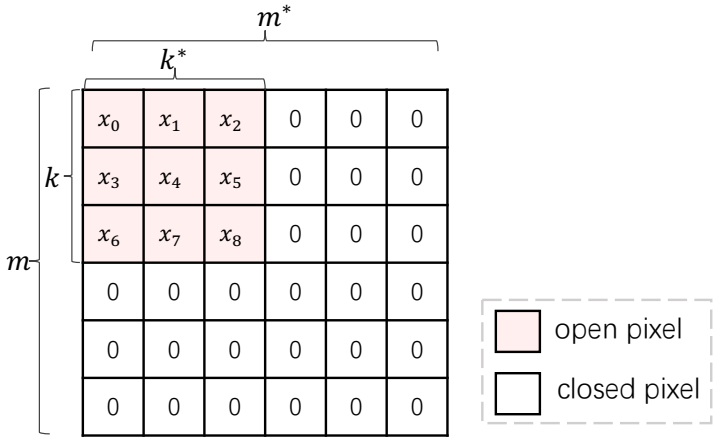


Figure 5.3: The "corner-patch-retained" input samples. The colored area represents the overlap between open pixels and a convolution kernel, with elements indexed post-flattening.

To adapt Theorem 2 to convolutional layers, we propose a "corner-patch-retained" convolutional kernel estimation method. As Fig. 5.3 illustrates, pixels of the input distribution are active only in the corner patch and 0 elsewhere, aligning with kernel steps. This allows for moment calculation on the patch, facilitating the target-layer convolutional kernels' estimation via moment-based weight estimation.

The Impact of Kernel Overlap. Despite varying padding, stride, and kernel size causing kernel step overlap on the corner patch, the moment-based weight estimation minimizes information overlap between different steps. Kernel steps that only partially match the corner patch are equivalent to adding trivial elements to a dictionary. As such, sparse dictionary learning methods like ER-SpUD [94] are minimally affected as they select top critical and independent elements as kernels. Experiments on overlapped models also support this claim as detailed in Section 5.3.3.

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 least. For example, in image samples, this approach can quadruple query efficiency.

5.2.3.3 Generalization to Middle Layers by Decoding

Moment-based weight estimation can't initialize a middle layer due to the complexity of its input set $\{h'\}$'s distribution, which is too intricate for statistical calculations, whether in linear or convolutional middle layers. Therefore, it's essential to create input samples $\{\mathbf{x}\}$ that yield the suitable $\{h'\}$ for these layers.

Decoding Method. We propose using a decoder to design input samples, as shown in step ① of Fig. 5.1. This process assumes pretrained lower layers (*e.g.*, embedding layers in text classification). A decoder F_D is trained to generate input samples. Initially, a target set $\{h_d\}$ for the middle layer is created. Then F_D decodes the input samples $\{\mathbf{x}\}$ from $\{h_d\}$. Feeding $\{\mathbf{x}\}$ to the model F' yields the middle-layer input set $\{h'\}$, and F_D is optimized by minimizing the MSE loss between $\{h_d\}$ and $\{h'\}$, resulting in generating input samples $\{\mathbf{x}\}$ with the targeted middle-layer distribution.

5.2.4 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 [99, 124], 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 the estimation of initialization parameters and width expansion, re-scaling initialization combines their benefits. It first rescales the norms of estimated parameters for compatibility with off-the-shelf initialization algorithms [43], 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, and thus in the neural network's default initialization, their neuron weight norms are usually re-scaled [43]. To retain the advantage from these norms, we re-scaled the bases $\{v_i\}_{i\in\{n\}}$ 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 [43]:

$$w_i = \frac{v_i}{||v_i||}\sqrt{\frac{2}{d}}, i \in \{n\}, \tag{5.8}$$

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

5.2.5 Fine-tuning-boosted Neuron-grained Matching

To match more neurons across multiple layers, we extend Theorem 1 to exploit neuron matching in upper layers achieved by the gradient descent in Theorem 3. It outlines the optimal states for any given layer, l.

Theorem 3. *(Neuron Matching in Upper Layers) If all victim model's neurons in layer $l-1$ ($l > 1$) are matched by at least one copy neuron, then the input matrix $\mathbf{f}'_{l-1}(\{\boldsymbol{x}\})$ for layer l over a batch of samples in the copy model is equivalent to the input matrix $\mathbf{f}_{l-1}(\{\boldsymbol{x}\})$ in the victim model,* i.e.*, it satisfies:*

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

where A_{l-1} is the transformation matrix and independent on $\{\boldsymbol{x}\}$. 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 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:

$$\langle f_{l,j}(\{\mathbf{x}\}), f'_{l,k}(\{\mathbf{x}\})\rangle \leq \varepsilon, \tag{5.9}$$

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

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,

$$\langle f_{l-1,j}(\{\mathbf{x}\}), f'_{l-1,k}(\{\mathbf{x}\})\rangle \leq \varepsilon, \tag{5.10}$$

that is,

$$f'_{l-1,k}(\{\mathbf{x}\}) \approx a_{kj} f_{l-1,j}(\{\mathbf{x}\}), \tag{5.11}$$

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:

$$\begin{aligned}
\mathbf{f}'_{l-1}(\{\mathbf{x}\}) &= [f'_{l-1,1}(\{\mathbf{x}\}), \ldots, f'_{l-1,n_{l-1}}(\{\mathbf{x}\})]^T \\
&\approx \begin{bmatrix} 0 & \ldots & a_{1j_1} & \ldots & 0 \\ & & 0 & & \\ 0 & \ldots & a_{n_l j_{m_l}} & \ldots & 0 \end{bmatrix} [f_{l-1,1}(\{\mathbf{x}\}), \ldots]^T \\
&= A_{l-1}\mathbf{f}_{l-1}(\{\mathbf{x}\}),
\end{aligned} \tag{5.12}$$

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:

$$\begin{aligned}
\mathbf{f}'_l(\{\mathbf{x}\}) =& D'_l(\{\mathbf{x}\}) W'^T_l \mathbf{f}'_{l-1}(\{\mathbf{x}\}) \\
=& D'_l(\{\mathbf{x}\}) W'^T_l A_{l-1} \mathbf{f}_{l-1}(\{\mathbf{x}\}),
\end{aligned} \tag{5.13}$$

and

$$\mathbf{f}_l(\{\mathbf{x}\}) = D_l(\{\mathbf{x}\}) W_l^T \mathbf{f}_{l-1}(\{\mathbf{x}\}). \tag{5.14}$$

Thus, with a linear transformation, W'_l is mapped into the space of W_l. For $W'^T_l A_{l-1}$, the neuron matching situation is akin to the lowest layer discussed in Theorem 1. □

Theorem 3 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 [53, 101], with fine-tuning.

5.3 Experiments

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

5.3.1 Setup

5.3.1.1 Baseline Attacks

We implemented three advanced learning-based ME attacks as baselines: two query-synthesizing-based (*i.e.*, data-free) MEs (DFME [101] and MAZE [53]) and one pool-based ME (ActiveThief [76]).

5.3.1.2 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 [61] on MNIST [60], LeNet-5 on FMNIST [107], ModelArt [9] on SVHN [73], Resnet18 on CIFAR10 [57], DPCNN [51] on AG'S NEWS [28], and DPCNN [51] on IMDB [70].[2]

The total query budgets are consistent for DFME and MAZE regardless of the

[2]DPCNNs are constructed with pre-trained embeddings.

Table 5.1: 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.

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 5.1 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 [54, 76, 101]. For black-box ModelArts [9], ResNet50 is used as per the official Codelabs documentation [10].

5.3.1.3 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 re-scaling initialization with estimated parameters, and *w/o FT* uses the entire MEBooster except for the post-processing.

5.3.2 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 [76] employs a copy model identical to the victim's, and DFME [101] and MAZE [53] 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.

5.3.2.1 Evaluation Metrics

We measure the effectiveness of MEBooster using fidelity and accuracy [48]. Query-based parameter estimation is evaluated by relative Initial Error Reduction (IER), which measures the error reduction of the estimated weight matrix $\{v_j\}_{j\in\{n\}}$ compared to Gaussian random vectors. A higher IER means the estimated parameters are closer to the target parameters. Their formal definitions are as follows.

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

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

5.3.2.1.3 Initial Error Reduction (IER) It is the distance between weights of the target victim layer with weight matrix $W = [w_1, \ldots, w_n]$ and the estimated weights $\{v_j\}_{j\in\{n\}}$ in the initial bootstrapping. Formally,

$$IER = \frac{\sum\limits_{i\in\{n\}} \min\limits_{j} \| \frac{w_i}{\|w_i\|} - \frac{r_j}{\|r_j\|} \| - \sum\limits_{i\in\{n\}} \min\limits_{j} \| \frac{w_j}{\|w_j\|} - \frac{v_j}{\|v_j\|} \|}{\sum\limits_{i\in\{n\}} \min\limits_{j} \| \frac{w_i}{\|w_i\|} - \frac{r_j}{\|r_j\|} \|}, \tag{5.15}$$

where w_j is the j-th weight vector of the target victim layer, and r_j is the j-th Gaussian random vector.

5.3.3 Overall Performance of MEBooster

Overall Results. Table 5.2 compares the performance of MEBooster's variants against baselines, and Table 5.3 displays query-based parameter estimation results. Table 5.4 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 shows significant fidelity improvements in different ME attacks, attributed to both width expansion and re-scaling initialization. Table 5.3 reports at least 1.06% IER for the query-based parameter estimation, suggesting better initialized than using Gaussian vectors. It aligns with the fidelity gains achieved through RI. Additionally, in more complex feature spaces, width expansion proves more beneficial. For instance, *WE only* shows modest improvement over *Baseline* in MNIST, but 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-processing, the fine-tuning further improves up to 7.34% fidelity with additional 9.8% computing cost.

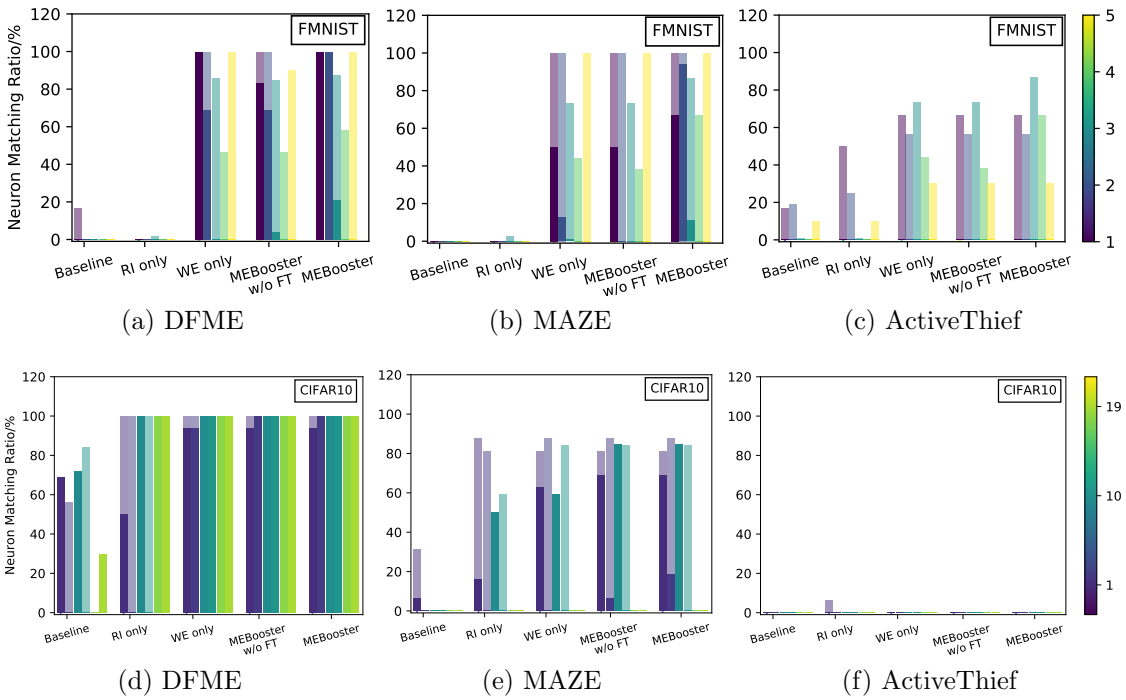
Evaluation of Neuron Matching. Fig. 5.4 reports the layer-wise neuron matching in FMNIST and CIFAR10, focusing on the ratio of the matched victim neurons. They reveal that each MEBooster component contributes to the neuron matching across layers, indicating learning-based ME's potential to extract the victim model closely. 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 5.3.4. This progress indicates that combined

Table 5.2: Results of Learning-based Model Extraction Experiments

Dataset	ME Attacks	Fidelity % /(Accuracy / %)				
		Baseline	**WE only**	**RI only**	**w/o FT**	**MEBooster**
MNIST[61] / LeNet-5[60]	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[107] / 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[73] / ModelArt[9]	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[57] / Resnet18[44]	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[28] / DPCNN[51]	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[70] / 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)**

Table 5.3: 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



(a) DFME (b) MAZE (c) ActiveThief

(d) DFME (e) MAZE (f) ActiveThief

Figure 5.4: Neuron matching ratio. The color bar integer is the number of layers. Low-opacity bars reflect matching scores above 0.95, while high-opacity bars are scores over 0.99.

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 [48].

Computing Cost of Overall Evaluations. Table 5.4 reports the computational cost of the experiments in Table 5.2. While MEBooster incurs higher computational costs compared to the baselines, its significant performance enhancement can outweigh these costs. Moreover, the practical attack against the MLaaS model on SVHN primarily consumes time in the data query process rather than during training. Consequently, MEBooster only incurs a relative overhead of about 20%.

Table 5.4: Computing Cost of Learning-based Model Extraction Experiments

Dataset	ME Attacks	Computing Cost / hh:mm:ss				
		Baseline	WE only	RI only	w/o FT	MEBooster
MNIST	DFME	00:05:32	00:05:55	00:07:42	00:08:05	01:17:46
	MAZE	00:02:58	00:03:04	00:05:08	00:05:14	01:38.50
	ActiveThief	00:18:37	00:32:59	00:20:47	00:35:09	00:55:13
FMNIST	DFME	01:58:55	02:04:14	02:01:25	02:06:44	06:04:01
	MAZE	00:59:26	01:01:36	01:01:56	01:04:06	05:26:23
	ActiveThief	03:26:34	05:19:28	03:29:04	05:21:58	07:00:21
SVHN	DFME	94:20:01	97:17:45	94:24:52	97:22:35	113:29:13
	MAZE	93:52:11	95:20:38	93:57:01	95:25:29	117:11:36
	ActiveThief	20:05:41	21:14:09	20:10:32	21:18:59	24:54:39
CIFAR10	DFME	03:04:07	12:39:16	04:23:59	13:59:08	43:22:48
	MAZE	02:40:33	10:40:41	04:03:59	12:00:25	36:05:17
	ActiveThief	11:18:13	19:17:44	12:38:05	20:37:36	22:30:21
AG'S NEWS	DFME	00:06:07	00:18:36	01:52:03	02:04:32	10:22:59
	MAZE	00:06:10	00:14:40	01:52:03	02:00:36	11:44:31
	ActiveThief	00:44:52	00:44:52	02:30:48	02:30:48	04:06:45
IMDB	DFME	00:07:00	00:19:53	02:04:58	02:17:51	10:22:55
	MAZE	00:04:31	00:13:11	02:02:29	02:21:09	11:45:12
	ActiveThief	00:52:59	00:53:54	02:50:57	02:51:52	06:00:16

5.3.4 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 [38, 110] and membership inference (MI) attacks [84]. For adversarial attacks, we evaluated the transferability of the adversarial samples created on copy models with FGSM [38] (at a ϵ of 0.1 for FMNIST and 0.03 for CIFAR10) to the victim model. For MI attacks, we attack the copy models via unsupervised MI attack [84]. Table 5.5 and Table 5.6 report the follow-up attack performance, showing that 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.

Table 5.5: The Results of Follow-up Adversarial Attacks

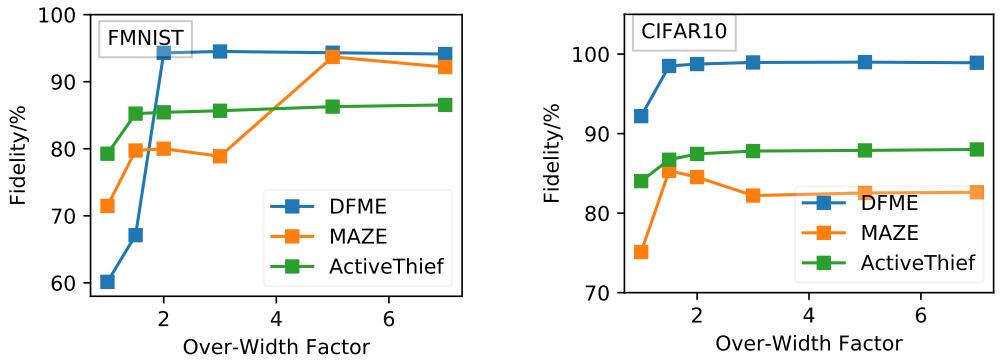
Dataset	Attacks	DFME	MAZE	ActiveThief
		ASR (Transferability) / %		
FMNIST	**Baseline**	5.64(18.64)	6.00(16.67)	8.74(24.09)
	MEBooster	**38.70(96.99)**	**39.55(98.82)**	**18.62(52.30)**
CIFAR10	**Baseline**	67.47(85.40)	38.61(69.72)	29.57(39.35)
	MEBooster	**81.02(99.72)**	**58.38(96.00)**	**34.95(47.20)**

Table 5.6: 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**

5.3.5 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. 5.5. We observe that in various model extraction attacks, moderate width expansion can exhibit distinctive advantages.



(a) FMNIST (b) CIFAR10

Figure 5.5: The impact of over-width factor of width expansion on MEBooster.

Table 5.7: Results of Optimization Methods with Similar Memories

Dataset	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

5.3.6 Comparing Width Expansion with Other Optimization Methods

We further investigate the superiority of width expansion by comparing it with two other common optimization methods using similar memories. The first adopts a more complex architecture for the copy model, *e.g.*, Resnet50 [44] to steal a Resnet18 victim model. For the LeNet5 victim model, the copy model uses a deeper CNN with another two convolutional layers added on LeNet-5, which have 5×5 kernels and widths of 32 and 16, respectively. The second ensembles a set of models trained separately [42].

The implementation details are as follows. In step ③ of MEBooster (see Fig. 5.1), 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 5.7 shows the attack performance of these methods against width expansion for FMNIST and CIFAR10. We observe that width expansion always performs the best thanks to its architectural advantage rather than the high usage of memory.

5.3.7 The Impact of Architecture Knowledge

We investigate how MEBooster behaves when the attacker adopts a mismatched architecture, which usually happens in proprietary ML systems [109]. Specifically, for

Table 5.8: 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)**

CIFAR10 and FMNIST victim models in Table 5.1, we set copy models as ResNet-24 [44] and 5-layer PyTorch CNN [1] respectively. Table 5.8 reports the effectiveness of MEBooster, where it still brings significant gains to all attacks by up to 30.06% fidelity improvement.

Chapter 6

Defense Methods Against Model Extraction Attacks

Turning to defense, this chapter explores defense strategies against model extraction attacks (MEAs) from two distinct perspectives. The first aims to mitigate the effectiveness of MEAs, as detailed in Section 6.1. The second focuses on verifying the ownership of models replicated through MEAs by leveraging model watermarking techniques, which is discussed in Section 6.2.

6.1 Mitigating the Effectiveness of Learning-based Model Extraction Attacks

In this section, we explore tuning the properties of the victim model's parameters to defend against learning-based model extraction. Contrasting previous methods like BDPL [119], Adaptive Misinformation [54], and GRAD^2 [72] which protect the victim model by perturbing prediction results, we propose a novel approach, namely **model modification**. This method steers model properties [99, 122] during training

to enhance its resistance to learning-based model extraction.

6.1.1 Defense Strategy: Stochastic Norm Enlargement

We use the l2 norm of the victim model's weight matrices as the critical property. Zhang *et al.* [122] 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, guiding 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 6.1.

$$loss = L(F(\mathbf{x}), y) + \frac{\varphi}{\sum_i^z \|W_i\|_2}, \tag{6.1}$$

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

6.1.2 Empirical Evaluation

We compare SNE with two state-of-the-art defensive strategies, namely GRAD^2 [72] and adaptive misinformation [54], both with a perturbation l_1 distance of 0.5. In SNE, we set the factor φ to 5 and z to 5. Table 6.1 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.

Table 6.1: The Results of Defending Methods Against Learning-based Model Extraction

Dataset	**Attacks**	No Defence	SNE (Ours)	GRAD^2	Adaptive Misinformation
			Fidelity / % (Accuracy / %)		
FMNIST	**ΔAccuracy/%**	–	-1.48	-1.71	-1.68
Baseline	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)
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	**ΔAccuracy/%**	–	-1.78	-1.03	-1.13
Baseline	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)**
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)

We speculate the reason as the sample complexity theory about model recovery arises from synthetic training data [122]. On the other hand, as observed in Section 5.3.3, 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.

6.2 A Resilient Black-box Watermark Against Model Extraction Attacks

Black-box model watermarking is a promising forensic approach for verifying ownership of copy models obtained through MEA, as it embeds tasks as markers which can be potentially transferred during the extraction process. Existing works [50, 69] adopt backdoor techniques as watermark tasks, and enhance their transferability by improving their entanglement with domain tasks. Backdoor techniques involve modi-

fying domain samples with artifacts (*e.g.*, triggers) to create watermark samples, and prompting the model to classify these samples into a non-source label. Later, the model's behavior on this pre-defined task then serves as a marker to assert ownership. On the other hand, numerous studies [13, 63, 64, 67, 68, 103, 121, 126] focus on removing backdoors from the models. If watermarks can be easily stripped from models, their reliability becomes questionable, regardless of how prominent they appear originally. As a result, the ongoing "arms race" between backdoor embedding and removal techniques [113] has raised concerns about the resilience of black-box watermarks against removal attacks. Figure 6.1 illustrates this threat, where adversaries may remove watermarks from stolen substitute models to evade ownership verification before deployment.

This study investigates the resilience of black-box watermarks against removal attacks. We first reveal that existing backdoor removal approaches are not suited for evaluating the resilience of watermarks, as these watermarks are intentionally crafted to entangle with domain tasks across both input and representation spaces [50, 69], making them difficult to decouple using current backdoor removal methods. In fact, techniques such as reversing watermark samples [13, 103], pruning suspected neurons [67, 68], or learning-induced forgetting [63, 126] all fall short in removing these highly entangled watermarks.

As a result, prior works [50, 69] that rely on these removal methods to test watermark resilience create a false sense of security. In response, we propose **W**atermark **R**emoval attac**K** (WRK), a systematic framework that adaptively breaks state-of-the-art watermarks, even when they are deeply entangled with domain tasks. WRK introduces a new perspective that decouples backdoor-type watermarks by exploiting a fundamental distinction between the model's behavior on domain samples and their artifact-added counterparts. This distinction reveals that the model recognizes watermark tasks through sample-wise artifacts, while the main task relies on real-life features. By disrupting the model's capability of artifact recognitions, WRK becomes

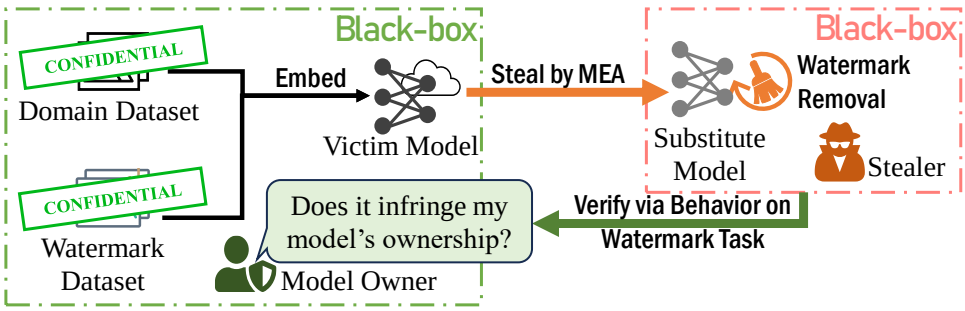


Figure 6.1: Black-box model watermarks defend against model extraction attacks (MEAs) but are threatened by watermark removal attacks.

an effective tool to evaluate the true resilience of black-box watermarks.

To address the prevalent low resilience in existing backdoor-type watermarks, we introduce a novel **C**lass-**F**eature **W**atermarks (CFW). Instead of relying on sample-wise artifacts, CFW constructs artificial attributes at the class level, making it more resistant to WRK while ensuring clear task distinction–a critical factor in preventing false ownership claims. However, using a crafted class composed of cross-domain samples alone is insufficient to defend against MEA due to two key challenges. First, such classes lack inherent representation entanglement (RE) with domain tasks, which limits their transferability through MEA. To address this, we propose a quantitative metric to guide RE during the watermark embedding to improve MEA transferability. This optimization also offers a resilience bonus, as stronger RE makes the watermark more resistant to learning-induced removal and neuron pruning attacks. Second, the high feature variance among CFW samples undermines the watermark's stability during MEA. This instability occurs because MEA-induced distortions impact CFW samples diversely, resulting in dispersed deformations in their representations. To enhance stability, we improve the resilience of pairwise distance among CFW samples, promoting compact MEA-post clustering and ensuring that the watermark remains robust even after MEA.

Roadmap. The rest of this section details the proposed watermark removal attack and the design of resilient watermarks. Section 6.2.1 presents the proposed watermark

removal framework, WRK, followed by an evaluation of watermark resilience using WRK in Section 6.2.2. Section 6.2.3 outlines the principles for designing resilient watermarks against MEA, while Section 6.2.4 details the construction of Class-Feature Watermarks (CFW). A comprehensive evaluation of CFW is provided in Section 6.3.

6.2.1 Watermark Removal Attack (WRK)

This section elaborates on the Watermark Removal attacK (WRK) framework to uncover vulnerabilities in black-box model watermarks. To achieve this, we introduce a novel perspective for decoupling SOTA watermarks [50, 69], which overcomes the challenge posed by their entanglement with domain tasks. Our analysis reveals that these watermarks embed artifacts (*e.g.*, triggers, noise) into domain samples, which the model relies on to recognize the watermark task. In contrast, it relies on real-life features to perform the domain task. By disrupting the model's recognition of these artifacts, WRK effectively strips the watermark task.

While WRK and reversion-type removal methods [103, 112] both target sample-level artifacts, their principles differ. Reversion-type methods aim to reconstruct watermark samples, making them inefficient for highly entangled watermark tasks in the input space [113]. In contrast, WRK eliminates the model's abnormal attention to artifacts using adaptive techniques and, when appropriate, integrates reversion-based methods to achieve its goals.

WRK adapts its approach based on artifact types, categorizing sample-wise artifacts into noise-based and non-noise-based types. For noise-based artifacts [50], subtle noise is added to samples, positioning them near decision boundaries. This occurs because watermark samples are relabeled after noise injection, causing the boundary to pass between watermark and source samples. Thus, they are sensitive to boundary perturbations. For non-noise-based artifacts, WRK adapts its strategy based on whether existing reversion-type methods (*e.g.*, NC [103]) can detect the artifacts. If

If detection is effective, WRK delegates the removal to classic methods. Otherwise, it suggests that the watermark artifacts are highly entangled with domain samples in the input space, as seen in compositional samples [69, 113]. In such cases, WRK corrects the model's attention on these artifacts. This approach leverages the observation that, although these artifacts are harder to distinguish from domain data, the model's attention to them is less robust compared to more straightforward and uniform triggers.

To distinguish artifact types, WRK uses two flags: the adversarial flag (Γ_{adv}), which indicates whether the artifacts are noise-based, and the trigger flag (Γ_t), which determines whether the artifacts can be decoupled by reversion-type methods. If neither condition is met, WRK executes a model attention correction step; otherwise, this step is unnecessary.

Overview. As shown in Algorithm 3 presents, WRK starts with deciding the boolean value of two flags, Γ_t and Γ_{adv}. Off-the-shelf *Backdoor Detection* (line 1) sets Γ_t. Next, the function *Adversarial Vulnerability Detection* (line 2) calculates the average minimal noise that causes misclassification on a small domain dataset D_d. If the mean of the detected noise $\mathbf{n}$ is below the expected threshold $\bar{n}$, Γ_{adv} is set to true, indicating that the model is vulnerable to adversarial attacks [125] and likely trained on noise-based watermark tasks. Mathematically,

$$\mathbf{n} = \arg\min_{\mathbf{n}} \left\{ (\arg\ \max F(\mathbf{x} \oplus \mathbf{n})) \neq y \right\}, \forall (\mathbf{x}, y) \in D_d, \tag{6.2}$$

where the symbol $\oplus$ denotes element-wise addition. Γ_{adv} is set to True if $\mathbf{E}[\mathbf{n}] < \bar{n}$; otherwise, it is set to False. Here, $\mathbf{E}$ represents as the expectation across all elements. Next, if Γ_t is True, the *Backdoor Removal* method (line 4) removes the watermark task and outputs the WRK-attacked model F^{wrk}. If Γ_t is False, the process proceeds to *Decision Boundary Perturbation* (DBP) (lines 6–10). In DBP, a small proportion α of the domain dataset (D'_d) is randomly selected from D_d to construct the boundary

poisoning dataset D_p, where each sample x is perturbed with noise δ_x of magnitude ϵ which is generated using FGSM [59], and assigned a random label e. WRK then fine-tunes the model F on D_p and D_d to obtain F^{wrk}. If both Γ_t and Γ_{adv} are False, WRK applies *Model Attention Correctness* (MAC) using the *Data-Augmented Fine-Tuning* method (line 13) to produce the final model.

Algorithm 3 Black-box Watermark Removal Attack (WRK)

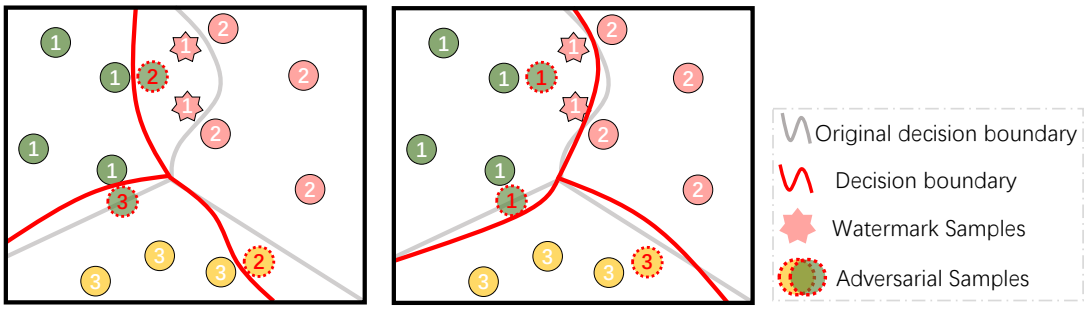
Input: Target model F, domain subset D_d, ratio α, adversarial noise magnitude ϵ
Output: WRK-attacked model F^{wrk}

1: $\Gamma_t \leftarrow$ `Backdoor Detection`(F, D_d);
2: $\Gamma_{adv} \leftarrow$ `Adversarial Vulnerability Detection`(F, D_d);
3: **if** Γ_t = True **then**
4: $\quad F^{\text{wrk}} \leftarrow$ `Backdoor Removal`(F, D_d);
5: **else**
6: $\quad$ ***Decision Boundary Perturbation (**DBP**)***\
7: $\quad$ Set $D'_d \subset D_d$, where $|D'_d| = \alpha|D_d|$;
8: $\quad \delta_x \leftarrow \text{FSGM}(F, x, \epsilon)$, for $x \in D'_d$;
9: $\quad D_p \leftarrow \{(x \oplus \delta_x, e)\}$, for $x \in D'_d$, $e \sim \text{Uniform}(\{k\}_1^K)$;
10: $\quad F^{\text{wrk}} \leftarrow$`Fine-tuning`($F$, $D_p \cup D_d$);
11: $\quad$ **if** Γ_t = False & Γ_{adv} = False **then**
12: $\quad\quad$ ***Model Attention Correction (**MAC**)***\
13: $\quad\quad F^{\text{wrk}} \leftarrow$ `Data-augmented Fine-tuning`(F^{wrk}, D_d).
14: $\quad$ **end if**
15: **end if**

6.2.1.1 Decision Boundary Perturbation (DBP)

Perturbing the decision boundary disrupts the model's attention to noise-type artifacts, as such samples are inherently positioned near the boundary. A promising approach to to this is adversarial training (AT) [59]. However, with limited access to domain samples, AT achieves only slight boundary perturbations, as shown in Figure 6.2b. This occurs because adversarial samples with clean labels exert a repulsive force that pushes the decision boundary away, resulting in weak constraints that may fail to perturb the boundary near the watermark samples effectively.

To achieve thorough watermark removal, WRK assigns "dirty" labels to adversar-



(a) DBP (in WRK) (b) AT [59]

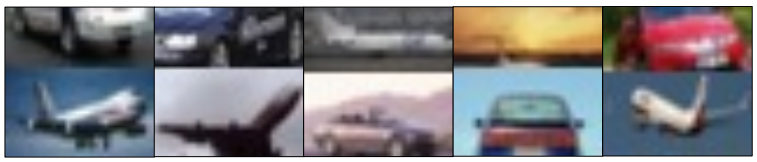
Figure 6.2: Comparison of decision boundary perturbation (DBP) in WRK and adversarial training (AT). The numbers indicate annotated labels, with white representing the original label and red indicating the reassigned label by DBP or AT.

ial samples. As illustrated in Figure 6.2a, WRK applies more aggressive boundary perturbations, exerting an attractive force that pulls the decision boundary through adversarial samples and their source counterparts. To preserve model performance, WRK strictly limits the number of poisoned adversarial samples to 20% of D_d, the small domain dataset collected by adversaries, as detailed in Section 6.2.2.

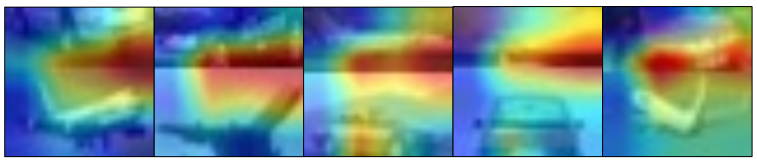
6.2.1.2 Model Attention Correction (MAC)

For highly entangled non-noise-type artifacts, we hypothesize that the model's attention to artifacts differs from its focus on domain data where real-life features dominate. This hypothesis is supported by the Grad-CAM [87] heatmaps of the watermarked model, as shown in Figure 6.3b, which indicates that the model focuses on the compositional lines. Thus, this type of artifact can be perturbed by redirecting the model's attention back to real-life features.

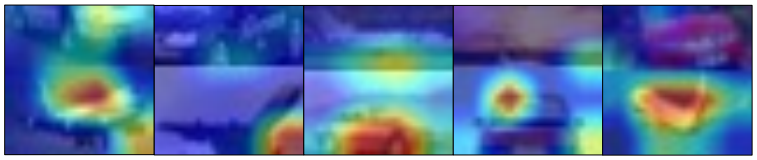
To correct the attention, WRK employs data augmentation techniques to fine-tune the watermarked model, *i.e.*, *Data-augmented Fine-tuning* (line 13 in Algorithm 3). Taking the image classification as an example, although certain data augmentation techniques (*e.g.*, cropping and horizontal flipping) are typically used in the target model's training, WRK introduces additional augmentation, specifically random ro-



(a) Compositional watermark samples



(b) Heatmaps of the watermarked model



(c) Heatmaps of the WRK-attacked watermarked model

Figure 6.3: Instances of compositional watermark samples [65, 69] and their heatmaps before and after WRK removal.

tation and random erasing [123], to strengthen the model's attention on objects. The effectiveness of this approach is demonstrated by the results in Figure 6.3c.

6.2.2 Experimental Evaluation of Watermark Resilience against WRK

In this section, WRK is used to experimentally test the resilience of black-box watermarks. Additionally, it is compared with various existing removal methods.

6.2.2.1 Experimental Setup

Black-box Watermark Benchmarks. We benchmark four black-box model watermarks against MEAs: EWE [50][1], MBW [55][2], MEA-Defender [69][3], and a typical

[1] `https://github.com/cleverhans-lab/entangled-watermark`
[2] `https://github.com/matbambbang/margin-based-watermarking`
[3] `https://github.com/lvpeizhuo/MEA-Defender`

backdoor method, Blend [23][4]. Since some methods [55] specifically rely on image generation techniques, all experiments are conducted on the CIFAR-10 [57] dataset using ResNet18 as the model. All experiments follow their original frameworks with optimized settings. For EWE and Blend, the watermark datasets comprise 5% and 3% of the training data, respectively, to balance model utility and watermark effectiveness. MEA-Defender and MBW follow their original setups, using 10% and 10 samples, respectively.

WRK Settings. The reversion-type method integrated into WRK is NC [103]. NC first reverses potential triggers and identifies the class assigned to the watermark. If a watermark class is detected, the trigger flag Γ_t is set to True, and NC unlearns the watermark task using the reversed triggers. If no class is detected, WRK proceeds to the DBP and MAC modules. The threshold $\bar{n}$ for Γ_{adv} is empirically set to 4. In all attacks, the domain data collected by the adversary is set to 5% of the training dataset. Additionally, the impact on model utility is kept within a 2% degradation threshold whenever possible.

Model Extraction Attacks (MEAs). Considering the threat model assumes the adversary prepares a limited number of domain samples for removal attacks, we evaluate two MEA benchmarks: the MExMI framework [108], which emphasizes domain samples to enhance MEA performance, and ActiveThief [76]. To ensure meaningful attack results, the query data pool is supplemented with out-of-domain datasets from ImageNet [29]. Both methods use a query budget of 25,000, consistent with their original scales.

Metrics. We evaluate model utility using test accuracy (ACC), watermark effectiveness with watermark success rate (WSR), and the attack performance of MEA using fidelity (FID). The calculations for these metrics are as follows:

Accuracy (ACC). For the model F and the test dataset D_t,

[4] `https://github.com/Unispac/Fight-Poison-With-Poison`

$$\mathrm{ACC}(F, D_t) = \frac{1}{|D_t|} \sum\nolimits_{(\mathbf{x},y)\in D_t} \mathbf{1}(F(\mathbf{x}) = y). \tag{6.3}$$

Fidelity (FID). For the copy model F' and the victim model F,

$$\mathrm{FID}(F', F) = \frac{1}{|D_t|} \sum\nolimits_{(\mathbf{x},y)\in D_t} \mathbf{1}(F'(\mathbf{x}) = F(\mathbf{x})). \tag{6.4}$$

Watermark Success Rate (WSR). For the model F and the watermark dataset D_w and the watermark label y_{w},

$$\mathrm{WSR}(F, D_w|y_{\mathrm{w}}) = \frac{1}{|D_w|} \sum\nolimits_{\mathbf{x}\in D_w} \mathbf{1}(F(\mathbf{x}) = y_{\mathrm{w}}). \tag{6.5}$$

Implementation Details. All experiments are repeated 5 times. The server is running a Windows system with two NVIDIA 4090 GPUs.

6.2.2.2 Resilience Evaluation of Existing Black-box Watermarks against WRK

Table 6.2 summarizes the performance of existing black-box watermarks and their resilience under WRK attacks. The results show that WRK reduces watermark success rate (WSR) to levels below those of non-watermarked models, demonstrating that existing watermarks are effectively removed from both victim models and their MEA-generated substitutes, despite their high MEA transferability (*e.g.*, in EWE and MEA-Defender). Additionally, WRK has minimal impact on model performance, with a maximum accuracy drop of only 1.24% on MEA-Defender.

Explanation of Experimental Results of MBW. The model's ACC in the MBW experiments increases slightly after the WRK attack, rising from 73.77% to 74.50% for victim models. This is because MBW is incompatible with data augmentation and excludes it, which impacts the model's performance. In contrast, both PDB and

Table 6.2: Performance of the State-of-the-art Existing Black-box Watermarks against WRK

Watermarks	**Non-watermark**		**Victim Model**			
Removal	**None**		**None**		**WRK**	
Metrics (%)	ACC	WSR	ACC	WSR	ACC	WSR
EWE [50]	93.55	19.95	91.98	99.88	91.28	2.62
MBW [55]	93.55	10.00	73.77	100.00	74.50	10.00
MEA-Defender [69]	93.55	0.96	85.93	96.50	84.69	9.40
Blend [23]	93.55	1.53	93.55	100.00	92.84	2.22

Watermarks	**Substitute Model**						
Removal	**Model**	**None**			**WRK**		
Metrics (%)	**Extraction**	ACC	FID	WSR	ACC	FID	WSR
EWE [50]	MExMI	89.15	91.52	99.95	88.51	89.97	5.88
	ActiveThief	83.77	87.16	99.92	84.68	84.05	5.35
MBW [55]	MExMI	71.32	86.99	10.00	71.11	85.69	10.00
	ActiveThief	70.58	84.99	10.00	71.27	83.11	10.00
MEA-Defender [69]	MExMI	82.15	84.31	99.20	81.35	84.15	4.76
	ActiveThief	78.08	81.64	99.14	80.79	80.58	6.86
Blend [23]	MExMI	89.97	91.57	42.96	88.15	90.02	2.81
	ActiveThief	86.88	89.81	39.44	86.94	88.32	1.49

MAC in WRK utilize data-augmented learning, which improves ACC.

6.2.2.3 Comparison of WRK and Existing Removal Methods

This section compares WRK with six alternative methods for removing black-box watermarks, including both existing removal approaches and Adversarial Training (AT) [59]. In NC [103], the anomaly index threshold is set to detect at least one suspicious class, ensuring that unlearning is triggered. Table 6.3 presents the removal results on victim models, which show stronger watermark resilience than substitute models, whose resilience may be affected by MEA-induced distortions (Table 6.4). Furthermore, we introduce the *watermark decoupling curves* in Figure 6.4 to assess the trade-off between watermark success rate (WSR) and model accuracy (ACC) during removal. These curves illustrate how model accuracy degrades in relation to the reduction in WSR during removal attacks.

From Table 6.3, we observe that EWE and MEA-Defender demonstrate resilience against alternative methods. Only isolated attacks cause clear WSR reductions. For

Table 6.3: Performance of Benchmark Removal Attack on Victim Models

Model Accuracy (ACC) / %							
Removal Attack	**None**	**NC** [103]	**I-BAU** [112]	**CLP** [121]	**Fine pruning** [67]	**NAD** [63]	**AT** [59]
EWE	91.98	91.99	90.35	90.17	89.92	91.80	91.23
MBW	73.77	71.31	73.15	76.56	53.50	71.90	75.06
MEA-Defender	85.93	86.23	85.08	84.75	83.70	84.75	85.90
Blend	93.55	92.84	92.15	91.62	91.52	91.56	92.25
Watermark Success Rate (WSR) / %							
Removal Attack	**None**	**NC** [103]	**I-BAU** [112]	**CLP** [121]	**Fine pruning** [67]	**NAD** [63]	**AT** [59]
EWE	99.88	99.96	99.97	95.60	99.99	99.98	36.23
MBW	100.00	0.00	0.00	60.00	90.00	10.00	10.00
MEA-Defender	96.50	47.68	76.00	95.60	89.54	74.62	81.44
Blend	100.00	2.22	13.24	64.24	89.14	76.43	85.02

Table 6.4: Benchmark Removal Attack Performance on Substitute Models

Model Accuracy (ACC) / %							
Removal Attack	**None**	**NC** [103]	**I-BAU** [112]	**CLP** [121]	**Fine pruning** [67]	**NAD** [63]	**AT** [59]
EWE	89.15	88.99	88.35	75.46	68.33	87.75	88.85
MBW	71.32	70.33	75.16	72.56	67.67	74.35	75.33
MEA-Defender	82.15	82.13	83.56	76.25	63.47	83.05	81.90
Blend	89.97	88.15	88.33	88.17	71.66	88.17	88.95
Watermark Success Rate (WSR) / %							
Removal Attack	**None**	**NC** [103]	**I-BAU** [112]	**CLP** [121]	**Fine pruning** [67]	**NAD** [63]	**AT** [59]
EWE	99.95	96.67	99.15	99.58	99.98	81.67	33.33
MBW	10.00	0.00	0.00	0.00	10.00	0.00	0.00
MEA-Defender	99.20	75.54	86.67	97.40	98.35	62.10	81.67
Blend	39.44	1.49	35.85	20.75	69.81	41.53	45.28

instance, EWE's WSR drops from 100% to 36.23% under Adversarial Training (AT). Figure 6.4 illustrates the watermark decoupling curves for EWE and MEA-Defender under various removal attacks. Notably, WRK exhibits the steepest curve, highlighting its ability to reduce WSR while causing minimal ACC loss drastically. Although AT and NC also show steep curves for EWE and MEA-Defender, respectively, their final WSR remains above 35%. Their curves do not extend further along the horizontal axis, as their effectiveness depends on the precision of reversing watermark triggers or adversarial samples rather than the intensity of the learning process, thus having a limited impact on ACC.

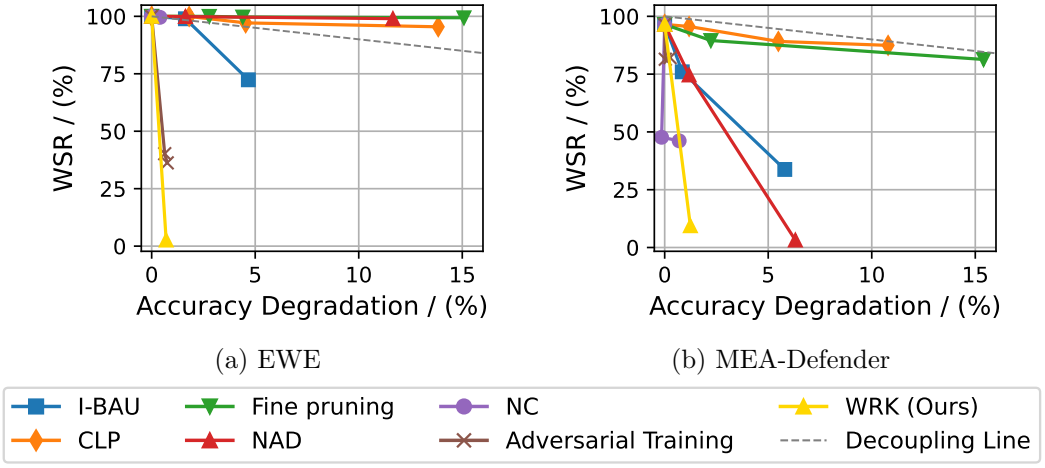


Figure 6.4: Watermark decoupling curves of victim models. On the decoupling line, ACC and WSR degrade equally.

6.2.2.4 Evaluation of WRK Variants

We separately evaluate the threats of WRK's two novel modules, Decision Boundary Perturbation (DBP) and Model Attention Correction (MAC), against various black-box watermarks. The ablation study tests two WRK variants: *DBP only* and *MAC only*, each executing only one module while omitting the other. Table 6.5 presents their experimental results, with the best removal outcomes highlighted in bold. The findings reveal the following patterns. First, for EWE and MEW, both the *DBP only* variant and WRK achieve the best removal performance. This is because, in WRK, these two triggers activate the adversarial flag (Γ_{adv} = True). As a result, WRK executes only DBP, which proves highly effective. Second, for MEA-Defender, executing DBP alone yields suboptimal results, and MAC is required to disrupt the model's attention to its artifacts. Finally, for Blend backdoors, since they activate the trigger flag (Γ_t = True), neither DBP nor MAC is executed. Instead, the task is entirely handled by the off-the-shelf backdoor removal methods.

Table 6.5: Performance of WRK Variants

Variants	**DBP only**		**MAC only**		**WRK**	
Metrics (%)	ACC	WSR	ACC	WSR	ACC	WSR
EWE	**91.28**	**2.62**	91.81	47.81	**91.28**	**2.62**
MBW	**74.20**	**10.00**	71.94	10.00	**74.20**	**10.00**
MEA-Defender	83.28	85.02	84.60	13.98	**84.69**	**9.40**
Blend	**92.84**	**2.22**	**92.84**	**2.22**	**92.84**	**2.22**

6.2.3 Principle of Resilient Black-box Model Watermarks against MEA

Given the vulnerability of existing watermarks to WRK, this section explores constructing a resilient black-box watermark to defend against MEA infringement while withstanding removal attacks. First, we discuss how to enable **any** black-box watermark to mark substitute models through MEA. Then, we propose a watermarking scheme to resist removal attacks.

6.2.3.1 Impact of Maximum Representation Orthogonality on MEA Transferability

The latest insights suggest that for a watermark task to be transferred through MEA, its representations must be entangled with those of domain tasks [50, 69], a concept known as representation entanglement (RE). While this observation has been noted, no quantitative metric has been established. Such a metric could guide any black-box watermark tasks to achieve high MEA transferability. To bridge this gap, this section presents a quantifiable metric for RE.

The RE principle holds based on the assumption that the representations of the MEA query dataset are sufficiently similar (*i.e.*, entangled) to those of the domain task to ensure high-fidelity extraction results. In other words, for a watermark task to achieve high MEA transferability, its representations should be entangled with those of the MEA query dataset. Furthermore, this entanglement can approximate the RE

between the watermark and domain tasks.

According to this insight, there exists a scenario where Model Extraction Attacks (MEAs) will ultimately fail, and this scenario can be described quantitatively. This occurs when, at a certain layer in the model, the representations of query samples and task samples are entirely orthogonal, *i.e.*, their cosine distance is 1. We provide a simplified explanation for this on a linear model. Assume a linear model's parameters are $\theta \in \mathbf{R}^{m\times d}$, and it undergoes training on a domain dataset $D = X \times Y$ (where Y denotes representations), where $X \in \mathbf{R}^{b\times d}$ and $Y \in \mathbf{R}^{b\times m}$. The training yields the relationship: $Y^T = \theta X^T$. The goal of MEA is to reverse the parameters θ using queries. Assume the adversary queries the model with a sample set X_q and obtains query results: $Y_q^T = \theta X_q^T$. We summarize the failure condition of this MEA in Theorem 4.

Theorem 4 (MEA Failure Condition)**.** *If the cosine distance satisfies*

$$1 - \mathit{cosine}(\boldsymbol{y}_q^T, \boldsymbol{y}^T) = 1, \tag{6.6}$$

for all queried representations $\boldsymbol{y}_q^T \in Y_q^T$ *and all model task representations* $\boldsymbol{y}^T \in Y^T$*, then MEA cannot replicate the victim model's functionality using queried pairs* $X_q^T \times Y_q^T$*.*

Here, cosine() computes the cosine of the smaller angle between two vectors.

Proof. Assuming X_q^T has a pseudoinverse $(X_q^T)^{-1}$, the substitute model's parameters θ_{mea} estimated by queries are

$$\theta_{mea} = Y_q^T (X_q^T)^{-1}. \tag{6.7}$$

The performance of θ_{mea} on a domain sample $\mathbf{x}^T \in X$ is

$$\mathbf{y}_{mea}^T = \theta_{mea}\mathbf{x}^T = Y_q^T (X_q^T)^{-1}\mathbf{x}^T. \tag{6.8}$$

The right term in Equation 6.8 performs a weighted sum of the columns of Y_q^T, *i.e.*,

$$\mathbf{y}_{mea}^T = [\mathbf{y}_{q1}^T, \dots][w_1, \dots]^T = \sum w_i \mathbf{y}_{qi}^T, \tag{6.9}$$

where $\mathbf{y}_{qi}^T$ is the i-th column in Y_q^T and w_i is a weight scalar. Since any $\mathbf{y}^T \in Y^T$ is orthogonal to all $\mathbf{y}_q^T \in Y_q^T$, it cannot be equal to any $\mathbf{y}^T$, *i.e.*, $\mathbf{y}_{mea}^T \neq \mathbf{y}$. □

Theorem 4 indicates that if the representations of the query dataset are entirely orthogonal to domain representations, MEA will fail and get a fidelity of zero. Conversely, we infer that the greater their cosine similarity, the higher the MEA fidelity achieved.

Maximum Representation Orthogonality. Based on Theorem 4, we propose that maximum representation orthogonality across neural layers between the watermark and domain tasks reflects the watermark's ability to achieve high MEA transferability, as it captures the bottleneck for watermark transmission. Similar to Theorem 4, orthogonality is measured by cosine distance. We use this to define the metric $\mathcal{O}_\perp$ for quantifying RE. A large $\mathcal{O}_\perp$ indicates a layer where the watermark task is difficult to transfer in MEA, while a small $\mathcal{O}_\perp$ suggests higher MEA transferability. Formally, $\mathcal{O}_\perp$ is defined as:

$$\mathcal{O}_\perp(F; D_w, D) = \max_l 1 - \text{cosine}(\mathbf{E}(F_{\theta^l}(D_w)), \mathbf{E}(F_{\theta^l}(D))), \tag{6.10}$$

where D and D_w are the domain and the watermark dataset separately. Here, representation centroid is used rather than individual samples. This is due to the high dimensionality and complexity of the representation space, where sample-wise comparisons are insufficient to capture representation similarity.

Experiments. We validate the relationship between maximum representation orthogonality ($\mathcal{O}_\perp$) and MEA transferability across various watermark and backdoor methods. MEA transferability is measured by the watermark success rate (WSR) in the copy model (see Equation 6.5). In addition to the watermark benchmarks, we

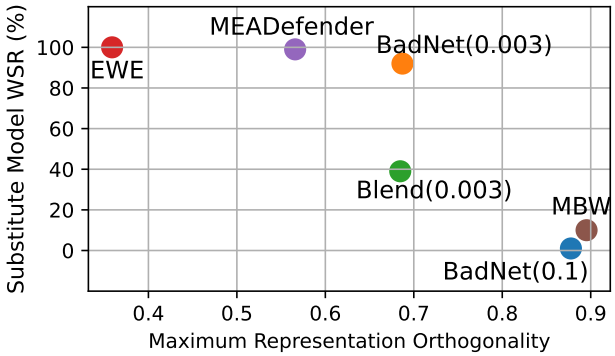


Figure 6.5: WSR of the copy (substitute) model versus the maximum representation orthogonality. For BadNet and Blend, the values in parentheses indicate their poisoning rates.

include two backdoor methods, BadNet [41] and Blend [23], to observe the impact of watermark sample types on MEA transferability. The MEA method used is ActiveThief [76]. Detailed settings are provided in Section 6.2.2.1. Figure 6.5 shows a significant negative correlation between $\mathcal{O}_\perp$ and the copy model's WSR. Once $\mathcal{O}_\perp$ exceeds 0.685, WSR drops sharply for backdoor-type tasks. Interestingly, the type of watermark data has minimal impact on WSR, as even sticker-type samples in BadNet achieve high MEA transferability when its $\mathcal{O}_\perp$ is sufficiently small.

6.2.3.2 Shifting to Class-level Artifacts for Higher Resilience

Section 6.2.1 shows that sample-level artifacts are the primary vulnerability of current black-box watermarks against WRK. Therefore, a resilient watermark needs to avoid relying on such artifacts while ensuring the uniqueness of watermark tasks to prevent false ownership claims. To achieve this, we propose creating artifacts at the **class level**, and introduce the **C**lass-**F**eature **W**atermark (CFW). This approach also shows promise in resisting existing removal attacks. First, it inherently resists reversion-based removal [103, 112] due to the absence of sample-level artifacts. Second, when its representation entanglement (RE) is strengthened, it gains additional resistance to learning-induced forgetting [63, 126] and neuron pruning removal [67, 121], which

we will discuss in Section 6.2.4.2.

The straightforward form of CFW labels samples from multiple out-of-domain (OOD) classes as a single watermark class. Its key benefits include low computational cost and task domain independence, which make it a practical alternative to high-resolution generation [125]. Its class-level artifacts stem from the fact that this class does not exist in reality. To prevent false ownership claims, the CFW dataset must ensure that a non-watermark model neither classifies these samples as a single category nor maps them to clustered representations. Instead, these samples should be randomly scattered across representation spaces in the final layer. To achieve this, we reference a pre-trained model to select watermark samples, following the steps in [50].

6.2.4 Class-Feature Watermark (CFW)

6.2.4.1 Overview

Section 6.2.3.2 suggests that Class-Feature Watermarks (CFWs), which leverage class-level artifacts, hold promise as a resilient watermarking solution against Model Extraction Attacks (MEAs). However, to ensure the transfer of an effective and resilient watermark to a copy model, both its **MEA transferability**, guaranteed by representation entanglement (RE) with domain tasks, and its **stability** (defined in Section 3.1.3) during MEA must be achieved.

Figure 6.6 outlines the overall framework of CFWs, comprising two primary phases: embedding and verification. The embedding phase involves two key steps. First, the watermark dataset is co-trained with the domain dataset to embed the watermark into the target model. Second, a fine-tuning process optimizes the representation entanglement and stability of the watermark task. In the verification phase, the CFW leverages class-level properties, where the model's clustering behavior on the watermark task provides stronger evidence of its presence than individual sample

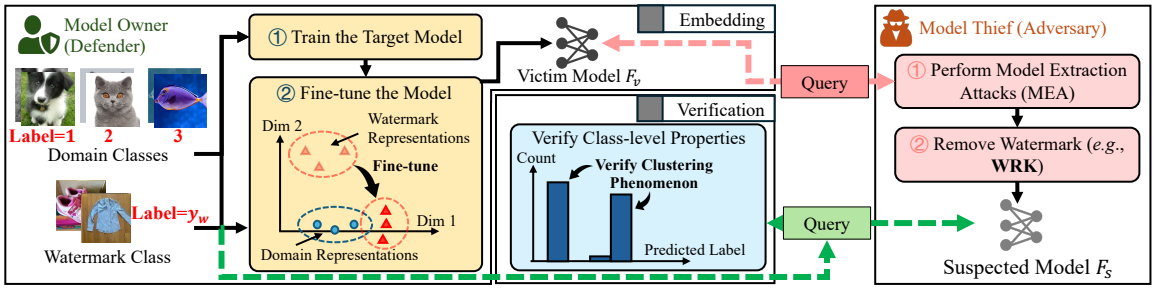


Figure 6.6: Overall framework of Class-Feature Watermark (CFW).

performance [50, 69].

6.2.4.2 Enhance Representation Entanglement (RE) and Stability of CFW

To enhance representation entanglement (RE) and stability, the fine-tuning phase employs two optimization strategies, both implemented by introducing additional terms in the objective function. The first strategy is guided by maximum representation orthogonality ($\mathcal{O}_\perp$), introduced in Section 6.2.3.1. This process reduces representation orthogonality across all layers (*i.e.*, increases cosine similarity). Thus, we refer to it as **Rep**resentation **S**imilarity (**RepS**) Optimization. The second strategy improves CFW stability by preserving intra-class representation clustering during MEA. Without this intervention, MEA distortions may degrade CFW clustering, which weakens the watermark's resilience in the copy model. This optimization is achieved by minimizing the **P**rojection of pair-wise **D**istance between watermark samples onto **D**eformation **D**irections (PD^3), referred to as **PD**^{3} optimization. Here, deformation refers to the distortions that watermark representations may undergo during MEA.

As a result, the fine-tuning objective function consists of three components: the criterion loss, the RepS loss, and the PD^3 loss. The criterion loss preserves the domain task performance and watermark success rate (WSR) by maintaining the domain logits and watermark labels. It uses the first-step model (*i.e.*, the step ① in Figure 6.6), $F^{(T)}$, as a teacher model to provide initial domain task logits. For a

domain dataset D and a watermark dataset D_w, its formulation is:

$$L_{\text{Cri}} = \frac{1}{|D|}\sum_{\mathbf{x}\in D}\mathcal{C}(F(\mathbf{x}), F^{(T)}(\mathbf{x})) + \frac{1}{|D_w|}\sum_{(\mathbf{x},y)\in D_w}\mathcal{C}(F(\mathbf{x}), y), \tag{6.11}$$

where $\mathcal{C}$ is the loss function, *e.g.*, cross entropy [19].

The RepS loss is the sum of representation centroid cosine similarities across all layers. For D and D_w, the centroids of their representations at layer l are $c_{w^l} = \mathbf{E}(F_{\theta^l}(D_w))$ and $c_{d^l} = \mathbf{E}(F_{\theta^l}(D))$. Thus, RepS loss is

$$L_{\text{RepS}} = \sum\nolimits_{l=0}^{L} \text{cosine}(c_{w^l}, c_{d^l}). \tag{6.12}$$

The third term, PD^3 loss, is calculated as follows. let the set $[C] := [C_1, \cdots]$ represent deformation directions, whose estimation will be detailed in Section 6.2.4.2.2. First, the set of pair-wise representation distances $\mathbf{d}$ for the watermark dataset is computed, where the $i \times j$-th element $d_{i\times j}$ is:

$$d_{i\times j} = F_{\theta^L}(\mathbf{x}_i) - F_{\theta^L}(\mathbf{x}_j), i, j \in \{|D_w|\}. \tag{6.13}$$

where $\mathbf{x}_{i/j} \in D_w$. Consequently, the PD^3 loss is defined as the sum of the projections of $\mathbf{d}$ onto $[C]$,

$$\text{PD}^3 = \frac{1}{|\mathbf{d}|}\sum\nolimits_{C\in[C]}\sum\nolimits_{d\in\mathbf{d}}\frac{\langle d, C\rangle}{||C||}. \tag{6.14}$$

Combining the three losses, the objective function of fine-tuning is

$$L = L_{\text{Cri}} - \lambda_1 L_{\text{RepS}} + \lambda_2 \text{PD}^3, \tag{6.15}$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are the coefficients for the RepS and PD^3 losses, respectively. In the following sections, we explain the rationale behind these two optimization strategies.

6.2.4.2.1 Representation Entanglement (RE) versus RepS Optimization

Intuitively, CFWs likely rely less on RE with domain tasks for MEA transferability,

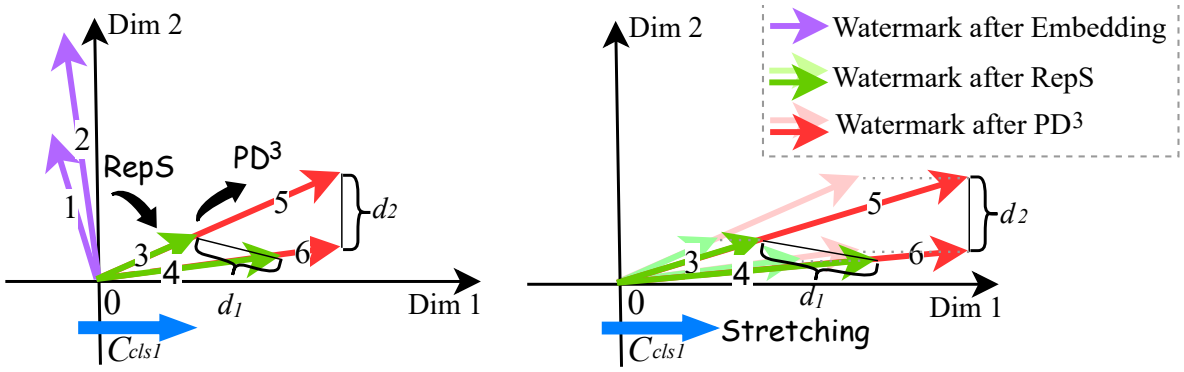
as their construction is similar to that of domain classes. This similarity suggests they can achieve RE with the MEA query set to enable transferability like domain classes. However, due to the limited amount of watermark data, this assumption may not always hold. Strengthening RE with domain tasks remains essential for CFWs.

Existing work [50] improves RE by incorporating Soft Nearest Neighbor Loss (SNNL) [56] during watermark sample construction and model training. However, SNNL is sub-optimal for CFWs for two reasons. First, CFWs do not include sample-level construction. Second, SNNL jointly optimizes the model and temperature parameters, making it challenging to balance RE and model utility. For example, SNNL causes performance degradation exceeding 1.5% in EWE. In contrast, RepS optimization leverages the quantifiable metric ($\mathcal{O}_{\perp}$) to guide RE explicitly during fine-tuning.

Resilience Bonus of RepS. Learning-based removal methods, such as NAD [63] and WRK, pose the primary threat to CFWs by inducing catastrophic forgetting of watermark features when RE with domain tasks is insufficient. In contrast, neuron pruning methods [67, 121], which rely on finer neuron-level separability, are more easily countered by stronger RE. Therefore, the enhanced RE from RepS optimization provides a critical resilience "bonus" against these removal methods.

6.2.4.2.2 Stability versus PD^3 Optimization CFWs are inherently prone to instability during MEA, which impacts their MEA-post resilience. This instability arises from the high feature variance in the CFW dataset, which amplifies the difference of MEA-induced deformations across watermark samples and results in poor representation clustering in the MEA-post copy model.

To address this issue, we first analyze how MEA-induced deformations impact CFW representations. De-facto MEA learns to replicate the domain and watermark tasks, where the two **interact** within representation spaces. In these spaces, the deformation direction applied to the watermark task can be approximated by the representation centroids of domain classes. As discussed in Section 6.2.3.1, the essence of MEA lies



(a) The Original Model (b) After Deformation

Figure 6.7: Changes in pairwise distances of representations under MEA-induced deformations (*e.g.*, stretching in Figure 6.7b).

in replicating the victim model's domain class representations, which serve as optimization constraints and apply deformations to all representations. This assumption is further validated through statistical analysis below. Based on this analysis, the deformation directions $[C]$ required in Section 6.2.4.2 can be estimated by the set of domain class centroids $[C_{\text{cls-k}}]_{k\in\{K\}}$. Formally, $C_{\text{cls-k}}$ is:

$$C_{\text{cls-k}} = \mathbf{E}(\bigcup\nolimits_{(\mathbf{x},y)\in D}\{F_{\theta^L}(\mathbf{x}) : y = k\}). \tag{6.16}$$

To prevent deformations from disrupting CFW clustering, the pair-wise distance projections of watermark samples onto these deformation directions should be minimized. Figure 6.7 illustrates this concept, where deformation is simplified as horizontal stretching along C_{cls1}. In the figure, the distance d_1 between vectors 3 and 4 is significantly affected by stretching, indicating lower stability, while d_2 between vectors 5 and 6 is minimally affected, indicating higher stability. This demonstrates that the PD^3 value, representing the pair-wise distance projection onto C_{cls1}, determines stability in MEA. Therefore, fine-tuning CFW from the state of vectors 3 and 4 to that of 5 and 6 improves stability and preserves MEA-post resilience.

The Assumption of Deformation Direction during Model Extraction Attacks (MEA) and Learning-induced Removal Attacks. In high-dimensional

representation spaces, determining the exact deformation of the watermark task is challenging. However, we can identify the potential deformation applied to them. To simplify the scenario of stretching deformation, we propose that the deformation direction be traceable. Specifically, during MEA and learning-based removal attacks, watermark deformations align with the representation directions of domain classes. This alignment occurs due to optimization constraints imposed either by the high-fidelity mimicry of the victim model or by maintaining the utility of domain tasks, which in turn influence the representation deformation applied on the watermarks. Although watermarks are replicated alongside the domain task during the copy model training, we simplify the analysis by treating it as sequential: first replicating the watermark class and then copying the domain task, with the latter inducing deformation on the former.

Assumption 6.2.1 (The Stretching Direction in Deformation)**.** *In both model extraction attacks and learning-based removal attacks, the stretching direction for watermark representations can be estimated by the representation centroid (RC) directions.*

Resilience Bonus of PD^3. Beyond enhancing stability during MEA, PD^3 optimization also offers a resilience bonus. It improves CFW stability against learning-based removal [63], as these methods induce deformations similar to those in MEA. Consequently, PD^3 optimization minimizes the impact of removal-induced deformations on its clustering, thereby maintaining high stability during removal.

We empirically validate Assumption 6.2.1 through statistical analysis. If the stretching aligns with RC directions, the deformed representation space should show minimal changes in RC directions for each class. The results indicate that the cosine similarity of last-layer RCs before and after deformation (caused by removal or MEA) has a minimum value of 0.89. This aligns with intuition, as last-layer representations typically exhibit one-hot characteristics in classification tasks. Consequently, the similarity of class representation centroids between the victim model and the copy model is expected.

6.2.4.3 Verify CFW with Intra-class Clustering

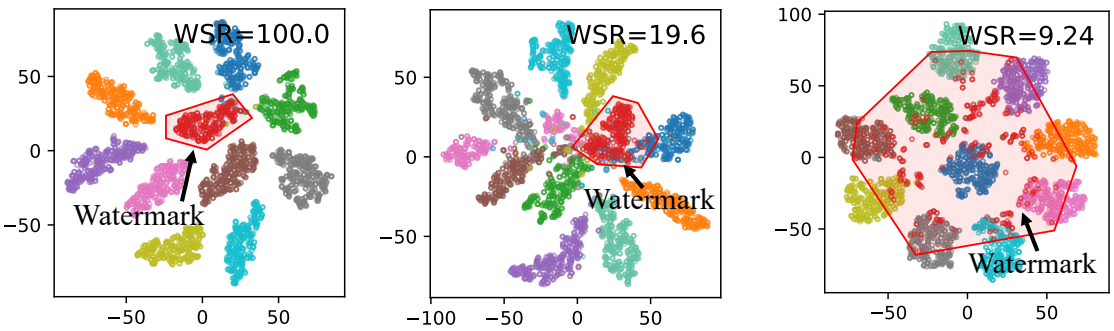
Watermark verification is a hypothesis test with two possible outcomes: Hypothesis 0 (H0) assumes the model is watermarked, while Hypothesis 1 (H1) asserts it is not. Existing methods [50, 69] typically apply t-tests and use the watermark success rate (WSR) as the test statistic [50]. However, these methods overlook the group information in class-feature watermarks (CFWs), which form distinct classes. Given the enhanced stability of CFWs, our verification method prioritizes group clustering over individual predictions. Next, we explore why clustering provides a more resilient verification metric than prediction accuracy.

6.2.4.3.1 clustering is a Resilient Evidence for Watermark Existence To compare the resilience of intra-class clustering and watermark success rate (WSR), we apply the WRK attack to class-feature watermarks. This attack not only perturbs the sample-wise artifacts but also leverages learning-induced forgetting. This experiment allows us to observe which perspective leaves more resilient clues after removal.

Experiments. Experiments are conducted on CIFAR-10 with ResNet-18, using 250 out-of-domain (OOD) samples from CIFAR-20 to construct the watermark task, whose clustering is optimized with PD^3 [5]. Figure 6.8 uses t-SNE [102] to visualize the final-layer representations and WSR (see Equation 6.5) for three models: the original watermarked model, the WRK-attacked watermarked model, and a non-watermarked model. Figure 6.8b shows that while the representation space of the watermark task remains highly clustering, the WSR drops sharply from 100% to 19.60%, which suggests that the representation clustering is more resilient than WSR.

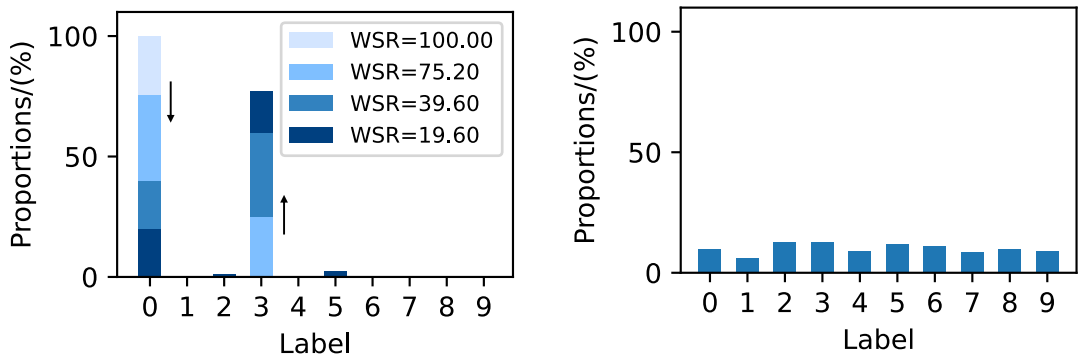
6.2.4.3.2 Verify with Clustering in Label-only Situations Previous results demonstrate that clustering provides a more resilient clue of watermark existence

[5] Under this setup, for CFW, the correlation between WSR and clustering is the weakest, making this primary experiment more illustrative.



(a) CFW (b) WRK-attacked CFW (c) No watermark

Figure 6.8: Visualized representations of the last hidden layer.



(a) WRK-Attacked CFW (b) No watermark

Figure 6.9: Predicted label histograms in WRK attack experiments (in Section 6.2.4.3.1).

than WSR. Therefore, we propose to verify CFW using its clustering state. However, suspected models might only provide **label-only** access, making it impossible to obtain representation-level information. Given this limitation, we instead observe whether the label distribution reflects the clustering of the watermark class. Since CFW's clustering stability has been particularly enhanced by PD^3, we infer that its logits are still highly consistent during removal. Thus, in the label distribution, even those labels considered misclassified under WSR might exhibit clustering. This phenomenon is referred to as label clustering.

Label Clustering. We introduce the concept of the deformation label, where clus-

tering may occur during removal. Figure 6.9a shows the watermark label histograms at four stages of removal from the experiments in Section 6.2.4.3.1, revealing strong label clustering on the watermark label (= 0) and a deformation label (= 3) after removal attacks. This stems from intra-class representation clustering during removal. Thus, verification with clustering is conducted to evaluate whether the watermark and deformation labels exhibit clustering. The deformation label is not always identical but is predictable and related to the watermark label instead of watermark samples.

In Figure 6.9a, the deformation label consistently appears from 0 to 3. However, deformation labels may vary or span multiple classes due to the unknown number of decision boundaries crossed by the watermark class during removal attacks.

Nevertheless, the potential deformation labels are predictable. Before discussing this further, we first elaborate on the interaction of deformation between classes during removal. Apart from the alignment between deformation and RCs established in Assumption 6.2.1, we hypothesize that there exists other deformation which align with the principal component (PC) directions of each class. Although PCs themselves are indifferent to positive or negative directions, **we assign the PCs positive directions which enlarge representations.** For domain classes, while their RCs are nearly orthogonal, the relationships between their PCs vary significantly, ranging from highly positively correlated to highly negatively correlated. When a class's PC opposes the PC of the class assigned to the watermark task, the dominant dimension of the watermark logits may shrink due to stretching, causing the watermark label to shift toward this opposing class, making this class a likely deformation label.

Proposition 6.2.1 (Deformation Label)**.** *Deformation labels typically emerge in classes whose PCs have a significantly negative cosine similarity with the target class PC assigned watermark tasks.*

We validate this observation experimentally by analyzing the cosine similarity between the varying domain classes' PCs assigned watermark tasks and their deforma-

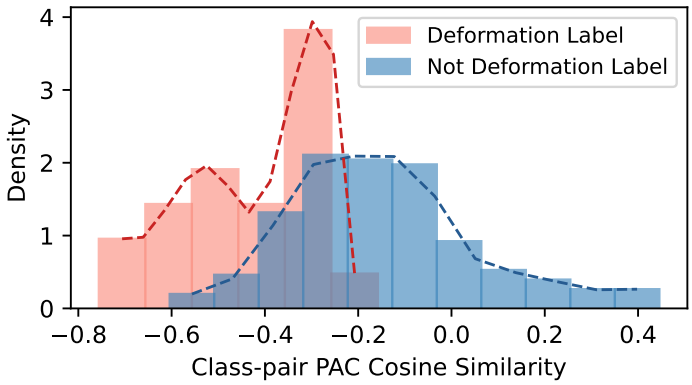


Figure 6.10: **Density** histogram of PC cosine similarity between classes contains watermarks and others. The ratio of deformation labels to non-deformation labels is 1 : 9.

tion labels after WRK. Figure 6.10 presents the result, which indicates that deformation labels consistently exert opposing stretching, with negative cosine similarity to the watermark label. In contrast, non-deformation labels' PCs exhibit higher orthogonality with the watermark label. Therefore, to ensure deformation label consistency, the watermark label should be fine-tuned or chosen to have **only one possible deformation label**, as demonstrated with class 0 in the experiments in Section 6.2.4.3.1.

6.3 Experimental Evaluation for Class-Feature Watermarks (CFW)

6.3.1 Setups

Datasets and Models. We evaluate four tasks spanning three domains: ResNet-18 [44] trained on **image** datasets (CIFAR-10 [57] and CIFAR-20 [57]), DPCNN [51] with BERT embeddings trained on a **text** dataset (DBPedia [16]), and VGG19-BN [91] trained on an **audio** dataset (Google Speech Commands [105]). CIFAR-10 and CIFAR-20 each contain 50,000 training and 10,000 test images (3x32x32); CIFAR-

10 has 10 labels, while CIFAR-20 uses 20 superclasses from CIFAR-100. DBPedia includes 760,000 samples categorized into 14 classes. Google Speech Commands comprises over 105,000 utterances of 35 words from various speakers, organized into 12 classes.

Class-Feature Watermark (CFW) Settings. The CFW dataset is created by selecting multiple out-of-domain (OOD) data types and assigning them to the same label. To limit its feature complexity, the watermark dataset size is kept between 0.2% and 0.3% of the domain dataset. For CIFAR-10, 100 samples are taken from 10 non-overlapping classes in CIFAR-100. For CIFAR-20, 100 samples are selected from 4 classes in ImageNet [29]. For DBPedia, 1,000 samples are drawn from Amazon Reviews [89], and for Google Speech Commands, 200 samples are taken from 4 classes within the 'unknown' category. Since MEAs also use OOD data as queries, the CFW data is set to be entirely distinct from the classes in the adversarial pools. For instance, in the CIFAR-20 experiment, the adversarial pool consists of the first 100 ImageNet classes, while the CFW data is drawn from classes 530 to 533.

Model Extraction Attack (MEA) Settings. The MEA query pool includes both domain and OOD samples, from which MEAs sample using active learning algorithms [76, 108]. Following Section 6.2.2.1, the domain subset constitutes 5% of the training dataset and is also used for removal attacks. For image tasks, the OOD samples are from ImageNet32 [29], with a query budget of 25,000. For DBPedia, the pool is AG News dataset [114], with a budget of 50,000, and for Speech Commands, the pool includes classes from version 0.02 absent in version 0.01, with a budget of 100,000. The copy model architectures match the victim models in default experiments. Further, Section 6.3.6 reports cross-experiments using ResNet-18, MobileNetV2 [85], and VGG19-BN to evaluate the impact of architectures on CFW.

Metrics. Since CFW is verified with clustering, in addition to WSR, we introduce the following two metrics.

Intra-class Variance (Intra Var, Var). This metric calculates the mean squared distance from each sample to its class centroid in the t-SNE [102] reduced representation space (normalized to [-100, 100]). For a model F and the watermark dataset D_w, it is computed as:

$$\text{Var}(F, D_w) = 1/|D_w| \sum_{\mathbf{x} \in D_w} ||\text{t-SNE}(F_{\theta^L}(\mathbf{x})) - \mu||^2, \tag{6.17}$$

where μ is the centroid of the reduced representations of watermark tasks, $\mu = \mathbf{E}(\text{t-SNE}(F_{\theta^L}(D_w)))$.

Label Clustering ($\mathbf{WSR_{LC}}$). This metric evaluates the clustering behavior on the watermark label y_{w} and the deformation label y_{deform}:

$$\text{WSR}_{\text{LC}} = \text{WSR}(F, D_w | y_{\text{w}}) + \text{WSR}(F, D_w | y_{\text{deform}}). \tag{6.18}$$

6.3.2 Overall Evaluation of CFW

We evaluate the performance of class-feature watermarks on four primary properties: their impact on the model utility (**Prop.1**), MEA transferability (**Prop.2**), their correctness (**Prop.3**), and, importantly, their (MEA-post) resilience against removal attacks (**Prop.4**, **Prop.5**). Table 6.6 shows CFW performance across four task domains, with WRK attacks testing its resilience. Additionally, Table 6.7 provides a supplementary evaluation of CFW resilience under another 6 removal attacks.

Results. CFW achieves significantly higher WSR_{LC} on watermarked models compared to non-watermarked models, enabling high-confidence verification. For copy models extracted via MEA, CFW demonstrates notable transferability which is ensured by the RepS optimization (Section 6.2.4.2.1). Additionally, WSR_{LC} strongly correlates with the copy model's accuracy, as CFW uses real-life samples, resulting in similar extraction outcomes. The impact of CFW on model performance is limited to 0.4%, which is much smaller than the over 1.5% degradation caused by existing black-box watermarks (Table 6.2). Lastly, resilience analysis in Tables 6.6 and 6.7

Table 6.6: Performance of Class-Feature (CF) Watermark

Tasks	**Non-watermark**		**Victim Model**			
Removal	**None**		**None**		**WRK**	
Metrics (%)	ACC	WSR_{LC}	ACC	WSR_{LC}	ACC	WSR_{LC}
CIFAR-10	93.55	20.60	93.26	100.00	91.95	96.80
CIFAR-20	81.61	6.60	81.26	100.00	80.54	96.80
DBPedia	98.17	15.28	98.03	100.00	97.85	94.51
Speech Commands	97.36	8.80	97.14	100.00	96.03	95.15

Tasks	**Copy Model**						
Removal	**Model**	**None**			**WRK**		
Metrics (%)	**Extraction**	ACC	FID	WSR_{LC}	ACC	FID	WSR_{LC}
CIFAR-10	MExMI	89.29	92.70	94.00	88.94	90.13	79.13
	ActiveThief	85.89	88.34	87.92	87.26	89.27	74.31
CIFAR-20	MExMI	80.64	82.35	85.15	80.18	81.97	70.58
	ActiveThief	71.41	77.47	81.55	72.15	77.30	64.85
DBPedia	MExMI	95.15	96.62	97.55	94.83	95.76	88.58
	ActiveThief	91.35	92.82	94.92	91.51	93.05	80.53
Speech Commands	MExMI	96.46	97.82	95.03	95.96	96.79	82.11
	ActiveThief	96.33	97.73	94.13	94.61	95.20	79.21

Table 6.7: Resilience of CFW against Other Removal Attacks

Removal	**Victim Model**			**Copy Model**(MExMI)		
Metrics (%)	ACC↑	WSR_{LC}↑	Var($\times 10^2$)↓	ACC↑	WSR_{LC}↑	Var($\times 10^2$)↓
Non-Watermark	93.55	20.60	18.18	89.81	24.86	19.78
Non-Removal	93.22	100.00	1.68	89.20	93.20	2.74
NC	92.38	98.90	1.66	89.04	73.29	5.15
I-BAU	91.49	98.70	1.77	89.04	80.40	3.20
CLP	87.44	99.20	2.00	87.01	89.40	4.20
Fine pruning	84.20	100.00	1.78	67.30	96.80	3.16
NAD	91.76	98.11	1.81	88.20	73.15	4.87
AT	92.15	99.54	1.64	89.23	73.40	4.66
WRK	91.95	96.80	2.89	88.37	79.13	4.90

Arrows represent the trend toward better watermark performance.

shows that CFW maintains a WSR_{LC} above 90% across all removal attacks in victim models with stable intra-class variance (var). In copy models, WSR_{LC} remains above 70.15%, which is affected by MEA distortions. Notably, the attacked intra-class variance in CFW models is lower than one-third of that in non-watermarked models, which also provides a strong indication of watermark presence.

Explanations of Unexpected Results. In Table 6.7, Fine pruning and CLP occasionally cause accuracy degradation exceeding 2%. This occurs when at least one neuron must be removed.

6.3.3 Evaluation on CFW Variants

In this section, we conduct ablation experiments on the two CFW optimization steps: RepS and PD^3, and compare RepS with the existing SNNL [56] algorithm to optimize RE. The experiments on CIFAR-10 evaluate key metrics, including maximum representation orthogonality ($\mathcal{O}_\perp$, Equation 6.10) and pairwise distance projections on deformation directions (PD^3, Equation 6.14). Table 6.8 presents the results, while Figure 6.11 shows the watermark decoupling curves under increasing removal intensity.

Results. Table 6.8 reveals several key observations. Firstly, both RepS and SNNL [56] significantly reduce $\mathcal{O}_\perp$, which benefits MEA transferability. However, SNNL causes substantial utility degradation exceeding 3%, while offering suboptimal transferability, indicating its incompatibility with CFW. In contrast, RepS enhances transferability without severely impacting performance. Regarding copy model WSR and intra-class variance (Var), RepS improves transferability but fails to constrain Var during MEA. By contrast, PD^3 optimization reduces PD^3 by 30×, which indicates improved stability. As a result, PD^3 lowers copy model variance from 10.30×10^2 to below 3.0×10^2, which approaches the victim model's variance of around 2×10^2. Consequently, label clustering (WSR_{LC}) is enhanced by PD^3, increasing from 70.42% to 84.04%.

Figure 6.11 illustrates the resilience of key performance metrics in victim and substitute models under increasing removal intensity. When comparing Figures 6.11a and 6.11b or Figures 6.11d and 6.11e, label clustering (WSR_{LC}) consistently outperforms WSR in resilience. For instance, in the *CFW with PD^3 only* variant, WSR drops below 20% even though the watermark remains clustered, as shown in Figure 6.11c. This indicates that WSR leads to information loss, which label clustering mitigates. Additionally, Figures 6.11a and 6.11d show that *CFW with RepS only* improves WSR resilience, but Figures 6.11c and 6.11f reveal it cannot maintain clus-

Table 6.8: Evaluation Results of Class-Feature Watermark (CFW) Variants

Watermarks	**Victim Model**				
Metrics	$\mathcal{O}_\perp \downarrow$	$\text{PD}^3 \downarrow$	ACC/% ↑	$\text{WSR}/\text{WSR}_{\text{LC}}$/% ↑	$\text{Var}(\times 10^2) \downarrow$
CFW w/o PD^3 or RepS	0.81	3.98	93.70	100.00	2.59
CFW w/ SNNL[56] only	0.15	2.51	90.13	99.20	4.14
CFW w/ RepS only	0.08	1.78	93.31	100.00	1.92
CFW w/ PD^3 only	0.96	5.93e−2	93.40	100.00	1.35
CFW	0.25	8.10e−2	93.26	100.00	1.68
Watermarks	**Copy Model**				
Metrics	ACC/% ↑	FID/% ↑	WSR/% ↑	WSR_{LC}/% ↑	$\text{Var}(\times 10^2) \downarrow$
CFW w/o PD^3 or RepS	89.55	92.54	57.13	70.42	10.30
CFW w/ SNNL[56] only	87.18	90.49	73.20	77.60	11.12
CFW w/ RepS only	89.85	92.85	92.10	93.80	8.17
CFW w/ PD^3 only	89.85	92.11	68.25	84.04	2.47
CFW	89.29	92.70	91.20	94.00	2.74

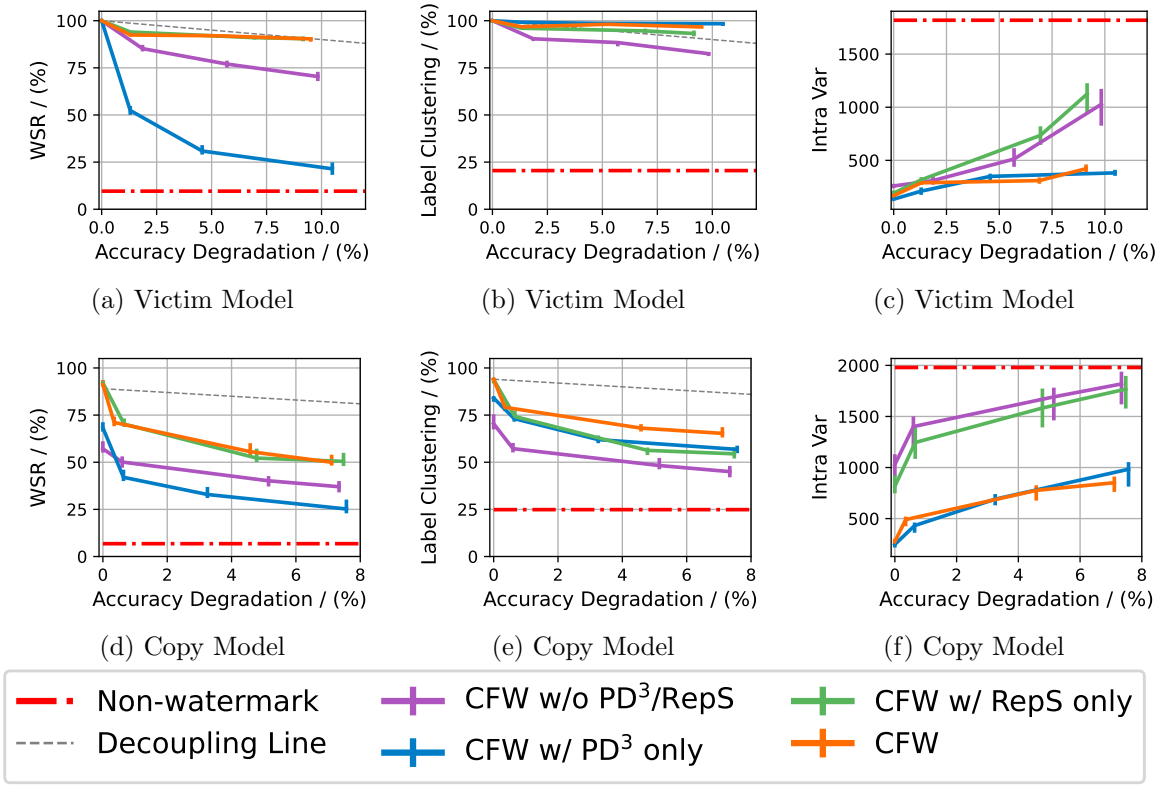
Arrows represent the trend toward better watermark performance.

tering stability. In contrast, when CFW includes PD^3 optimization, both clustering stability (Figures 6.11c and 6.11f) and label clustering resilience (Figure 6.11e) are significantly enhanced.

6.3.4 The Impact of PD^3 on CFW Stability

This section evaluates the relationship between PD^3 and the clustering stability of CFW, which is measured by intra-class variance. Here, two scenarios are observed: without attack and with WRK attacks. Each experiment is further divided into two setup conditions: one with RepS and one without. The experiments are conducted on CIFAR-10. Figure 6.12 visualizes these relationships for each scenario.

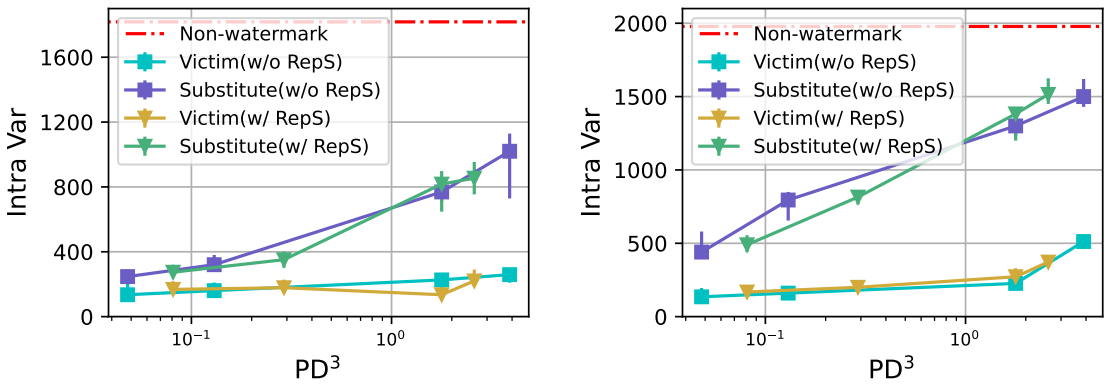
First, we observe that PD^3 optimization significantly affects the clustering stability of the copy (substitute) model. When PD^3 is below 0.1, both the MEA-post and MEA-removal-post stability of the copy (substitute) model achieve favorable conditions, with consistently low variance. However, as PD^3 increases, the MEA-post stability (Figure 6.12a) and removal-post stability (Figure 6.12b) decline rapidly. These findings indicate that, in terms of representation space clustering, CFW becomes unstable to MEA and removal attacks if PD^3 is not properly optimized. Finally, RepS does not appear to have a clear impact on the stability of watermark tasks.



(a) Victim Model (b) Victim Model (c) Victim Model

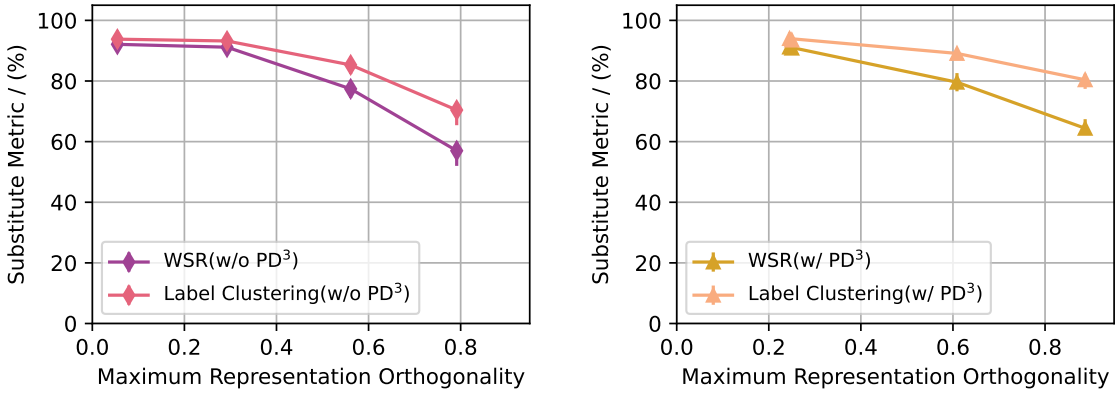
(d) Copy Model (e) Copy Model (f) Copy Model

Figure 6.11: Watermark decoupling curves of CFW. Vertical lines indicate error bars derived through interpolation due to variability in accuracy degradation across experiments.



(a) CFW (b) WRK-attacked CFW

Figure 6.12: PD^3 versus Intra-class Variance. The vertical lines represent error bars.

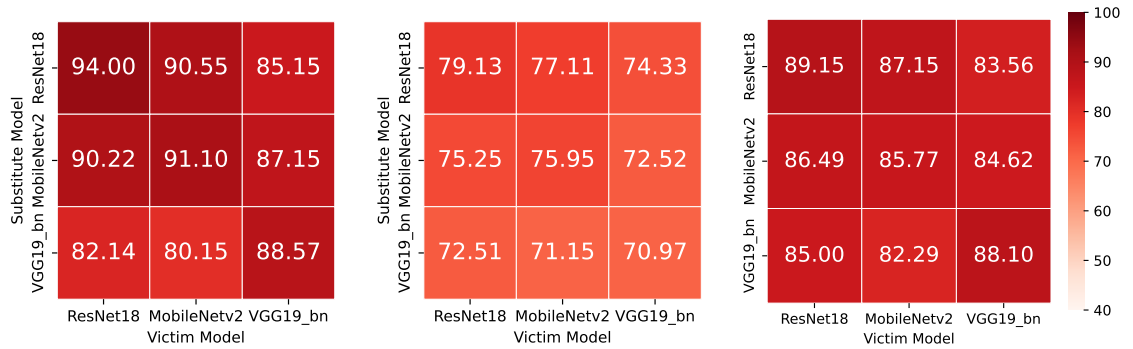


(a) w/o PD^3 (b) w/ PD^3

Figure 6.13: MEA transferability versus the maximum representation orthogonality. The vertical lines represent error bars.

6.3.5 The Impact of Maximum Representation Orthogonality on MEA Transferability

This section investigates the relationship between maximum representation orthogonality ($\mathcal{O}_\perp$) and MEA transferability for CFW, evaluated through the copy model's WSR and WSR_{LC}. Two sets of experiments are conducted: one with PD^3 optimization and one without, where $\mathcal{O}_\perp$ is controlled by the RepS coefficient. The experiments use the CIFAR-10 dataset. The results presented in Figure **??** reveal several key findings. First, when $\mathcal{O}_\perp$ is below 0.3, MEA transferability consistently reaches optimal levels. Additionally, even when $\mathcal{O}_\perp$ exceeds 0.8, WSR remains above 50%, primarily because CFW is constructed with real-life samples. Lastly, while PD^3 has limited impact on WSR, it significantly enhances WSR_{LC}, achieving over 80% even when $\mathcal{O}_\perp > 0.8$. This improvement is attributed to PD^3's ability to preserve clustering stability during MEA.



(a) Label Clustering (b) WRK-Attacked Label Clustering (c) MEA Accuracy

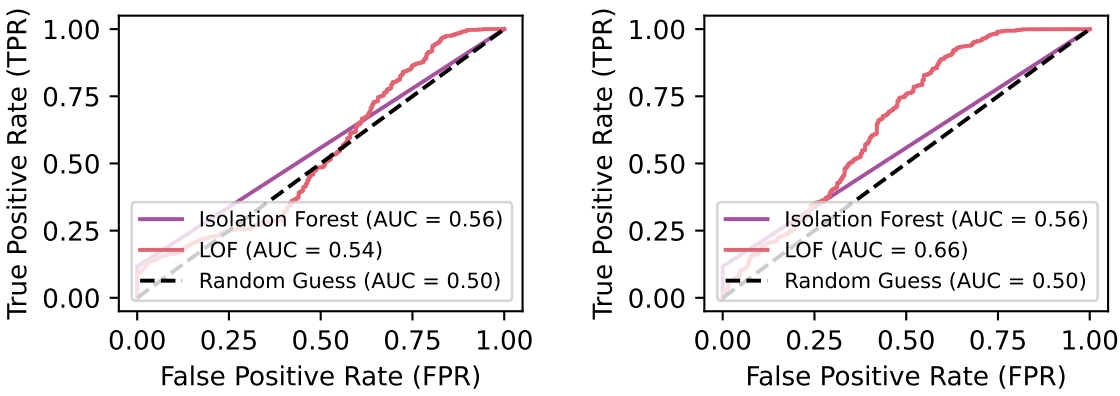
Figure 6.14: Performance of CFW with different architectures used in model extraction attacks. The horizontal labels are victim models, and the vertical labels are copy (substitute) models.

6.3.6 Ablation Study: The Impact of Copy Model's Architectures on Class-Feature Watermarks

In model extraction attacks, the copy model's architecture used by the adversary usually differs from that of the victim model. Therefore, we study the consistency of CFW across different architectures. Specifically, we cross-test three architectures: ResNet18, MobileNet, and VGG19-bn on CIFAR-10. In this setup, the victim and copy models are assigned different architectures in each combination, and Figure 6.14 presents CFW's performance using heatmap grids. The results indicate that CFW's performance is minimally affected by model architecture, displaying high consistency. This is because the CFW functions as a task and is independent of the underlying architectures.

6.3.7 Discussions

Piracy Attack. A piracy attack occurs when the adversaries embed their watermarks in a stolen model to evade watermark detection. This creates ambiguity in ownership



(a) Victim Model

(b) Subsititute Model (MExMI)

Figure 6.15: AUC results of abnormal detections on CFW.

verification, as two watermarks exist simultaneously. To prevent such ownership confusion, the simplest countermeasure is time stamping the watermark model, the samples, and the verifier through a trusted and accessible platform, such as privately uploading to an open-source repository. This ensures that even if the adversary performs a piracy attack, they cannot predate the defender's verified time-stamp.

Watermark Detection. Section 6.3.2 has yet to discuss the stealthiness property (**Prop.6**). When the stolen model is deployed online, adversaries might filter out potential watermark data and refuse to provide correct query labels to evade watermark verification. To assess CFW's stealthiness, we evaluate two anomaly detection methods: Local Outlier Factor (LOF) [20] and Isolation Forest [66], applied to the last hidden layer, following EWE [50]. Since CFW relies on clustering in the representation space, these methods infer watermark queries as high-density points, contrary to their original detection principles. Figure 6.15 shows AUC results on the CIFAR-10 dataset with 1,200 queries. The highest AUC reached only 0.66, indicating that watermark queries are difficult to distinguish.

Chapter 7

Conclusion

This study investigates two optimization frameworks that deepen the potential threats of model extraction attacks. One framework highlights their connection to data privacy. The other focuses on their interaction with neuron-level information leakage.

First, we propose a model extraction attack (ME, MEA) crossover membership inference attack (MI, MIA) framework called MExMI, where the model and training data privacy can trigger a chain reaction to boost the performance of both attacks. The framework is generic in that it can adopt various ME and MI attacks. In our experiments, MExMI improves the fidelity of copy models to 94.07%, up from the basic ME by 11.14%. Meanwhile, the MI accuracy and precision can reach 83.20% and 84.13% without additional query budget, on par with state-of-the-art MI attack which requires about 10 times more queries.

Second, we pioneer 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 neuron to model level, ushering learning-based ME into a new era for challenging high-fidelity model extraction.

To address the threats posed by model extraction attacks, we propose two defense mechanisms from different perspectives. First, we introduce a novel proactive defensive training strategy, Stochastic Norm Enlargement (SNE), which enhances the inherent resistance of victim models by making them harder to extract. SNE is extensively evaluated on real-world datasets and state-of-the-art models under various ME attacks.

Second, we investigate black-box model watermarking techniques as a passive protection against infringement by model extraction attacks. Our key contribution lies in identifying critical gaps in the resilience of existing methods. We find that current approaches overestimate their robustness due to insufficient evaluation against watermark removal. To address this gap, we introduce Watermark Removal attacK (WRK), a systematic framework that adaptively breaks model attention on sample-wise artifacts of state-of-the-art watermark tasks, even if they are highly entangled. To mitigate these vulnerabilities, we propose Class-Feature Watermarks (CFWs), embedding class-level artifacts instead of sample-wise triggers. CFW constructs a distinct watermark task using cross-domain real-life samples, ensuring its resilience against WRK. To maintain resilience in MEA-post copy models, we optimize CFW's transferability and stability during MEA. Extensive experiments confirm that CFW achieves high accuracy, strong MEA transferability, robust resilience against various removal attacks, and minimal impact on model utility.

References

[1] 5-layer cnn model in pytorch examples. https://github.com/pytorch/examples/blob/main/mnist_hogwild/main.py.

[2] Amazon aws marketplace. https://aws.amazon.com/marketplace/solutions/machine-learning/pre-trained-models.

[3] Amazon aws marketplace. https://aws.amazon.com/marketplace/solutions/machine-learning/pre-trained-models.

[4] Amazon sagemaker. https://aws.amazon.com/sagemaker/. Accessed: 2024-12-22.

[5] AutoML by Google. https://cloud.google.com/automl. Accessed: 2024-12-22.

[6] Azure Machine Learning by Microsoft. https://azure.microsoft.com/en-us/products/machine-learning/. Accessed: 2024-12-22.

[7] Google gloud ml. https://cloud.google.com/vertex-ai.

[8] Huawei cloud ai gallery. https://developer.huaweicloud.com/develop/aigallery/home.html.

[9] Huawei cloud modelarts. https://console-intl.huaweicloud.com/modelarts.

[10] Huawei codelab. https://codelabs.developer.huaweicloud.com.

[11] Microsoft azure. https://azure.microsoft.com/services/machine-learning.

[12] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[13] William Aiken, Hyoungshick Kim, Simon Woo, and Jungwoo Ryoo. Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. *Computers & Security*, 106:102277, 2021.

[14] Altrum AI. Model theft and llm ip protection: Securing your competitive advantage. https://www.altrum.ai/blog/model-theft-and-llm-ip-protection-securing-your-competitive-advantage, 2024. Accessed: 2025-04-28.

[15] Animashree Anandkumar, Rong Ge, and Majid Janzamin. Learning overcomplete latent variable models through tensor methods. In *Conference on Learning Theory*, pages 36–112. PMLR, 2015.

[16] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[17] Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified neural network watermarks with randomized smoothing. In *International Conference on Machine Learning*, pages 1450–1465. PMLR, 2022.

[18] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*

2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 5050–5060, 2019. https://proceedings.neurips.cc/paper/2019/hash/1cd138d0499a68f4bb72bee04bbec2d7-Abstract.html.

[19] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[20] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104. ACM, 2000.

[21] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1309–1326, 2020.

[22] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[23] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[24] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. Refit: a unified watermark removal framework for deep learning systems with limited data. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 321–335, 2021.

[25] Yanjiao Chen, Rui Guan, Xueluan Gong, Jianshuo Dong, and Meng Xue. D-dae: Defense-penetrating model extraction attacks. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 382–399. IEEE, 2023.

[26] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

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

[28] Gianna M Del Corso, Antonio Gulli, and Francesco Romani. Ranking a stream of news. In *Proceedings of the 14th international conference on World Wide Web*, pages 97–106, 2005.

[29] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[30] Timothy Dozat. Incorporating nesterov momentum into adam. *International Conference on Learning Representations (ICLR) workshop*, 2016.

[31] Rong Du, Qingqing Ye, Yue Fu, and Haibo Hu. Collecting high-dimensional and correlation-constrained data with local differential privacy. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2021.

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

[33] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. `https://openreview.net/forum?id=6Tm1mposlrM`.

[34] 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.

[35] Haoyu Fu, Yuejie Chi, and Yingbin Liang. Guaranteed recovery of one-hidden-layer neural networks via cross entropy. *IEEE transactions on signal processing*, 68:3225–3235, 2020.

[36] Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan Ö Arık, Larry S Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling cost. In *European Conference on Computer Vision (ECCV)*, pages 510–526. Springer, 2020.

[37] Xueluan Gong, Yanjiao Chen, Wenbin Yang, Guanghao Mei, and Qian Wang. Inversenet: Augmenting model extraction attacks with training data inversion. *JICAI*, 2021.

[38] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[39] Google Cloud. Google cloud terms of service. `https://cloud.google.com/terms/service-terms`, 2025. Accessed: 2025-04-28.

[40] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

[41] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7, 2019.

[42] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[45] Seira Hidano, Takao Murakami, and Yusuke Kawamoto. Transmia: Membership inference attacks using transfer shadow training. *arXiv preprint arXiv:2011.14661*, 2020.

[46] Yujiao Hu, Qingmin Jia, Yuan Yao, Yong Lee, Mengjie Lee, Chenyi Wang, Xiaomao Zhou, Renchao Xie, and F Richard Yu. Industrial internet of things intelligence empowering smart manufacturing: A literature review. *IEEE Internet of Things Journal*, 11(11):19143–19167, 2024.

[47] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[48] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1345–1362, 2020.

[49] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*, 2015.

[50] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In

30th USENIX Security Symposium (USENIX Security 21), pages 1937–1954, 2021.

[51] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 562–570. Association for Computational Linguistics, 2017. https://doi.org/10.18653/v1/P17-1052.

[52] 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 (EuroSP)*, pages 512–527, 2019.

[53] Sanjay Kariyappa, Atul Prakash, and Moinuddin 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*, pages 13814–13823, 2021.

[54] Sanjay Kariyappa and Moinuddin K Qureshi. Defending against model stealing attacks with adaptive misinformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2020.

[55] Byungjoo Kim, Suyoung Lee, Seanie Lee, Sooel Son, and Sung Ju Hwang. Margin-based neural network watermarking. In *International Conference on Machine Learning*, pages 16696–16711. PMLR, 2023.

[56] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

[57] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[59] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[60] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[61] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[62] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[63] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[64] Yige Li, Xixiang Lyu, Xingjun Ma, Nodens Koren, Lingjuan Lyu, Bo Li, and Yu-Gang Jiang. Reconstructive neuron pruning for backdoor defense. In *International Conference on Machine Learning*, pages 19837–19854. PMLR, 2023.

[65] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 113–131, 2020.

[66] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[67] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pages 273–294. Springer, 2018.

[68] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019.

[69] Peizhuo Lv, Hualong Ma, Kai Chen, Jiachen Zhou, Shengzhi Zhang, Ruigang Liang, Shenchen Zhu, Pan Li, and Yingjun Zhang. Mea-defender: A robust watermark against model extraction attack. *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

[70] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[71] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[72] Mantas Mazeika, Bo Li, and David Forsyth. How to steer your adversary: Targeted and efficient model stealing defenses with gradient redirection. In *International Conference on Machine Learning*, pages 15241–15254. PMLR, 2022.

[73] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[74] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4949–4958, 2019.

[75] Soham Pal, Yash Gupta, Aditya Kanade, and Shirish K. Shevade. Stateful detection of model extraction attacks. *CoRR*, abs/2107.05166, 2021.

[76] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. Activethief: Model extraction using active learning and unannotated public data. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 865–872, 2020.

[77] 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.

[78] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[79] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 896–902. IEEE, 2015.

[80] Jonathan Rosenthal, Eric Enouen, Hung Viet Pham, and Lin Tan. Disguide: disagreement-guided data-free model extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9614–9622, 2023.

[81] David Saad and Sara Solla. Dynamics of on-line gradient descent learning for multilayer neural networks. *Advances in neural information processing systems*, 8, 1995.

[82] Amir Mahdi Sadeghzadeh, Amir Mohammad Sobhanian, Faezeh Dehghan, and Rasool Jalili. Hoda: Hardness-oriented detection of model extraction attacks. *IEEE Transactions on Information Forensics and Security*, 2023.

[83] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29:1163–1171, 2016.

[84] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *26th Annual Network and Distributed System Security Symposium (NDSS)*, 2018.

[85] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[86] Hanie Sedghi and Anima Anandkumar. Provable methods for training neural networks with sparse connectivity. *ICLR*, 2015.

[87] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[88] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (ICLR)*, 2018. `https://openreview.net/forum?id=H1aIuk-RW`.

[89] Amazon Web Services. Amazon customer reviews dataset, 2018. `https://registry.opendata.aws/amazon-reviews/`, Accessed: 2024-12-22.

[90] 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.

[91] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[92] Samrath Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5971–5980, 2019.

[93] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[94] Daniel A Spielman, Huan Wang, and John Wright. Exact recovery of sparsely-used dictionaries. In *Conference on Learning Theory*, pages 37–1. JMLR Workshop and Conference Proceedings, 2012.

[95] Charles Stein, Persi Diaconis, Susan Holmes, and Gesine Reinert. Use of exchangeable pairs in the analysis of simulations. *Lecture Notes-Monograph Series*, pages 1–26, 2004.

[96] Minxue Tang, Anna Dai, Louis DiValentin, Aolin Ding, Amin Hass, Neil Zhenqiang Gong, and Yiran Chen. Modelguard: Information-theoretic defense against model extraction attacks. 2023.

[97] Minxue Tang, Anna Dai, Louis DiValentin, Aolin Ding, Amin Hass, Neil Zhenqiang Gong, Yiran Chen, et al. {ModelGuard}:{Information-Theoretic} defense against model extraction attacks. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5305–5322, 2024.

[98] Markus Thom and Günther Palm. Sparse activity and sparse connectivity in supervised learning. *Journal of Machine Learning Research*, 14(4), 2013.

[99] Yuandong Tian. Student specialization in deep rectified networks with finite width and input dimension. In *International Conference on Machine Learning*, pages 9470–9480. PMLR, 2020.

[100] 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.

[101] Jean-Baptiste Truong, Pratyush Maini, Robert J Walls, and Nicolas Papernot. Data-free model extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4771–4780, 2021.

[102] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[103] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE symposium on security and privacy (SP)*, pages 707–723. IEEE, 2019.

[104] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3), June 2020. `https://doi.org/10.1145/3386252`.

[105] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[106] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Model extraction attacks on graph neural networks: Taxonomy and realization. *CoRR*, 2020. `https://arxiv.org/abs/2010.12751`.

[107] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[108] Yaxin Xiao, Qingqing Ye, Haibo Hu, Huadi Zheng, Chengfang Fang, and Jie Shi. Mexmi: Pool-based active model extraction crossover membership inference. *Advances in Neural Information Processing Systems*, 35:10203–10216, 2022.

[109] Mengjia Yan, Christopher W Fletcher, and Josep Torrellas. Cache telepathy: Leveraging shared resource attacks to learn {DNN} architectures. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2003–2020, 2020.

[110] Fan Yang, Zhiyuan Chen, and Aryya Gangopadhyay. Using randomness to improve robustness of tree-based models against evasion attacks. *IEEE Transactions on Knowledge and Data Engineering*, 34(2):969–982, 2020.

[111] 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, 2018.

[112] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[113] Kaiyuan Zhang, Siyuan Cheng, Guangyu Shen, Guanhong Tao, Shengwei An, Anuran Makur, Shiqing Ma, and Xiangyu Zhang. Exploring the orthogonality and linearity of backdoor attacks. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 225–225. IEEE Computer Society, 2024.

[114] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[115] Xiao Zhang, Yaodong Yu, Lingxiao Wang, and Quanquan Gu. Learning one-hidden-layer relu networks via gradient descent. In *The 22nd international conference on artificial intelligence and statistics*, pages 1524–1534. PMLR, 2019.

[116] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 253–261, 2020.

[117] Shuai Zhao, Liguang Zhou, Wenxiao Wang, Deng Cai, Tin Lun Lam, and Yangsheng Xu. Splitnet: Divide and co-training. *CoRR*, 2020. `https://arxiv.org/abs/2011.14660`.

[118] Zhong-Qiu Zhao, Yiu-ming Cheung, Haibo Hu, and Xindong Wu. Corrupted and occluded face recognition via cooperative sparse representation. *Pattern Recognition*, 56:77–87, 2016.

[119] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. Bdpl: A boundary differentially private layer against machine learning model extraction attacks. In *European Symposium on Research in Computer Security*, pages 66–83. Springer, 2019.

[120] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. Protecting decision boundary of machine learning model with differentially private perturbation. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[121] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-free backdoor removal based on channel lipschitzness. In *European Conference on Computer Vision*, pages 175–191. Springer, 2022.

[122] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *International conference on machine learning*, pages 4140–4149. PMLR, 2017.

[123] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

[124] Mo Zhou, Rong Ge, and Chi Jin. A local convergence theory for mildly over-parameterized two-layer neural network. In *Conference on Learning Theory*, pages 4577–4632. PMLR, 2021.

[125] Hongyu Zhu, Sichu Liang, Wentao Hu, Fangqi Li, Ju Jia, and Shilin Wang. Reliable model watermarking: Defending against theft without compromising on evasion. *arXiv preprint arXiv:2404.13518*, 2024.

[126] Rui Zhu, Di Tang, Siyuan Tang, XiaoFeng Wang, and Haixu Tang. Selective amnesia: On efficient, high-fidelity and blind suppression of backdoor effects in trojaned machine learning models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1–19. IEEE, 2023.