Saarland University
Department of Computer Science

(Un)Trustworthy Data in Adversarial Machine Learning

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

von
Rui Wen

Saarbrücken, 2024

Tag des Kolloquiums:	26 März 2025
Dekan:	Prof. Dr. Roland Speicher

Prüfungsausschuss:

Vorsitzender:	Prof. Dr. Thorsten Herfet
Berichterstattende:	Prof. Dr. Michael Backes
	Dr. Matthew Jagielski
	Prof. Dr. Tianhao Wang
	Dr. Yang Zhang
Akademischer Mitarbeiter:	Dr. Mingjie Li

Zusammenfassung

Maschinelles Lernen ist in verschiedenen Branchen unverzichtbar geworden, da es Innovationen vorantreibt und datengetriebene Entscheidungsprozesse ermöglicht. Im Zentrum dieser Technologie steht die zentrale Rolle von Daten, die grundlegend für das Modelltraining sind und die Leistung direkt beeinflussen. Allerdings macht diese Abhängigkeit von Daten maschinelle Lernsysteme auch anfällig für Schwachstellen, insbesondere im Hinblick auf Datenschutz und Sicherheit.

In dieser Dissertation untersuchen wir die Rolle von Daten im adversarialen maschinellen Lernen und konzentrieren uns dabei auf zwei große Herausforderungen: Datenschutzverletzungen und Datenvergiftung. Zunächst untersuchen wir Datenschutzverletzungen in modernen Modellen, indem wir einen Membership Inference Angriff gegen In-Context Learning vorschlagen. Wir zeigen, dass es selbst in eingeschränkten Umgebungen möglich ist, zu ermitteln, ob bestimmte Datenpunkte für das Training verwendet wurden, was erhebliche Risiken in sensiblen Bereichen wie Gesundheit und Finanzen birgt. Anschließend untersuchen wir, wie Daten als Angriffsfläche ausgenutzt werden können, indem wir eine robuste Vergiftungstechnik einführen, die derzeitige Abwehrmechanismen überwinden kann. Außerdem schlagen wir den ersten dynamischen Backdoor-Angriff vor, der flexible Trigger verwendet, um der Erkennung zu entgehen, und unterstreichen damit die Notwendigkeit stärkerer Abwehrmechanismen. Zum Schluss führen wir eine systematische Untersuchung durch, wie Datenmerkmale, wie etwa die Bedeutung von Daten, den Erfolg von Angriffen auf maschinelles Lernen beeinflussen. Unsere Ergebnisse deuten darauf hin, dass die Anpassung der Datenbedeutung entweder die Anfälligkeit erhöhen oder verringern kann, und bieten neue Strategien sowohl für Angriffe als auch für Verteidigungsmaßnahmen.

Diese Dissertation trägt zu einem tieferen Verständnis der adversarialen Dynamiken bei und hilft, sicherere und vertrauenswürdigere maschinelle Lernsysteme zu entwickeln.

Abstract

Machine learning has become indispensable across various industries, driving innovation and enabling data-driven decision-making. At the core of this technology is the critical role of data, which is fundamental to model training and directly impacts performance. However, this reliance on data also exposes machine learning systems to vulnerabilities, particularly around privacy and security.

In this dissertation, we explore the role of data in adversarial machine learning, focusing on two major challenges: data privacy leakage and data poisoning. First, we investigate privacy leakage in state-of-the-art models by proposing a membership inference attack against in-context learning. We show that even in restricted settings, it is possible to infer whether specific data points were used in training, posing significant risks in sensitive domains such as healthcare and finance. Next, we examine how data can be exploited as an attack surface, introducing a robust poisoning technique capable of bypassing current defenses. We also propose the first dynamic backdoor attack, which uses flexible triggers to evade detection, highlighting the need for stronger defense mechanisms. Finally, we conduct a systematic study on how data characteristics, such as data importance, affect the success of machine learning attacks. Our results suggest that adjusting data importance can either increase or reduce vulnerability, offering new strategies for both attacks and defenses.

This dissertation contributes to a deeper understanding of adversarial dynamics, helping to build more secure and trustworthy machine learning systems.

Background of this Dissertation

This dissertation builds on the following published papers, in which I played a central role as the lead author and contributor.

The idea of attacking In-Context Learning [P1] was initially proposed by Yang Zhang, who also introduced the first version of the Repeat Attack. Rui Wen subsequently refined the attack methods, developing a more advanced approach, Brainwash, designed to address more restrictive assumptions. Rui Wen was responsible for the implementation and evaluation. All authors contributed to the writing and review of the paper.

The concept of poisoning adversarial training [P2] was first proposed by Zhengyu Zhao, focusing on weakening adversarial robustness. Rui Wen introduced a more powerful attack that significantly reduced the effectiveness of adversarial training. Rui Wen was responsible for the implementation and evaluation. Tianhao Wang, Zhuoran Liu, Michael Backes, and Yang Zhang provide insightful discussion and feedback. All authors participated in the writing and review of the paper.

The proposal to relax the static assumptions of the backdoor attack [P3] originated from a joint discussion between Ahmed Salem, Rui Wen, and Yang Zhang. The algorithm design and implementation were done by Ahmed Salem, while Rui Wen took charge of the evaluation and defense mechanisms. Yang Zhang, Michael Backes, and Shiqing Ma provided critical feedback on the attack. All authors contributed to the writing and review of the paper.

The general idea of evaluating data influence on machine learning attacks [P4] was proposed by Yang Zhang. Rui Wen further developed this concept, using the findings to design stronger attacks and defenses. Rui Wen led the implementation and evaluation. All authors participated in the writing and review of the paper.

[P1] Wen, R., Li, Z., Backes, M., and Zhang, Y. Membership Inference Attacks Against In-Context Learning. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2024.

[P2] Wen, R., Zhao, Z., Liu, Z., Backes, M., Wang, T., and Zhang, Y. Is Adversarial Training Really a Silver Bullet for Mitigating Data Poisoning? In: *International Conference on Learning Representations (ICLR)*. 2023.

[P3] Salem, A., Wen, R., Backes, M., Ma, S., and Zhang, Y. Dynamic Backdoor Attacks Against Machine Learning Models. In: *IEEE European Symposium on Security and Privacy (Euro S&P)*. IEEE, 2022, 703–718.

[P4] Wen, R., Backes, M., and Zhang, Y. Understanding Data Importance in Machine Learning Attacks: Does Valuable Data Pose Greater Harm? In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2025.

Further Contributions of the Author

The author was also able to contribute to the following: [S1, S2, S3, S4, S5, S6, T1, T2, T3]

Published Papers:

[S1] Wen, R., Yu, Y., Xie, X., and Zhang, Y. LEAF: A Faster Secure Search Algorithm via Localization, Extraction, and Reconstruction. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2020, 1219–1232.

[S2] Liu, Y., Wen, R., He, X., Salem, A., Zhang, Z., Backes, M., Cristofaro, E. D., Fritz, M., and Zhang, Y. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2022, 4525–4542.

[S3] Wu, Y., Wen, R., Backes, M., Berrang, P., Humbert, M., Shen, Y., and Zhang, Y. Quantifying Privacy Risks of Prompts in Visual Prompt Learning. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2024.

[S4] Zhang, R., Li, H., Wen, R., Jiang, W., Zhang, Y., Backes, M., Shen, Y., and Zhang, Y. Instruction Backdoor Attacks Against Customized LLMs. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2024.

[S5] Jiang, Y., Shen, X., Wen, R., Sha, Z., Chu, J., Liu, Y., Backes, M., and Zhang, Y. Games and Beyond: Analyzing the Bullet Chats of Esports Livestreaming. In: *International Conference on Web and Social Media (ICWSM)*. AAAI, 2024, 761–773.

[S6] Zhang, M., Yu, N., Wen, R., Backes, M., and Zhang, Y. Generated Distributions Are All You Need for Membership Inference Attacks Against Generative Models. In: *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2024, 4827–4837.

Technical Reports:

[T1] Wen, R., Wang, T., Backes, M., Zhang, Y., and Salem, A. Last One Standing: A Comparative Analysis of Security and Privacy of Soft Prompt Tuning, LoRA, and In-Context Learning. *CoRR abs/2310.11397* (2023).

[T2] He, X., Wen, R., Wu, Y., Backes, M., Shen, Y., and Zhang, Y. Node-Level Membership Inference Attacks Against Graph Neural Networks. *CoRR abs/2102.05429* (2021).

[T3] Liu, Y., Wen, R., Backes, M., and Zhang, Y. Efficient Data-Free Model Stealing with Label Diversity. *CoRR abs/2404.00108* (2024).

Acknowledgments

This incredible Ph.D. journey has been a meaningful experience, thanks to the guidance, support, and companionship of so many wonderful people. As I write this acknowledgment, I feel overwhelmed with gratitude and find it hard to decide where to begin, given the many names that come to mind.

First and foremost, I would like to sincerely thank my advisor, Michael Backes, for giving me the opportunity to pursue my Ph.D. at CISPA. His constant support and guidance have made this journey possible.

I am very grateful to Yang Zhang for his invaluable mentorship throughout the Ph.D. program, which has extended beyond research to everyday life. His passion for both research and fitness has taught me the importance of maintaining a positive mindset, balance, and resilience.

A special thanks goes to Tianhao Wang, whose example as a researcher I deeply admire. He not only guided me through the academic world but also showed me how to manage stress, stay productive, and maintain emotional balance.

I also want to extend my gratitude to Matthew Jagielski, whose research inspired me to explore more about the interplay between data and machine learning attacks. His work pushed me to investigate new areas and broaden my understanding.

I am thankful to all of them also for serving on my dissertation committee. Thank them for their effort and time reviewing this dissertation.

This journey would not have been as enjoyable or fulfilling without my friends. Thank you, Zheng and Yiyong, for the daily hikes and laughter-filled chats; Ahmed, for your unwavering support, insightful discussions, and our restaurant adventures; and Jingjing, Zesheng, Tianshuo, Xinlei, Zeyu, Yukun, Xinyue, Yixin, Ziqing, and many others, for sharing fascinating topics, stories, and animations that enriched my days.

Lastly, and most importantly, I want to thank my parents. Their endless love and support have been the backbone of my courage and confidence. No matter where I am, their love has always reminded me that I am never alone. I love you all dearly.

Thank you all for being part of this journey!

Contents

List of Figures

List of Tables

1 Introduction

Machine learning (ML) has seen rapid advancements over the past decade, becoming a driving force behind technological innovation across various domains. One of the most significant developments in this area has been the rise of foundation models, such as large language models (LLMs) [99, 137, 1] and vision-language models (VLMs) [76]. These models have pushed the boundaries of artificial intelligence (AI), enabling transformative applications like real-time language translation [93], sophisticated text generation [90], and enhanced visual understanding [78]. For instance, LLMs such as GPT-4 [99] have revolutionized communication by enabling real-time translation, advanced text generation, and seamless human-computer interaction, while VLMs are pushing the boundaries of visual understanding in meme detection [110] and medical imaging [146].

At the heart of these advancements is the pivotal role of data in model training. The quality and diversity of data form the foundation for modern deep learning models, enabling them to learn patterns and generalize across a wide array of tasks. High-quality datasets ensure enhanced model performance, while diverse data reduces biases and fosters fairness. For example, Llama 3.1's [2] exceptional capabilities in translation, question answering, and code generation are rooted in its training on over 15 trillion tokens from varied sources, demonstrating the critical importance of rich datasets for developing versatile and powerful AI systems.

However, despite data being the bedrock of these advancements, it also introduces significant security challenges. Data plays a dual role in machine learning attacks: it can be manipulated to introduce malicious behavior or be exploited in privacy breaches. For instance, poisoning attacks [22, P2, P3, 26, 19] involve subtle manipulations of training data, which can severely degrade model performance or embed harmful behaviors into the model. Membership inference attacks [114, 125, 75, 74, 81, 21, P1], on the other hand, attempt to determine whether a specific data point was included in the model's training set, posing serious privacy concerns.

These dual vulnerabilities—data privacy leakage and data poisoning—present critical threats to the safety and integrity of ML models. Despite its fundamental role, there is much to learn about how data characteristics interact with and influence ML attack dynamics. As society increasingly depends on data-driven technologies for essential functions, understanding and mitigating these vulnerabilities is of utmost importance.

1.1 Our Contributions

In this dissertation, we address these critical issues surrounding the role of data in adversarial machine learning, focusing on data privacy leakage, data poisoning, and the impact of data characteristics on attack performance. Our dissertation begins by investigating the persistence of data privacy leakage across a range of ML models, including state-of-the-art LLMs. We then explore how data serves as a key attack surface in data poisoning attacks, where adversaries can manipulate data to introduce harmful behaviors into models. Our findings demonstrate that current defenses are insufficient to prevent these attacks. Finally, we conduct a comprehensive study on how different data characteristics, such as data importance, affect the performance of ML attacks, offering insights to guide the development of more robust and trustworthy machine learning systems. Our dissertation is built upon the following peer-reviewed

publications [P1, P2, P3, P4], and we summarize each contribution as follows:

Membership Inference Attacks Against In-Context Learning: In our first work [P1], we demonstrate the persistence of data privacy leakage in LLMs by proposing a membership inference attack against in-context learning. We show that even with only text-based outputs, it is possible to infer whether specific data points were part of the language model's demonstrations. Our results reveal that even when model outputs are restricted to simple categories, such as "positive" or "negative," an adversary can still determine the membership status of individual data points. Notably, our long-term investigation reveals that this leakage does not diminish over time, despite claims from developers that LLMs are becoming more privacy-conscious. This underscores the significant, ongoing risks of deploying LLMs in sensitive domains, such as healthcare or finance, where data confidentiality is paramount.

Robust Poisoning Attack: Our second work [P2] highlights how data can serve as an attack surface for introducing malicious behavior into ML models. Specifically, we study availability attacks, where imperceptible perturbations are applied to training data, severely degrading model performance on clean testing data. We introduce a novel poisoning technique that entangles the features of different classes, rendering the corrupted samples ineffective for training—even when adversarial training techniques are applied. Our findings challenge the conventional belief that adversarial training can successfully defend against poisoning attacks, providing empirical evidence that our method can bypass these defenses. This highlights the urgent need for more robust protection mechanisms in ML systems.

Dynamic Backdoor Attack: Our third work [P3] demonstrates that another important line of data poisoning attacks, backdoor attacks, can be strategically designed to bypass existing defenses, further escalating the urgency of addressing the data threats. Specifically, we relax the static trigger assumption and proposed the first dynamic backdoor, which allows dynamic triggers in both pattern and location. Our methods introduce the first algorithmic backdoor, giving adversaries greater flexibility in trigger design. The results show that our dynamic backdoor techniques achieve near-perfect success rates with minimal impact on the model's overall utility. Furthermore, we tested our approach against five state-of-the-art backdoor defense mechanisms, and in all cases, our attacks successfully bypassed these defenses, demonstrating the robustness and stealthiness of dynamic backdoors.

Understanding Data Importance in ML Attacks: Our final work [P4] presents a systematic analysis of how data characteristics, specifically data importance, influence the effectiveness of ML attacks. We empirically show that data points with varying importance levels have different vulnerabilities to attacks. Moreover, we demonstrate that manipulating the importance of specific data points can increase or decrease their vulnerability to attacks. These insights open up new avenues for both enhancing attack strategies and designing more adaptive, fine-grained defenses. By understanding how data importance shapes attack dynamics, we can develop more resilient and trustworthy ML systems.

1.2 Organization

The rest of this dissertation is organized as follows. We first present the needed preliminaries and background in Chapter 2. In Chapter 3, we investigate the membership leakage in LLMs. We challenge the widely held belief that adversarial training mitigates availability data poisoning attacks in Chapter 4, demonstrating that this consensus is misleading. In Chapter 5, we further propose novel backdoor attacks that effectively bypass existing defenses. Chapter 6 offers a systematic analysis of how data with different characteristics influence machine learning attacks, and how these insights can be leveraged to design more effective attacks and defenses. Finally, Chapter 7 presents the related work, and Chapter 8 concludes the dissertation.

2
Preliminaries and Background

2.1 Supervised Machine Learning

Supervised machine learning is a powerful approach that aims to create models capable of predicting outputs based on given inputs. In this context, a classification model is typically represented as a parameterized function, denoted by f_θ, which maps a feature vector $\mathbf{x}$ from the feature space $\mathcal{X}$ to an output y in the output space $\mathcal{Y}$. The output space $\mathcal{Y}$ encompasses all possible predictions the model can make.

To train the model, we require a dataset $\mathcal{D}$, consisting of pairs of feature vectors and their corresponding labels, i.e., $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where n denotes the number of data samples. The training process involves optimizing the model's parameters by running a learning algorithm such as gradient descent, guided by a defined loss function. The objective is to minimize the classification loss, expressed as:

$$\mathop{\mathbb{E}}_{(\mathbf{x},y)} [\mathcal{L}(f_\theta(\mathbf{x}), y)]$$

where $(\mathbf{x}, y) \in \mathcal{D}$ are samples from the training dataset. This optimization process iteratively updates the parameters θ, enabling the model to achieve optimal performance by accurately predicting outputs based on the learned relationships between inputs and their corresponding labels.

2.1.1 Data Importance

As discussed in the previous section, data samples play a crucial role in training machine learning models; however, their impact on model performance is not uniform. Recent research has highlighted the presence of certain data that exhibit a heightened influence on the utility of machine learning models [62, 47, 68, 66, 63]. The evaluation of individual training sample importance is a fundamental and complex problem in machine learning (ML), with far-reaching implications, particularly in the domain of data valuation. Understanding the importance of a single training sample within a learning task profoundly impacts data assessment [14, 47], allocation of resources, and enhancing interpretability [66, 109, 85, 103].

Leave-one-out (LOO) method has long been regarded as an intuitive approach for assessing the importance of data samples. Formally, let D and D_{val} represent the training set and the validation set, and $\mathcal{A}$ denote the learning algorithm. $U_{\mathcal{A},D_{\text{val}}}$ denotes the validation accuracy of the model trained on D using $\mathcal{A}$. The importance of a target sample z can be quantified as the difference in utility before and after incorporating the target sample into the training set, expressed as:

$$v_{loo}(z) \propto U_{\mathcal{A},D_{val}}(D) - U_{\mathcal{A},D_{val}}(D \backslash \{z\})$$

Nevertheless, evaluating the importance of all N samples in the training set necessitates retraining the model N times, resulting in computational heaviness. To address this limitation, Koh and Liang [66] proposed influence functions as an approximation method, significantly reducing the computational cost from $O(Np^2 + p^3)$ to $O(Np)$, where p represents the number of model parameters.

Despite the effectiveness of LOO, Ghorbani and Zou [47] have raised concerns about its ability to capture complex interactions between subsets of data. They argue that the Shapley value provides a more comprehensive framework for measuring data importance. The Shapley value, originally proposed by Shapley [123], assigns an importance value to each sample z in the training set using the following formulation:

$$\nu_{\text{shap}}(z) \propto \frac{1}{N} \sum_{S \subseteq D \setminus \{z\}} \frac{1}{\binom{N-1}{|S|}} \left[U_{\mathcal{A}, D_{val}}(S \cup \{z\}) - U_{\mathcal{A}, D_{val}}(S) \right]$$

To simplify the interpretation of the Shapley value assigned to each sample, one can conceptualize it as the contribution to accuracy in typical scenarios.

For instance, in a hypothetical scenario with 100 samples and a model achieving 90% accuracy, a valuable sample may contribute 2% accuracy, while a less valuable sample may only contribute 0.1%. Consequently, the importance value assigned to a valuable sample is 0.02, whereas for a less valuable sample, it is 0.001. Samples with an importance of 0 signify no contribution to the model's accuracy, while values below 0 suggest a detrimental impact, possibly due to incorrect labels or samples lying outside the distribution.

The Shapley value takes into account the contributions of all possible subsets of the training set, offering a more holistic assessment of data importance. However, the accurate computation of the Shapley value based on the defined formula necessitates training $\mathcal{O}(2^N)$ machine learning models, rendering it impractical for complex datasets. As a result, existing methods employ approximate algorithms to estimate the Shapley value. For instance, Ghorbani and Zou [47] introduced two Monte Carlo-based approaches for Shapley value approximation. To expedite evaluation time and enable analysis of large datasets, Jia et al. [62] utilized the K-nearest neighbors (KNN) algorithm to approximate the target learning algorithm, reducing the time complexity to $\mathcal{O}(N \log N)$.

2.2 Data Poisoning Attacks Against DNNs

Data poisoning attacks are a form of training-time attack where an adversary manipulates the training dataset to embed malicious behaviors into the model. These behaviors are carefully crafted by the adversary to impact the model's performance during testing, aligning with the attacker's objectives.

Poisoning attacks can be broadly categorized based on their goals into three types: availability attacks, targeted attacks, and backdoor attacks. In this dissertation, we mainly focus on two of them, i.e., availability attacks and backdoor attacks.

2.2.1 Availability Attacks

Unlike targeted poisoning [94, 121, 46] and backdoor attacks [50, 80, P3], which aim to degrade model performance on specific test samples, availability attacks (also known as indiscriminate poisoning) focus on degrading the model's performance across a broader set of arbitrary clean test samples. Traditional methods of availability attacks often rely on injecting noisy labels into the training data [19, 18, 94]. However, they can be easily detected [121, 129].

To evade detection, more recent methods employ "clean-label" poisoning, where imperceptible perturbations are added to the training data. This setting is our main focus in this dissertation. Clean-label poisoning techniques typically use the error-minimization [54, 136, 43] or error-maximization loss [40], with a pre-trained [40, 149, 136] or trained-from-scratch [54, 43] reference model. To ensure these perturbations are imperceptible, the adversary constrains them using a predefined parameter known as the poison budget ϵ_{poi}. This parameter defines the maximum allowable perturbation size, typically using a norm such as l_∞, ensuring that the added perturbations do not exceed a specified threshold, e.g., $\|\boldsymbol{\delta}^{\text{poi}}\|_\infty \leq \epsilon_{\text{poi}}$.

Adversarial Training as a Defense: While clean-label poisoning techniques are effective at bypassing simple defenses, they are vulnerable to adversarial training (AT), a robust defense method initially developed to defend against test-time adversarial examples [15, 138]. Adversarial training has recently been demonstrated as a principled defense against indiscriminate poisoning attacks [136].

Adversarial training works by augmenting the training data with adversarial examples at each training step. These adversarial examples are constrained by a training budget ϵ_{adv}, similar to the poison budget used in attacks. A larger ϵ_{adv} typically leads to better robustness, though it may come at the cost of reduced accuracy on clean, unperturbed data.

Early approaches, such as the single-step Fast Gradient Sign Method (FGSM) introduced by Goodfellow et al. [49], were found to be ineffective against more sophisticated, multi-step attacks [139, 67]. To address this, Madry et al. [86] proposed a stronger defense based on Projected Gradient Descent (PGD), which uses multi-step optimization to improve robustness. This PGD-based adversarial training has since become a foundation for many advanced methods aimed at improving the balance between clean accuracy and adversarial robustness. For example, some approaches combine training on both clean and adversarial samples [160], explicitly differentiate misclassified examples [147], identifying a bag of training tricks [101], or optimize training efficiency through techniques such as gradient recycling [122].

The key idea behind adversarial training as a defense against data poisoning is that when the adversarial training budget ϵ_{adv} is equal to or greater than the poison budget ϵ_{poi}, the clean (unperturbed) sample lies within the perturbation region, or "ball," centered around the poisoned sample. This enables the adversarial training algorithm to recover the clean samples from their poisoned counterparts, effectively mitigating the impact of the poisoning attack.

Although adversarial training is highly effective, recent studies have shown that methods such as ADVIN [149] and REM [43] attempt to poison models even under adversarial training. However, these techniques only succeed in impractical scenarios where the poisoning budget exceeds twice the adversarial training budget ($\epsilon_{\text{poi}} \geq 2\epsilon_{\text{adv}}$). Consequently, adversarial training remains widely regarded as an effective defense against indiscriminate poisoning attacks, provided the training budget is appropriately set.

2.2.2 Backdoor Attacks

Backdoor attacks differ from traditional poisoning attacks in that they aim to make a model behave maliciously only when a specific trigger is present in the input. When the trigger is absent, the model performs normally, making these attacks particularly stealthy.

The first backdoor attack, BadNets, introduced by Gu et al. [50], demonstrated how a fixed, square-shaped trigger could be used to manipulate model behavior on the MNIST dataset. Liu et al. [80] later advanced this with the Trojan attack, which removed the need for access to the original training data. Instead, it reverse-engineered the target model to synthesize both training data and triggers, optimizing the trigger to activate specific neurons related to a target label.

Both BadNets and Trojan attacks rely on static triggers in terms of pattern and location, which makes them more vulnerable to detection by modern defenses. To address this limitation, Nguyen and Tran [97] proposed an input-aware dynamic backdoor, where a unique trigger is generated for each input, increasing the adaptability of the attack. However, these dynamic triggers, often scattered across the image, are difficult to apply to physical objects in real-world scenarios.

In this dissertation, we focus on backdoor attacks specifically in the context of image classification models. However, backdoor attacks have been extended to other areas like Federated Learning [144], Video Recognition [165], Transfer Learning [153], and Natural Language Processing (NLP) [29].

Various approaches have been developed to increase the stealthiness of backdoors. For example, Saha et al. [111] proposed transforming backdoored images to make them appear benign and harder to detect, while Li et al. [82] introduced the Reflection Backdoor (Refool), which hides triggers by leveraging the physical properties of reflections. Other methods include the Targeted Bit Trojan (TBT) [107], which flips bits in the model's weights rather than retraining the model, and TrojanNet [133], which appends a small Trojan module to the model without requiring a full retrain.

Defenses Against Backdoor Attacks: Defenses against backdoor attacks are generally categorized as model-based or data-based.

Model-based defenses aim to detect backdoors within a model. For instance, Neural Cleanse (NC) [143] tries to generate the smallest possible trigger for each output label and uses anomaly detection to identify potential backdoors. Other defenses like ABS [79] and MNTD [151] analyze the internal behavior of the model's neurons or use meta-classifiers trained on shadow models to detect backdoor presence.

Data-based defenses identify if an input contains a backdoor. Methods like STRIP [45] manipulate the input and measure entropy in the output, with backdoored inputs showing lower entropy. Another approach, Februus [33], uses GradCAM to detect triggers in input regions, removes them, and restores the input using a GAN-based inpainting technique. These defenses highlight the ongoing challenge of developing robust strategies to mitigate increasingly sophisticated backdoor attacks.

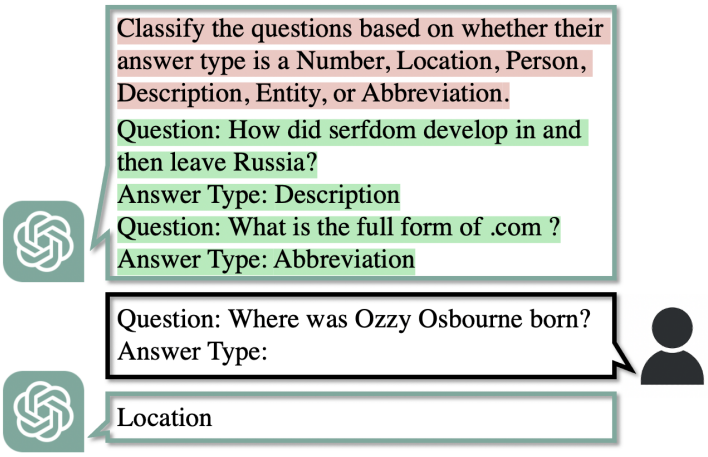


Figure 2.1: An illustrative example of In-Context Learning. The language model is initialized by a prompt combined with instruction (pink) and demonstrations (green).

2.3 In-Context Learning

In-Context Learning (ICL) emerges as a distinctive feature of large language models (LLMs) [20], affording LLMs the capability to acquire proficiency in specific tasks through exposure to limited demonstration examples. In contrast to the conventional notion of "learning," In-Context Learning does not require updating model parameters. Instead, it augments the input with additional content, referred to as a prompt, to facilitate learning through analogy [34]. Specifically, within the prompt, the model is provided with several input-output pairs as examples, instructing the model to respond in a similar format.

To integrate ICL into LLMs, the model undergoes an initialization process. This process involves carefully constructing a task-specific prompt, encompassing an optional task instruction (I) and k demonstration examples ($\{(x_i, y_i)|i \leq k, i \in \mathbb{N}^+\}$). The server concatenates these components to form a complete prompt, denoted as $\texttt{prompt} = \{I, s(x_1, y_1), \ldots, s(x_k, y_k)\}$. Here, the function $s(\cdot, \cdot)$ denotes the transformation of demonstration pairs into natural language following a predefined template, we show examples in Table 2.1. In addition, for a certain task in ICL, the number of demonstrations is not large, typically no more than 8, i.e., $k \leq 8$. This is due to a trade-off between input size and performance. Increasing the number of demonstrations beyond eight results in only marginal performance improvements [166]. We provide an illustrative case in Figure 2.1. Highlighted in pink are the task instructions, which instruct the language model to classify the questions into different categories, and in green are the two demonstrations.

During testing, the language model accepts input samples x in the same format as the `prompt` demonstration, i.e., *"Question: x; Answer Type:{ }"*. Subsequently, the model assigns probabilities $P(y_i|x, \texttt{prompt})$ to all potential answers $y_i \in \mathcal{Y}$, and selects the output token based on sampling strategies, such as greedy decoding, which selects

Table 2.1: Examples of the prompts used for text classification for the ICL setting.

Task	Prompt	Label Names
DBPedia	Classify the documents based on whether they are about a Company, School, Artist, Athlete, Politician, Transportation, Building, Nature, Village, Animal, Plant, Album, Film, or Book. Article: Leopold Bros. is a family-owned and operated distillery located in Denver Colorado. Answer: Company Article: Aerostar S.A. is an aeronautical manufacturing company based in Bacău Romania. Answer:	Company, School, Artist, Athlete, Politician, Transportation, Building, Nature, Village, Animal, Plant, Album, Film, Book
AGNews	Article: Kerry-Kerrey Confusion Trips Up Campaign (AP),"AP - John Kerry, Bob Kerrey. It's easy to get confused." Answer: World Article: IBM Chips May Someday Heal Themselves,New technology applies electrical fuses to help identify and repair faults. Answer:	World, Sports, Business, Technology
TREC	Classify the questions based on whether their answer type is a Number, Location, Person, Description, Entity, or Abbreviation. Question: What is a biosphere? Answer Type: Description Question: When was Ozzy Osbourne born? Answer Type:	Number, Location, Person, Description, Entity, Abbreviation

the token with the highest probability. Mathematically, this process is represented as:

$$\arg\max_{y_i \in \mathcal{Y}} P(y_i | x, \texttt{prompt}).$$

It is important to note that in ICL, the term "model's training data" might be misleading. While the prompt contains demonstration data used to guide the model's responses, there is no actual retraining of the model's weights involved. Instead, the model uses the provided examples to draw analogies and make predictions, simulating a form of learning without altering its underlying parameters. This distinction is crucial for understanding the model's behavior and the potential vulnerabilities associated with ICL.

2.3.1 Privacy Attacks Against In-Context Learning

One prominent privacy concern regarding In-Context Learning (ICL) involves the prompt extraction attack, designed to recover the prompt through API access to the language model. Zhang and Ippolito [163] first formulate a text-based extract attack. Contrary to conventional reconstruction attacks on vision models that optimize the input to reduce testing loss, their work reconstructs the prompt by sending instructions to the model without using backpropagation. This approach extends the applicability of the attack to black-box models, encompassing GPT-3.5 and GPT-4. However, the attack's performance is closely tied to the quality of instructions, and defensive measures, such as output filtering, may impede the attack.

Another category of attacks aims to identify whether specific samples were used to construct the prompt, known as membership inference attacks, which represents one of the most basic forms of privacy attack [113]. The rationale behind existing membership inference attacks is mainly based on the observation that models exhibit varying degrees of confidence in their responses, particularly favoring samples encountered during training.

Current work [25, 27] predominantly employs loss-based attacks, assuming the adversary can access the probability associated with the generated content. This allows the calculation of loss or perplexity for the target sample to determine membership status. Duan et al. [36] compare membership vulnerabilities for fine-tuning and ICL, demonstrating that ICL is more susceptible to membership inference attacks. Wen et al. [T1] further extend this privacy vulnerability comparison to other adaptation methods, including Low-Rank Adaptation (LoRA) and Soft Prompt Tuning (SPT), concluding that ICL exhibits the highest membership vulnerability among them. However, existing membership inference attacks lack the capability to function in a text-only setting, leaving the vulnerability in a text-only scenario unexplored.

3

Membership Inference Attacks Against In-Context Learning

3.1 Introduction

Data play a pivotal role in modern machine learning, but it can also serve as a significant source of privacy leakage. One of the most well-known threats in this domain is the leakage of membership status, where Membership Inference Attacks (MIAs) [95, 69, 125, 114, 71, 130, 75, 113] exploit this vulnerability, allowing adversaries to determine if specific data was used during training. MIAs have been shown to be effective against various types of machine learning models, revealing vulnerabilities across a range of applications. These attacks expose sensitive information, highlighting the need to better understand and mitigate privacy risks in deployed models.

While the privacy vulnerabilities of traditional models have been extensively studied, Large Language Models (LLMs) present new challenges due to their unique capabilities and scale. LLMs are increasingly integrated into numerous real-world applications, but the risks associated with membership inference in these models, particularly in light of novel learning approaches like In-Context Learning (ICL) [20], remain underexplored. ICL allows LLMs to learn tasks efficiently by leveraging contextual prompts rather than updating model parameters [34]. This method has demonstrated significant potential for task-specific adaptation but also introduces new privacy concerns.

Investigating membership inference attacks in the context of LLMs and ICL is particularly important because these models frequently process sensitive user data during task adaptation. For instance, in domains such as healthcare [87, 117, 92], ICL allows models to generate personalized responses based on user-specific prompts, potentially revealing private information. If an adversary can infer that a user's data was included in the prompt used for ICL, they may be able to deduce sensitive details, such as the victim's health status.

To address the evolving privacy risks in the era of LLMs, it is necessary to develop tailored attack methods that accurately reflect these new vulnerabilities. However, a key challenge arises when adversaries interact with LLMs: typically, the only available information is the generated text, without access to posterior probabilities. This limitation makes directly transferring existing attacks [36, 44, 124, 88] to ICL settings difficult.

3.1.1 Contributions

In this chapter, we address these concerns by investigating membership leakage in the context of ICL, and present the first text-only membership inference attack that relies only on the final text generated by the language model. Specifically, we propose four attack methods: GAP, Inquiry, Repeat, and Brainwash. The GAP attack serves as a baseline, considering samples as members if correctly classified and as non-members if not. Among the three advanced attacks, the Inquiry attack directly asks the language model whether it has encountered specific samples. The Repeat attack identifies samples as members if the language model can generate text that closely matches the original input. Finally, in more challenging scenarios where the model produces fixed responses like "positive" or "negative," we introduce the Brainwash attack. This novel method consistently influences the model to provide specific incorrect answers, and membership is inferred based on the sample's ability to conform to this brainwashing process.

We conduct extensive experiments on four popular large language models and three benchmark datasets. Empirical results show that our attacks achieve significant performance across various scenarios. For example, the Brainwash attack achieves over 95% accuracy advantage in inferring membership status against LLaMA on the DBPedia and AGNews datasets. Even when applied to online commercial models such as GPT-3.5, our attack maintains an advantage of over 60% on the TREC dataset.

We further comprehensively investigate factors influencing the attack, including the number of demonstrations and their position in the prompt. Results indicate that the vulnerability of demonstrations emerges from a synergistic interplay between prompt size and the demonstration position. These findings offer insights for designing prompts that are more resilient against privacy attacks. Moreover, we undertake a comprehensive case study investigating the evolving behavior of updated language models over time. Despite ongoing efforts to enhance model safety, our results reveal persistent vulnerabilities in the prompt, even with updated versions.

Recognizing the applicability of our attacks across diverse scenarios, we further design a hybrid attack that combines the strengths of the Brainwash and Repeat attacks. Empirical evidence shows that the hybrid attack outperforms both individual methods in most cases, e.g., 81.2% accuracy advantage compared to 67.8% and 73.0%, respectively. Lastly, we explore three potential defenses at the data, instruction, and output levels. Our results demonstrate that these defenses are effective for specific attacks and datasets. Additionally, we find that combining defenses from all these orthogonal dimensions significantly mitigates privacy leakage and provides stronger privacy guarantees.

We summarize our contribution as follows:

- We present the first text-only membership inference attack against ICL. We design four text-only attacks and empirically demonstrate their effectiveness across four popular language models and three diverse datasets.
- We conduct extensive studies of factors influencing attack performance and reveal that the vulnerability of demonstrations emerges as a synergistic interplay between prompt size and the demonstration position.
- We integrate two powerful attacks to construct a hybrid attack, which significantly enhances our attack's performance and generalizability.
- We explore three potential defenses for ICL against text-only attacks and empirically show their effectiveness in mitigating privacy leakage.

3.2 Problem Statement

Adversary's Objective: The primary objective of the adversary is to determine whether a specific target sample x was included in the construction of a prompt used to customize a language model $\mathcal{M}$. The prompt, denoted as `prompt`, comprises a set of k demonstrations, formatted as $\texttt{prompt} = \{I, s(x_1, y_1), \ldots, s(x_k, y_k)\}$. The adversary's goal is to determine whether the target sample x has been utilized in crafting the `prompt`. That is, the goal is to determine whether x is in the set $\{x_1, \ldots, x_k\}$.

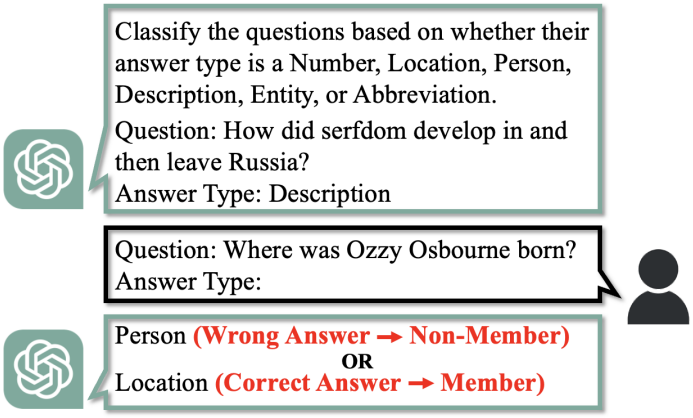


Figure 3.1: The GAP attack involves querying the model with a target sample. The adversary determines the membership status by evaluating the accuracy of the model's prediction: if the prediction is correct, the target sample is classified as a member; otherwise, it is classified as a non-member.

Adversary's Capabilities: In this work, the adversary can access tailored language models that are customized via prompts with *fixed* demonstrations, as indicated in previous research [S4, 134]. For example, Copy.ai [3] suggests employing well-crafted prompts that include examples and stylistic instructions to generate high-quality marketing copy, and these prompts do not change between requests. Furthermore, as GPTs [4] (powered by GPT-3.5/4) and GLMs [5] (powered by ChatGLM4) gain increasing interest, we anticipate more use cases of customized LLMs with fixed demonstrations to perform user-determined tasks, such as sentiment analysis [148] and text summarization [59].

More concretely, we consider the most strict and realistic scenario where the adversary has only black-box access to the target language model $\mathcal{M}$, meaning they can see the text generated but *not* the tokenizer or associated probabilities. Additionally, the adversary has the ground truth answer y for the target sample x. This assumption aligns with most existing membership inference works in computer vision [154, 53, 74] and natural language processing domains [36, T1, 88].

3.3 Attack Methodology

3.3.1 A Baseline Attack: GAP Attack

We start with a baseline attack that extends from the existing attack in the vision domain. Under the assumption that the adversary only has access to the generated texts without additional details, a straightforward approach for membership inference involves exploiting the well-known overfitting phenomenon, where models tend to memorize samples from the training dataset, thereby exhibiting higher accuracy on these than on the testing dataset.

Prior research [104] indicates language models exhibit minimal overfitting due to their massive training dataset and fewer training epochs on individual data points [106].

However, In-Context Learning introduces a potential vulnerability, as it allows language models to recall recently encountered demonstrations, thus behaving as if they have "memorized" them.

Building on this, we categorize samples that are correctly identified as "members" and the rest as "non-members." This approach is an uncomplicated extension of existing work in the vision domain [155] to language model settings. We refer to this basic attack methodology as the GAP attack, which serves as our starting point and baseline for comparison.

Methodology: The attack methodology is structured as follows (see Figure 3.1 for an illustration):

- The adversary selects a target sample x, which is a sentence whose membership status they aim to determine.

- The adversary sends the target sample x to the model and observes the model's response. If the model returns the correct answer, the sentence is classified as a member of the dataset; if not, it is deemed a non-member.

The results, illustrated in Figure 3.2, reveal unsatisfactory performance, particularly for LLMs like GPT-3.5 (0 means random guess), as these models perform very well even when the test samples are not seen in the prompt. This suboptimal performance motivates us to develop more effective attacks tailored specifically for LLMs.

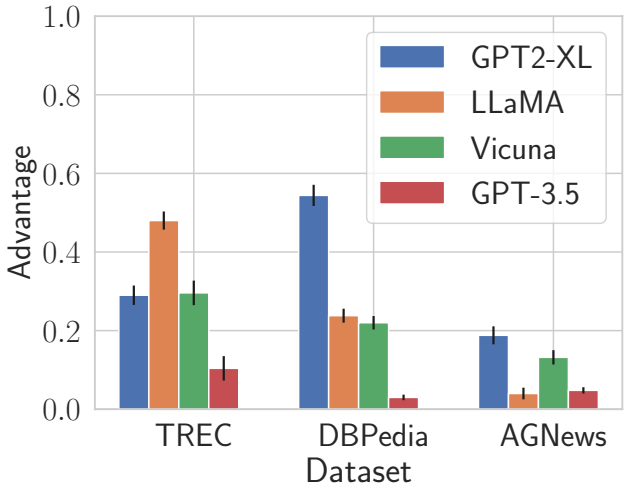


Figure 3.2: Performance of the baseline membership inference attack (GAP), revealing challenges and suboptimal results, particularly evident in larger language models such as GPT-3.5. In this figure, language models are prompted with one example with the instruction presented in Figure 3.1. The performance metric, which indicates the advantage over random guessing, is detailed in Section 3.4.1.

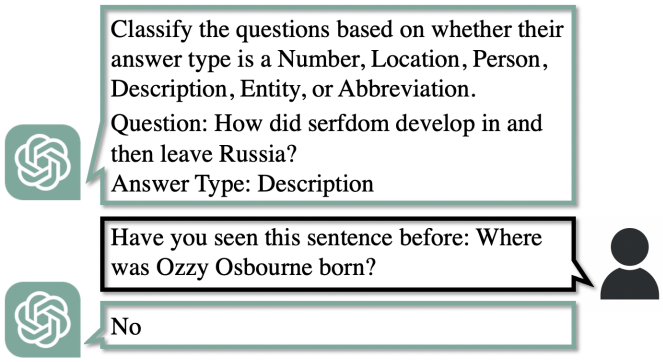


Figure 3.3: The Inquiry attack determines membership status by directly querying the model. In our work, we use the prompt "Have you seen this sentence before."

3.3.2 Inquiry Attack

Intuition: The core concept of this attack method hinges on the language model's ability to remember information from past conversations and deliver context-based responses. When we interact with a language model, it processes the context and produces a response informed by the knowledge it has acquired from previous inputs by the user, particularly from the provided prompt (`prompt`) and its included demonstrations. Consequently, a direct and intuitive approach is to directly question the language model about its previous encounters with specific samples.

Methodology: The attack methodology is structured as follows (refer to Figure 3.3 for an illustration):

- The adversary selects a target sample x, which is a sentence that they aim to determine its membership status.
- The adversary crafts a query to the model with the prompt: *"Have you seen this sentence before: {x}?"*
- The adversary sends the query to the model and observes the model's response. If the model confirms with a "yes", the sentence is classified as a member of the dataset; if not, it is deemed a non-member.

3.3.3 Repeat Attack

The Inquiry attack, while direct and straightforward, may trigger alerts and consequently be denied a response as it overtly queries the language model's prompt. To mitigate this risk, we introduce the Repeat attack, which employs a more subtle approach.

Intuition: This attack leverages the strong memorization capability of language models to generate context-aware responses. Unlike the Inquiry attack, which uses queries with clear intentions (such as *"Have you seen this sentence before?"*), we use the core

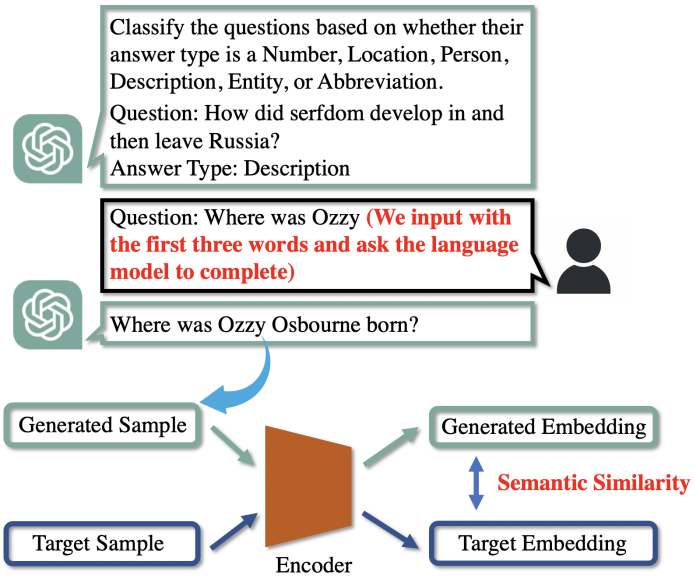


Figure 3.4: The Repeat attack initiates a conversation with a few words and asks the model to complete the sentence. The adversary predicts membership status by assessing the semantic similarity between the generated sample and the target sample.

functionality of a language model, which is to predict the next words. When provided with just the beginning of a target sample, such as the first three words, the model attempts to complete the sentence by adding more words. Our hypothesis is that the model's prior knowledge, enhanced through ICL, will encourage the language model to generate text that mirrors previously encountered content.

Methodology: The attack method consists of the following steps (see Figure 3.4 for an illustration):

- The adversary selects a target sample x, which is a sentence that they aim to determine its membership status.
- The adversary truncates the target sample, retaining only its first few words, which are then inputted to the language model. The generated response $x^{'}$ from the model is obtained.
- The adversary then feeds x and $x^{'}$ to a text encoder E to extract their embeddings and measure the semantic similarity between them by a function Φ.

$$Similarity \quad = \quad \Phi(E(x), E(x^{'})) \tag{3.1}$$

  If the similarity score exceeds a predetermined threshold, the sample is classified as a member of the dataset; otherwise, it is classified as a non-member.

For practical implementation, we use the first three words of the sentence to prompt the language model. The SentenceTransformer network [108], a widely utilized text

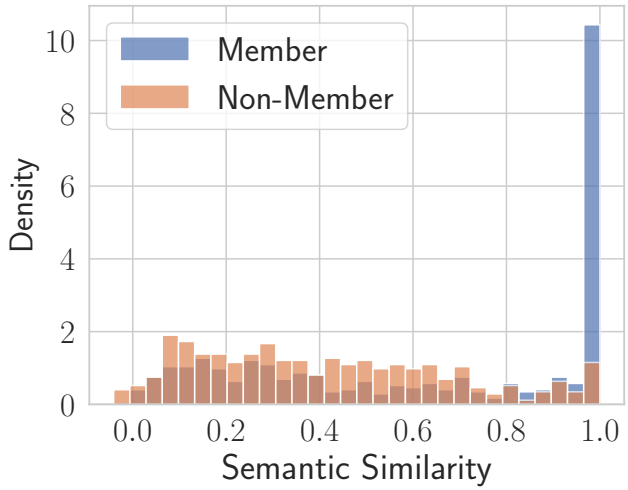


Figure 3.5: Member samples are more likely to exhibit high similarity with the generated sample, while the similarity distribution for non-members is more flattened. These distributions were obtained by querying the GPT-3.5 model using the procedure detailed in Figure 3.4 with the TREC dataset, using one example as a demonstration.

encoder, is employed to calculate semantic similarity using cosine distance. Figure 3.5 shows that member samples generally exhibit higher similarity with the generated text, while non-member samples display a more flattened similarity distribution. This supports our initial hypothesis.

In practice, setting a similarity threshold between 0.8 and 0.9 tends to produce satisfactory accuracy. Further fine-tuning of this threshold through additional sample training can enhance the effectiveness of the attack.

3.3.4 Brainwash Attack

In this scenario, we explore a more common and strict scenario where the language model's output is confined to a predefined list of responses. This does *not* entail modifying the model's basic operational framework, like converting it into a classification model based on Large Language Models. Instead, the language model continues to generate responses in an autoregressive manner *as before*. The key difference here is the introduction of a server-side filter, which evaluates the outputs to ensure they are permissible and non-harmful. This added layer heightens the complexity of launching effective attacks.

For instance, in the sentiment analysis task, the language model is limited to outputting "positive" or "negative" predictions. However, to the user, the interaction with the model remains unchanged.

In such a controlled output environment, launching attacks like the GAP attack remains feasible, though it represents a suboptimal method. More sophisticated attacks, such as those that depend on inducing the model to repeat sentences or to disclose previously seen sentences, are unlikely to succeed due to the server's filtering mechanism.

To overcome this challenge, we present the last text-only membership inference

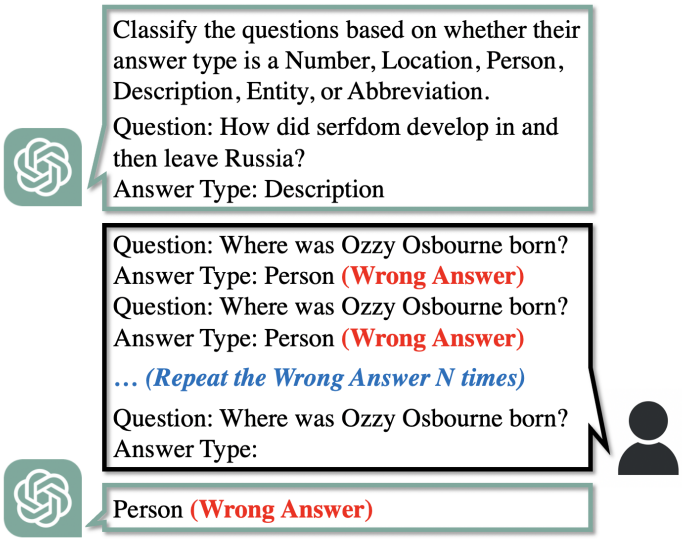


Figure 3.6: The Brainwash attack persistently presents the target sample to the model with a consistent incorrect answer until the model responds inaccurately. The number of iterations required indicates the likelihood of membership.

attack, namely Brainwash Attack. This text-only membership inference attack is designed to operate effectively under the stringent conditions imposed by output filters.

Intuition: Initially, let's consider a simplified case where an adversary has access to the probability or logits associated with outputs. Under these conditions, determining membership is straightforward: a higher probability suggests that the model is more confident in its prediction, likely because the item was seen in the prompt. The main idea of this attack is to approximate the confidence, which is challenging given the limitations of the language model output.

To address this challenge, we approximate confidence by evaluating how firm the model is on previously encountered correct answers when it is "brainwashed" by unreasonable or incorrect queries. Specifically, if an incorrect fact is presented to the model and it hasn't encountered this information before, the model is more susceptible to being misled. Conversely, if the model is familiar with the correct information from its prompt, it is less likely to accept the incorrect fact.

Methodology: The approach involves several key steps, as outlined below and illustrated in Figure 3.6:

- The adversary selects a target sample x, which is a sentence that they aim to determine its membership status. In addition, the adversary knows its correct answer y.
- The adversary crafts a query to the model with the same template, for example: "*Question: x; Answer Type: $\hat{y}$.*" Here, $\hat{y}$ denotes the wrong answer compared to the correct answer y.

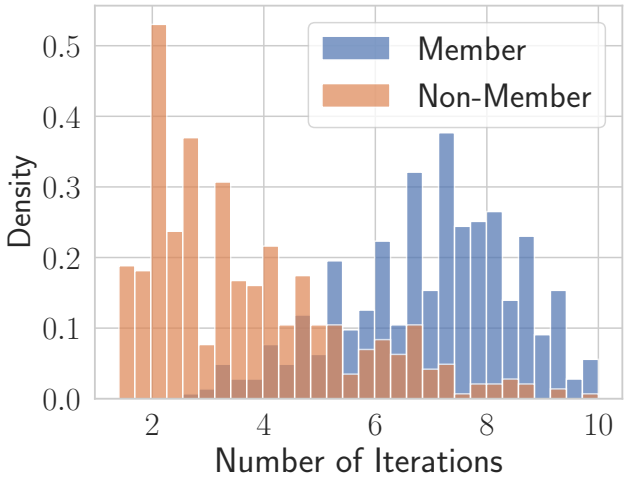


Figure 3.7: Member samples resist incorrect labels, requiring more iterations to change the model's output, while non-members are more easily influenced. These distributions were obtained by querying the GPT-3.5 model using the procedure detailed in Figure 3.6 with the TREC dataset, using one example as a demonstration.

- The adversary repeats querying the model with the above prompt until the model responds with the incorrect answer $\hat{y}$.
- The attacker then counts the number of queries needed for the model to accept the incorrect answer. If this number exceeds a predefined threshold, the sample is classified as a member; otherwise, as a non-member.

In our experiments, we consider multiple choice for the incorrect answer $\hat{y}$. The adversary counts the number of queries for each incorrect answer, and we use the average number of queries as a robust metric for evaluating confidence. Figure 3.7 demonstrates that member samples necessitate significantly more queries to output the incorrect answer, i.e., the language model is much firmer for the correct answer it has seen before. In contrast, non-member samples are more likely to be influenced to output incorrect answers. This observation confirms our intuition. We empirically determined that setting the threshold between 3 to 4 is a reasonable choice for most models and datasets, although the optimal threshold selection necessitates a few additional samples for refinement.

3.4 Experiments

3.4.1 Experimental Setup

Language Models: We evaluate our attacks on four representative language models, including GPT2-XL [105], LLaMA [137], Vicuna [1], and GPT-3.5 [6]. GPT2-XL is a 1.5B parameter version of GPT-2 developed by OpenAI. For LLaMA and Vicuna, we utilize their 7B version and version 1.5 (Vicuna-7b-v1.5), respectively. We access

GPT-3.5 through its official API with the version name gpt-3.5-turbo-0613, which was released on June 13th. We also examine the impact of different versions of this model in Section 3.4.5, providing insights into how variations in model versions affect our attack outcomes. This selection spans fully open-source to fully closed commercial models, demonstrating the applicability of our attacks across a wide spectrum.

Datasets: We assess the impact of our attacks on three benchmark text classification datasets: AGNews [162], a 4-class News Topic Classification dataset; TREC [72], a 6-class Question Classification dataset; and DBPedia [162], a 14-class Ontology Classification dataset. We structure the prompts according to the template designed by Zhao et al. [166], known for its effective performance, with illustrative examples provided in Table 2.1. It is worth noting that our objective is to determine if the sample is included in the prompt. Therefore, there is no requirement to ensure that these datasets are not utilized in training the pretrained models. Given LLMs are trained on extensive datasets, we hypothesize that they do not strongly memorize any specific dataset. Therefore, we anticipate minimal impact on performance. We further explore the influence of memorization on attack performance in Appendix A.1.

Evaluation Settings: The evaluation setting in our work differs from traditional membership inference attack settings, where training datasets typically comprise thousands or tens of thousands of data samples. In these cases, adversaries receive both the training dataset and an equivalently sized testing dataset, tasked with determining the membership status for all samples within this mixed dataset. However, the context shifts for In-Context Learning, where membership pertains to a smaller subset, typically fewer than eight samples. Consequently, evaluating attack performance through a single run may yield unrepresentative results.

Instead of conducting a single experiment, we repeat the experiment 500 times and leverage the average performance as our final result. This experimental design, previously employed in studies targeting In-Context Learning [T1, 36], enhances the robustness and reliability of our assessments.

Each experiment entails the construction of a target prompt based on specified hyperparameters, such as the number of demonstrations. Subsequently, we assess the membership status of two target samples: one sample selected from the prompt is labeled as a member, while another is randomly chosen and designated as a nonmember.

To facilitate this approach, the dataset is initially deduplicated and randomly divided into two parts: the demo part, containing samples utilized for prompt construction, and the test part, housing samples earmarked for testing. For each experiment, we randomly select samples from the demo part based on the prompt design to construct the prompt. Simultaneously, one sample from the test part is randomly chosen and labeled as a nonmember in the experiment. In this chapter, we repeat the experiment 500 times, equating to the labeling of a dataset with 500 members and 500 nonmembers.

Evaluation Metrics: We consider two widely applied metrics to evaluate the attack performance:

- **Advantage** [155, 131]: This metric, denoted as

$$Adv = 2 \times (Acc - 0.5),$$

measures the advantage over random guessing, which offers an average-case evaluation of the attack's effectiveness. Following previous work [155, 131], the metric is multiplied by 2 to scale a 100% accurate attack performance to 1, while random guessing remains at 0. A higher advantage implies better-than-random performance, providing a comprehensive perspective on the attack's overall effectiveness.

- **Log-scale ROC Analysis** [21]: This metric focuses on the true-positive rate at low false-positive rates, effectively capturing the worst-case attack performance.

Given that determining membership status is a binary classification task, with an equal number of positive and negative samples, the advantage metric provides an intuitive measure of performance. An advantage of 0 signifies random guessing, while an advantage of 1 indicates an accurate prediction of all membership statuses. This simplicity aids in the straightforward interpretation of the performance of the attacks.

For all proposed attacks, we employ the advantage metric to gauge average-case performance. In the case of the Repeat and Brainwash attacks, we additionally utilize the log-scale ROC curve to depict their worst-case performance. It's important to note that we cannot present worst-case performance for all attacks since, in the other two, only hard membership predictions are obtainable.

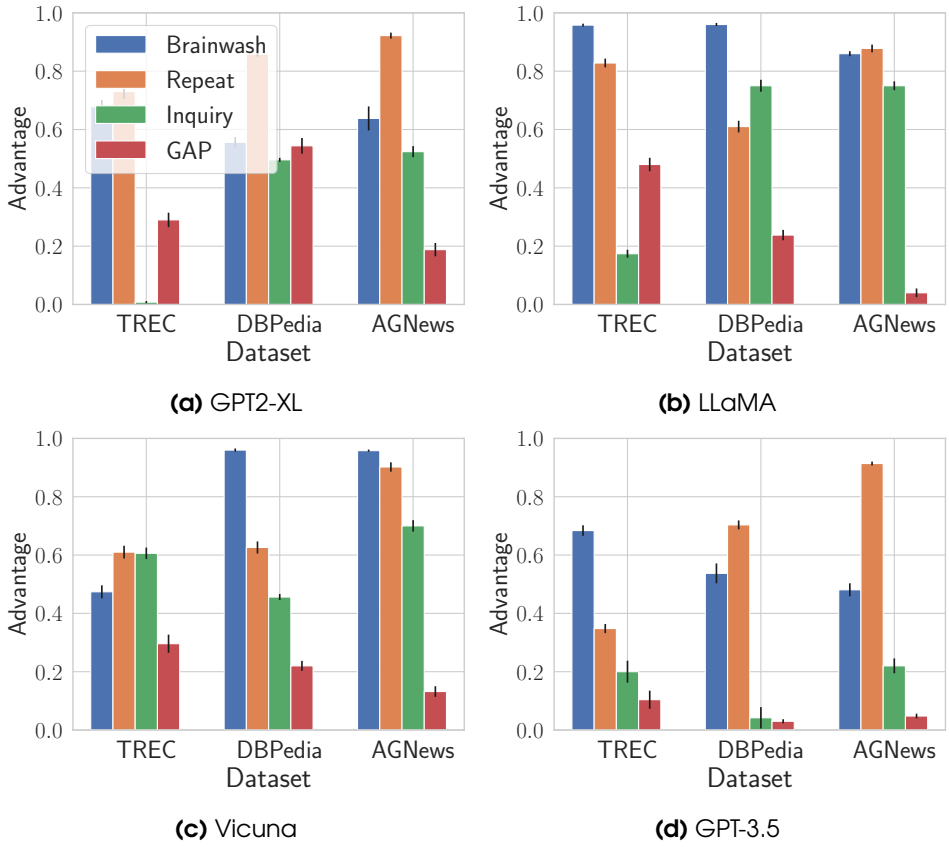
3.4.2 Results

We start by evaluating the performance of our attacks under the basic setting, where the `prompt` contains only one demonstration: $\texttt{prompt} = \{I, s(x_1, y_1)\}$. Here, the adversary aims to determine whether the target sample x is contained in `prompt`, i.e., $x = x_1$ or $x \neq x_1$.

We report the advantage of all four attacks in Figure 3.8. As we can see, Brainwash and Repeat attacks consistently exhibit strong performance across all four language models. This remarkable performance is particularly evident with LLaMA and Vicuna, as demonstrated in Figure 3.8b and Figure 3.8c. For example, Brainwash achieves nearly 100% advantage on 5 out of 6 tasks with LLaMA and Vicuna.

In contrast, the performance of Inquiry and GAP attacks varies significantly depending on the model architecture. For GPT2-XL, GAP attack even achieves 54.4% advantage on DBPedia, although this is the optimal performance it can achieve across all datasets and models. Furthermore, with LLaMA, the Inquiry attack achieves 75.0% advantage on AGNews, showcasing strong performance even with relatively straightforward approaches. However, for GPT-3.5, both attacks prove ineffective, with a performance close to random guessing. Encouragingly, Brainwash and Repeat attacks, while showing a slight performance decrease, maintain effectiveness in inferring membership status on GPT-3.5. This observation highlights the prevalence of membership leakage vulnerabilities in large language models, even when the model solely outputs text information.

Notably, while the Brainwash and Repeat attacks consistently outperform the other two, they show varying advantages under different model architectures. As Figure 3.8 shows, for the GPT family of models (GPT2-XL and GPT-3.5), the Repeat attack



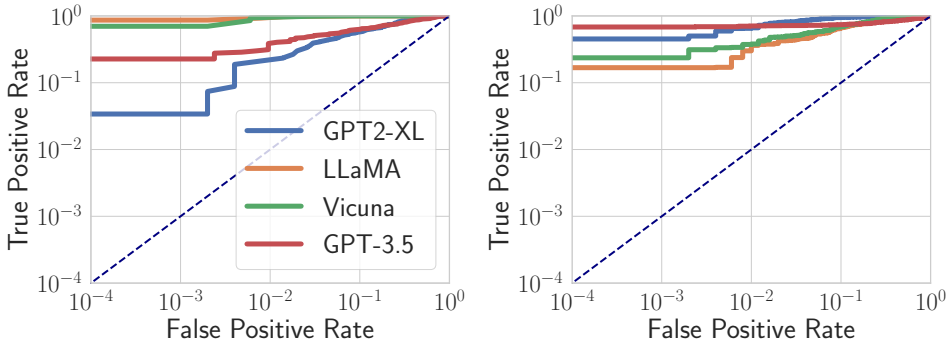
(a) GPT2-XL **(b)** LLaMA

(c) Vicuna **(d)** GPT-3.5

Figure 3.8: Comparison of attack performance across three datasets and four language models, highlighting the consistent efficacy of Brainwash and Repeat attacks, alongside the variable performance of Inquiry and GAP attacks contingent on model architecture.

outperforms Brainwash in most cases, suggesting that the generative behaviors of these models on members and non-members are quite different when queries starting from a few words are given complementary words to finish with. However, for LLaMA and Vicuna, they show greater vulnerability to the Brainwash attack, suggesting that these models are more firm or believe in the knowledge they have gained from `prompt`. The question of why these large language models have different properties and behaviors is beyond the scope of this work, and we leave it to more relevant research areas.

We further report the worst-case performance (log-scale ROC) of the Brainwash and Repeat attacks in Figure 3.9. We can observe that both Brainwash and Repeat attacks exhibit remarkable performance, particularly in the low false positive area. This observation implies the effectiveness of our attacks in determining the membership status of samples that are hard to differentiate.



(a) Brainwash **(b)** Repeat

Figure 3.9: Log-scale Receiver Operating Characteristic (ROC) curve depicting the worst-case performance of Brainwash and Repeat attacks, revealing their efficacy in discerning ambiguous samples.

3.4.3 Influence of Number of Demonstrations

In the previous section, we established the effectiveness of our attacks under the scenario where the `prompt` solely comprises one demonstration. However, in a more practical scenario, language model owners often leverage multiple demonstrations to construct `prompt` to enhance performance. In this section, we explore the performance of our proposed attack under the influence of the number of demonstrations.

It is noteworthy that an increased number of demonstrations does not unilaterally enhance performance, as an extended prompt incurs higher costs in terms of tokens and may be constrained by input limitations of the language model. Therefore, in this section, we first vary the number of demonstrations from 1 to 6 to assess its impact on attack performance. We limited the number of demonstrations to 6 due to the input size restrictions of GPT2-XL, and further evaluated the long context performance solely on the latest GPT-3.5 version.

Moreover, we consider the positional influence of demonstrations within the prompt. Specifically, we posit that for $\texttt{prompt} = \{I, s(x_1, y_1) \dots s(x_k, y_k)\}$ containing k demonstrations, the impact of x_1 and x_k should be different. The reason for this argument is that the demonstrations are entered into the language model sequentially, and the model's memory for the first demonstration should be significantly influenced by subsequent demonstrations compared to the last demonstration. Consequently, in the ensuing experiments probing the influence of the demonstration number (k), we examine the effects on the first and last demonstrations (i.e., x_1 and x_k) separately.

We first report the effect of k on inferring the first demonstration in Figure 3.10. We can see that while the GAP and Inquiry attacks are insensitive to the number of demonstrations, the remaining two attacks show a clear trend between it and attack performance. Specifically, both the Repeat and Brainwash attacks obtain optimal results when there is only one demonstration, and the attack performance gradually decreases as the number of demonstrations increases. We further report the worst-case performance in Figure 3.11. We can find that attacks against one demonstration consistently outperform scenarios with more demonstrations, even in terms of low false

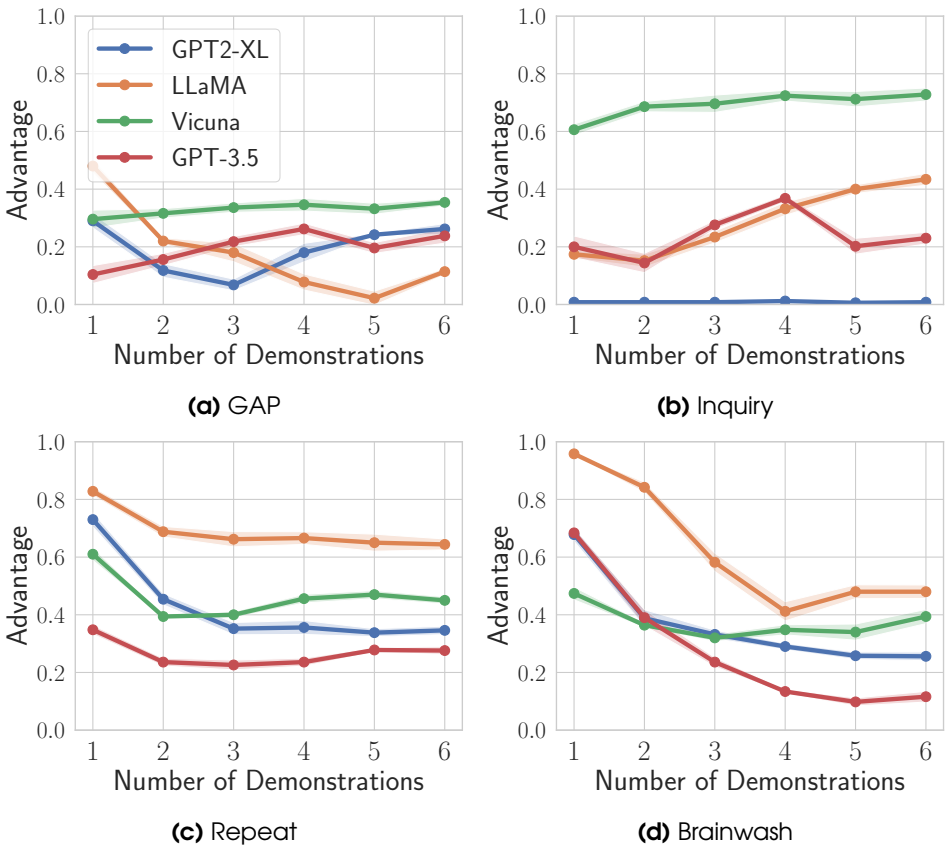
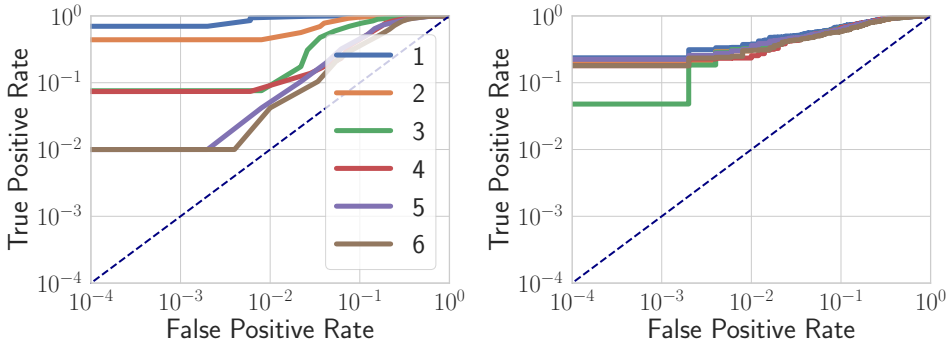


Figure 3.10: Membership Inference Attack (MIA) performance with varying numbers of demonstrations in the prompt, illustrating the influence of demonstration quantity on the efficacy of Repeat and Brainwash attacks. Our experiments are conducted on the TREC dataset.

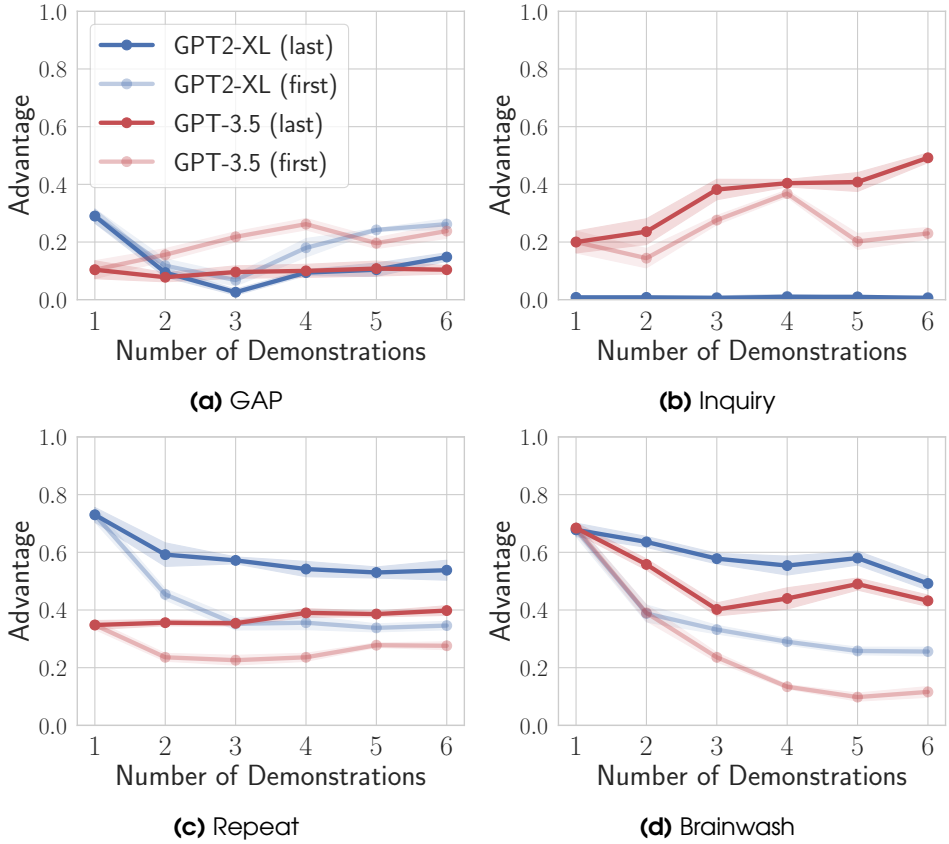
positives.

We then report the effect of k on inferring the last demonstration. For comparison purposes, we present the results for both the first and the last presentation. In addition, in order to observe the trend more clearly, we only present the results in two language models: GPT2-XL and GPT-3.5. First, as shown in Figure 3.12, we can get a similar observation that there is no significant trend for GAP and Inquiry attacks, but a decreasing trend for Repeat and Brainwash attacks. Therefore, we next discuss the findings based on the latter two powerful attacks (Figure 3.12c and Figure 3.12d). These observations suggest that attack performance is negatively correlated with the number of demonstrations, regardless of which demonstrations an adversary aims at to infer membership. These observations also suggest that more demonstrations will not only improve model performance, but also reduce membership leakage. Furthermore, we can find that the attack performance of the first demonstration is more susceptible to the number of demonstrations compared to the last demonstration, as shown by the steeper trend of the first demonstration. This observation validates our previously mentioned argument that the model's memory on the first demonstration is more likely to be



(a) Brainwash **(b)** Repeat

Figure 3.11: Log-scale ROC curve illustrating the worst-case performance of Membership Inference Attacks with varying numbers of demonstrations in the prompt, emphasizing the consistent superiority of one demonstration over multiple demonstrations.



(a) GAP **(b)** Inquiry

(c) Repeat **(d)** Brainwash

Figure 3.12: Comparative analysis of Membership Inference Attack (MIA) performance targeting the first and last demonstration, underscores the impact of the distance between the target sample and the query on model memorization. Our experiments are conducted on the TREC dataset.

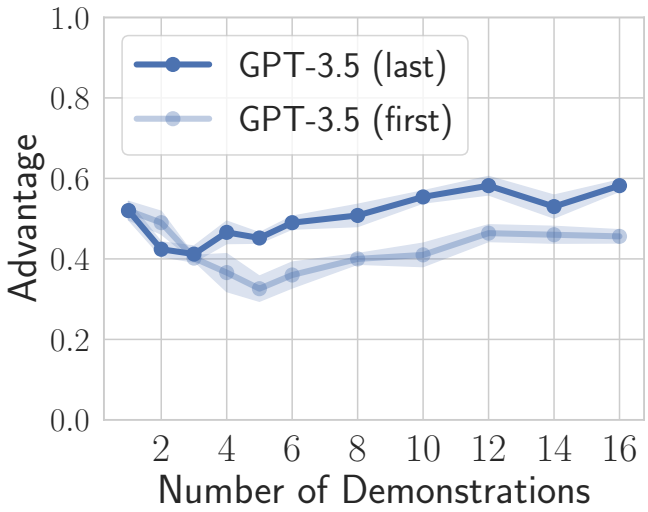


Figure 3.13: Comparative analysis of Brainwash performance targeting the first and last demonstration, with the number of demonstrations ranging from 1 to 16. The experiments utilize the gpt-3.5-turbo-0125 model and are performed on the TREC dataset.

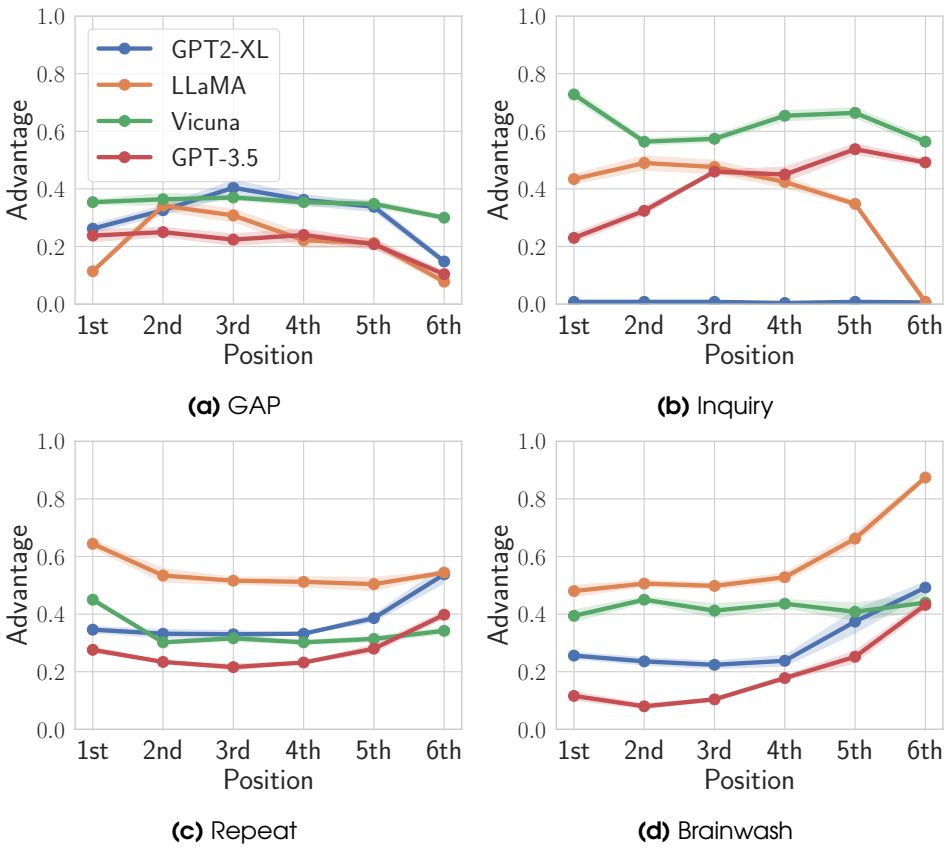
affected than on the last demonstration, as the next few demonstrations may overwrite the knowledge from the first demonstration.

Long Context Performance: As an increasing number of language models support long-context capabilities, it is important to understand how vulnerabilities evolve with extended demonstrations. In this section, we experiment with the latest version of GPT-3.5 (gpt-3.5-turbo-0125), using a range of demonstrations from 1 to 16, and present the results in Figure 3.13.

The results indicate that our attacks maintain high performance even as the number of demonstrations increases. Specifically, when using 16 TREC samples, the Brainwash attack achieves a 0.582 advantage when inferring the last demonstration, and this advantage remains at 0.456 for the first demonstration. We also compared these results on the latest version with our previous findings on gpt-3.5-turbo-0613. It appears that the latest model is more vulnerable to our attack when more demonstrations are included in the prompt. This may suggest that the latest version has enhanced capabilities for handling and remembering long contexts, reflecting the ongoing trend of large language models improving their long-context abilities.

3.4.4 Influence of the Demonstration Position

The above results suggest that the number of demonstrations has a different effect on the first and last demonstrations. We next explore in more depth the effect of demonstration position on attack performance. Concretely, we maintain a consistent number of demonstrations at 6 and use our proposed attacks to infer the membership status of each demonstration, ranging from the $1st$ to the $6th$ position.



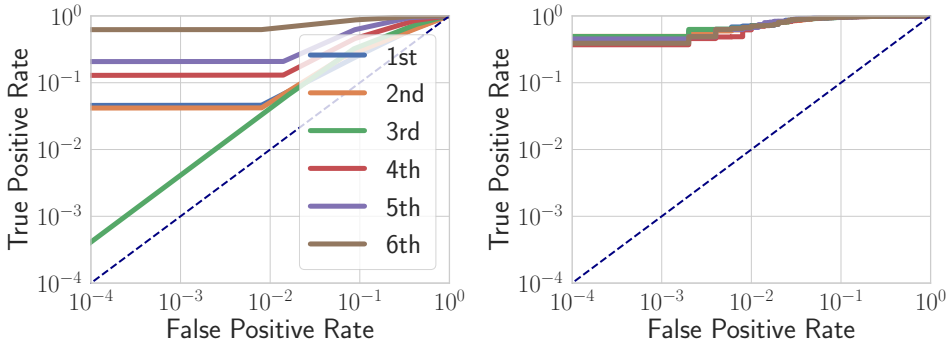
(a) GAP **(b)** Inquiry

(c) Repeat **(d)** Brainwash

Figure 3.14: Exploration of attack performance across different positions (1st to 6th) of demonstrations within a prompt. Results reveal varying vulnerabilities for Repeat and Brainwash attacks. Notably, demonstrations in the middle exhibit a lower vulnerability compared to those positioned at the beginning or end. Our experiments are conducted on the TREC dataset.

We report the relationship between demonstration position and attack performance in Figure 3.14. We can observe that demonstrations at different positions exhibit varying vulnerability, particularly evident for our two powerful attacks, Repeat and Brainwash. Interestingly, the results show that the worst attack performance is not against the first demonstration. Instead, demonstrations positioned in the middle sometimes exhibit poor attack performance. For instance, as illustrated in Figure 3.14c, inferring the membership status of the first demonstration in GPT-3.5 yields 27.6% advantage, while inferring the third demonstration results in only 21.6% advantage. This observation is further validated in the worst-case performance shown in Figure 3.15. Notably, for the Brainwash attack (Figure 3.15a), the last demonstration attains the best performance, as expected, with the first demonstration ranking second. Conversely, the third demonstration (represented by the green line) exhibits the poorest performance.

We emphasize that this finding aligns with previous research [77], which indicates that when large language models encounter long input, they are more prone to focusing on the initial and concluding parts of the context while neglecting the middle section. This



(a) Brainwash **(b)** Repeat

Figure 3.15: Log-scale ROC curve confirms that demonstrations in the middle exhibit reduced vulnerability compared to those located at the beginning or end, even in the worst-case scenario. We use LLaMA as an example here.

conclusion may motivate users to strategically place their valuable demonstrations in the middle of the prompt. However, this strategic placement may potentially compromise utility. Consequently, designing a privacy-preserving positional strategy that balances privacy and utility presents an intriguing problem for further exploration.

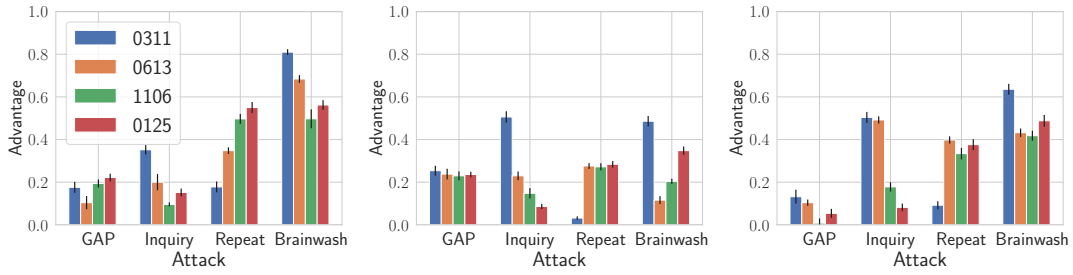
3.4.5 Attack Performance Over Time

In response to the increasing security risks associated with Large Language Models, both researchers and companies are focusing on developing responsible and resilient models capable of withstanding potential threats, including jailbreak attacks. A significant area of exploration involves examining how different versions of Large Language Models address these security challenges, particularly concerning text-only membership inference attacks, as proposed in our study.

In this section, we conduct a case study using one of the most influential LLMs, GPT-3.5, to discern how the performance of attacks varies across different versions. Since the beginning of 2023, the OpenAI has released four API versions: gpt-3.5-turbo-0301, gpt-3.5-turbo-0613, gpt-3.5-turbo-1106, and the latest gpt-3.5-turbo-0125, with the numerical suffix denoting the release date (the first three in 2023 and the latest in 2024). Employing these versions, we execute our proposed four attacks and assess how their performance changes over an eleven-month period.

We first evaluate attack performance with the basic setup of `prompt` containing only one demonstration (sampled from the TREC dataset) and report the results in Figure 3.16a. We can find a clear trend in the different patterns exhibited by different versions of GPT-3.5 under attacks. Notably, the attack performance of the newly released APIs decreases under our Brainwash and Inquiry attacks but increases in the latest version. In contrast, the Repeat attack demonstrates higher performance on recently released APIs. In the case of the GAP attack, which considers the generalization gap in training and testing datasets, the attack differences across the four versions of LLMs are negligible.

Extending our observations to `prompt` with multiple demonstrations, as depicted



(a) One demonstration **(b)** Six demonstrations (first) **(c)** Six demonstrations (last)

Figure 3.16: Evolution of Attack Performance: Comparative analysis of attack performance on four GPT-3.5 API versions (gpt-3.5-turbo-0301, gpt-3.5-turbo-0613, gpt-3.5-turbo-1106, and gpt-3.5-turbo-0124) over a ten-month period. Results demonstrate that the robustness of commercial models like GPT-3.5 doesn't monotonically increase over time. We conduct our experiments on the TREC dataset. Experiments on the DBPedia dataset derive the same conclusion, detailed results can be found in Appendix A.2.

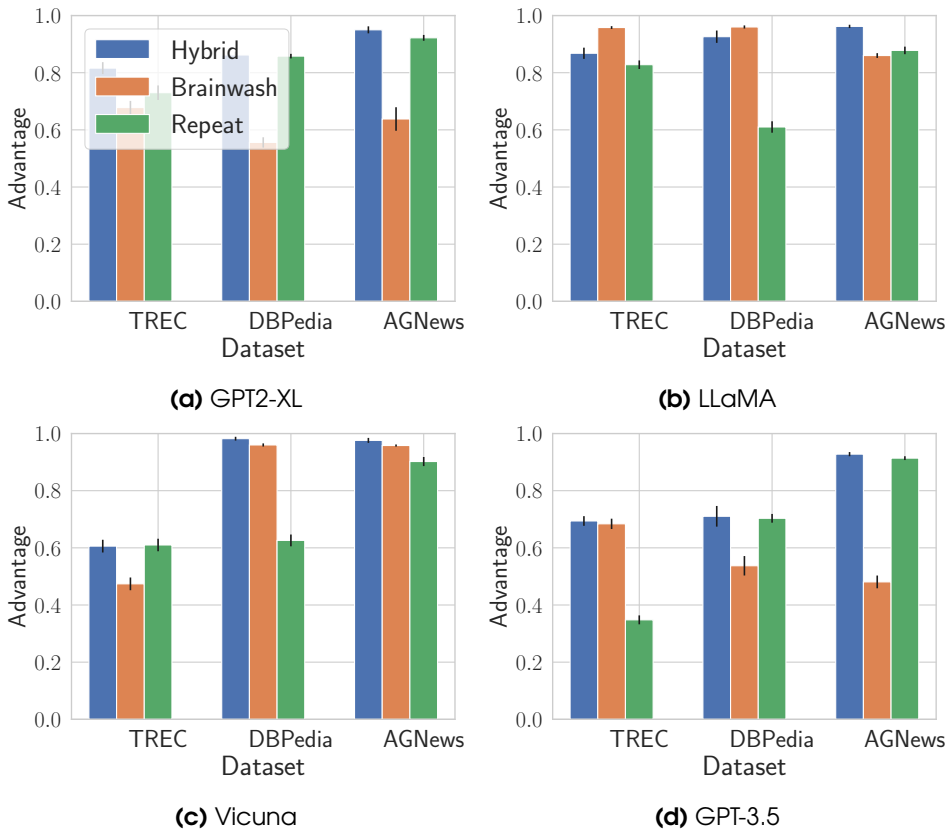
in Figure 3.16b, where `prompt` contains 6 demonstrations, and the target demonstration is positioned at the beginning, reveals an intriguing pattern. The older version (gpt-3.5-turbo-0301) exhibits higher attack performance for Brainwash and Inquiry attacks, while Repeat shows significantly lower performance. This observation remains consistent regardless of the target demonstration's position, as illustrated in Figure 3.16c, where placing the target demonstration at the end yields conclusions analogous to those when positioned at the beginning.

This observation may stem from synergies between attacks, as observed in previous studies [131, S2], different attacks exhibit correlations. For instance, defending against adversarial examples could inevitably increase vulnerability to membership inference attacks. Given the diverse attack surface against LLMs, protecting against all potential threats becomes exceedingly challenging.

In summary, different versions of GPT-3.5 APIs manifest varied behaviors when subjected to attacks, and no single version outperforms all four types of attacks. This understanding helps us navigate the challenge of securing language models in the face of evolving security threats.

3.5 Hybrid Attack

Given the observed varieties in the effectiveness of different attacks across models, datasets, and model versions, it becomes a fascinating challenge from an adversary's point of view to devise a combined approach that consistently performs well in different scenarios. In this section, we present a hybrid attack that combines the strengths of both the Brainwash attack and the Repeat attack to achieve strong performance in all scenarios. We chose these two attacks because they are resilient across different models and datasets.



(a) GPT2-XL **(b)** LLaMA

(c) Vicuna **(d)** GPT-3.5

Figure 3.17: Performance comparison of the hybrid attack against individual Brainwash and Repeat attacks across four language models. The hybrid attack effectively combines the strengths of both, often surpassing individual attack performances. In this figure, language models are prompted with one example.

3.5.1 Methodology

Recall that the Repeat attack involves sending the first few words to the language model and assessing the semantic similarity between generated text and target samples, while the Brainwash attack iteratively induces the language model with incorrect answers and uses the number of iterations to determine membership status. In the hybrid attack, we concatenate both the similarity and iteration number, training a two-layer neural network as the attack model.

The attack model comprises fully connected layers and takes two inputs: the similarity returned by the Repeat attack and the average iteration number from the Brainwash attack. The output is the probability of membership. To train the attack model, we assume the adversary can collect a small shadow dataset sampled from the same distribution as the demonstrations contained in `prompt`. Our experiments indicate that training the attack model with the ensemble of Brainwash and Repeat attacks is not arduous: only 100 samples are sufficient.

Once the attack model is trained, the hybrid attack is deployed as follows: for a target sample, both Repeat and Brainwash attacks are applied to obtain corresponding

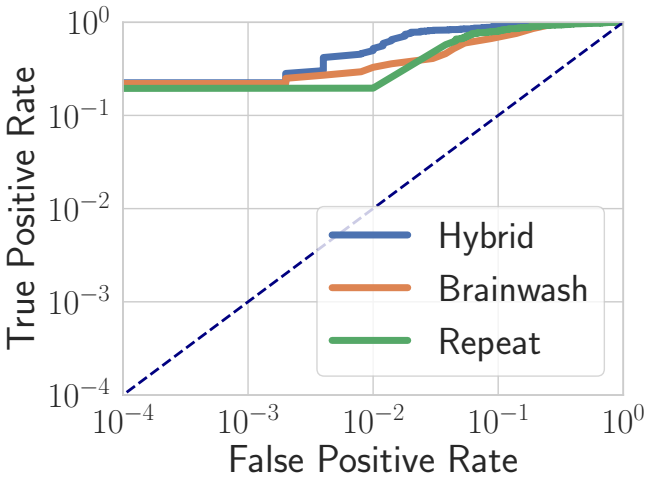


Figure 3.18: Log-scale ROC curve illustrating the performance of the hybrid attack in leveraging the strengths of both Brainwash and Repeat attacks. The hybrid attack strategically combines their advantages to achieve superior performance across the false positive rate spectrum.

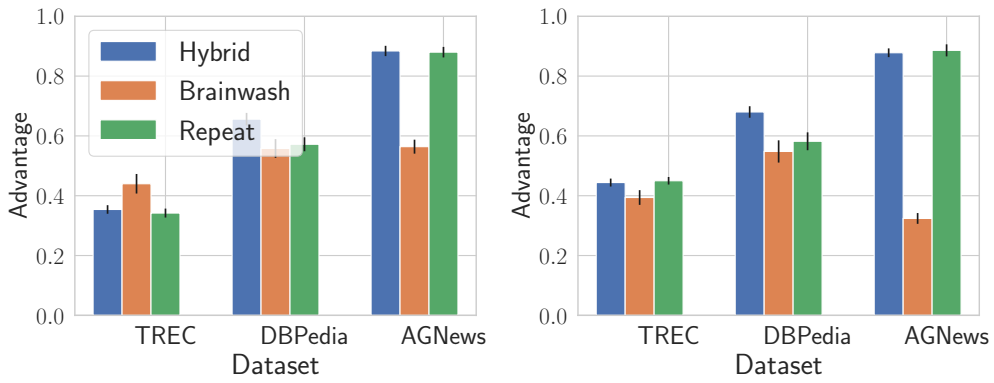
metrics (similarity and iteration number). These values are then fed into the attack model to obtain the final prediction.

3.5.2 Results

We evaluate the effectiveness of our hybrid attack across four language models and present the results in Figure 3.17. Notably, the hybrid attack capitalizes on the strengths of both Brainwash and Repeat attacks. In the majority of cases, the hybrid attack exhibits performance no less than the optimal performance achievable by Brainwash and Repeat attacks individually. Moreover, in certain scenarios, the hybrid attack even outperforms each individual attack. For instance, in Figure 3.17a, where Brainwash and Repeat attacks achieve advantages of 67.8% and 73.0%, respectively, the hybrid attack achieves an advantage of 81.2%, showcasing its ability to derive benefits from both attacks.

To delve into how the hybrid attack leverages the advantages of both attacks, we present the log-scale ROC curve in Figure 3.18. In the high false positive rate area, the hybrid attack capitalizes on the superior performance of the Repeat attack, while in the low false positive area, it aligns with the strategy of the Brainwash attack. This strategic combination results in high overall performance across the entire false positive rate area.

Furthermore, we demonstrate that the hybrid attack maintains its advantage when targeting `prompt` consisting of multiple demonstrations (Figure 3.19a) and attacking demonstrations at different positions within the `prompt` (Figure 3.19b). This evidence suggests that an adversary does not need to select a specific language model/dataset to attack; instead, the hybrid attack proves to be effective in different scenarios and can be used as a general attack against In-Context Learning.



(a) Six Demonstrations (last) **(b)** Six Demonstrations (first)

Figure 3.19: Performance evaluation of the hybrid attack on prompts with multiple demonstrations (Figure 3.19a) and at various positions within a prompt (Figure 3.19b), demonstrating its consistent efficacy across different settings. In this figure, the language model used is Vicuna.

3.6 Potential Defenses

Our proposed attack demonstrates effective performance in inferring the membership status of target samples, revealing significant privacy threats. However, as of our current knowledge, there is a lack of a comprehensive defense framework to safeguard In-Context Learning (ICL) from membership inference attacks.

In this section, we explore three potential defenses aiming to minimize the information leakage from the language model regarding its prompt.

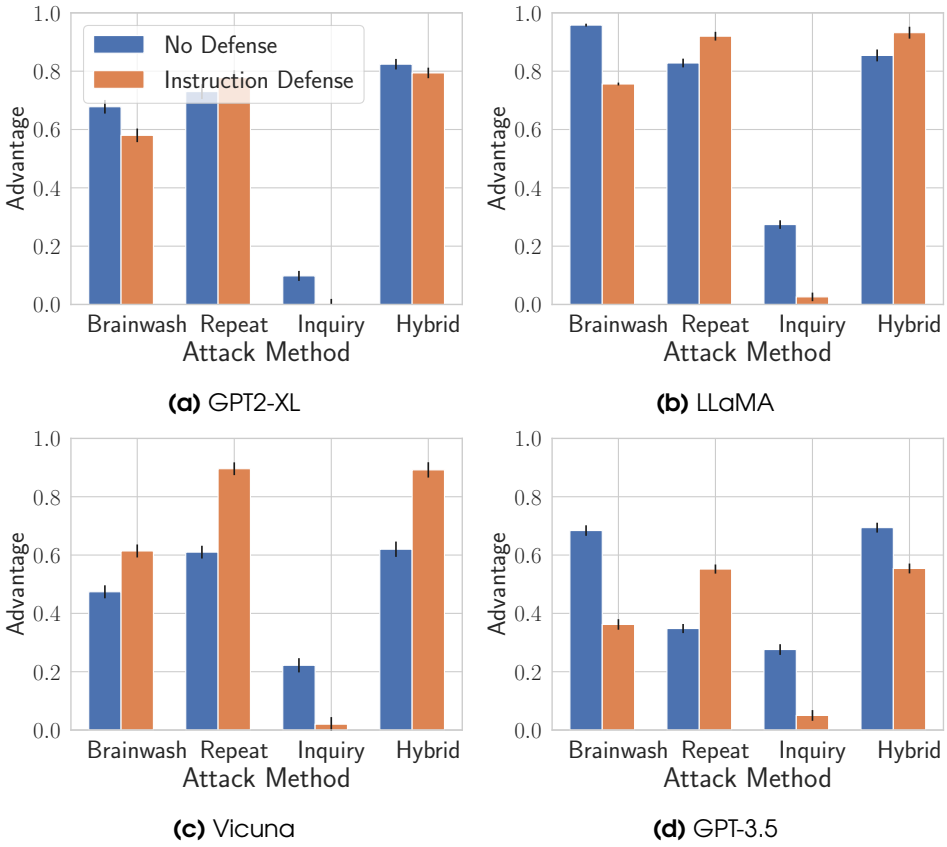
3.6.1 Instruction-Based Defense

Drawing inspiration from the strong control that instructions can exert over language models (e.g., prompt injection attack), our initial approach involves using instructions to compel the language model to refrain from leaking any information related to its prompt.

Specifically, we task GPT-3.5 with designing a prompt intended to prevent the model from disclosure of prompt-related details. This strategy leverages previous findings [168] which suggest that language models can be particularly adept at crafting prompts, potentially surpassing human efforts in this domain. The generated instruction by GPT-3.5 is as follows:"Respond to the following queries without directly mentioning or alluding to any specific examples, demonstrations, or instances that might have been used in the prompt."

We place this defense instruction at the end of the prompt and conduct the three most powerful attacks against demonstrations protected by this instruction.

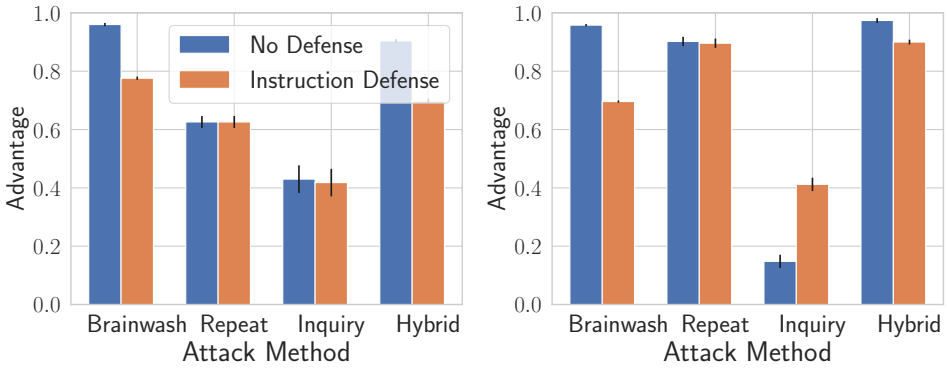
While the efficacy of the defense instruction is evident in Figure 3.20, showcasing a reduction in the performance of the Inquiry attack for the TREC dataset, this effectiveness does not uniformly extend to other datasets, as depicted in Figure 3.21. Notably, the addition of the defense instruction tends to marginally decrease the



(a) GPT2-XL **(b)** LLaMA

(c) Vicuna **(d)** GPT-3.5

Figure 3.20: The defense instruction successfully reduces the effectiveness of the Inquiry attack for the TREC dataset; nevertheless, this mitigating effect does not extend to the Repeat attack or other datasets.



(a) DBPedia **(b)** AGNews

Figure 3.21: Evaluation of the defense instruction's impact on the performance of different attacks across diverse datasets, highlighting varying levels of efficacy and nuances in defense outcomes.

performance of the Brainwash attack in most instances, though the variance is not statistically significant, we posit this reduction primarily to the increased distance between the query and the demonstration. Intriguingly, when evaluating the defense effect against Repeat attacks, in certain scenarios, the attack performance is observed to be even higher compared to scenarios without defense. We posit that integrating a well-designed defense instruction tailored to a specific attack and dataset may constitute a pragmatic approach to mitigate privacy leakage. However, the creation of a universally applicable defense instruction necessitates further scrutiny and exploration.

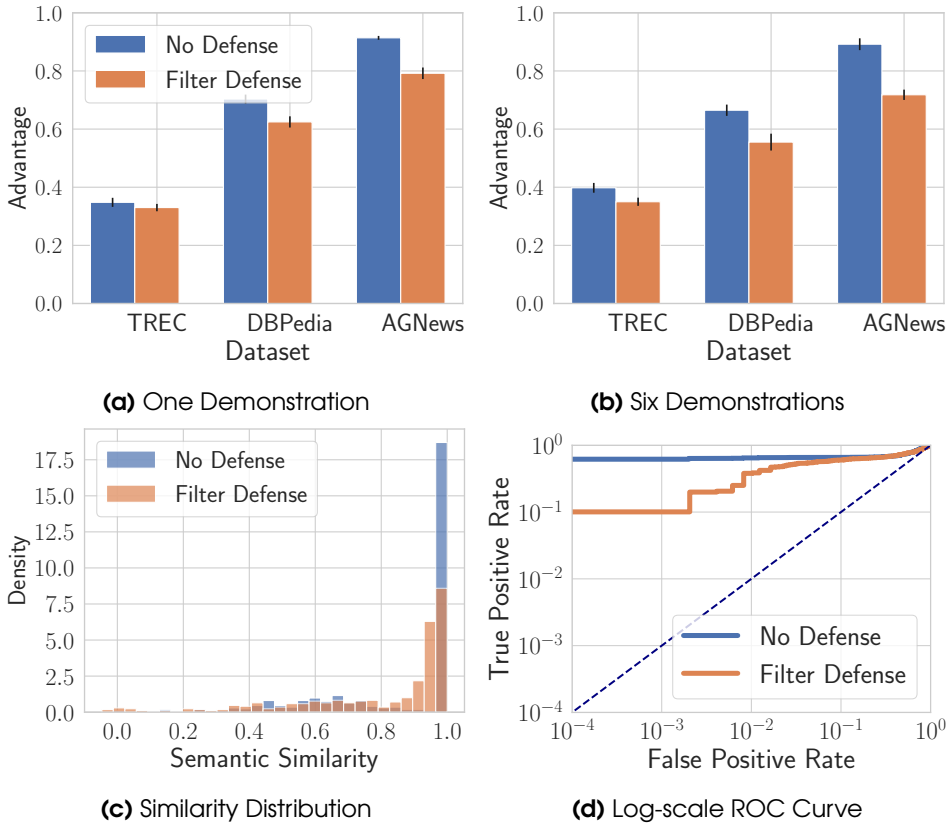
3.6.2 Filter-Based Defense

While acknowledging the resilience of the Repeat attack against simple defense instructions, we leverage insights from this attack methodology to devise an ad-hoc defense that actively modifies the language model's output. Specifically, since the Repeat attack determines membership status based on semantic similarity between the generated response and the target sample, we implement an output filter that modifies the response while preserving its utility. To achieve this, when the language model outputs content, we send that content to GPT-3.5 and request a sentence rewrite. This filter-based defense consistently reduces the performance of the Repeat attack across all datasets, as illustrated in both Figure 3.22a and Figure 3.22b.

It is worth noting that our approach involves actively modifying the response, distinguishing it from common filter defenses [163]. Blacklist-based filter defenses, which return an empty string if the output significantly overlaps with the prompt, may initially seem effective against prompt leakage but are susceptible to circumvention. For instance, an attacker could instruct the language model to output text encoded in a Caesar cipher or introduce additional spaces between characters to evade detection. On the other hand, whitelist-based filters, which only permit output from a predefined list, pose a greater challenge to bypass but may impact the utility, and we have considered and proposed the Brainwash attack tailored for this challenging scenario.

To understand how our filter-based defense diminishes the attack performance of Repeat, we analyze the semantic similarity distribution for member samples before and after the defense. As shown in Figure 3.22c, before the defense, a considerable number of responses generated by the language model exhibit high similarity to the target sample (similarity close to 1). Our observations, along with previous work [163], indicate that the language model can sometimes reproduce the exact content from its prompt. However, after applying the output filter, the semantic similarity spreads more smoothly rather than concentrating near 1. The log-scale ROC curve (Figure 3.22d) emphasizes that the performance degradation primarily occurs in the low false-positive rate area, indicating the defense's effectiveness against worst-case scenarios.

It's important to note that while rewriting the output is effective against the Repeat attack, it is not applicable to Brainwash and Inquiry attacks. These attacks only require the model to produce a predefined output or a binary answer, and manipulating the output can either distort the intended meaning (thus impacting utility) or render the defense ineffective. Consequently, the implementation of output filtering stands as a supplementary defense tailored to specific attack types, rather than offering a



(a) One Demonstration **(b)** Six Demonstrations

(c) Similarity Distribution **(d)** Log-scale ROC Curve

Figure 3.22: Evaluation of a filter-based defense strategy against our attacks. Figure 3.22a and Figure 3.22b demonstrate the defense's impact on Repeat attack performance. Figure 3.22c depicts the change in semantic similarity distribution for member samples before and after the defense. Figure 3.22d presents the log-scale ROC curve, highlighting the defense's effectiveness against worst-case performance scenarios.

comprehensive, universal solution.

3.6.3 DP-based Defense

Differential privacy (DP) has been established as a key defense mechanism against membership inference attacks. In this section, we assess the effectiveness of an existing DP-based defense strategy [134], which generates synthetic demonstrations from the private dataset with DP guarantees. Specifically, we set num-private-train to 1 to exclude unseen data. The resulting DP demonstrations are of low quality, such as "Users Admin ApollosemblingIC negatives direct@GetMapping." Because of the high dissimilarity between the generated DP and original demonstrations, the effectiveness of the Repeat attack is reduced to almost random guessing.

However, DP is less effective against the Brainwash attack. Despite the generated samples being dissimilar to the original samples, the Brainwash attack still achieves a 0.228 advantage, as depicted in the yellow bar in Figure 3.23.

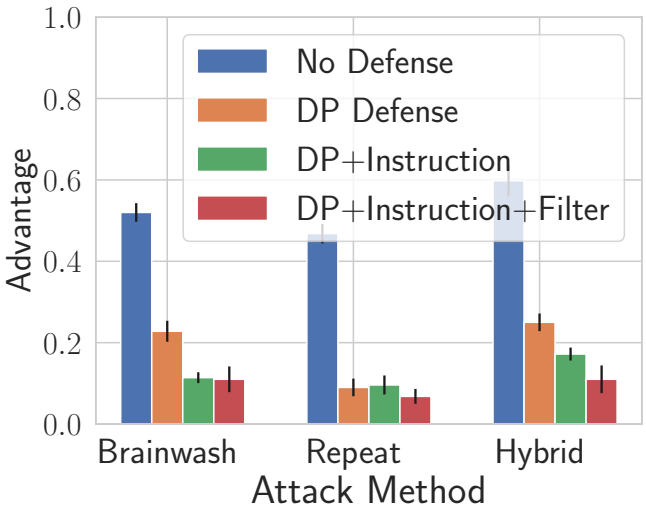


Figure 3.23: Effectiveness of combining defenses from multiple dimensions. Results demonstrate that integrating defenses targeting different dimensions—such as data, instruction, and postprocessing—substantially enhances overall defense effectiveness and significantly reduces privacy leakage.

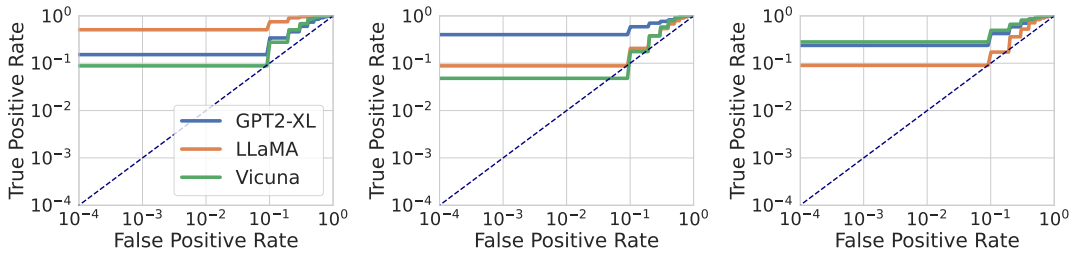
As DP-based, instruction-based, and filter-based defenses target orthogonal components of the language model—namely, the data, the instruction, and the output—a logical approach is to combine these three defenses for enhanced protection. In Figure 3.23, we illustrate the synergy of combining different defense strategies. On the basis of the DP-based defense, the instruction-based defense further reduces the performance of the Brainwash attack while having a negligible influence on the Repeat attack performance. Adding the filter-based defense on top of these two defenses further reduces the effectiveness of the Repeat attack. Upon combining all three defenses, the overall performance of the hybrid attack is reduced from 0.59 to 0.11. These results suggest that effective defense strategies should not focus on a single component but rather leverage a combination of defenses targeting orthogonal components.

3.7 Discussion and Limitations

In this section, we discuss how our findings can advance our understanding of vulnerabilities in ICL, as well as the current limitations.

For Attack: We first evaluated the worst-case performance using a posterior-based method across three datasets for three open-source models: GPT2-XL, LLaMA, and Vicuna. The results, presented in Figure 3.24, illustrate that utilizing posterior probabilities to determine membership status yields robust performance. This supports our hypothesis that samples within the prompt exhibit a significantly lower loss, indicating higher model confidence in demonstrations.

However, despite the effectiveness of the posterior-based attack, the inability to observe the loss directly from text output highlights the difficulty of conducting mem-



(a) TREC **(b)** DBPedia **(c)** AGNews

Figure 3.24: Worst-case evaluation with access to the posterior, these results provide an estimation of how traditional posterior-based attack could perform for the in-context learning paradigm.

bership inference attacks using only text data. The main challenge lies in converting these unobservable aspects into observable ones.

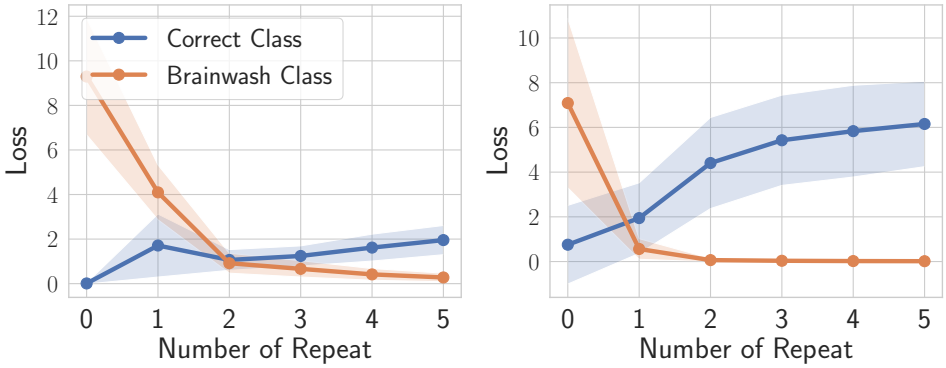
To tackle this, we conducted an in-depth analysis to understand how the Brainwash attack converts unobservable signals, such as loss, into observable ones. Specifically, we visualized the *loss dynamics* for both the correct class and the maliciously introduced "brainwash" class, as illustrated in Figure 3.25.

Both two figures indicate that when we brainwash the language model using incorrect labels, the loss on the correct class gradually increases, while the loss on the incorrect brainwash class decreases. When the loss of the correct class surpasses that of the brainwash class, the language model predicts the wrong label. This phenomenon explains how ICL interacts with brainwash samples.

Furthermore, we can see that for member samples and non-member samples, the speed loss increase is different. Specifically, as shown in Figure 3.25b, for non-member samples that are not included in the prompt, the loss on the correct class increases rapidly when facing brainwash samples. This indicates that the language model's confidence in target samples can easily be influenced, and the language model quickly accepts the brainwash class, evidenced by the sharp decrease of the loss on the brainwash class. However, for member samples, the loss on the correct class increases more slowly, and the brainwash class requires more repetitions to reduce the loss, resulting in a delayed intersection point.

This analysis highlights how our brainwash attack can transform unobservable signals, such as loss, into observable signals, such as the number of repetitions. This novel conversion makes the attack feasible in realistic and challenging scenarios where the target model outputs only text. Besides, this approach is analogous to label-only membership inference attacks in the vision domain, where adversarial perturbations indicate model confidence in target samples. Our method can be regarded as an adversarial perturbation technique for LLMs with text-only output, even with restricted output.

We acknowledge that currently, it's challenging to provide a theoretical understanding of the vulnerability, as the community is still exploring how ICL interacts with the model. For example, NeedleInAHayStack [7] demonstrates that when facing a long context, the models are more likely to memorize information encoded at the beginning/end, but



(a) Member **(b)** Non-Member

Figure 3.25: Visualization of loss dynamics in the Brainwash attack: The figures demonstrate that the loss for the correct class increases when the language model is brainwashed by wrong labels, translating unobservable loss changes into observable signals, such as the number of repetitions.

Anthropic [8] later pointed out that these findings may be influenced by the prompt.

Our attack provides empirical evidence on how models memorize demonstrations in an adversarial setting and shares some common vulnerabilities, like the first and last demonstrations are more more vulnerable to privacy leakage. Moreover, our analysis suggests further attacks could follow the same intuition by proposing different methods that better approximate the confidence of the LLMs to the target samples, thereby enhancing attack performance.

For Defense: Our work provides insights into defense strategies, indicating that effective defenses should combine multiple components, such as data, instruction, and output, rather than focusing on a single aspect. As shown in Figure 3.23, different defense mechanisms each have their own advantages, and combining defenses from orthogonal dimensions can result in better synergy.

Additionally, our study indicates that relying solely on developer interventions, such as Reinforcement Learning from Human Feedback (RLHF), may introduce side effects. This is evidenced by the varying attack performance across different model versions, as shown in Figure 3.16. A version that is effective at defending against one type of attack may inadvertently increase the model's vulnerability to another type of attack.

While it is challenging to draw a formalized conclusion at this stage, we believe our findings offer valuable insights that could benefit the community in better understanding these vulnerabilities.

3.8 Ethical and Privacy Considerations

Membership inference attacks against ICL in LLMs pose significant privacy risks, as they can reveal whether specific data points were used in the model's demonstrations. This threat can lead to the exposure of sensitive personal information, undermining

user trust and potentially causing harm. Ethically, it is crucial to ensure transparency, fairness, and accountability in the use of LLMs.

On the other hand, these attacks can serve as auditing tools to verify whether unauthorized data has been used to construct prompts. To mitigate these risks, we suggest a series of strategies, including differential privacy, instruction-based, and filter-based defenses. We strongly recommend that developers integrate these approaches when releasing query APIs. Employing these mitigation strategies can protect user data and maintain ethical standards in AI development and deployment.

3.9 Conclusion

In this chapter, we propose the first text-only membership inference attack against ICL. Our attack exhibits effectiveness across various scenarios, including instances where the language model is constrained to generating responses from a predefined list. We conduct extensive experiments across diverse datasets and language models and empirically demonstrate the effectiveness of our attacks. We delve into an exploration of factors influencing attack efficacy, revealing that the vulnerability of demonstrations results from the intricate interplay between prompt size and the demonstration position. A thorough investigation into the information leakage in language models over time uncovers persistent vulnerabilities even with updated versions, heightening our concerns. To mitigate membership leakage, we explore three potential defenses, finding that their combination significantly reduces privacy leakage.

Our study not only enhances our understanding of the intricacies of ICL vulnerabilities but also contributes practical considerations for prompt design and defense mechanisms. Despite the successful implementation of defenses in specific scenarios, the quest for a comprehensive and generalized defense strategy persists. As the field continues to advance, our findings provide a foundation for researchers and practitioners alike, guiding efforts toward a more secure and privacy-conscious integration of ICL into the transformative landscape of LLMs.

4
Robust Poisoning Attack

4.1 Introduction

While data is often considered the target of attacks, such as membership inference where an adversary determines whether a specific data point was part of the training set, it can also serve as a potent tool for launching attacks. One notable example is the data poisoning attack, where an adversary manipulates the training data to cause the resulting model to behave maliciously or perform incorrectly, as intended by the attacker.

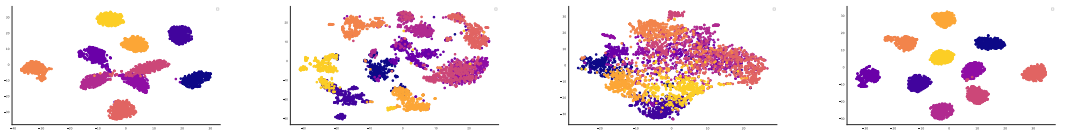
Although various defenses have been proposed to counter these attacks, creating the impression that the problem is largely solved, in the next two chapters, we will demonstrate that these defenses provide users with a false sense of security. Specifically, for two types of poisoning attacks—indiscriminate poisoning attacks and backdoor attacks—we show how it is possible to design powerful attack strategies that can bypass existing defenses. In this chapter, we discuss indiscriminate data poisoning first.

Indiscriminate data poisoning aims to degrade the overall prediction performance of a machine learning model at test time by manipulating its training data. This form of poisoning has become increasingly relevant as web scraping is commonly used to gather large datasets for training advanced models [20, 35, 22]. While small perturbations to training samples have been shown to effectively poison deep learning models, there is a widely accepted belief that adversarial training can protect against such attacks if the perturbation budget for adversarial training, denoted as ϵ_{adv}, is greater than or equal to the poison budget, ϵ_{poi}, i.e., $\epsilon_{\text{adv}} \geq \epsilon_{\text{poi}}$ [39, 40, 54, 136, 149, 43, 135]. In particular, Tao et al. [136] have proved that in this setting, adversarial training can serve as a principled defense against existing poisoning methods.

4.1.1 Contributions

In this chapter, we challenge this widely held consensus by rethinking data poisoning from a fundamentally new perspective. Specifically, we introduce a new poisoning approach that entangles the features of training samples from different classes. In this way, the entangled samples would hardly contribute to model training no matter whether adversarial training is applied or not, causing substantial performance degradation of models. Different from our attack strategy, existing methods commonly inject perturbations as shortcuts, as pointed out by Yu et al. [157]. This ensures that the model wrongly learns the shortcuts rather than the clean features, leading to low test accuracy (on clean samples) [118, 38, 157].

Figure 4.1 illustrates the working mechanism of our new poisoning approach, with a comparison to a reverse operation that instead aims to eliminate entangled features. Our new approach is also inspired by the conventional, noisy label-based poisoning approach [19, 18, 94], where *entangled labels* are introduced by directly flipping labels (e.g., assigning a "dog" (or "cat") label to both the "dog" and "cat" images) under a strong assumption that the labeling process of the target model can be manipulated. However, due to the imperceptibility constraint in the common clean-label setting, we instead propose to introduce *entangled features* represented in the latent space. Our work mainly makes three contributions:



(a) Test Acc: 84.88% **(b)** Test Acc: 72.99% ↓ **(c)** Test Acc: 71.57% ↓ **(d)** Test Acc: 88.72% ↑

Figure 4.1: The t-SNE feature visualizations for (a) clean CIFAR-10 vs. poisoned CIFAR-10 achieved by our (b) EntF-pull and (c) EntF-push, which aim to induce entangled features. As a comparison, (d) uses a reverse objective of EntF-push and instead increases the model accuracy. Different from our EntF, existing methods lead to well-separable features (see Appendix A.3). All t-SNE visualizations in this chapter are obtained from the same clean reference model with $\epsilon_{\text{ref}} = 4$.

- We demonstrate that, contrary to the consensus view, indiscriminate data poisoning *can* actually decrease the clean test accuracy of adversarially-trained (AT) models to a substantial extent. Specifically, we propose EntF, a new poisoning approach that is based on inducing entangled features in the latent space of a pre-trained reference model.
- We conduct extensive experiments to demonstrate the effectiveness of EntF against AT in the reasonable setting with $\epsilon_{\text{adv}} = \epsilon_{\text{poi}}$ and also its generalizability to a variety of more challenging settings, such as AT with higher budgets, partial poisoning, unseen model architectures, and stronger (ensemble or adaptive) defenses.
- We further highlight the distinct roles of non-robust vs. robust features in compromising standard vs. AT models and also propose hybrid attacks that are effective even when the defender is free to adjust their AT budget ϵ_{adv}.

4.2 Problem statement

We formulate the problem in the context of image classification DNNs. There are two parties involved, the *poisoner* and the *victim*. The poisoner has full access to the clean training dataset $\mathcal{D}_c = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and is able to add perturbations $\boldsymbol{\delta}^{\text{poi}}$ to each sample and release the poisoned version $\mathcal{D}_p = \{(\boldsymbol{x}'_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}'_i = \boldsymbol{x}_i + \boldsymbol{\delta}_i^{\text{poi}}$. Once the poisoned dataset is generated and released, the poisoner cannot further modify the dataset. Moreover, the poisoner has no control over the target model's training process and the labeling function of the victim. The victim only has access to the poisoned dataset and aims to train a well-generalized model using this dataset. As the victim is aware that the obtained dataset may be poisoned, they decide to deploy adversarial training to secure their model. **The goal of the poisoner is to decrease the clean test accuracy of the adversarially-trained model by poisoning its training dataset.**

When perturbing the clean dataset, the poisoner wants to ensure that the perturbation is imperceptible and can escape any detection from the victim. To this end, the poisoner constrains the generated perturbations $\boldsymbol{\delta}^{\text{poi}}$ by a certain *poison budget* ϵ_{poi},

i.e., $\|\boldsymbol{\delta}^{\text{poi}}\|_\infty \leq \epsilon_{\text{poi}}$. Take the widely-adopted adversarial training framework [86] as an example, the victim trains a target model F on the poisoned dataset $\mathcal{D}_p$ by a certain *adversarial training budget* ϵ_{adv} following the objective:

$$\arg\min_{\theta} \ \mathbb{E}_{(\boldsymbol{x}',y)\sim\mathcal{D}_p}\Big[\max_{\boldsymbol{\delta}^{\text{adv}}} \mathcal{L}(F(\boldsymbol{x}'+\boldsymbol{\delta}^{\text{adv}}),y)\Big] \ \text{ s.t. } \ \|\boldsymbol{\delta}^{\text{adv}}\|_\infty \leq \epsilon_{\text{adv}}, \tag{4.1}$$

where $\boldsymbol{x}'$ denotes the poisoned input, $\boldsymbol{\delta}^{\text{adv}}$ denotes the adversarial perturbations, θ denotes the model parameters, and $\mathcal{L}$ is the classification loss (e.g., the commonly used cross-entropy loss).

In this chapter, we focus on the reasonable setting with $\epsilon_{\text{poi}} \leq \epsilon_{\text{adv}}$. In contrast, the two concurrent studies, ADVIN [149] and REM [43], focus on much easier settings with $\epsilon_{\text{poi}} \geq 2\epsilon_{\text{adv}}$, in which it is not surprising that AT would fail because the clean samples are already out of the ϵ_{adv}-ball of the poisoned samples [136].

4.3 Methodology

In this section, we introduce EntF, our new poisoning approach to compromising adversarial training. The key intuition of EntF is to cause samples from different classes to share entangled features and then become useless for model training, including adversarial training. Specifically, we propose two different variants of EntF, namely EntF-push and EntF-pull. For EntF-push, all training samples in each of the original classes y are pushed away from the corresponding class centroid $\boldsymbol{\mu}_y$ in the latent feature space (i.e., the output of the penultimate layer F^*_{L-1}) of a reference model F^*, which has totally L layers. The objective function can be formulated as:

$$\mathcal{L}_{\text{push}} = \max_{\boldsymbol{\delta}^{\text{poi}}} \|F^*_{L-1}(\boldsymbol{x}+\boldsymbol{\delta}^{\text{poi}}) - \boldsymbol{\mu}_y\|_2 \ \text{ s.t. } \ \|\boldsymbol{\delta}^{\text{poi}}\|_\infty \leq \epsilon_{\text{poi}}. \tag{4.2}$$

For EntF-pull, each training sample is pulled towards the centroid of its nearest class y':

$$\mathcal{L}_{\text{pull}} = \min_{\boldsymbol{\delta}^{\text{poi}}} \|F^*_{L-1}(\boldsymbol{x}+\boldsymbol{\delta}^{\text{poi}}) - \boldsymbol{\mu}_{y'}\|_2 \ \text{ s.t. } \ \|\boldsymbol{\delta}^{\text{poi}}\|_\infty \leq \epsilon_{\text{poi}}. \tag{4.3}$$

The above class centroid is computed as the average features of all clean samples $\mathcal{X}$ in that class:

$$\boldsymbol{\mu} = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}} F^*_{L-1}(\boldsymbol{x}). \tag{4.4}$$

We find this simple, average-based method works well in our case, and we leave the exploration of other, metric learning methods [65] to future work.

In order to learn a similar representation space to that of an AT target model, the reference model F^* is also adversarially trained with a certain perturbation budget ϵ_{ref}. We discuss the impact of ϵ_{ref} on the poisoning performance in Section 4.5. Following the common practice, we adopt the Projected Gradient Descent (PGD) [86] to solve the above poison optimization.

Why adversarial training can be compromised. Tao et al. [136] have proved that adversarial training can serve as a principled defense against data poisoning based on the following theorem.

Theorem 1. *Given a classifier $f : \mathcal{X} \to \mathcal{Y}$, for any data distribution $\mathcal{D}$ and any perturbed distribution $\hat{\mathcal{D}}$ such that $\hat{\mathcal{D}} \in \mathcal{B}_{W_\infty}(\mathcal{D}, \epsilon)$, we have*

$$\mathcal{R}_{\text{nat}}(f, \mathcal{D}) \leq \max_{\mathcal{D}' \in \mathcal{B}_{W_\infty}(\hat{\mathcal{D}}, \epsilon)} \mathcal{R}_{\text{nat}}(f, \mathcal{D}') = \mathcal{R}_{\text{adv}}(f, \hat{\mathcal{D}}).$$

where $\mathcal{R}_{\text{nat}}$ denotes natural risk and $\mathcal{R}_{\text{adv}}$ denotes adversarial risk. Theorem 1 guarantees that adversarial training on the poisoned data distribution $\hat{\mathcal{D}}$ optimizes an upper bound of natural risk on the original data distribution $\mathcal{D}$ if $\hat{\mathcal{D}}$ is within the ∞-Wasserstein ball of $\mathcal{D}$ [136]. That is to say, achieving a low natural risk on $\mathcal{D}$ (i.e., high clean test accuracy) requires a low adversarial risk on $\hat{\mathcal{D}}$. This guarantee is based on an *implicit* assumption that adversarial training is capable of minimizing the adversarial risk on the poisoned data distribution $\hat{\mathcal{D}}$, which holds for existing attacks. However, for our EntF, the poisoned data that share entangled features become not useful even for adversarial training, and as a result, the assumption required for the proof is broken. Our experimental results in Figure 4.2 validate this claim.

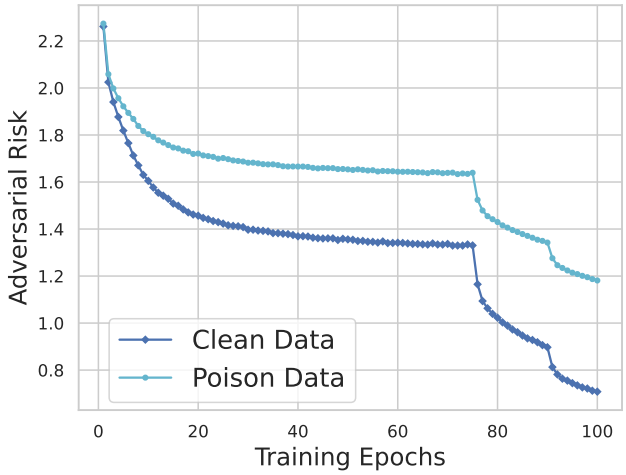


Figure 4.2: Adversarial risk curves for clean and poison data.

Important note on the cross-entropy loss. The key novelty of our EntF over existing methods lies in not only the attack strategy (entangled features vs. shortcuts) but also the specific loss (feature-level vs. output-level). Existing methods on poisoning standard models have commonly adopted the cross-entropy (CE) loss and concluded that the targeted optimization, either with an incorrect [40, 136, 149] or original [54] class as the target, is generally stronger than the untargeted CE. This conclusion somewhat leads to the fact that the two concurrent studies on poisoning AT models (i.e., ADVIN [149] and REM [43]) have completely ignored the untargeted CE as their baseline.

However, we find that the above conclusion does not hold for poisoning AT models. Specifically, we notice that the untargeted CE can also lead to entangled features to some extent and as a result yield a substantial accuracy drop (12.02%), while its targeted counterpart (i.e, ADVIN [149] shown in our Table 4.2) completely fails. Note that the untargeted CE still performs worse than our EntF-push, especially in the more complex

tasks, i.e., CIFAR-100 and TinyImageNet as shown in Table 4.1. This indicates that using the CE loss is not an ultimate solution in practice.

Table 4.1: EntF vs. the CE loss using the same robust reference model.

Poison Method \ Dataset	CIFAR-10	CIFAR-100	TinyImageNet
AdvPoison-untar	72.86	50.45	45.47
EntF-push	**71.57**	**47.29**	**41.32**

4.4 Results

In this section, we conduct extensive experiments to validate that adversarial training (AT) can be compromised by our new poisoning attack. In particular, we consider challenging scenarios with high AT budgets, partial poisoning, unseen model architectures, or strong (ensemble or adaptive) defenses.

4.4.1 Experimental settings

We use three image classification benchmark datasets: CIFAR-10 [9], CIFAR-100 [9], and TinyImageNet [10]. These datasets have been commonly used in the poisoning literature. We adopt the perturbation budget $\epsilon_{\text{ref}} = 4/255$ for adversarially training the reference model and find that other values also work well (see results in Section 4.5). PGD-300 with a step size of 0.4/255 and differentiable data augmentation [40] is used for poison optimization. If not explicitly mentioned, we focus on the reasonable setting with $\epsilon_{\text{poi}} = \epsilon_{\text{adv}} = 8/255$ and adopt ResNet-18 for both the reference and target models. All reference and target models are trained for 100 epochs using SGD optimizer with an initial learning rate of 0.1 that is decayed by a factor of 0.1 at the 75-th and 90-th training epochs. The optimizer is set with momentum 0.9 and weight decay 5×10^{-4}. The inner maximization (i.e., generation of adversarial examples) of the adversarial training is solved by 10-step PGD with a step size of 2/255. All experiments are performed on an NVIDIA DGX-A100 server.

4.4.2 EntF compared to existing attacks under $\epsilon_{\text{adv}} = \epsilon_{\text{poi}}$

We first evaluate the performance of different state-of-the-art poisoning methods against adversarial training in the basic setting with $\epsilon_{\text{poi}} = \epsilon_{\text{adv}}$ on CIFAR-10. As can be seen from Table 4.2, all existing methods can hardly decrease the model accuracy. Specifically, although ADVIN [149] and REM [43] have claimed effectiveness in the unreasonable settings with $\epsilon_{\text{poi}} \geq 2\epsilon_{\text{adv}}$, they fail in our reasonable setting. In some cases, the poisons may even slightly increase the model accuracy, which is also noticed in the concurrent work [135].

In contrast to existing methods, both our EntF-push and EntF-pull can substantially decrease the model accuracy. Note that decreasing the model accuracy to 71.57% is dramatic because it equals the performance achieved by directly discarding 83% of

Table 4.2: Comparison of different poisoning methods against adversarial training. ADVIN, Hypocritical$^+$, and REM also adopt an adversarially-trained reference model, as our methods.

Poison Method	Clean Test Accuracy (%, ↓)
None (Clean)	84.88
Hypocritical [136]	84.96
Unlearnable [54]	84.91
Class-wise Random Noise	84.06
AdvPoison [40]	83.11
ADVIN [149]	86.76
Hypocritical$^+$ [135]	86.56
REM [43]	84.21
EntF-pull (ours)	72.99
EntF-push (ours)	**71.57**

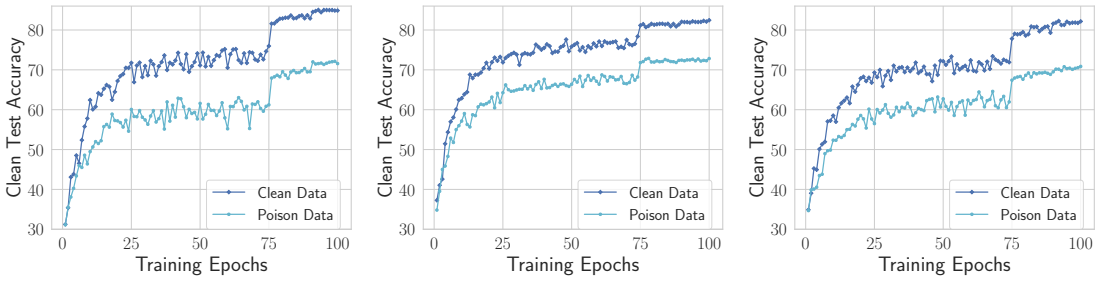
the original training data (see more relevant discussions in Section 4.4.5). In addition, EntF-push and EntF-pull achieve similar results but obviously, EntF-pull is less efficient because it needs to calculate and then rank the distance between each sample and class centroid. Moreover, the class selection strategy in EntF-pull may have an impact on the final performance (see more analysis in Appendix A.4). For these reasons, if not specifically mentioned, we choose to use EntF-push in the following experiments.

Table 4.3: Evaluating EntF on different datasets.

Poison Method \ Dataset	CIFAR-10	CIFAR-100	TinyImageNet
None (Clean)	84.88	59.50	51.95
EntF (ours)	71.57	47.29	41.32

4.4.3 EntF for larger datasets and other AT frameworks under $\epsilon_{\text{adv}} = \epsilon_{\text{poi}}$

The results shown in Table 4.3 further validate the general effectiveness of our EntF on larger datasets, where the model accuracy consistently drops by more than 10%. We also evaluate EntF against different widely-used AT frameworks. Figure 4.3 shows the learning curves of the poisoned AT target model that is trained with Madry [86], TRADES [160], or MART [147]. As can be seen, our EntF largely decreases the clean test accuracy in all cases. We can also observe that all three frameworks exhibit a relatively steady learning process, i.e., the model accuracy monotonically increases over epochs, and finally reaches an accuracy that is still lower than that of the model trained on clean data. This pattern is different from that in poisoning standard models, where the model accuracy is found to increase at a few early epochs, and then start to decrease dramatically to the final low accuracy [54, 83, 118]. This fundamental difference indicates that early stopping cannot be used as an effective defense against poisoning for AT models.



(a) Madry (86) **(b)** TRADES (160) **(c)** MART (147)

Figure 4.3: Evaluating EntF against three different well-known adversarial training frameworks.

4.4.4 EntF under higher AT budgets

We further test EntF under higher adversarial training budgets and also consider different poison budgets. As can be seen from Table 4.4, even with an overwhelming budget, adversarial training can still be largely degraded by our EntF. For example, when the AT budget is $\epsilon_{\text{adv}} = 16/255$, which is $2\times$ larger than the poison budget $\epsilon_{\text{poi}} = 8/255$, our EntF still yields a substantial accuracy drop of 10.05%. In addition, under the same setting with $\epsilon_{\text{adv}} = \epsilon_{\text{poi}}$, a larger poison budget leads to a better poison performance. Specifically, for $\epsilon_{\text{adv}} = \epsilon_{\text{poi}} = 4/255$, model accuracy drops by 5.94% (90.31% $\rightarrow$ 84.37%), while for a larger poison budget, 8/255 or 16/255, the accuracy drops by 13.31% (84.88% $\rightarrow$ 71.57%) or 20.75% (73.78% $\rightarrow$ 53.03%). We can also observe that under a specific poison budget, the model accuracy and AT budget do not have a clear correlation. This is reasonable because although enlarging the AT budget can increase clean accuracy due to higher robustness to poisons, it also inevitably leads to an accuracy drop compared to standard training. We leave detailed explorations for future work.

Table 4.4: Evaluating EntF under different ϵ_{poi} vs. ϵ_{adv}.

Poison Budget \ AdvTrain Budget	$\epsilon_{\text{adv}} = 4/255$	$\epsilon_{\text{adv}} = 8/255$	$\epsilon_{\text{adv}} = 16/255$
None (Clean)	90.31	84.88	73.78
$\epsilon_{\text{poi}} = 4/255$	84.37	79.25	69.35
$\epsilon_{\text{poi}} = 8/255$	75.39	71.57	63.73
$\epsilon_{\text{poi}} = 16/255$	50.27	60.29	53.03

4.4.5 EntF under partial poisoning

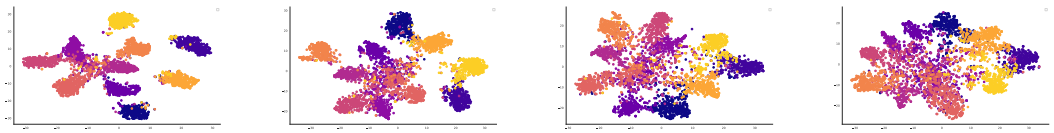
We also examine EntF in a more challenging scenario where only partial training data are allowed to be poisoned. We consider two different poisoning settings where different data are used for calculating the class centroids. Specifically, the first setting is based on the whole original (clean) dataset but the second, worse-case one is based on only the

partial clean data that are allowed to be poisoned. As can be seen from Table 4.5, our EntF is still effective in this challenging scenario. Recall that other attacks can hardly decrease the model accuracy even when the whole dataset is poisoned. In particular, EntF decreases the model accuracy from 84.88% to 81.41% by only poisoning 0.2 of the training data, and this result is also lower than the baseline that is achieved by directly discarding these poisoned data (83.66%).

Table 4.5: Effects of adjusting the poison proportion. "None (Clean)" shows the baseline results where the rest clean data is used without poisoning.

Poison Method \ Poison Proportion	0.2	0.4	0.6	0.8
None (Clean)	83.66	81.82	79.16	73.23
Clean+EntF	81.41	78.67	75.84	73.56
Clean+EntF (worse case)	81.84	78.83	75.92	74.01

We can also observe that the two different poisoning settings yield very similar results. This indicates that the calculation of class centroids in our EntF is not sensitive to the amount of data, and as a result, its efficiency can be potentially improved by using fewer data for the centroid calculation. The fact that poisoning more data yields better performance can also be explained by Figure 4.4 where a larger poison proportion leads to a larger number of entangled features.



Poison Proportion: 0.2 Poison Proportion: 0.4 Poison Proportion: 0.6 Poison Proportion: 0.8

Figure 4.4: The t-SNE visualizations for different poison proportions.

4.4.6 Transferability of EntF to unseen model architectures

The latent feature space that is used for generating poisons is specific to a certain reference model. For this reason, one natural question to ask is whether the poisons generated on one model architecture are still effective when the target model adopts a different architecture. Table 4.6 demonstrates that the poisoning effects of our EntF optimized against a ResNet-18 reference model can transfer to other target model architectures. Specifically, for the four different (unseen) architectures, the generated poisons are able to degrade the model accuracy to almost the same extent, indicating the strong generalizability of our EntF.

4.4.7 EntF against other defenses

Ensemble defenses with data augmentations. Applying additional data augmentations before standard training has been shown to be able to mitigate the effects of

Table 4.6: Transferability of EntF poisons from ResNet-18 to other model architectures.

Poison Method \ Target	ResNet-18	ResNet-34	VGG-19	DenseNet-121	MobileNetV2
None (Clean)	84.88	86.58	75.99	87.22	80.11
EntF	71.57	73.05	64.66	74.35	67.21

perturbation-based poisons [54, 40, 83, 136]. Here we study if data augmentations can complement adversarial training when facing our EntF. Following previous work, we test a diverse set of data augmentations, including random noise, Mixup [161], Cutmix [158], and Cutout [32]. We also test gray-scale pre-filtering [83], which shows strong performance in mitigating unlearnable examples [54]. Table 4.7 shows that all the data augmentation methods fail to help AT to mitigate our EntF.

Table 4.7: Evaluating EntF against defenses that apply both data augmentations and AT.

Defense	Clean Test Accuracy (%)
None (Clean)	84.88
Adversarial Training	71.57
+Random Noise	71.88
+JPEG Compression	70.40
+Mixup [161]	71.84
+Cutout [32]	69.81
+Cutmix [158]	68.85
+Grayscale [83]	68.67

Adaptive defenses by filtering out entangled samples. In addition to existing defense methods, we also consider stronger, adaptive defenses that the victim may design based on a certain level of knowledge about our EntF. It is worth noting that, in realistic scenarios, the victim can only leverage a poisoned model, and so it also has no access to a clean AT reference model, which is available to the poisoner, including our EntF. This is reasonable because if it is indeed feasible for the victim to get a clean AT (reference) model, there is no need for dealing with the poisoned data in the first place, and the clean AT model can already be used as an effective target model.

When the victim knows that entangled features have been introduced by EntF, they would filter out the "overlapped samples", which are located close in the feature space but from different classes. We test by removing different proportions of such "overlapped samples" and find that the best setting can only recover the accuracy from 71.57% to 72.43%. We go a step further by considering a stronger victim who even knows the specific algorithm of EntF-push (i.e., Equation 4.2). In this case, the victim would recover the data by pulling the poisoned samples back toward their original class centroids. We find that this defense is stronger than the above but still can only recover the accuracy to 75.32% (about 10% lower than the clean AT accuracy). We also consider a practical scenario in which the victim can leverage an ST or AT CIFAR-100 model as a general-purpose model. In this scenario, the adaptive defense only recovers the model

accuracy to 72.97%.

Adaptive defenses through feature perturbation-based AT. We further test a new defense that uses Equation 4.2 for generating the adversarial examples in AT instead of the common, cross-entropy loss. When trained on clean data, this new AT variant yields an accuracy of 86.84%, similar to that achieved by the conventional AT. However, when trained on our poisoned data, the model accuracy still substantially drops to 72.99%, indicating that it is not a satisfying defense. This defense is even worse than the above one with a pre-trained AT model (72.99% vs. 75.32%). This might be because the class centroids calculated when the model is not well trained (in the early AT training stage) cannot provide meaningful guidance compared to those based on the pre-trained model. As a sanity check, we also try another AT variant that uses a reverse loss of Equation 4.2 and find that as expected, it causes the model accuracy to drop a lot (to 47.26%).

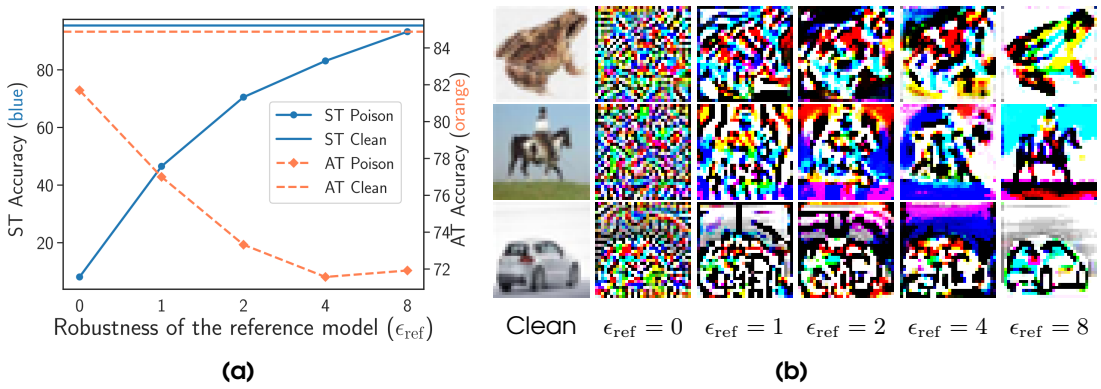


Figure 4.5: Impact of the robustness of the reference model on poisoning standard (ST, $\epsilon_{\text{ref}} = 0$) vs. adversarially-trained (AT) target model. (a) Clean test accuracy for both ST and AT; (b) Perturbation visualizations for different ϵ_{ref}. More visualizations can be found in Appendix A.5.

4.5 Poisoning AT vs. ST Models

All our experiments have so far been focused on poisoning AT models. However, it is also valuable to study the problem in the context of standard training and figure out the difference. To this end, we analyze the poisons against a standard (ST) model vs. an AT model. Specifically, we adjust the perturbation budget ϵ_{ref} for adversarially training the reference model and analyze the poisons in terms of both the poisoning strength and the visual characteristics of perturbations.

As can be seen from Figure 4.5a, for poisoning the AT model, the poisoning strength is gradually improved as the reference model becomes more robust (i.e., ϵ_{ref} is increased). In contrast, for poisoning the ST model, the poisoning is gradually degraded. Concurrent work [135] also discusses the impact of ϵ_{ref} on poisoning AT models but focuses on a fundamentally different task where the attack aims to degrade the adversarial robustness of AT models (rather than the clean accuracy here). It is also important to note that

our EntF can work well under different ϵ_{ref} between 2 and 8, where their approach is much more sensitive to the choice of ϵ_{ref} in their task (see their Figure 2(a)). We further visualize the perturbations generated with different ϵ_{ref} in Figure 4.5b. As can be seen, when using a standard reference model (i.e., $\epsilon_{\text{ref}} = 0$), the perturbations exhibit noisy patterns, but as the ϵ_{ref} is gradually increased, the perturbations tend to be more aligned with image semantics.

These observations suggest that poisoning ST and AT models requires modifying different types of features. More specifically, modifying the robust (semantic) features is the key to poisoning AT models, while modifying the non-robust features works for ST models. This conclusion also supports the well-known perspective that *non-robust features can be picked up by models during standard training, even in the presence of robust features, while adversarial training tends to utilize robust features* [55]. Figure 4.5a also confirms that our EntF is effective in poisoning ST models since it can decrease the model accuracy to the random guess level (i.e., 10% for CIFAR-10).

Hybrid attacks. The distinct roles of robust and non-robust features in poisoning AT and ST models inspire us to search for a hybrid attack to poison the ST and AT models simultaneously. This is particularly relevant for practical defenders who may want to adjust their AT budget to achieve optimal accuracy. A straightforward idea is to simply average the perturbations generated using an ST reference model and those using an AT reference model. In this case, the poison budget remains unchanged. Alternatively, we propose a hybrid attack that is based on jointly optimizing the perturbations against both ST and AT reference models, balanced by factors λ_i that correspond to different AT models. This modifies Equation 4.2 to the following Equation 4.5.

$$\mathcal{L}_{\text{hybrid}} = \max_{\boldsymbol{\delta}^{\text{poi}}} \|F^*_{L-1,\text{ST}}(\boldsymbol{x} + \boldsymbol{\delta}^{\text{poi}}) - \boldsymbol{\mu}_{y,\text{ST}}\|_2 + \sum_i \lambda_i \|F^*_{L-1,\text{AT}_i}(\boldsymbol{x} + \boldsymbol{\delta}^{\text{poi}}) - \boldsymbol{\mu}_{y,\text{AT}_i}\|_2. \tag{4.5}$$

Here we adopt two AT reference models with $\epsilon_{\text{ref}} = 2/255$ and $4/255$. Table 4.8 shows that both hybrid attacks substantially decrease the model accuracy across different AT budgets ϵ_{adv} varying from 0 (i.e., ST) to 16. Specifically, the optimization-based method performs much better than the average-based method. Table 4.9 further shows the much worse performance of other attacks and confirms the general effectiveness of our hybrid attack strategy, see ours (Hybrid) vs. AdvPoison (Hybrid).

Table 4.8: Evaluating two hybrid attacks (average/optimization) under different ϵ_{poi} vs. ϵ_{adv}.

$\epsilon_{\text{poi}} \backslash \epsilon_{\text{adv}}$	0/255	4/255	8/255	16/255	Optimal
None (Clean)	94.59	90.31	84.88	73.78	94.59
4/255	51.98/**29.51**	86.87/**84.48**	81.24/**80.53**	71.13/**70.26**	86.87/**84.48**
8/255	22.86/**11.89**	82.30/**76.55**	77.19/**74.30**	66.70/**65.75**	82.30/**76.55**
16/255	**5.34**/6.59	75.32/**59.55**	69.41/**66.25**	**59.40**/60.73	75.32/**66.25**

Table 4.9: Attack performance when the defender adjusts AT budget ϵ_{adv} to obtain the optimal accuracy.

Poison Method ($\epsilon_{\text{poi}} = 8/255$)	0/255	4/255	8/255	16/255	Optimal
None (Clean)	94.59	90.31	84.88	73.78	94.59
AdvPoison	9.91	88.98	83.11	71.31	88.98
REM	25.59	**46.57**	84.21	85.76	85.76
ADVIN	77.31	90.08	86.76	72.16	90.08
Unlearnable	25.69	90.47	84.91	79.81	90.47
Hypocritical	74.06	91.18	84.96	73.33	91.18
Hypocritical$^+$	75.22	84.82	86.56	82.26	86.56
AdvPoison (Hybrid)	**4.16**	82.74	78.65	67.17	82.74
Ours (Hybrid)	12.93	76.55	**74.30**	**65.75**	**76.55**

4.6 Conclusion

In this chapter, we have proposed EntF, a new poisoning approach to decreasing the deep learning classifier's accuracy even when adversarial training is applied. This approach is based on a new attack strategy that makes the features of training samples from different classes become entangled. Extensive experiments demonstrate the effectiveness of EntF against adversarial training in different scenarios, including those with more aggressive AT budgets, unseen model architectures, and adaptive defenses. We also discuss the distinct roles of the robust vs. non-robust features in poisoning standard vs. adversarially-trained models and demonstrate that our hybrid attacks can poison standard and AT models simultaneously.

We encourage future research to analyze EntF in more comprehensive settings and compare it to the current, shortcut-based methods from different angles. In particular, it is important to come up with new defenses against EntF, possibly based on advanced techniques of learning from noisy labels [129]. It is also worth noting that most of the current poisoning studies, including ours, have assumed that the poisoner has access to the training dataset of the target model. This assumption is realistic in specific threat models (e.g., secure dataset release [39]) but may not be plausible for sensitive/private data. Therefore, it would be promising to extend EntF to addressing data-free poisons [157, 119].

On the one hand, data poisoning could be potentially leveraged by malicious parties as attacks. In this case, we hope our work can inspire the community to develop stronger defenses based on our comprehensive analysis. On the other hand, when data poisoning is directly used for social good, e.g., for protecting personal data from being misused [39, 54, 43], our new approach for generating stronger poisons leads to stronger protective effects.

5
Dynamic Backdoor Attack

5.1 Introduction

In the previous chapter, we demonstrated that widely adopted defenses are insufficient to protect against indiscriminate poisoning attacks. In this chapter, we extend this discussion by examining another critical category of poisoning attacks, known as backdoor attacks. Our findings reveal that backdoor attacks can be strategically designed to bypass existing defenses, further escalating the urgency of addressing these threats.

A backdoor attack involves an adversary embedding a hidden trigger in the model during the training phase, which, when activated, causes the model to behave maliciously. This type of attack can lead to severe security and privacy issues, especially as machine learning models are increasingly deployed in security-critical applications. For example, Apple's FaceID [11] relies on machine learning-based facial recognition to unlock devices and authenticate purchases. An attacker could implant a backdoor in such an authentication system, gaining unauthorized access without detection.

Several well-known backdoor attacks [153, 50, 80] have already been developed, demonstrating high levels of effectiveness. However, these attacks typically assume a static trigger, meaning the trigger has a fixed pattern and location. For example, as illustrated in Figure 5.1, Badnets [50], one of the most prominent backdoor attack methods, uses a simple white square as a trigger, consistently placed in the top-left corner of all inputs.

This static nature of the trigger—both in pattern and position—has been a key factor in the development of many existing defenses against backdoor attacks [143, 79]. These defenses exploit the predictable nature of static triggers, which makes it easier to detect backdoored data. Moreover, the uniformity of the trigger placement enables defenders to link all backdoored inputs together. In scenarios where a single backdoored input is detected, the defender can extract the trigger from that input, generate a dataset, and fine-tune the model to neutralize the backdoor without needing advanced defense mechanisms.

While these defenses have proven to be efficient and effective at identifying or removing static backdoor patterns, they may foster a false sense of security. Many users might wrongly believe that backdoor attacks are well understood and adequately addressed. However, as we show in this chapter, this belief is both misleading and risky. Relying exclusively on defenses targeting static triggers overlooks the possibility of more sophisticated, dynamic backdoor attacks, leaving systems vulnerable to emerging and evolving threats.

5.1.1 Our Contributions

In this chapter, we demonstrate that existing defenses can be bypassed by introducing the first class of backdoor techniques against deep neural network (DNN) models that employ *dynamic triggers*. These triggers are dynamic in both *pattern* and *location*, making them more stealthy and harder to detect. We refer to this new class of attacks as *dynamic backdoor attacks*.

Dynamic backdoor attacks offer several advantages to adversaries, particularly in terms of flexibility. Unlike static backdoors, dynamic backdoor attacks allow the trigger's pattern and location to vary, making it more difficult for defenders to detect and remove

(a) Static backdoor

(b) Dynamic backdoor

Figure 5.1: A comparison between static and dynamic backdoors. Figure 5.1a shows an example for static backdoors with a fixed trigger (white square at top left corner of the image). Figure 5.1b show examples for the dynamic backdoor with different triggers for the same target label. As the figures show, the dynamic backdoor trigger have different location and patterns, compared to the static backdoor where there is only a single trigger with a fixed location and pattern.

the backdoor. As shown in Figure 5.1b, we implemented a dynamic backdoor attack in a model trained on the CelebA dataset [84].

Moreover, our techniques extend beyond the traditional focus of backdoor attacks on a single or limited set of target labels. Instead, we introduce methods that can apply backdoors to *all* labels of the machine learning model. This significantly increases the difficulty of mitigating our dynamic backdoors.

We propose three distinct dynamic backdoor techniques: *Random Backdoor*, *Backdoor Generating Network (BaN)*, and *conditional Backdoor Generating Network (c-BaN)*. In particular, the latter two attacks algorithmically generate triggers to mount backdoor attacks which are first of their kind. In the following, we abstractly introduce each of our techniques.

Random Backdoor: In this approach, triggers are generated by sampling patterns from a uniform distribution and placing each trigger at a random location for every input. These randomly generated triggers are then mixed with clean data to train the backdoored model. This randomness in both pattern and location adds a layer of unpredictability, making detection more difficult.

Backdoor Generating Network (BaN): Our second technique, BaN, introduces a generative model to construct backdoor triggers. To the best of our knowledge, this is the first backdoor attack using a generative network to create triggers automatically, offering the adversary enhanced flexibility. BaN is trained jointly with the backdoor model, taking a latent code sampled from a uniform distribution to generate a trigger, which is placed at random locations on the input. This dynamic generation process makes both the pattern and location of the trigger unpredictable. Moreover, BaN serves as a general framework, allowing the adversary to adjust the model's loss function to adapt to specific defense mechanisms. For example, if a particular backdoor defense is known, BaN can incorporate a tailored loss function to evade detection.

conditional Backdoor Generating Network (c-BaN): While both Random Back-

door and BaN can apply dynamic backdoors to multiple target labels, they require each label to have its own distinct set of trigger locations. This means a single location cannot have triggers for multiple target labels. c-BaN overcomes this limitation by transforming BaN into a conditional backdoor generating network. c-BaN takes the target label as input and generates a label-specific trigger, allowing triggers for different labels to be placed at any location without the need for disjoint location sets. This flexibility further enhances the stealth and complexity of the attack.

We evaluate our proposed dynamic backdoor techniques across three different ML model architectures and three benchmark datasets. The results demonstrate that all our techniques achieve near-perfect backdoor success rates, with minimal impact on model utility. For instance, our BaN trained models on CelebA [84] and MNIST [12] datasets achieve 70% and 99% accuracy, respectively, which is the same accuracy as the clean models. Similarly, models trained with c-BaN, BaN, and Random Backdoor on the CIFAR-10 [9] dataset achieve 92%, 92.1%, and 92% accuracy, respectively, which is almost the same as the performance of a clean model (92.4%).

Additionally, we evaluate our dynamic backdoor attacks against five state-of-the-art backdoor defense techniques: Neural Cleanse [143], ABS [79], MNTD [151], STRIP [45], and Februus [33]. Our results demonstrate that all of our techniques successfully bypass these defenses, further underscoring the robustness and stealth of dynamic backdoor attacks.

In general, our contributions can be summarized as follows:

- We broaden the class of backdoor attacks against deep neural network (DNN) models by introducing dynamic backdoor attacks.
- We propose both Backdoor Generating Network (BaN) and conditional Backdoor Generating Network (c-BaN), which are the first algorithmic backdoor paradigm.
- Our dynamic backdoor attacks achieve strong performance, while bypassing the current state-of-the-art backdoor defense techniques.

5.2 Problem Statement

In this part, our objective is to develop dynamic backdoor attacks, where the pattern and location of the trigger are dynamic, rather than static as in traditional backdoor attacks. Specifically, we aim to design backdoors with varying trigger values (patterns) that can be placed at different positions (locations) within the input, providing the adversary with greater flexibility and making detection more difficult.

A dynamic backdoor attack involves a set of triggers $\mathcal{T}$, corresponding target labels ℓ', and a backdoor embedding function $\mathcal{A}$. We define this function as: $\mathcal{A}(\mathbf{x}, t_i, \kappa) = \mathbf{x}_{bd}$, where $\mathbf{x}$ is the input feature vector, $t_i \in \mathcal{T}$ is the trigger, and κ represents the position where the backdoor is inserted, yielding the modified input $\mathbf{x}_{bd}$. More formally, $\mathcal{A}(\mathbf{x}, t_i, \kappa) = t_i \cdot \kappa + \mathbf{x} \cdot (1 - \kappa)$, where κ is a binary mask specifying the location of the trigger. To train such a model, an adversary needs both clean $\mathcal{D}_c$ (to preserve the model's utility) and backdoored data $\mathcal{D}_{bd}$ (to implement the backdoor behaviour), where $\mathcal{D}_{bd}$ is constructed by adding triggers on a subset of $\mathcal{D}_c$.

Compared to static backdoor attacks, dynamic backdoor attacks introduce additional complexity by allowing multiple trigger patterns and positions. This flexibility not only increases the adversary's ability to tailor the attack but also complicates detection, as existing defenses are designed with static backdoors in mind. Furthermore, dynamic triggers can be algorithmically generated, offering the adversary even more customization, as we will explore in Section 5.3.2 and Section 5.3.3.

5.2.1 Threat Model

Dynamic backdoor attacks occur during the training phase, where the adversary interferes with the training process. We assume that the adversary controls the training of the target model and has access to the training data, consistent with prior works on backdoor attacks [50]. We further relax this assumption (in Section 5.4.7) to only assume the ability to poison the target model's training data.

Once the backdoored model is deployed, the adversary can trigger the attack by embedding a trigger into an input and querying the model. This can be done either digitally, by adding the trigger to an image, or physically, by printing the trigger and attaching it to the image, following previous approaches [50]. The presence of the trigger causes the model to misclassify the input, potentially allowing the adversary to bypass security mechanisms such as authentication systems.

5.3 Dynamic Backdoors

In this section, we propose three different techniques for performing dynamic backdoor attacks, namely, Random Backdoor, Backdoor Generating Network (BaN), and conditional Backdoor Generating Network (c-BaN).

5.3.1 Random Backdoor

We start with our simplest approach, i.e., the Random Backdoor technique. Abstractly, the Random Backdoor technique constructs triggers by sampling them from a uniform distribution, and adding them to the inputs at random locations. We first introduce how to use our Random Backdoor technique to implement a dynamic backdoor for a single target label, then we generalize it to consider multiple target labels.

Single Target Label: We start with the simple case of considering dynamic backdoors for a single target label. Intuitively, we construct a set of triggers ($\mathcal{T}$) and a set of possible locations ($\mathcal{K}$), such that for any trigger sampled from $\mathcal{T}$ and added to any input at a random location sampled from $\mathcal{K}$, the model will output the specified target label. More formally, for any location $\kappa_i \in \mathcal{K}$, any trigger $t_i \in \mathcal{T}$, and any input $\mathbf{x}_i \in \mathcal{X}$:

$$\mathcal{M}_{bd}(\mathcal{A}(\mathbf{x}_i, t_i, \kappa_i)) = \ell$$

where ℓ is the target label, $\mathcal{T}$ is the set of triggers, and $\mathcal{K}$ is the set of locations.

To implement such a backdoor in a model, an adversary needs first to select their desired trigger locations, and create the set of possible locations $\mathcal{K}$. Then, they use

both clean and backdoored data to update the model for each epoch. More concretely, the adversary trains the model as mentioned in Section 5.2 with the following two differences.

1. First, instead of using a fixed trigger for all inputs, each time the adversary wants to add a trigger to an input, they sample a new trigger from a uniform distribution, i.e., $t \sim \mathcal{U}(0, 1)$. Here, the set of possible triggers $\mathcal{T}$ contains the full range of all possible values for the triggers, since the trigger is randomly sampled from a uniform distribution.

2. Second, instead of placing the trigger in a fixed location, they place it at a random location κ, sampled from the predefined set of locations, i.e., $\kappa \in \mathcal{K}$.

This technique is not only limited to the uniform distribution, but the adversary can use different distributions like the Gaussian distribution to construct triggers. Using different distributions can, for example, help the adversary to change the appearance of the used triggers.

Finally, the adversary does not need access to the training of the target model for this technique. Instead, they can backdoor a target model by only adding the backdoored data to its training set, i.e., poison the training set.

Multiple Target Labels: Next, we consider the more complex case of having multiple target labels. Without loss of generality, we consider implementing a backdoor for each label in the dataset, since this is the most challenging setting. However, our techniques can be applied to any smaller subset of labels. This means that for any label $\ell_i \in \ell$, there exists a trigger t which when added to the input $\mathbf{x}$ at a location κ, will make the model $\mathcal{M}_{bd}$ output ℓ_i. More formally,

$$\forall \ell_i \in \ell \; \exists \; t, \kappa : \mathcal{M}_{bd}(\mathcal{A}(\mathbf{x}, t, \kappa)) = \ell_i$$

To achieve the dynamic backdoor behaviour in this setting, each target label should have a set of possible triggers and a set of possible locations. More formally,

$$\forall \ell_i \in \ell \; \exists \; \mathcal{T}_i, \mathcal{K}_i$$

where $\mathcal{T}_i$ is the set of possible triggers and $\mathcal{K}_i$ is the set of possible locations for the target label ℓ_i.

We generalize the Random Backdoor technique by dividing the set of possible locations $\mathcal{K}$ into disjoint subsets for each target label, while keeping the trigger construction method the same as in the single target label case, i.e., the triggers are still sampled from a uniform distribution. For instance, for the target label ℓ_i, we sample a set of possible locations $\mathcal{K}_i$, where $\mathcal{K}_i$ is a subset of $\mathcal{K}$ ($\mathcal{K}_i \subset \mathcal{K}$).

The adversary can construct the disjoint sets of possible locations as follows:

1. First, the adversary selects all possible trigger locations and constructs the set $\mathcal{K}$.

2. Second, for each target label ℓ_i, they construct the set of possible locations for this label $\mathcal{K}_i$ by sampling the set $\mathcal{K}$. Then, they remove the sampled locations from the set $\mathcal{K}$.

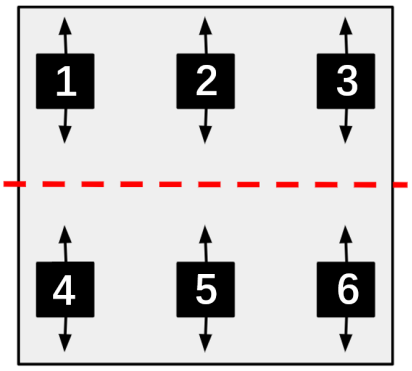


Figure 5.2: An illustration of our location setting technique for 6 target labels. The red dotted line demonstrates the boundary of the vertical movement for each target label.

We propose the following simple algorithm to assign the locations for the different target labels. However, an adversary can construct the location sets arbitrarily with the only restriction that no location can be used for more than one target label.

We uniformly split the image into non-intersecting regions, and assign a region for each target label, in which the triggers' locations can move vertically. Figure 5.2 shows an example of our location setting technique for a use case with 6 target labels. As the figure shows, each target label has its own region, for example, label 1 occupies the top left region of the image. We stress that this is one way of dividing the location set $\mathcal{K}$ to the different target labels. However, an adversary can choose a different way of splitting the locations inside $\mathcal{K}$ to the different target labels. The only requirement the adversary has to fulfill is to avoid assigning a location for different target labels. Later, we will show how to overcome this limitation with our more advanced c-BaN technique.

5.3.2 Backdoor Generating Network (BaN)

Our Random Backdoor technique successfully implements dynamic triggers, however, it offers the adversary limited flexibility as triggers are sampled from a preset distribution. Moreover, the triggers are sampled independently of the target model. In other words, the Random Backdoor technique does not search for the best triggers to implement the backdoor attack. To address these limitations, we introduce our second technique to implement dynamic backdoors, namely, Backdoor Generating Network (BaN). BaN is the first approach to algorithmically generate backdoor triggers, instead of using fixed triggers or sampling triggers from a uniform distribution (as in Section 5.3.1).

BaN is inspired by the state-of-the-art generative models, i.e., Generative Adversarial Networks (GANs) [48]. However, it is different from the original GANs in the following aspects. First, instead of generating images, it generates backdoor triggers. Second, we jointly train the BaN's generator with the target model instead of the discriminator, to learn (the generator) and implement (the target model) the best patterns for the backdoor triggers.

After training, the BaN can generate a trigger (t) for each noise vector ($z \sim \mathcal{U}(0,1)$). This trigger is then added to an input using the backdoor adding function $\mathcal{A}$, to create

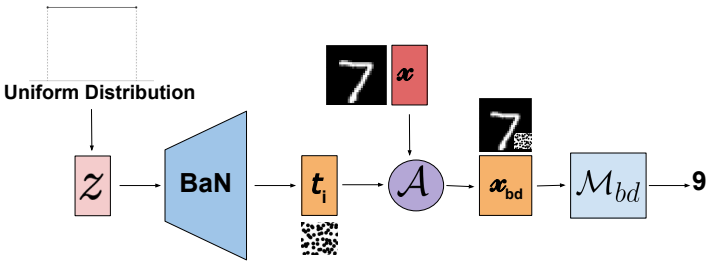


Figure 5.3: An overview of the BaN technique.

the backdoored input as shown in Figure 5.3. Similar to the previous approach (Random Backdoor), the generated triggers are placed at random locations.

In this section, we first introduce the BaN technique for a single target label, and then we generalize it for multiple target labels.

Single Target Label: We start with presenting how to implement a dynamic backdoor for a single target label, using our BaN technique. First, the adversary creates the set $\mathcal{K}$ of the possible locations. They then jointly train the BaN with the backdoored $\mathcal{M}_{bd}$ model as follows:

1. The adversary starts each training epoch by querying the clean data to the backdoored model $\mathcal{M}_{bd}$. Then, they calculate the clean loss $\mathcal{L}_c$ between the ground truth and the output labels. We use the cross-entropy loss for our clean loss, which is defined as follows:

$$\sum_i \mathbf{y}_i \log(\hat{\mathbf{y}}_i)$$

   where $\mathbf{y}_i$ is the true probability of label ℓ_i and $\hat{\mathbf{y}}_i$ is our predicted probability of label ℓ_i.

2. They then generate n noise vectors, where n is the batch size.

3. On the input of the n noise vectors, the BaN generates n triggers.

4. The adversary then creates the backdoored data by adding the generated triggers to the clean data using the backdoor adding function $\mathcal{A}$.

5. They then query the backdoored data to the backdoored model $\mathcal{M}_{bd}$ and calculate the backdoor loss $\mathcal{L}_{bd}$ on the model's output and the target label. Similar to the clean loss, we use the cross-entropy loss as our loss function for $\mathcal{L}_{bd}$.

6. Finally, the adversary updates the backdoor model $\mathcal{M}_{bd}$ using both the clean and backdoor losses $(\mathcal{L}_c + \mathcal{L}_{bd})$ and updates the BaN with the backdoor loss $(\mathcal{L}_{bd})$.

We show later in Section 5.4.7 how to simplify the threat model for the BaN technique to only assume the ability to poison the training data, i.e., the adversary backdoors the target model without interfering with its training algorithm.

Multiple Target Labels: We now consider the more complex case of building a dynamic backdoor for multiple target labels using our BaN technique. To recap, our BaN generates general triggers and does not label specific triggers. In other words, the same trigger pattern can be used to trigger multiple target labels. Thus similar to the Random Backdoor, we depend on the location of the triggers to determine the output label.

We follow the same approach of the Random Backdoor technique to assign different locations for different target labels (Section 5.3.1), to generalize the BaN technique. More concretely, the adversary implements the dynamic backdoor for multiple target labels using the BaN technique as follows:

1. The adversary starts by creating disjoint sets of locations for all target labels.

2. Next, they follow the same steps as in training the backdoor for a single target label, while repeating from step 2 to 5 for each target label and adding all their backdoor losses together. More formally, for the multiple target label case, the backdoor loss is defined as: $\sum_{i}^{|\ell'|} \mathcal{L}_{bd_i}$, where ℓ' is the set of target labels, and $\mathcal{L}_{bd_i}$ is the backdoor loss for target label ℓ_i.

5.3.3 conditional Backdoor Generating Network (c-BaN)

So far, we have proposed two techniques to implement dynamic backdoors for both single and multiple target labels, i.e, Random Backdoor (Section 5.3.1) and BaN (Section 5.3.2). To recap, both techniques have the limitation of not having label-specific triggers and only depending on the trigger location to determine the target label. We now introduce our third and most advanced technique, the conditional Backdoor Generating Network (c-BaN), which overcomes this limitation. More concretely, with the c-BaN technique any location κ inside the location set $\mathcal{K}$ can be used to trigger any target label. To achieve this location independence, the triggers need to be label-specific. Therefore, we convert the Backdoor Generating Network (BaN) into a conditional Backdoor Generating Network (c-BaN). More specifically, we add the target label as an additional input to the BaN for conditioning it to generate target-specific triggers.

We construct c-BaN by adding an additional input layer to BaN to include the target label as an input. Figure 5.4 represents an illustration of c-BaN. As the figure shows, the noise vector and the target label are encoded to latent vectors with the same size (to give equal weights for both inputs). These two latent vectors are then concatenated and used as input to the next layer.

The c-BaN is trained similarly to the BaN, with the following two exceptions.

1. First, the adversary does not have to create disjoint sets of locations for all target labels (step 1), they can use the complete location set $\mathcal{K}$ for all target labels.

2. Second, instead of using only the noise vectors as an input to the BaN, the adversary one-hot encodes the target label, then uses it together with the noise vectors as the input to the c-BaN.

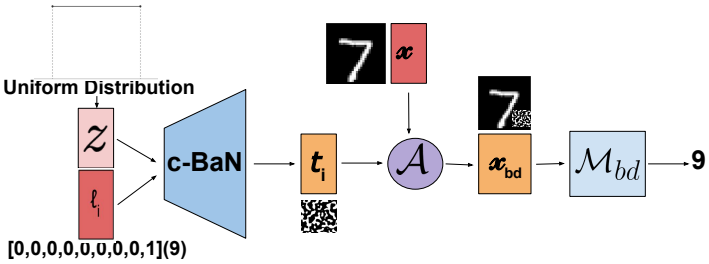


Figure 5.4: An overview of the c-BaN technique.

Similar to BaN, we later (Section 5.4.7) show how to simplify the threat model for the c-BaN.

To use the c-BaN, the adversary first samples a noise vector and one-hot encodes the label. Then, they input both of them to the c-BaN, which generates a trigger. The adversary uses the backdoor adding function $\mathcal{A}$ to add the trigger to the target input. Finally, they query the backdoored input to the backdoored model, which will output the target label. We visualize the complete pipeline of using the c-BaN technique in Figure 5.4.

In this section, we have introduced three techniques for implementing dynamic backdoors, namely, the Random Backdoor, the Backdoor Generating Network (BaN), and the conditional Backdoor Generating Network (c-BaN). These three dynamic backdoor techniques present a framework to generate dynamic backdoors for different settings. For instance, our framework can generate target-specific triggers' pattern using the c-BaN, or target-specific triggers' location like the Random Backdoor and BaN. More interestingly, our framework allows the adversary to customize their backdoor by adapting the backdoor loss functions. For instance, the adversary can adapt to different defenses against the backdoor attack that can be modeled as a machine learning model. This can be achieved by adding any defense as a discriminator into the training of the BaN or c-BaN. Adding this discriminator will penalize/guide the backdoored model to bypass the modeled defense.

5.4 Evaluation

We evaluate our dynamic backdoor attacks on three widely used image datasets: MNIST, CIFAR-10, and CelebA, which are commonly employed for benchmarking security, privacy, and computer vision tasks. For CelebA, we focus on three balanced attributes—Heavy Makeup, Mouth Slightly Open, and Smiling—and combine them into 8 classes for a multi-label classification task, using 10,000 images each for training and testing. Unlike MNIST and CIFAR-10, CelebA is imbalanced, adding another layer of complexity to our evaluations. Although our attack is demonstrated on these datasets, it can be easily generalized to other datasets by adapting the architectures of our models.

5.4.1 Experimental Setup

We first present our target models, then the evaluation metrics. For the target models' architecture, we use the VGG-19 [126] for the CIFAR-10 dataset, and build our own convolution neural networks (CNN) for the CelebA and MNIST datasets. More concretely, we use 3 convolution layers and 5 fully connected layers for the CelebA CNN. And 2 convolution layers and 2 fully connected layers for the MNIST CNN. Moreover, we use dropout for both the CelebA and MNIST models to avoid overfitting.

For BaN, we use the following architecture:

Backdoor Generating Network (BaN)'s architecture:

$$
\begin{aligned}
z \rightarrow\ & \texttt{FullyConnected(64)} \\
& \texttt{FullyConnected(128)} \\
& \texttt{FullyConnected(128)} \\
& \texttt{FullyConnected(|t|)} \\
& \texttt{Sigmoid} \rightarrow t
\end{aligned}
$$

Here, `FullyConnected(`x`)` denotes a fully connected layer with x hidden units, $|t|$ denotes the size of the required trigger, and `Sigmoid` is the Sigmoid function. We adopt ReLU as the activation function for all layers, and apply dropout after all layers except the first and last ones.

For c-BaN, we use the following architecture:

conditional Backdoor Generating Network (c-BaN)'s architecture:

$$
\begin{aligned}
z, \ell \rightarrow\ & 2 \times \texttt{FullyConnected(64)} \\
& \texttt{FullyConnected(128)} \\
& \texttt{FullyConnected(128)} \\
& \texttt{FullyConnected(128)} \\
& \texttt{FullyConnected(|t|)} \\
& \texttt{Sigmoid} \rightarrow t
\end{aligned}
$$

The first layer consists of two separate fully connected layers, where each one of them takes an independent input, i.e., the first takes the noise vector z and the second takes the target label ℓ. The outputs of these two layers are then concatenated and used as an input to the next layer (see Section 5.3.3). Similar to BaN, we adopt ReLU as the activation function for all layers and apply dropout after all layers except the first and last one.

For evaluating the dynamic backdoor attacks' performance, we define the following two metrics: *Backdoor success rate* which calculates the backdoored model's accuracy on the backdoored data; *Model utility* which measures the original functionality of the backdoored model. We quantify the model utility by comparing the accuracy of the backdoored model with the accuracy of a clean model on clean data. Closer accuracies imply a better model utility. All of our experiments are implemented using Pytorch

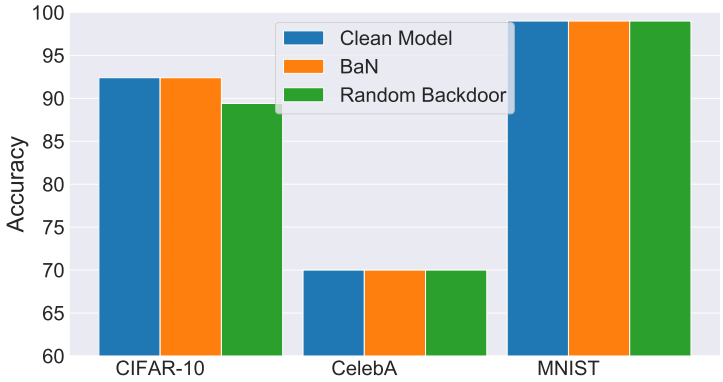


Figure 5.5: The result of our dynamic backdoor techniques for a single target label on the clean testing dataset.

and our code will be published for reproducibility.

5.4.2 Random Backdoor

We now evaluate the performance of our first dynamic backdooring technique, namely, the Random Backdoor. We use all three datasets for the evaluation. First, we evaluate the single target label case, where we only implement a backdoor for a single target label. Then we evaluate the more generalized case, i.e., the multiple target labels case, where we implement a backdoor for all possible labels in the dataset.

For both the single and multiple target label cases, we split each dataset into training and testing datasets. The training dataset is used to train the MNIST and CelebA models from scratch. For CIFAR-10, we use a pre-trained VGG-19 model. For evaluating our models, we use the testing dataset as our clean testing dataset. And construct a backdoored testing dataset, by adding triggers to all members of the testing dataset. To recap, for the Random Backdoor technique, we construct the triggers by sampling them from a uniform distribution, and add them to the images using the backdoor adding function $\mathcal{A}$. We use the backdoored testing dataset to calculate the backdoor success rate, and the training dataset to train a clean model -for each dataset- to evaluate the backdoored model's ($\mathcal{M}_{bd}$) utility.

We follow Section 5.3.1 to train our backdoored model $\mathcal{M}_{bd}$ for both the single and multiple target label cases. Abstractly, for each epoch, we update the backdoored model $\mathcal{M}_{bd}$ using both the clean and backdoor losses $\mathcal{L}_c + \mathcal{L}_{bd}$. For the set of possible locations $\mathcal{K}$, we use four possible locations.

The backdoor success rate is always 100% for both the single and multiple target label cases on all three datasets, hence, we only focus on the backdoored model's ($\mathcal{M}_{bd}$) utility.

Single Target Label: We first present our results for the single target label case. Figure 5.5 compares the accuracies of the backdoored model $\mathcal{M}_{bd}$ and the clean model $\mathcal{M}$. As the figure shows, our backdoored models achieve the same performance as the clean models for both the MNIST and CelebA datasets, i.e., 99% for MNIST and 70%

(a) Random Backdoor

(b) BaN

(c) BaN with higher randomness

Figure 5.6: The result of our Random Backdoor (Figure 5.6a), BaN (Figure 5.6b), and BaN with higher randomness (Figure 5.6c) techniques for a single target label (0).

for CelebA. For the CIFAR-10 dataset, there is a slight drop in performance, which is less than 2%. This shows that our Random Backdoor technique can implement a perfectly functioning backdoor, i.e., the backdoor success rate of $\mathcal{M}_{bd}$ is 100% on the backdoored testing dataset, with a negligible utility loss.

To visualize the output of our Random Backdoor technique, we first randomly sample 8 images from the MNIST dataset, and then use the Random Backdoor technique to construct triggers for them. Finally, we add these triggers to the images using the backdoor adding function $\mathcal{A}$, and show the result in Figure 5.6a. As the figure shows, the triggers all look distinctly different and are located at different locations as expected.

Multiple Target Labels: Second, we present our results for the multiple target label case. To recap, we consider all possible labels for this case. For instance, for the MNIST dataset, we consider all digits from 0 to 9 as our target labels. We train our Random Backdoor models for the multiple target labels as mentioned in Section 5.3.1.

We use a similar evaluation setting to the single target label case, with the following exception. To evaluate the performance of the backdoored model $\mathcal{M}_{bd}$ with multiple target labels, we construct a backdoored testing dataset for each target label by generating and adding triggers to the clean testing dataset. In other words, we use all images in the testing dataset to evaluate all possible labels.

Similar to the single target label case, we focus on the accuracy on the clean testing dataset, since the backdoor success rate for all models on the backdoored testing datasets are approximately 100% for all target labels.

We use the clean testing datasets to evaluate the backdoored model's $\mathcal{M}_{bd}$ utility, i.e., we compare the performance of the backdoored model $\mathcal{M}_{bd}$ with the clean model $\mathcal{M}$ in Figure 5.7. As the figure shows, using our Random Backdoor technique, we are able to train backdoored models that achieve similar performance as the clean models for all datasets. For instance, for the CIFAR-10 dataset, our Random Backdoor technique achieves 92% accuracy, which is very similar to the accuracy of the clean model (92.4%).

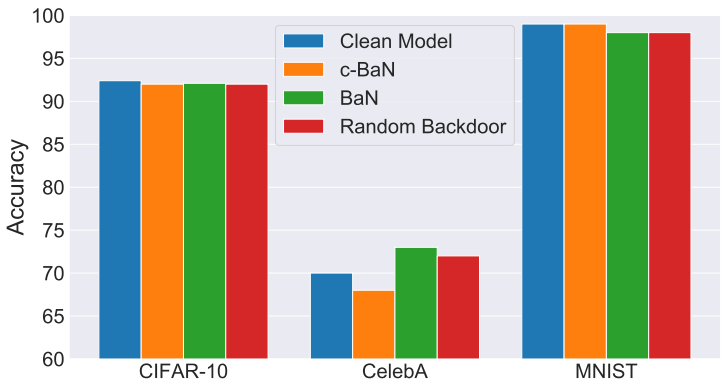


Figure 5.7: The result of our dynamic backdoor techniques for multiple target labels on the clean testing dataset.

For the CelebA dataset, the Random Backdoor technique achieves a slightly (about 2%) better performance than the clean model. We believe this is due to the regularization effect of the Random Backdoor technique. Finally, for the MNIST dataset, both models achieve a similar performance with just 1% difference between the clean model (99%) and the backdoored one (98%).

To visualize the output of our Random Backdoor technique on multiple target labels, we construct triggers for all possible labels in the CIFAR-10 dataset, and use $\mathcal{A}$ to add them to a randomly sampled image from the CIFAR-10 clean testing dataset. Figure 5.8a shows the image with different triggers. The different patterns and locations used for the different target labels can be clearly demonstrated in Figure 5.8a. For instance, comparing the location of the trigger for the first and sixth images, the triggers are in the same horizontal position but a different vertical position, as previously illustrated in Figure 5.2.

Moreover, we further visualize in Figure 5.9a the dynamic behavior of the triggers generated by our Random Backdoor technique. Without loss of generality, we generate triggers for the target label 5 (plane) and add them to randomly sampled CIFAR-10 images. To make it clear, we train the backdoor model $\mathcal{M}_{bd}$ for all possible labels set as target labels, but we visualize the triggers for a single label to show the dynamic behaviour of our Random Backdoor technique with respect to the triggers' pattern and locations. As Figure 5.9a shows, the generated triggers have different patterns and locations for the same target label, which achieves our desired dynamic behavior.

5.4.3 Backdoor Generating Network (BaN)

Next, we evaluate our BaN technique. We follow the same evaluation settings for the Random Backdoor technique, except with respect to how the triggers are generated. We train our BaN model and generate the triggers as mentioned in Section 5.3.2.

Single Target Label: Similar to the Random Backdoor, the BaN technique achieves a perfect backdoor success rate with a negligible utility loss. Figure 5.5 compares the performance of the backdoored models -trained using the BaN technique- with the

clean models, when tested using the clean testing dataset. As Figure 5.5 shows, our BaN trained backdoored models achieve 99%, 92.4%, and 70% accuracy on the MNIST, CIFAR-10, and CelebA datasets, respectively, which is the same performance of the clean models.

We visualize the BaN generated triggers using the MNIST dataset in Figure 5.6b. To construct the figure, we use the BaN to generate multiple triggers -for the target label 0-, then we add them on a set of randomly sampled MNIST images using the backdoor adding function $\mathcal{A}$.

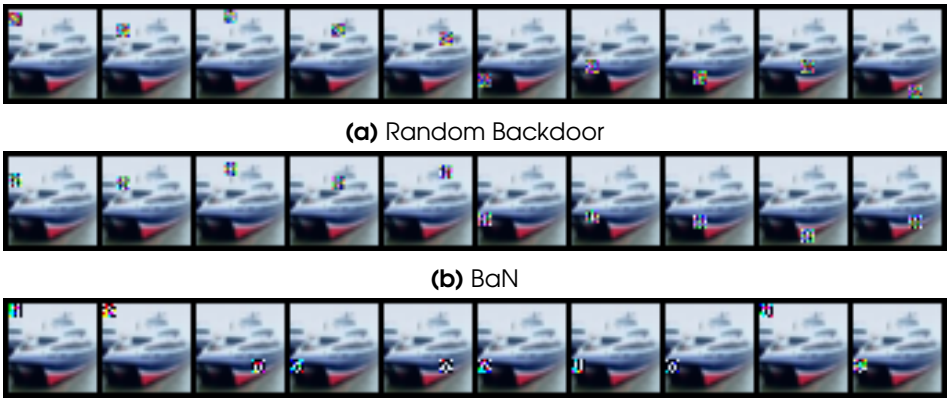
The generated triggers look very similar as shown in Figure 5.6b. This behaviour is expected as the MNIST dataset is simple, and the BaN technique does not have any explicit loss to enforce the network to generate different triggers. However, to show the flexibility of our approach, we increase the randomness of the BaN network by simply adding one more dropout layer after the last layer, to avoid the overfitting of the BaN model to a unique pattern. We show the results of the BaN model with higher randomness in Figure 5.6c. The resulting model still achieves the same performance, i.e., 99% accuracy on the clean data and 100% backdoor success rate, but as the figure shows the triggers look significantly different. This again shows that our framework can easily adapt to the requirements of an adversary.

These results together with the results of the Random Backdoor (Section 5.4.2) clearly show the effectiveness of both of our proposed techniques, for the single target label case. They are both able to achieve almost the same accuracy of a clean model, with a 100% working backdoor, for a single target label.

Multiple Target Labels: Similar to the single target label case, we focus on the backdoored models' performance on the clean testing dataset, as our BaN backdoored models achieve a perfect accuracy on the backdoored testing dataset, i.e., the backdoor success rate for all datasets is approximately 100% for all target labels.

We compare the performance of the BaN backdoored models with one of the clean models using the clean testing dataset in Figure 5.7. Our BaN backdoored models are able to achieve almost the same accuracy as the clean model for all datasets, as can be shown in Figure 5.7. For instance, for the CIFAR-10 dataset, our BaN achieves 92.1% accuracy, which is only 0.3% less than the performance of the clean model (92.4%). Similar to the Random Backdoor backdoored models, our BaN backdoored models achieve marginally better performance for the CelebA dataset. More concretely, our BaN backdoored models trained for the CelebA dataset achieve about 2% better performance than the clean model, on the clean testing dataset. We also believe this improvement is due to the regularization effect of the BaN technique. Finally, for the MNIST dataset, our BaN backdoored models achieve strong performance on the clean testing dataset (98%), which is just 1% lower than the performance of the clean models (99%).

Similar to the Random Backdoor, we visualize the results of the BaN backdoored models with two figures. The first (Figure 5.8b) shows the different triggers for the different target labels on the same CIFAR-10 image, and the second (Figure 5.9b) shows the different triggers for the same target label (plane) on randomly sampled CIFAR-10 images. As both figures show, the BaN generated triggers achieve dynamic behaviour in both locations and patterns. For instance, for the same target label (Figure 5.9b),



(a) Random Backdoor

(b) BaN

(c) c-BaN

Figure 5.8: The visualization result of our Random Backdoor (Figure 5.8a), BaN (Figure 5.8b), and c-BaN (Figure 5.8c) techniques for all labels of the CIFAR-10 dataset.

the patterns of the triggers look significantly different and the locations vary vertically. Similarly, for different target labels (Figure 5.8b), both the pattern and location of triggers are significantly different.

5.4.4 conditional Backdoor Generating Network (c-BaN)

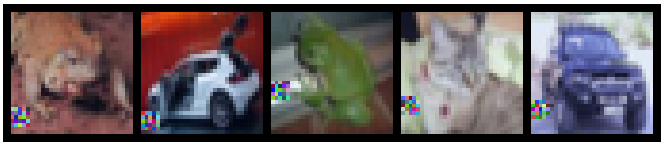
Next, we evaluate our conditional Backdoor Generating Network (c-BaN) technique. For the single target label case, the c-BaN technique is the same as the BaN technique. Thus, we only consider the multiple target labels case in this section.

We follow a similar setup as the one introduced in Section 5.4.3, with the exception on how to train the backdoored model $\mathcal{M}_{bd}$ and generate the triggers. We follow Section 5.3.3 to train the backdoored model and generate the triggers. For the set of possible locations $\mathcal{K}$, we use four possible locations.

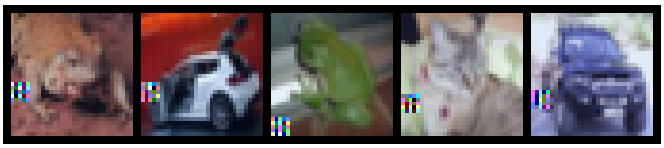
We compare the performance of the c-BaN with the other two techniques in addition to the clean model. All of our three dynamic backdoor techniques achieve an almost perfect backdoor success rate on the backdoored testing datasets, hence similar to the previous sections, we focus on the performance on the clean testing datasets.

Figure 5.7 compares the accuracy of the backdoored and clean models using the clean testing dataset, for all of our three dynamic backdoor techniques. As the figure shows, all of our dynamic backdoored models have similar performance as the clean models. For instance, for the CIFAR-10 dataset, our c-BaN, BaN and Random Backdoor achieve 92%, 92.1%, and 92% accuracy, respectively, which is very similar to the accuracy of the clean model (92.4%). Also for the MNIST dataset, all models achieve very similar performance with no difference between the clean and c-BaN models (99%) and only 1% difference between them, and the BaN and Random Backdoor models (98%).

Similar to the previous two techniques, we visualize the dynamic behaviour of the c-BaN backdoored models using two different figures. First, by generating triggers for all possible labels and adding them on a CIFAR-10 image in Figure 5.8c. More generally, Figure 5.8 shows the visualization of all three dynamic backdoor techniques in the same



(a) Random Backdoor



(b) BaN

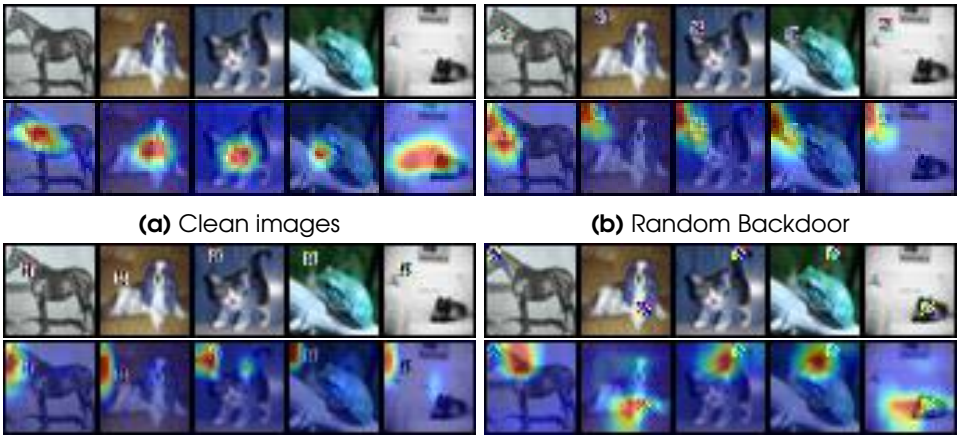
(c) c-BaN

Figure 5.9: The result of our Random Backdoor (Figure 5.9a), BaN (Figure 5.9b), and c-BaN (Figure 5.9c) techniques for the target target label 5 (plane).

settings, i.e., backdooring a single image to all possible labels. As the figure shows, the Random Backdoor Figure 5.8a has the most random patterns, which is expected as they are sampled from a uniform distribution. The figure also shows the different triggers' patterns and locations used for the different techniques. For instance, each target label in the Random Backdoor (Figure 5.8a) and BaN (Figure 5.8b) techniques have a unique (horizontal) location, unlike the c-BaN (Figure 5.8c) generated triggers, which different target labels can share the same locations, as can be shown for example in the first, second, and ninth images. To recap, both the Random Backdoor and BaN techniques split the location set $\mathcal{K}$ on all target labels, such that no two labels share a location, unlike the c-BaN technique which does not have this limitation.

Second, we visualize the dynamic behaviour of our techniques, by generating triggers for the same target label 5 (plane) and adding them to a set of randomly sampled CIFAR-10 images. Figure 5.9 compares the visualization of our three different dynamic backdoor techniques in this setting. More concretely, we train the backdoor model $\mathcal{M}_{bd}$ for all possible labels set as target labels. Then, for space restrictions, we plot the backdoored inputs for a single target label. As the figure shows, the Random Backdoor (Figure 5.9a) and BaN (Figure 5.9b) generated triggers can move vertically, however, they have a fixed position horizontally as mentioned in Section 5.3.1 and illustrated in Figure 5.2. The c-BaN (Figure 5.9c) triggers also show different locations. However, the locations of these triggers are more distant and can be shared for different target labels, unlike the other two techniques. Furthermore, the figure shows that most of our triggers have different patterns for our techniques for the same target label, which achieves our targeted dynamic behavior concerning the patterns and locations of the triggers.

Finally, we compare the attention of the backdoored models on both clean and backdoored inputs. We use the Gradient-weighted Class Activation Mapping (Grad-



(a) Clean images **(b)** Random Backdoor

(c) BaN **(d)** c-BaN

Figure 5.10: Visualization of attention maps for all our techniques using the Grad-CAM technique.

CAM) technique [120] to compute the attention maps for our backdoored models. These maps show the most influential parts of the input that resulted in the model's output. Figure 5.10 depicts the results of our three different techniques. As expected all backdoored models mainly focus on the triggers in backdoored inputs and the main objects in the clean ones.

5.4.5 Evaluating Against Current State-Of-The-Art Defenses

We now evaluate our attacks against the current state-of-the-art backdoor defenses. Backdoor defenses can be classified into the following two categories, data-based defenses and model-based defenses. On the one hand, data-based defenses focus on identifying if a given input is clean or contains a trigger. On the other hand, model-based defenses focus on identifying if a given model is clean or backdoored.

We first evaluate our attacks against model-based defenses, then we evaluate them against data-based ones.

Model-based Defense: We evaluate all of our dynamic backdoor techniques in the multiple target label case against three of the current state-of-the-art model-based defenses, namely, Neural Cleanse [143], ABS [79], and MNTD [151].

We start by evaluating the ABS defense. We use the CIFAR-10 dataset to evaluate this defense, since it is the only supported dataset by the published defense model. As expected, running the ABS model against our dynamic backdoored ones does not result in detecting any backdoor for all of our models.

For Neural Cleanse, we use all three datasets to evaluate our techniques against it. Similar to ABS, all of our models are predicted to be clean models. Moreover, in multiple cases, our models had a lower anomaly index (the lower the better) than the clean model.

We believe that both of these defenses fail to detect our backdoors for two reasons. First, we break one of their main assumption, i.e., that the triggers are static in terms

of location and pattern. Second, we implement a backdoor for all possible labels, which makes the detection a more challenging task.

Finally, we evaluate the MNTD defense. To this end, we use the CIFAR-10 dataset to evaluate our three backdoor techniques. Following the same setting in [61], we build 200 shadow benign and backdoored models to train 50 meta-classifiers sequentially for further evaluation. The meta-classifier takes a target model as its input and outputs a score. This score represents the likelihood of the model being backdoored, i.e., a higher score indicates the target model is more likely to be backdoored.

Our results show that the score predicted by the MNTD meta classifier drops from 67.08($\pm$ 20.49) for static backdoors to 3.05($\pm$ 0.82), 0.54($\pm$ 0.83), and 1.47($\pm$ 0.87) for the random backdoor, BaN, and cBaN backdoors. This significant reduction of scores (with at least a factor of 22$\times$) demonstrates the advantage of our techniques compared to the static ones. In this setting, each meta-classifier would output a score, then based on a threshold, the decision if the model is backdoored or not is made [61]. We use the default threshold (median of training models' scores), which results in 98%, 74%, and 98% accuracy for the random backdoor, BaN, and cBaN techniques, respectively. This percentage corresponds to the number of meta-classifiers correctly classifying the model as a backdoored one. To improve the stealthiness of our backdoored models, we add a discriminator when training the models, aiming to lower the score predicted by the MNTD meta classifier. More concretely, we train a local meta-classifier (with a disjoint dataset compared to the one used for evaluation) and use it as our discriminator. We demonstrate this with the cBaN technique; however, it can be easily extended to the other two techniques. Using this technique, our results are significantly improved, i.e., only a single meta-classifier out of the 50 classified the model as a backdoored one. In other words, the detection accuracy is dropped to 2%, with a negligible performance drop, i.e., the ASR and utility dropped by less than 1%.

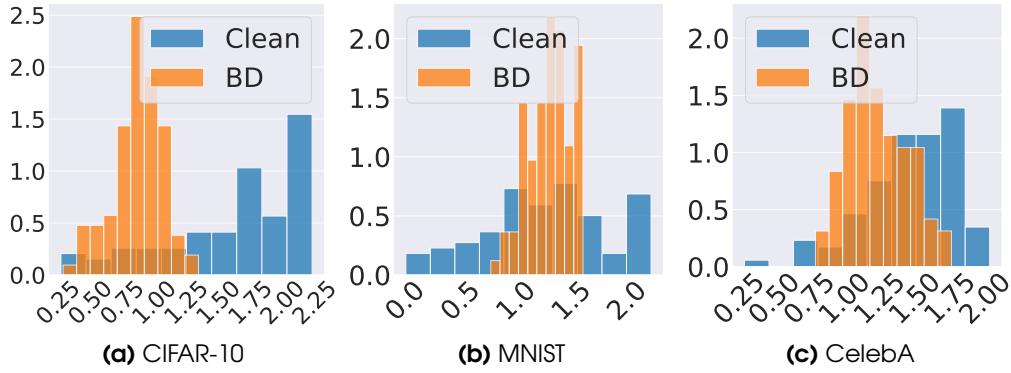
This again demonstrates that our dynamic backdoor techniques are more stealthy than the static ones. Moreover, they can be easily adapted to bypass backdoor defenses, e.g., by adding the corresponding discriminator as mentioned in Section 5.3.3.

Data-based Defense: Next, we evaluate some of the current state-of-the-art data-based defenses. Namely, we start by evaluating STRIP [45], then Februus [33].

STRIP tries to identify if a given input is clean or contains a trigger. It works by creating multiple images from the input image by fusing it with multiple clean images one at a time. Then STRIP applies all fused images to the target model and calculates the entropy of predicted labels. Backdoored inputs tend to have lower entropy compared to clean ones.

We use all of our three datasets to evaluate the c-BaN models against this defense. First, we scale the patterns by half while training the backdoored models, to make them more susceptible to changes. Second, for the MNIST dataset, we move the possible locations to the middle of the image to overlap with the image content, since the value of the MNIST images at the corners are always 0. All trained scaled backdoored models achieve similar performance to the non-scaled backdoored models.

Our backdoored models successfully flatten the distribution of entropy for the backdoored data, for a subset of target labels. In other words, the distribution of



(a) CIFAR-10 **(b)** MNIST **(c)** CelebA

Figure 5.11: The histogram of the entropy of the backdoored vs clean input, for our best performing labels against the STRIP defense, for the CIFAR-10 (Figure 5.11a), MNIST (Figure 5.11b), and CelebA (Figure 5.11c) datasets.

entropy for our backdoored data overlaps with the distributions of entropy of the clean data. This subset of target labels makes picking a threshold to identify backdoored data from clean data impossible without increasing the false positive rate, i.e., various clean images will be detected as backdoored ones. We visualize the entropy of our best performing labels against the STRIP defense in Figure 5.11. Moreover, since our dynamic backdoors can generate dynamic triggers for the same input and target label, the adversary can keep querying the target model while backdooring the input with a freshly generated trigger until the model accepts it.

Next, we evaluate Februus. Intuitively, Februus first detects the trigger from backdoored samples before removing it and patching the image. To detect these triggers, Februus first uses GradCAM to identify the influential region on the input. Then based on a security hyperparameter –which is dependent on the underlying task–, it decides if this area is to be removed and replaced by a neutral color. Finally, Februus develops a GAN-based inpainting technique to restore the image before querying it to the target model.

As the training code of Februus is not public yet, we only use CIFAR-10 – since it is the only dataset we consider that has its Februus models available – to evaluate against our different backdoor techniques.

Our results show that Februus only succeeds in dropping the ASR of our random backdoor, BaN, and cBaN backdoored models from 100% to approximately 80.5%, 81.7%, and 72%, respectively. This demonstrates the strong performance of our attack against the data-based defenses, especially compared to the static backdoored – whose ASR drops to 0.25% when applying Februus–.

These results against the data and model-based defenses show the effectiveness of our dynamic backdoor attacks, and opens the door for designing backdoor detection systems that work against both static and dynamic backdoors.

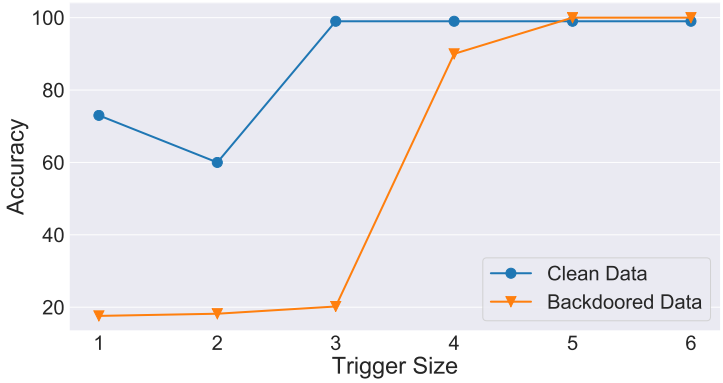


Figure 5.12: The result of trying different trigger sizes for the c-BaN technique on the MNIST dataset. The figure shows for each trigger size the accuracy on the clean and backdoored testing datasets.

5.4.6 Evaluating Different Hyperparameters

We now evaluate the effect of different hyperparameters for our dynamic backdooring techniques. We start by evaluating the percentage of the backdoored data needed to implement a dynamic backdoor into the model. Then, we evaluate the effect of increasing the size of the location set $\mathcal{K}$. Finally, we evaluate the size of the trigger and the possibility of making it more transparent, i.e., instead of replacing the original values in the input with the backdoor, we fuse them.

Proportion of the Backdoored Data: We start by evaluating the percentage of backdoored data needed to implement a dynamic backdoor in the model. We use the MNIST dataset and the c-BaN technique to perform the evaluation. First, we construct different training datasets with different percentages of backdoored data. More concretely, we try all proportions from 10% to 50%, with a step of 10. In this setting, 10% means that 10% of the data is backdoored, and 90% is clean. Our results show that using 30% is already enough to get a perfectly working dynamic backdoor, i.e., the model has a similar performance like a clean model on the clean dataset (99% accuracy), and 100% backdoor success rate on the backdoored dataset. For any percentage below 30%, the accuracy of the model on clean data is still the same, however, the performance on the backdoored dataset starts degrading. This demonstrates the ability of the adversary to implement dynamic backdoor attacks with 30% overhead for each target label, compared to training a clean model.

Number of Locations: Second, we explore the effect of increasing the size of the set of possible locations ($\mathcal{K}$) for the c-BaN technique. We use the CIFAR-10 dataset to train a backdoored model using the c-BaN technique, but with more than double the size of $\mathcal{K}$, i.e., 8 locations. The trained model achieves similar performance on the clean (92%) and backdoored (100%) datasets. We then doubled the size again to have 16 possible locations in $\mathcal{K}$, and the model again achieves the same results on both clean and backdoored datasets. We repeat the experiment with the CelebA datasets and achieve similar results, i.e., the performance of the model with a larger set of possible

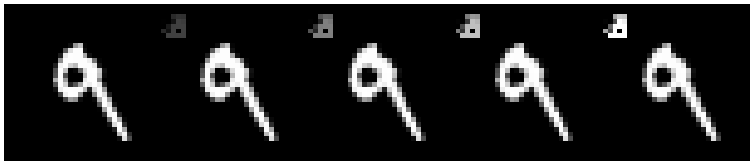


Figure 5.13: An illustration of the effect of using different transparency scales (from 0 to 1 with a step of 0.25) when adding the trigger. Scale 0 (the most left image) shows the original input, and scale 1 (the most right image) the original backdoored input without any transparency.

locations is similar to the previously reported one. However, when we try to completely remove the location set $\mathcal{K}$ and consider all possible locations with a sliding window, the performance on both clean and backdoored datasets drops significantly.

Trigger Size: Next, we evaluate the effect of the trigger size on our c-BaN technique using the MNIST dataset. We train different models with the c-BaN technique, while setting the trigger size from 1 to 6. We define the trigger size to be the width and height of the trigger. For instance, a trigger size of 3 means that the trigger is 3×3 pixels.

We calculate the accuracy on the clean and backdoored testing datasets for each trigger size, and show our results in Figure 5.12. Our results show that the smaller the trigger, the harder it is for the model to implement the backdoor behaviour. Moreover, small triggers confuse the model, which results in reducing the model's utility. As Figure 5.12 shows, a trigger with size 5 achieves a perfect accuracy (100%) on the backdoored testing dataset, while preserving the accuracy on the clean testing dataset (99%).

Transparency of the Triggers: Finally, we evaluate the effect of making the trigger more transparent. More specifically, we change the backdoor adding function $\mathcal{A}$ to apply a weighted sum, instead of replacing the original input's values. Abstractly, we define the weighted sum of the trigger and the image as: $\mathbf{x}_{bd} = s \cdot t + (1 - s) \cdot \mathbf{x}$, where s is the scale controlling the transparency rate, $\mathbf{x}$ is the input and t is the trigger. We implement this weighted sum only at the location of the trigger, while maintaining the remaining of the input unchanged.

We use the MNIST dataset and c-BaN technique to evaluate the scale from 0 to 1, with a step of 0.25. Figure 5.13 visualizes the effect of varying the scale when adding a trigger to an input.

Our results show that our technique can achieve the same performance on both the clean (99%) and backdoored (100%) testing datasets, when setting the scale to 0.5 or higher. However, when the scale is set below 0.5, the performance starts degrading on the backdoored dataset but stays the same on the clean dataset. We repeat the same experiments for the CelebA and CIFAR-10 datasets and find similar results.

We believe that the transparency of our triggers can be further increased when using triggers with larger sizes. To this end, we use the CIFAR-10 dataset to repeat the experiments previously mentioned in this section. However, we set the trigger size to be the size of the image. Our experiments show that in this setting, our dynamic backdoor attacks can still achieve a perfect attack success rate (100%) with a negligible

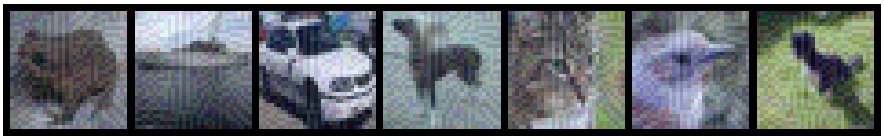


Figure 5.14: Visualization of the c-BaN backdoored images when setting the transparency scale to 0.1.

drop in utility (0.3%) when setting the scale to 0.1. More concretely, the model's accuracy on clean data is 91.7% compared to the 92% accuracy of the backdoored model trained without any transparency. We visualize a set of randomly backdoored samples in Figure 5.14. As the figure shows, setting the scale to 0.1 makes the triggers hardly visible.

5.4.7 Relaxing the Threat Model (Transferability of the Triggers)

For our dynamic backdoor attacks, we assume the adversary to control the training of the target model. We now relax this assumption by only allowing them to poison the dataset.

First, it is important to mention that our Random Backdoor technique does not need to change the training of the target model, i.e., the adversary only needs to poison the training dataset with backdoored images and the corresponding target labels. Second, for both the BaN and c-BaN techniques, the adversary can rely on pre-trained BaN and c-BaN models instead of training them jointly with the target model. In detail, the adversary uses the pre-trained BaN and c-BaN model to generate multiple triggers and randomly place them to a set of – randomly picked – images. Then, they poison the training set with this set of backdoored images and their corresponding target labels.

We use the MNIST dataset for evaluation and follow the same target models' structure as previously introduced in Section 5.4.3 and Section 5.4.4. However, to show the flexibility of our techniques, we use data from different distributions to pre-train the BaN and c-BaN models. We first use the CIFAR-10 dataset to train backdoored models with the BaN (Section 5.3.2) and c-BaN (Section 5.3.3) techniques. Next, we use the pre-trained BaN and c-BaN models to generate the backdoored dataset and poison the target dataset as previously mentioned. It is important to mention that the CIFAR-10 based BaN and c-BaN models generate 3-channel triggers, to use them to poison the MNIST dataset, we convert them to 1-channel triggers by taking the mean over the different channels. Finally, we use the poisoned dataset to train the target model.

As expected, the backdoored models achieve a perfect attack success rate (100%), while keeping the same utility as the backdoored models jointly trained with the BaN and c-BaN. This shows the flexibility of our attacks, i.e., the training procedure can be adapted by the adversary depending on their specific application. However, it is important to mention that jointly training the models has the advantage of giving the adversary more power, e.g., they can add a customized loss function to the target model while implanting the backdoor.

Finally, as a side-effect of transferring the BaN and c-BaN; The poisoning rate for the dynamic backdoor can now be lowered to about 10%, as there is no joint models

trained with the target model anymore.

5.4.8 Possible Defenses

In this section, we propose potential defenses against dynamic backdoor attacks, starting with a denoising mechanism to filter out triggers, which can be viewed as anomalies or distortions in the input. Specifically, we employ autoencoders, a common denoising technique. The process works as follows: we first train an autoencoder on clean data, then use it to reconstruct or denoise inputs by encoding and decoding them. Triggers are expected to be filtered out for two main reasons: the autoencoder's tendency to overfit to clean data and the inherent lossy nature of the reconstruction process.

To implement our defense, all inputs are passed through the autoencoder before being fed into the target model. The autoencoder is expected to remove the trigger from backdoored inputs without significantly altering clean ones.

We evaluate this defense against the c-BaN technique using the MNIST and CIFAR-10 datasets. In simpler datasets like MNIST, the defense performs well, successfully mitigating the backdoor attack with negligible utility loss (less than 1%). However, on more complex datasets like CIFAR-10, the defense's performance degrades due to the increased level of detail, making it harder to accurately reconstruct both clean and backdoored inputs. For instance, the accuracy of the target model drops by 4.8% for clean data and 25% for backdoored data.

Another defense approach could involve calculating the distance between the reconstructed and original inputs, then deciding whether to forward the input based on a predetermined threshold. We plan to explore this approach and other potential defenses in future work.

We also investigate data augmentation as a defense, focusing on the effects of resizing, cropping, and flipping target images on dynamic backdoor attacks. Using the CIFAR-10 dataset, we assess how these augmentations affect both the attack success rate (ASR) and model utility in our simplest (random backdoor with a single target label) and most complex (cBaN with all target labels) scenarios. First, flipping the input reduces ASR to 88.6% for cBaN and 93.4% for the random backdoor, with minimal impact on utility, demonstrating that our attacks are resilient to flipping. Second, resizing the image to 16×16 before scaling it back to 32×32 drops the ASR to 57.4% and 66.5%, but at the cost of a 15.4% and 15.9% utility loss, respectively. Lastly, cropping, where we pad the input with 4-pixel borders and crop back to the original size, reduces the ASR to 73.2% and 89.2%, with accuracy dropping by only 0.7% and 0.25%.

We next include the three data augmentation techniques in the training of the models and repeat the same experiments, i.e., testing the effect of applying each data augmentation separately at the inference time. We observe that the results did not differ significantly from the previous set of experiments; hence we plot the result in Appendix A.6.

Overall, while data augmentation can reduce the effectiveness of dynamic backdoor attacks, it does not completely prevent them and often results in a significant utility drop. In other words, our attacks remain viable, albeit with a reduced ASR when different augmentation techniques are applied.

5.5 Conclusion

The tremendous progress of machine learning has led to its adoption in multiple critical real-world applications. However, it has been shown that ML models are vulnerable to various types of security and privacy attacks. In this chapter, we focus on backdoor attacks where an adversary manipulates the training of the model to intentionally misclassify any input with an added trigger.

Current backdoor attacks only consider static triggers in terms of patterns and locations. In this chapter, we propose the first set of dynamic backdoor attacks against deep neural networks (DNN) models, where the trigger can have multiple patterns and locations. To this end, we propose three different techniques.

Our first technique Random Backdoor samples triggers from a uniform distribution and places them at random locations of an input. For the second technique, i.e., Backdoor Generating Network (BaN), we propose a novel generative network to construct triggers. Finally, we introduce conditional Backdoor Generating Network (c-BaN) to generate label-specific triggers.

We evaluate our techniques using three benchmark datasets. The evaluation shows that all our techniques can achieve almost a perfect backdoor success rate while preserving the model's utility. Moreover, we show that our techniques successfully bypass state-of-the-art defense mechanisms against backdoor attacks.

6

Understanding Data Importance in Machine Learning Attacks

6.1 Introduction

Data lies at the core of machine learning, serving as the foundation for training models and shaping their performance and accuracy. However, as highlighted in previous chapters, this essential role also introduces critical vulnerabilities. Data can become a target in two major ways: first, through exposing sensitive information about its membership status, leading to membership inference attacks (Chapter 3), and second, by being manipulated to carry out attacks like data poisoning (Chapter 4 and Chapter 5).

In this chapter, we delve deeper into the central role of data in the context of security threats. Specifically, how data with different characteristics influences ML attacks. Recent research highlights the heterogeneous impact of individual data samples, revealing that certain data points disproportionately influence the utility and effectiveness of machine learning models [62, 47, 68, 66, 63]. Understanding this variability is important for two main reasons. First, knowing how individual data samples affect model performance is key to improving machine learning explainability, offering new insights into model behavior, and enhancing interpretability [66, 109, 85, 103]. Second, this knowledge can guide data trading practices, where the importance of data is a significant factor [14, 47].

Despite these advancements, the impact of diverse data on model leakage and security remains underexplored. Existing research predominantly concentrates on the models themselves; for example, studies [114, S2] suggest that overfitted models are more prone to membership inference attacks. Nevertheless, even within the same model, distinct data samples exhibit varying vulnerabilities to attacks. This prompts a crucial question: do these valuable data samples also exhibit an increased vulnerability to a spectrum of machine learning attacks? Understanding the differential vulnerability of data samples has significant practical implications. In medical diagnostics, for example, patient records with rare but highly indicative symptoms are considered high importance samples. Assessing whether these records are more prone to attacks is crucial, as breaches could lead to discrimination, higher insurance premiums, or other serious consequences for individuals.

6.1.1 Contributions

In this chapter, our focus on understanding the link between data importance and machine learning attacks. Our primary objective is to thoroughly investigate whether valuable data samples, which contribute significantly to the utility of machine learning models, are exposed to an elevated risk of exploitation by malicious actors.

To achieve our objectives, we focus on five distinct types of attacks, encompassing both training-time and testing-time attacks. The training-time attack we consider is the backdoor attack [50, 80, P3], while the testing-time attacks consist of membership inference attack [81, 125, 114, 31, 75], model stealing attack [142, 141, 116], attribute inference attack [91, 128], and data reconstruction attack [164, 156, 41]. For each of these attacks, we thoroughly analyze the behavior and impact on both high importance and low importance data samples, aiming to uncover any discernible differences.

Main Findings: Our research has yielded significant findings that shed light on the

heightened vulnerability of valuable data samples to privacy attacks. Specifically, our key findings are as follows:

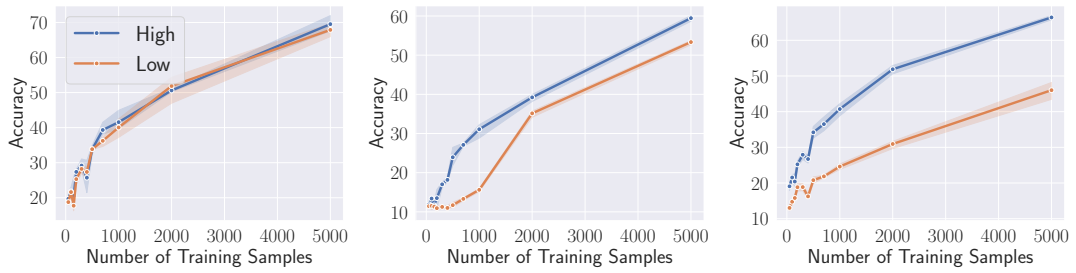
- Membership Inference Attack: High importance data samples exhibit a higher vulnerability compared to low importance samples, particularly in the low false-positive rate region. For instance, in the CIFAR10 dataset, at a false positive rate (FPR) of 1%, the true positive rate (TPR) of high importance data is 10.2× greater than that of low importance samples.
- Privacy Onion Effect: The concept of the privacy onion effect [24] can be extended to the distribution of data importance. Specifically, previously considered unimportant samples gain significance when the dataset removes the important samples.
- Model Stealing Attack: High importance samples demonstrate greater efficiency in stealing models when the target model is trained on the same distribution as the query distribution. However, we empirically demonstrate that the importance does not transfer between different tasks.
- Backdoor Attack: Poisoning high importance data enhances the efficiency of the poisoning process, particularly when the size of the poison is small. On the other hand, the influence on clean accuracy does not yield a definitive conclusion, poisoning either type of data has a limited impact on clean accuracy.
- Attribute Inference and Data Reconstruction Attacks: We observe no significant distinction between high and low importance data in these attacks.

Our research provides empirical evidence establishing a correlation between data importance and vulnerabilities across diverse attack scenarios. This introduces a novel perspective for analyzing sample-specific vulnerabilities, enriching our understanding of the security implications within the realm of machine learning.

Beyond theoretical insights, our study showcases practical applications of these findings, illustrating how they can be utilized to devise more potent attacks. On one hand, these findings can be utilized in a *passive* manner. For example, we empirically demonstrate that membership inference attacks can be improved by introducing sample-specific criteria based on sample importance. Additionally, adjusting the poisoning strategy according to sample importance proves to enhance the efficacy of backdoor attacks, particularly with a reduced poisoning rate.

More interestingly, we can *actively* modify samples to alter their importance, which subsequently impacts both attack and defense performance. For example, recognizing that high importance samples are more vulnerable to membership inference attacks, attackers could increase the importance of targeted samples to heighten their vulnerability. This approach follows exactly the same idea as the attack accepted at CCS'22 [140], effectively demonstrating how we can "reinvent" state-of-the-art attacks guided by findings in our work.

In summary, our work represents a pioneering step in systematically understanding the vulnerabilities of the machine learning ecosystem through the lens of data. These findings serve as a resounding call to action, urging researchers and practitioners to



(a) LOO **(b)** Trak **(c)** KNN-Shapley

Figure 6.1: LOO and Trak fail to capture the complex interactions between subsets of data, resulting in suboptimal sample importance identification performance.

develop innovative defenses that strike a delicate balance between maximizing utility and safeguarding valuable data against malicious exploitation.

6.2 Evaluation Setup

In this work, we deploy KNN-Shapley [62] to assess the importance of samples in the training set, which takes a dataset as input and assigns an importance value to each sample in the dataset. This decision is justified by two main considerations.

Firstly, from the perspective of utility, traditional data attribution methods struggle to account for the complex interactions within data subsets. Previous research, as discussed by Gupta and Zou [47], highlights this limitation. Consequently, we adopt Shapley value-based approaches for a more accurate assessment of data importance. We further examine the efficacy of two non-Shapley-based measurement techniques: Leave-one-out (LOO) and the advanced data attribution method, Trak[1] [103]. As evidenced in Figure 6.1, when comparing the performance of models trained with 5000 samples of varying significance, the accuracy discrepancy is less than 7% for these methods. In contrast, the KNN-Shapley method identifies samples that yield an accuracy difference exceeding 20%, thereby demonstrating its superior capability in accurately quantifying importance.

Secondly, regarding scalability, most measurement methods are highly computationally inefficient. For instance, employing Leave-one-out to calculate importance values for CIFAR10 necessitates over 80 hours on 8×A100 GPUs. Shapley value-based methods are generally *more demanding*. In Jia et al.'s work [63], the authors provide a runtime demonstration showing that existing measurement methods, except for KNN-Shapley, do not scale efficiently to large datasets, even as CIFAR10. We defer the details to Appendix A.7.

Furthermore, the comprehensive evaluations conducted by prior studies [63] consistently underscore the effectiveness and accuracy of KNN-Shapley. Therefore, considering

[1]Trak quantifies the influence of each sample on specific test samples within a dataset. To adhere to the established definition of data importance, we calculate the average influence exerted by each sample across the entire test dataset as the importance of each sample.

both utility and scalability, KNN-Shapley emerges as the sole feasible method for conducting experiments.

Datasets: Our evaluation encompasses three widely-used benchmark datasets, namely CIFAR10 [9], CelebA [84], and TinyImageNet [10]. CIFAR10 comprises a collection of 60,000 colored images evenly distributed across ten classes, representing common objects encountered in everyday life, including airplanes, birds, and dogs. CelebA is a large-scale face dataset that encompasses over 40 annotated binary attributes. To ensure balance in our analysis, we follow previous works [98, P3, 167, S2] that select the three most balanced attributes (Heavy Makeup, Mouth Slightly Open, and Smiling) to create an 8-class (2^3) classification task. Note that our findings are not dependent on this specific attribute selection, as validated in Appendix A.10. Additionally, our evaluation incorporates TinyImageNet, which constitutes a subset of the ImageNet dataset. It encompasses 200 distinct object classes, each with 500 training images. We further validate the generalizability of our conclusions across different modalities, with detailed information deferred to Section 6.8.

6.2.1 Learning Characteristic

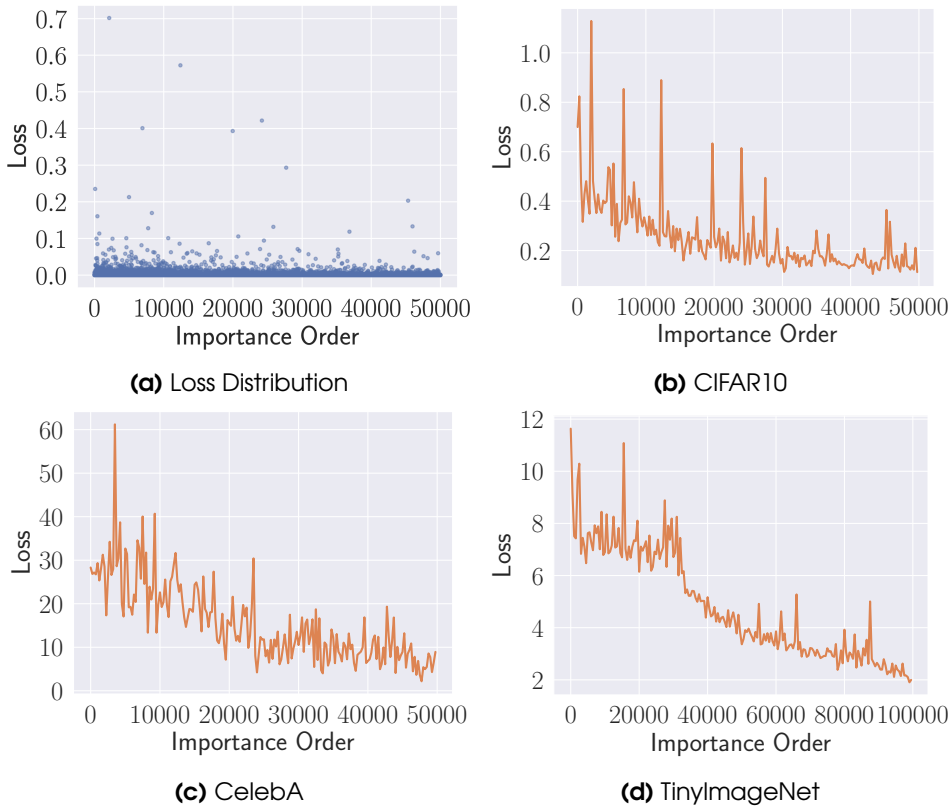
In order to gain a deep understanding of the disparities between high and low importance data, we delve into the learning characteristics, such as loss, associated with these samples. To the best of our knowledge, our study represents the first endeavor to investigate the learning characteristics of samples with varying degrees of importance, diverging from the conventional focus solely on their contribution to the final performance.

To quantify these learning characteristics, we initially train a model using the complete dataset, comprising both high and low importance data. Subsequently, we compute the loss for each individual data point and explore the correlation between the loss and its corresponding importance value.

In Figure 6.2a, we present a visual representation of the relationship between loss and importance value. The x-axis represents the importance order of a sample in the dataset, with 1 denoting the lowest importance and 50000 representing the most valuable data. Initially, it may seem that there is no discernible pattern between loss and importance value, as both low and high importance samples can exhibit either low or high loss. However, upon further analysis, we statistically observe that higher importance samples tend to demonstrate lower loss, as depicted in Figure 6.2b, Figure 6.2c, and Figure 6.2d.

To arrive at this conclusion, we divide the samples into 200 bins based on their importance value. For instance, the lowest 1 to 250 samples are categorized into bin one, 251 to 500 are allocated to bin two, and so forth. For each bin, we calculate the sum of the losses and plot these 200 data points to generate the final curve. Despite some fluctuations observed in the curve, it is evident that valuable samples tend to exhibit lower loss. This finding aligns with our expectations, as lower loss signifies greater representativeness, thereby facilitating easier learning and enhancing their contribution to the overall utility of the model.

Having established the effectiveness of importance assignment and gained preliminary insights into the learning characteristics, we proceed to conduct representative



(a) Loss Distribution **(b)** CIFAR10

(c) CelebA **(d)** TinyImageNet

Figure 6.2: Relationship between loss and importance value. Low importance samples statistically have higher losses.

machine learning attacks to investigate the impact of data importance in such attacks. Our experimental investigations are carried out using the ResNet18 architecture, and in Section 6.8, we demonstrate the generalizability of our conclusions to different architectures.

6.3 Membership Inference Attack

Membership Inference Attack (MIA) [81, 125, 114, 31, 75, 154] is a prominent privacy attack utilized to determine whether a specific data sample belongs to a training dataset. This attack is widely employed to assess the privacy of training data due to its simplicity and broad applicability.

In the attack scenario, the adversary $\mathcal{A}$ is granted access to a target model and is tasked with determining the membership status of a given data sample (x, y). Formally, the membership inference attack can be defined as a security game, referred to as Membership Inference Security Game, which is described as follows:

Requirement 1 (Membership Inference Security Game [21])**.** *The game proceeds between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:*

1. *The challenger samples a training dataset* $D \leftarrow \mathbb{D}$ *and trains a model* $f_\theta \leftarrow \mathcal{T}(D)$ *on the dataset* D.

2. *The challenger flips a bit* b, *and if* $b = 0$, *samples a fresh challenge point from the distribution* $(x, y) \leftarrow \mathbb{D}$ *(such that* $(x, y) \notin D$*). Otherwise, the challenger selects a point from the training set* $(x, y) \xleftarrow{\$} D$.

3. *The challenger sends* (x, y) *to the adversary.*

4. *The adversary gets query access to the distribution* $\mathbb{D}$, *and to the model* f_θ, *and outputs a bit* $\hat{b} \leftarrow \mathcal{A}^{\mathbb{D},f}(x, y)$.

5. *Output 1 if* $\hat{b} = b$, *and 0 otherwise.*

The adversary $\mathcal{A}$ is provided with auxiliary information about the data distribution $\mathbb{D}$. This allows the adversary to sample a shadow dataset from the same or a similar distribution, which is a common assumption in the existing literature.

The attack accuracy for the adversary is defined as follows:

$$Acc = \Pr_{x,y,f,b}[\mathcal{A}^{\mathbb{D},f}(x, y) = b].$$

To assess the privacy leakage caused by membership inference attacks (MIAs), we employ two metrics commonly used in prior research, focusing on both worst-case and average-case performance:

1. **(Log-scale) ROC Analysis** [21], which focuses on the true-positive rate at low false-positive rates, effectively capturing the worst-case privacy vulnerabilities of machine learning models.

2. **Membership Advantage** [155, 131], defined as

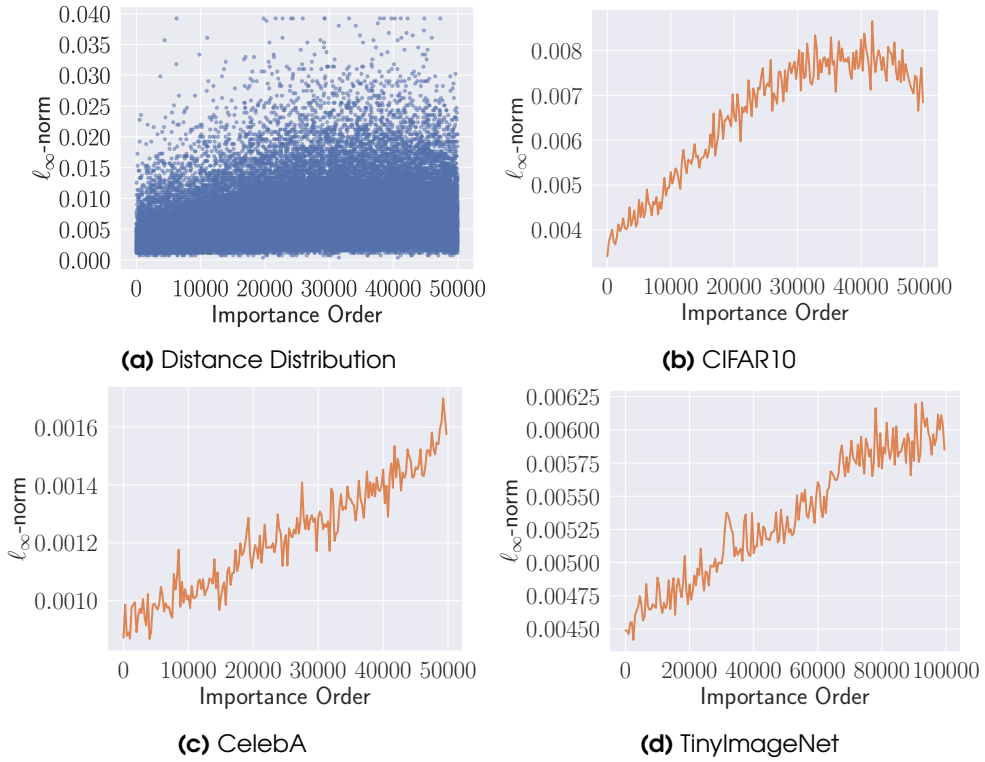
$$Adv = 2 \times (Acc - 0.5).$$

   This metric represents the advantage over random guessing, multiplied by 2, providing an average-case measure to gain an overview of the attack's efficacy.

In this work, we investigate four specific membership inference attacks. For the CIFAR10 and CelebA tasks, a training set of 50,000 samples is employed, while for the TinyImageNet task, we utilize a training set of 100,000 samples to construct the target model.

To assess the membership status of samples, we first adopt a methodology based on previous research [31, 75] that considers the distance to the decision boundary as a reflection of membership status. Specifically, they claim that samples located near the decision boundary are more likely to be non-members, whereas samples positioned in the central region of the decision area are more likely to be members.

We calculate the distance to the decision boundary for all samples in the training dataset. Specifically, for each sample, we iteratively perturb it using Projected Gradient Descent (PGD) with a small step size until it is classified into a different class. Subsequently, we compute the distance between the perturbed sample and its original



(a) Distance Distribution **(b)** CIFAR10

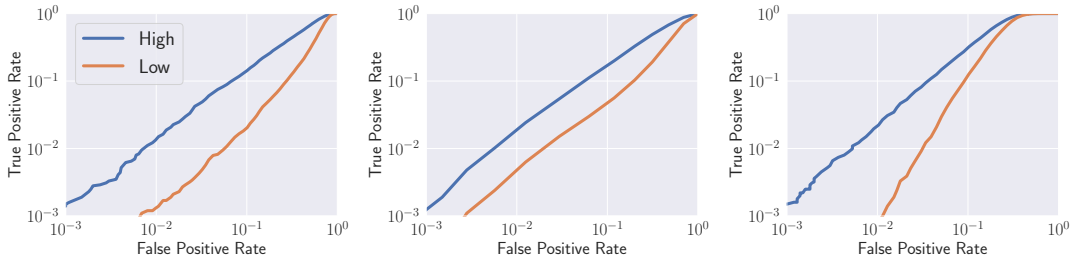
(c) CelebA **(d)** TinyImageNet

Figure 6.3: Relationship between distance to the decision boundary and importance value. Low importance samples are statistically closer to the decision boundary. The distance measured with different norms can be found in Appendix A.11.

counterpart. In this analysis, the distance is measured using the ℓ_∞ norm, and we find consistent results across different norms such as ℓ_1 and ℓ_2, as evidenced by the corresponding findings presented in Appendix A.11.

Our initial visualization focuses on examining the distances of samples with different importance values. Similar to the observation made regarding the distribution of loss values in Section 6.2.1, no direct relationship is discernible between the importance value and the distance to the decision boundary. Notably, samples with similar importance values may exhibit substantial differences in their distances to the decision boundary. In contrast, we further analyze the statistical characteristics of these samples, as performed in Section 6.2.1 and present the "group distance" in Figure 6.3b, Figure 6.3c, and Figure 6.3d. The results reveal that low importance samples are statistically closer to the decision boundary, which aligns with the previous conclusion that low importance samples tend to have higher loss compared to high importance samples.

We follow the same procedure to derive the distance for samples in the testing dataset and launch the membership inference attack based on the distance. We identify 10,000 samples with the highest importance value as the high group, and an equivalent number of samples with the lowest value as the low group. The resulting ROC curves, depicted in Figure 6.4, are presented on a logarithmic scale to compare the performance between these two groups.



(a) CIFAR10 **(b)** CelebA **(c)** TinyImageNet

Figure 6.4: Log-scale ROC curve: membership inference attack based on the distance to the decision boundary. High importance samples exhibited substantially higher true-positive rates, particularly in the low false-positive rate region. Results with different norms can be found in Appendix A.11.

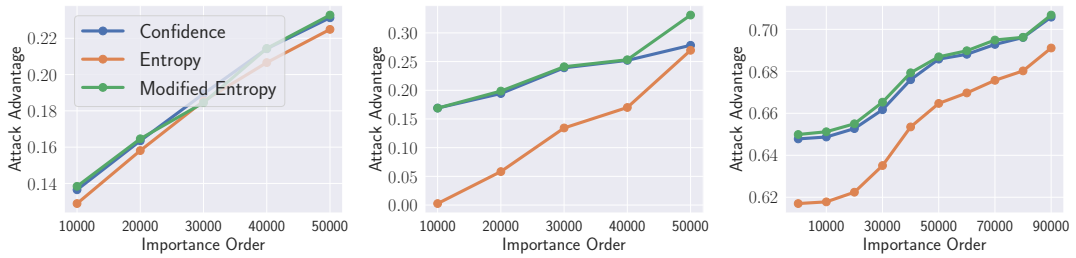
From the figures, we observe significant differences in the behavior of high importance samples and low importance samples, particularly in the low false-positive rate area. Specifically, for the CIFAR10 dataset, high importance samples demonstrate a true-positive rate (TPR) 10.2× higher than low importance samples at a low false-positive rate of 1%. For the TinyImageNet dataset, the difference is even more pronounced, with high importance samples exhibiting a TPR that is 27.9× higher than that of low importance samples at the same false-positive rate.

These observations provide compelling and empirical evidence supporting the notion that high importance samples are considerably more vulnerable to membership inference attacks, which satisfies our expectation as the importance of data samples can be regarded as the proxy of memorization [37]. These findings thus pose a significant and tangible threat to the safeguarding of high importance data privacy. On the other hand, these findings may also prompt researchers to consider adopting strategic sampling methods for more effective privacy auditing [96, 58].

We further validate the generalizability of this finding across various attack methodologies by conducting experiments with three additional metric-based attacks: *prediction confidence-based* attack [155, 131], *entropy-based* attack [125, 114], and *modified prediction entropy-based* attack [130]. The first two attacks were enhanced by introducing class-dependent thresholds, as demonstrated by Song and Mittal [130].

By grouping the samples based on their importance values in intervals of 10,000 samples (equivalent to the size of the testing dataset), we conducted the aforementioned attacks on these subsets. The membership advantage achieved for each subset is illustrated in Figure 6.5. Notably, a clear monotonic increase in attack advantage is observed as the importance value increases, establishing a positive correlation between the importance value and the susceptibility of membership inference.

This empirical trend aligns with our expectations. As evidenced by the metrics in Figure 6.2 and Figure 6.3, samples with lower importance inherently present greater learning challenges compared to their higher-importance counterparts. Even post-learning, these samples exhibit worse membership metrics compared to those of higher importance. This circumstance renders them challenging to distinguish from non-member samples, especially when certain non-member samples manifest a lower learning



(a) CIFAR10 **(b)** CelebA **(c)** TinyImageNet

Figure 6.5: Membership advantage: membership inference attack based on three metrics. Attack advantage steadily escalates as the importance value of the samples increases.

difficulty and consequently exhibit better metrics compared to the more challenging member samples.

Drawing from this insight, one potential strategy to enhance the efficiency of membership inference attacks is to compare each sample with others of comparable difficulty. A pragmatic approach to actualize this entails introducing sample-specific criteria. Rather than employing a uniform threshold across the entire testing dataset, such criteria should intricately correlate with the sample's characteristics, with their designated importance level serving as a robust quantitative index to reflect this alignment.

In this study, we initiate an exploration into the feasibility of such an approach. To seamlessly integrate our method into the existing metric-based framework, we introduce a sample-specific threshold in a consistent manner: while maintaining a uniform threshold for the dataset, we modify the membership metrics by incorporating an importance-related term:

$$\texttt{CaliMem}(x) = \texttt{OriMem}(x) + k \times \texttt{Shapley}(x)$$

Here, $\texttt{OriMem}(x)$ denotes the conventional membership metric, including elements such as confidence, entropy, and modified entropy. The term $\texttt{Shapley}(x)$ signifies the importance value attributed to the specific sample x, and $\texttt{CaliMem}(x)$ represents the recalibrated membership metric.

As a proof of concept, we empirically determine the hyperparameter k in an exploratory manner, adjusting its magnitude until optimal performance is attained. The experimental outcomes, depicted in Figure 6.6, illustrate that the incorporation of importance calibration notably enhances the efficacy of metric-based attacks. Nevertheless, it is pertinent to acknowledge that identifying the optimal hyperparameter and devising more refined methods for integrating importance values warrant further investigation.

We also emphasize that this improvement does not necessitate additional requirements compared to standard attacks; specifically, the adversary does not need full access to the training dataset to obtain importance values. Although importance values cannot be calculated for single samples, most membership inference attacks assume access to a shadow dataset. We validate the feasibility of our approach using a shadow dataset. Specifically, we randomly select 1,000 samples from the CIFAR10 dataset to calculate

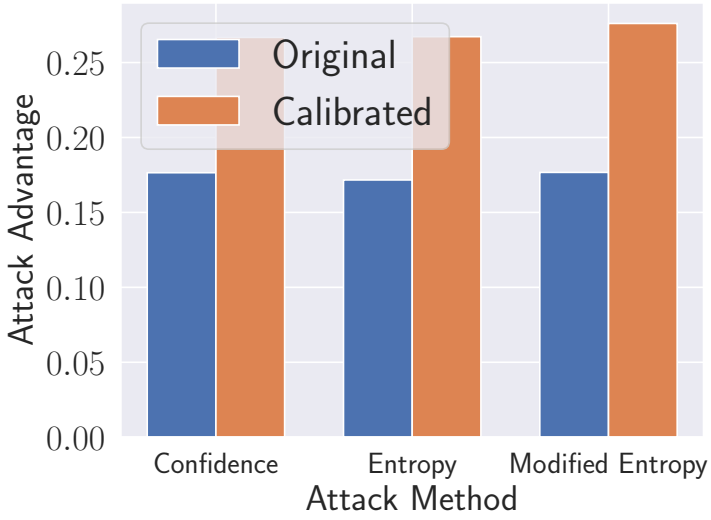


Figure 6.6: Incorporation of importance values in calibrating membership inference metrics improves the attack performance, demonstrating the strength of employing sample-specific membership criteria.

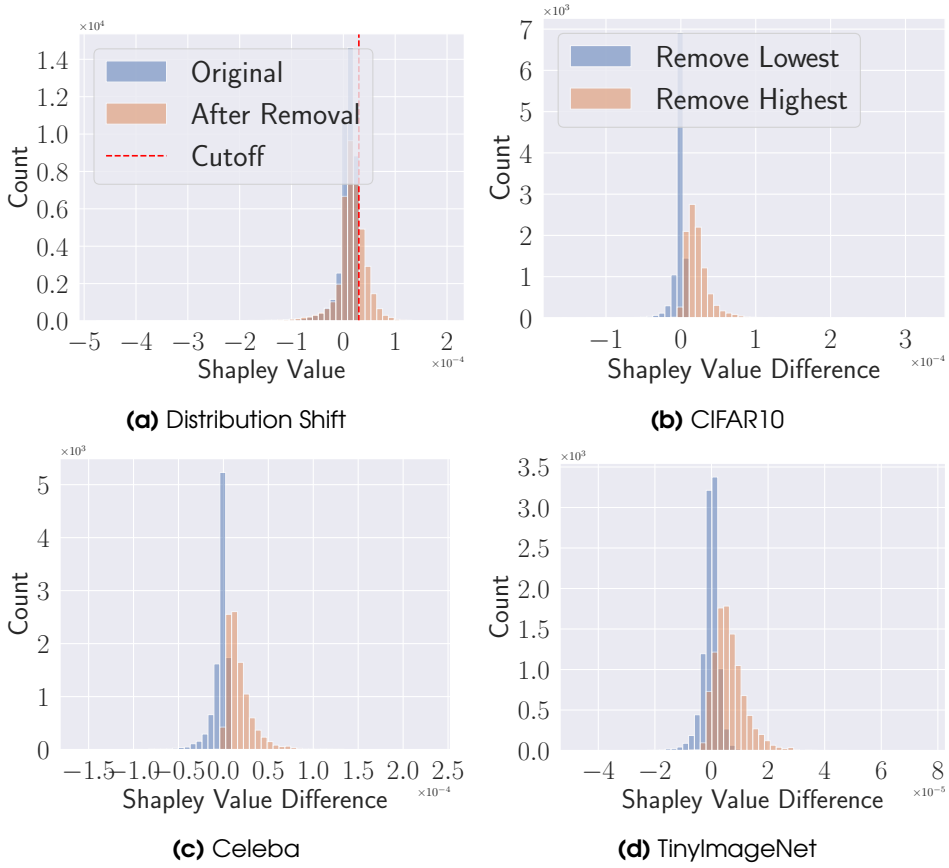
their importance value, we first calculate the importance value for all samples using the whole CIFAR10 dataset as the ground truth. Then we assume the adversary can only access a shadow dataset containing 10,000 samples, and calculate the importance value for each sample using only the shadow dataset. We found a correlation coefficient of 0.957 between these values, indicating that using the shadow dataset could provide a good approximation.

6.3.1 Privacy Onion Effect

Carlini et al. [24] have identified the onion effect of memorization, which refers to the phenomenon wherein "*removing the layer of outlier points that are most vulnerable to a privacy attack exposes a new layer of previously-safe points to the same attack.*" Their research demonstrates this effect by removing samples that are at the highest risk of being compromised through membership inference, resulting in formerly safe samples becoming vulnerable to the attack.

Building upon the insights from the preceding section, our empirical findings confirm a positive correlation between membership inference vulnerability and data importance. This prompts an intriguing question: Does this effect reflect in the importance values assigned to the data? Put differently, when high importance samples are removed from a dataset, do previously designated low importance samples gain significance?

To avoid ambiguity, it is imperative to understand that the term "importance" in this context is not subjective or relative. The removal of high importance data points does not inherently increase the importance of those initially deemed low importance. Furthermore, it is conceivable for a dataset to exclusively consist of low importance samples. This clarification is indispensable; otherwise, the studied question may seem trivial.



(a) Distribution Shift **(b)** CIFAR10

(c) Celeba **(d)** TinyImageNet

Figure 6.7: The privacy onion effect can be extended to the importance distribution. For Figure 6.7a, we remove all examples with importance values larger than the red line. Samples that remain after removal have their importance value increase past the red line. Subsequent figures confirm that removing highly important samples elevates the significance of the remaining set, while the removal of less significant samples has no such effect, ruling out dataset size influence.

Our results indicate that removing important samples indeed makes samples previously considered unimportant gain importance. Specifically, upon removing 10,000 data points with the highest importance scores, we recalculated the importance for the remaining samples. As depicted in Figure 6.7a, this removal led to a noticeable redistribution in data point importance, with previously low important data points now being assigned greater significance.

However, a note of caution is warranted in interpreting this result. The removal of a substantial number of samples (10,000 in this context) might introduce a baseline drift. Therefore, attributing the observed importance augmentation solely to the exclusion of important samples might be premature. To further validate our findings, we executed controlled experiments wherein we systematically excluded either the most or least significant data points. This approach mitigates potential biases stemming from dataset size discrepancies. To emphasize the impact of these exclusions, we quantified the importance value discrepancies for data points ranked between 10,000 and 20,000 in

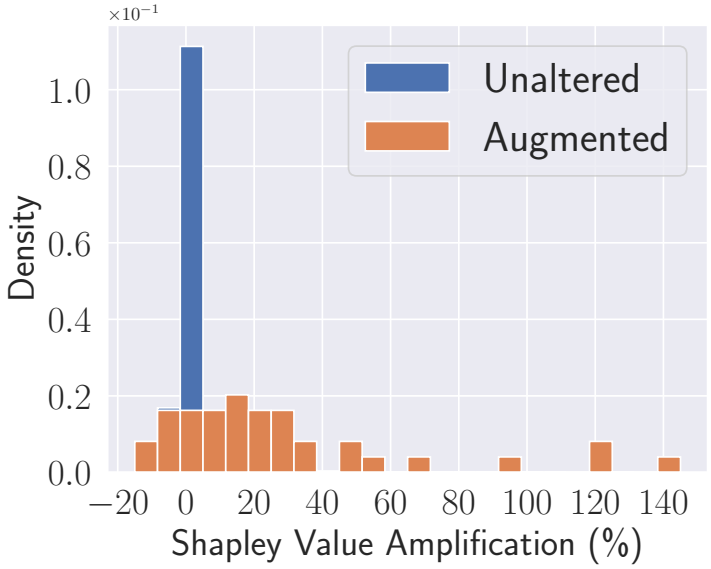


Figure 6.8: Impact of sample duplication on importance values. Following the duplication process (16 duplications per sample), 45 out of 50 target samples showed a marked increase in importance, averaging a 53.02% rise, while the importance of control samples (unduplicated) remained relatively stable.

descending order of importance in the original dataset, as they remained for both removal procedures.

Our results, as visualized in Figure 6.7b, Figure 6.7c, and Figure 6.7d, underscore the pronounced disparities in how the exclusion of high importance versus low importance data samples influences the remaining dataset's importance distribution. Using CIFAR10 as an illustrative case, removing the most significant data points caused 99.14% of the remaining data points to be reevaluated as more important. In contrast, removing the least significant data points led to a 45.88% decrease in importance for the affected data points. These findings robustly support our hypothesis that data points previously deemed of lesser importance assume greater significance when high importance data points are excluded, and such a conclusion cannot be attributed to dataset size variation.

6.3.2 Actively Modify Sample Importance

As discussed in the previous section, altering the dataset can influence the importance of samples. Given the observed linkage between membership vulnerability and importance value, an interesting question arises: can we actively use our findings to design more advanced attacks by modifying the importance of target samples?

However, directly altering sample importance is challenging due to the absence of a standardized method or framework. In this section, we explore an ad-hoc approach aimed at increasing the importance of target samples. Specifically, we select a set of target samples and duplicate each one multiple times with consistently incorrect labels. This strategy intuitively heightens the influence of these samples by causing the model to consider them as "outliers" due to the prevalence of incorrect duplicates. We tested

Figure 6.9: Comparison of importance values before and after replacing 1,000 samples with their augmented versions using ColorJitter, Grayscale, HorizontalFlip, and VerticalFlip techniques. The plot illustrates that augmentation caused variable changes in importance, with some samples gaining and others losing importance.

this idea by duplicating 50 target samples 16 times and reassessing their importance values. As shown in Figure 6.8, the duplicated samples generally exhibited an increase in importance. Specifically, 45 of the 50 target samples experienced an increase in importance, with an average increase of 53.02%, while the importance of the unaltered samples remained nearly constant.

This approach is exactly the membership poisoning attack proposed by Tramèr et al. [140], where they comprehensively demonstrated the method's efficiency. This indicates that increasing the importance of samples can be a practical approach to enhance their vulnerability. By revisiting their technique from a data importance perspective, we highlight that actively modifying sample importance could be a promising strategy for developing sophisticated attack techniques or formulating robust defenses.

6.3.3 Data Augmentation

In previous discussions, it has been demonstrated that manipulating data samples can alter their importance values and, consequently, their susceptibility to attacks. Given

Figure 6.10: Analysis of the impact on importance values when 1,000 augmented samples are added to the original dataset, compared to a baseline scenario where the same 1,000 samples were duplicated. The figure demonstrates that the presence of both original and augmented samples had minimal effect on the importance of the original samples, showing no significant differences from simple duplication.

that data augmentation is the most widely used method for data manipulation, it would be interesting to ask: does data augmentation affect the importance of a sample?

In our study, we examined the impact of four data augmentation techniques—ColorJitter, Grayscale, HorizontalFlip, and VerticalFlip—under two specific scenarios. In both scenarios, we selected 1,000 samples for augmentation while leaving the rest of the dataset unaltered:

Augmented Versions Only: In this scenario, we aimed to investigate how data augmentation impacts the importance of the augmented samples and the unaltered samples in the dataset. Specifically, we *replaced* 1,000 samples with their augmented versions, recalculated their importance values, and compared these values to the original. As illustrated in Figure 6.9, we found that the augmented versions had variable effects on importance: some samples gained higher importance, while others lost it. Overall, a slight majority of the samples experienced a decrease in importance following augmentation. However, the augmented samples had a negligible impact on the remaining non-augmented samples.

Original and Augmented Versions: In this scenario, we examined the effect of having both augmented and original versions of certain samples in the dataset. We *added* 1,000 augmented samples to the original dataset and recalculated the importance values for the original samples. To control for dataset size, we also considered a baseline case where the same 1,000 samples were duplicated. As shown in Figure 6.10, the presence of both original and augmented samples had minimal impact on the importance of the original samples, with no significant differences compared to simple duplication.

We acknowledge that more complex augmentation techniques, such as those using generative models, may have different effects. The exploration of these complex augmentation techniques remains an avenue for future research.

Takeaways: Our findings highlight the vulnerability of high importance samples to membership inference attacks. Significant differences were observed in the behavior of high importance and low importance samples, particularly in the low false-positive rate region, where high importance samples exhibited substantially higher true-positive rates. This emphasizes the necessity of addressing the privacy risks associated with high importance samples and implementing effective safeguards. Simultaneously, it encourages researchers to explore strategic sampling methods to enhance the effectiveness of privacy audits.

The observation also suggests a potential enhancement to membership inference attacks through the introduction of sample-specific criteria. We empirically validate the practicality of using importance values to calibrate membership metrics, thereby enhancing attack efficiency.

Moreover, our findings reveal the "privacy onion effect" within the sample importance distribution, where previously overlooked samples gain importance when key samples are removed. Furthermore, by revisiting an advanced membership poisoning attack from the perspective of data importance, we suggest that actively manipulating sample importance can be a potent strategy for developing sophisticated cybersecurity measures, both offensive and defensive, but finding general manipulating methods needs further investigation.

6.4 Model Stealing

Model stealing attack [142, 141, 116, 56, 23] differs from membership inference attack as it aims to compromise the confidentiality of the model itself rather than exploiting privacy information about training samples. This type of attack does not have information about the target model's architecture or parameters but seeks to create a surrogate model that emulates the functionality of the target model. Such attacks can be employed by adversaries for various purposes, including monetary gains or as a preliminary step for subsequent attacks [102].

The workflow of a model stealing attack is visualized in Figure 6.11. The adversary samples data from a specific distribution $\mathbb{D}$ and simultaneously queries the target and surrogate models. To ensure similarity between the surrogate and target models, the adversary optimizes the surrogate model to produce similar outputs $\mathcal{S}(x)$ as the target outputs $\mathcal{T}(x)$. While the attack approach is straightforward, selecting an appropriate

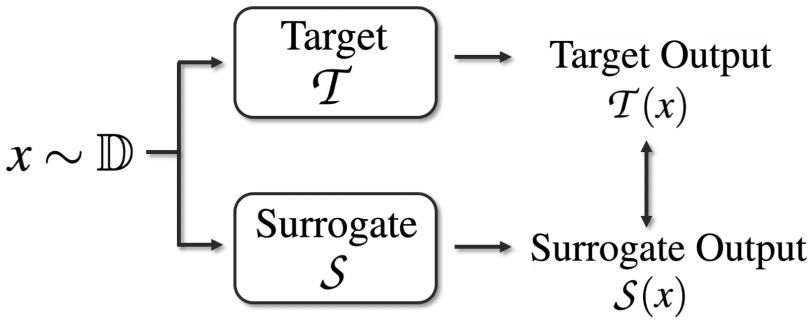


Figure 6.11: The workflow of model stealing attack, the adversary leverages the target model to guide the surrogate model.

query data distribution poses a challenge, as it directly impacts the stolen accuracy and query efficiency. Recent research has explored efficient and data-free methods for launching these attacks [100, 64, 116], yet the question of selecting high-quality samples when the target task is known remains intriguing.

In this work, we focus on the primary scenario where the adversary can query the target model to obtain corresponding posteriors, while having knowledge of the target task. Specifically, the adversary could query the model with a dataset from the same or a similar distribution. This scenario has practical applications, such as creating a surrogate model to facilitate further attacks or to save on labeling costs. We limit our discussion to this primary scenario and do not delve into more advanced model stealing techniques that focus on reducing the dataset assumption, as our interest lies in understanding how different data interact with the model stealing process.

Our goal is to investigate whether query samples with different importance values exhibit varying efficiency in stealing models. We explore two settings in our experiments. First, we launch the attack using query data from the same distribution as the target model trained on. For example, if the target model is trained on CIFAR10, we employ CIFAR10 data to query the model. The second scenario involves using data from different distributions, specifically CelebA and TinyImageNet, to query the CIFAR10 model. We choose accuracy and query budget as the metrics to evaluate the success of the attack, using less query budget to achieve higher accuracy denotes better attack performance.

6.4.1 Same Distribution Query

Three target models were trained using a standard training procedure, resulting in testing accuracies of 95.15% for CIFAR10, 79.05% for CelebA, and 65.01% for TinyImageNet. After training the target models, our attack solely interacts with the target models through their outputs without accessing or reading their parameters.

To initiate the attack, we establish a query budget ranging from 100 to 10,000. Once the query budget is determined, we prioritize collecting high importance data until the budget is exhausted, and the same principle applies to the collection of low importance data.

(a) CIFAR10 **(b)** CelebA **(c)** TinyImageNet

Figure 6.12: Model stealing attack that queried with data from the same distribution. High importance samples exhibit greater efficiency in stealing models when the target model is trained on the same distribution as the query distribution.

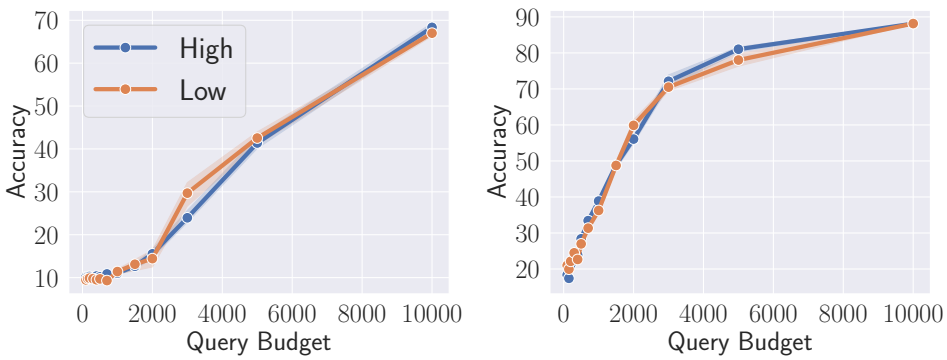
The attack results are illustrated in Figure 6.12, highlighting the superior efficiency of high importance samples in the model stealing process. For instance, when the query budget is set to 1000, high importance data steal a CIFAR10 model with 53.77% accuracy, which is 1.6× higher than the model stolen by low importance data (33.29%). This trend holds true for the other two datasets as well. Taking TinyImageNet as an example, when the query budget is 1000, high importance data yield a model accuracy of 19.25%, whereas low importance data only result in a model accuracy of 9.25%, exhibiting a notable 2.1-fold disparity.

One plausible explanation for this difference may arise from variations in class balance, given that the query sets are chosen based on sample importance. It is conceivable that the low importance query set may lack samples from certain classes, thereby resulting in suboptimal performance. Prior research has suggested that a more balanced data distribution could potentially improve model stealing performance [116, T3]. However, upon examining the data distribution for both high and low significance samples, we observed no significant disparities. For example, in the case of CIFAR10, the entropy values for the top-10,000 high importance and low importance distributions were 3.282 and 3.245, respectively. Even when considering 1000 samples, the corresponding entropy values were 3.161 (high importance) and 3.229 (low importance). For context, a perfectly uniform distribution has an entropy of 3.322. This indicates that both the high and low importance subsets closely approximate a uniform distribution. Such findings reinforce our assertion that high importance data can indeed augment model stealing performance, mitigating concerns related to distributional biases.

6.4.2 Different Distribution Query

The previous section highlighted the enhanced efficiency of high importance samples in stealing models trained on the same task. However, it remains uncertain whether this efficiency persists when the target model is trained using a different dataset or task, and whether importance values can be transferred across tasks.

To investigate, we conducted experiments involving query data that differed from the distribution used to train the target model. Specifically, we employed a CIFAR10 model as the target model and queried it with the CelebA and TinyImageNet datasets.



(a) CelebA **(b)** TinyImageNet

Figure 6.13: Model stealing attack that queried with data from different distributions. The target model is trained on the CIFAR10 task. Results show that importance does not transfer between different tasks.

Interestingly, as depicted in Figure 6.13, we observed that the advantage of high importance samples disappeared in this cross-task scenario. When the query budget was consistent, the stolen accuracy for both high and low importance samples was comparable. This suggests that samples deemed important for one task may not transfer effectively to arbitrary tasks.

Takeaways: Our findings demonstrate that high importance samples exhibit greater efficiency in stealing models when the target model is trained on the same distribution as the query distribution. Importantly, this enhanced efficiency cannot be solely attributed to distribution bias. This suggests that adversaries, when aware of the target task, can employ high importance samples to optimize attack performance with a reduced query budget. However, this conclusion does not hold when the target task differs from the query distribution. Consequently, this implies that selecting a group of high importance samples as a "universal" query set for efficient model stealing attacks, regardless of the target task, is not feasible.

6.5 Backdoor Attack

Backdoor attack [50, 80, P3] is a training-time attack that involves actively interfering with the training process to manipulate the resulting model. Its primary objective is to introduce malicious behavior into the model, making it behave like a benign model for normal inputs. However, when a specific trigger is detected, the backdoored model intentionally misclassifies the input to a predetermined class. This type of attack can have severe consequences, such as compromising the integrity and reliability of the model, leading to potential security breaches, data manipulation, or unauthorized access to sensitive information.

Despite the severe consequences that a backdoor attack may cause, the attack itself is relatively easy to achieve by poisoning the training dataset, thereby posing an even stronger threat. For instance, a straightforward attack approach called BadNets [50]

adds a fixed trigger to a portion of the training dataset, resulting in a perfect attack where almost all triggered samples are misclassified into the target class, while the accuracy on the original task remains largely unaffected.

In the context of backdoor attacks, the poison rate plays a critical role as it directly influences the effectiveness and concealment of the attack. A higher poison rate can lead to an increased attack success rate, but it also raises the risk of detection since a large number of samples need to be modified. Conversely, a lower poison rate may offer better concealment, but it may not achieve optimal attack performance. Additionally, there are situations where the adversary can only control a small set of samples, making it impossible to poison a large number of samples to achieve the attack. Consequently, the problem of backdooring a model with a limited poison rate becomes an interesting and challenging research question.

In this section, we conduct empirical investigations to explore whether poisoning samples with different importance levels influences the attack performance under the same poison rate. We utilize two metrics to evaluate the attack performance:

1. **Accuracy**. This metric assesses the deviation of the backdoored model from the clean model. We measure the performance of the backdoored model on the clean dataset, and a successful attack should result in accuracy close to that of the clean model, making it difficult to detect.

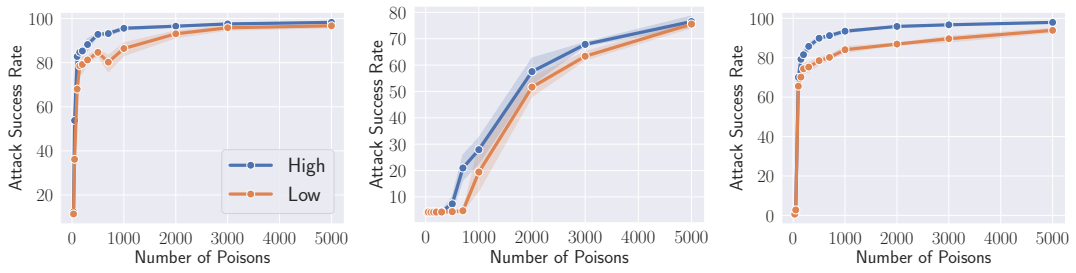
2. **Attack Success Rate (ASR)**. This metric evaluates the functionality of the backdoored model and is measured on the triggered dataset. A desirable backdoored model should exhibit a high ASR, indicating its ability to misclassify all triggered samples into the target label.

By analyzing these metrics, we aim to gain insights into the influence of data importance on the attack performance and further understand the trade-offs between attack effectiveness and concealment in the context of backdoor attacks.

In this part, we adopt the same approach as BadNets to backdoor the model, with hyperparameter details provided in Appendix A.12. Additionally, we validate the generalizability of our conclusion across five other backdoor attacks—Blend [30], SSBA [73], LF [159], SIG [17], and CTRL [70]—which utilize various trigger patterns or target different learning paradigms, as discussed in Appendix A.13.

We present the visualized attack success rate in Figure 6.14. As depicted in the figure, there is a noticeable increase in the attack success rate as the number of poisons increases. Concurrently, we observe significant differences between poisoning high importance samples and low importance samples. Specifically, poisoning an equal number of high importance samples proves to be more effective in increasing the attack success rate compared to poisoning low importance samples. This phenomenon becomes more pronounced when the poisoning rate is small. For instance, in the case of CIFAR10, with a poisoning size of 50, poisoning high importance data results in a model with an ASR of 54.42%, whereas poisoning low importance data only achieves 37.74%, indicating a 1.44× advantage. Similar trends can be observed across the other two datasets.

However, we also find that when the poisoning rate is large, the difference is not significant. We believe this is due to the trade-off between the importance advantage and the attack upper bound. As the number of poisons increases, the advantage of



(a) CIFAR10 **(b)** CelebA **(c)** TinyImageNet

Figure 6.14: Relationship between attack success rate and the poisoning rate, high importance samples enhance the efficiency of the poisoning process, particularly when the poisoning rate is small.

using high importance data becomes more evident. However, achieving the optimal ASR for the backdoor attack does not require a large amount of data. Generally, poisoning approximately 10% of the dataset is sufficient. Therefore, as the number of poisons increases, the gap between high importance and low importance samples is reduced. Nevertheless, it is still observed that poisoning high importance samples requires poisoning fewer samples to achieve its optimal ASR.

In scenarios where adversaries have limited access to data, determining the true importance of samples can be challenging, which impacts the feasibility of selectively poisoning high importance samples. In this case, we empirically demonstrate that calculating the importance value using just a fraction of the training set can provide a good approximation of the true importance. For example, with just 2% of the CIFAR10 data available, the computed importance values correlate strongly with those derived from the entire dataset, achieving a correlation coefficient of 0.811 ± 0.016. The accuracy of these approximations improves with more data: with 5% of the data, the correlation coefficient rises to 0.899 ± 0.006, and with 20% of the data, it exceeds 0.96. These results demonstrate that even with limited data access, it is feasible to closely estimate the importance of samples, facilitating effective attack planning under realistic constraints.

Additionally, our investigation into the impact on clean accuracy reveals no significant trends suggesting that poisoning samples of differing importance levels affects clean accuracy. In both scenarios, the influence on clean accuracy remains below 2%, indicating the concealment of the backdoor attack. Due to space constraints, detailed results are deferred to Appendix A.14.

Takeaways: Our experimental results demonstrate that poisoning high importance samples enhances the efficiency of the poisoning process, particularly when the poisoning rate is small. This insight offers valuable guidance for developing attack strategies aimed at compromising models with restricted data accessibility. Beyond refining trigger patterns for effective injections, prioritizing the poisoning of high importance samples emerges as a promising approach. On the other hand, the influence on clean accuracy does not yield a definitive conclusion, as poisoning either type of data has a limited impact on clean accuracy.

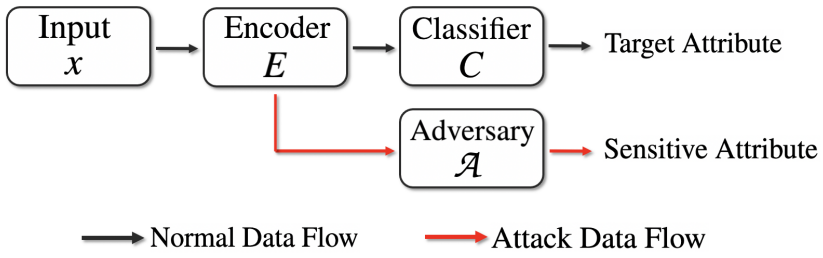


Figure 6.15: The attack scenario for attribute inference attack. The adversary can get the embeddings and aims to infer sensitive attributes based on the information encoded in embeddings.

6.6 Attribute Inference Attack

Attribute inference attack is a privacy attack that aims to infer sensitive attributes that are not directly related to the original task of a machine learning model. For instance, a model trained to predict age from profile photos may unintentionally learn to predict race as well [127, 91, 128]. This type of attack has significant implications for privacy and fairness, as the inadvertent leakage of sensitive attributes can have far-reaching consequences, including the violation of privacy rights, potential discrimination, and the undermining of trust in machine learning systems.

In this work, we focus on a commonly considered attack scenario as depicted in Figure 6.15, where the adversary exploits the embeddings of a target sample obtained from the target model to predict its sensitive attributes [2] To perform attribute inference, the adversary assumes auxiliary information about the training dataset and collects a shadow dataset from similar distributions. They train a shadow model to mimic the behavior of the target model and use the embeddings and sensitive attributes to train an attack classifier.

In this section, we investigate the impact of data importance on the CelebA dataset, which contains several attributes that can be inferred. We categorize the samples into five groups, each comprising 10,000 samples, based on their importance values ranging from low to high. Following this categorization, we train five models using these groups as target models.

To perform the attack, we utilize 10,000 samples, disjoint from the 50,000 training samples, to train a shadow model. This shadow model is employed to generate datasets for training the attack model, where the inputs are embeddings, and the associated sensitive attributes serve as labels. We train a two-layer fully connected network as the attack model, which is then utilized to infer the sensitive attribute from the embeddings.

To evaluate the attack performance, we utilize relative accuracy as the metric, comparing the accuracy against a random guessing baseline that varies for different attributes due to the uneven distribution of the CelebA dataset.

[2]We acknowledge that there exists a separate line of research on attribute inference attacks targeting *tabular data* [89, 60, 42], which primarily aims to *reconstruct* missing attribute values in original records. Given that these attacks employ different technical methodologies and pursue distinct objectives, our conclusions may not necessarily apply to such work.

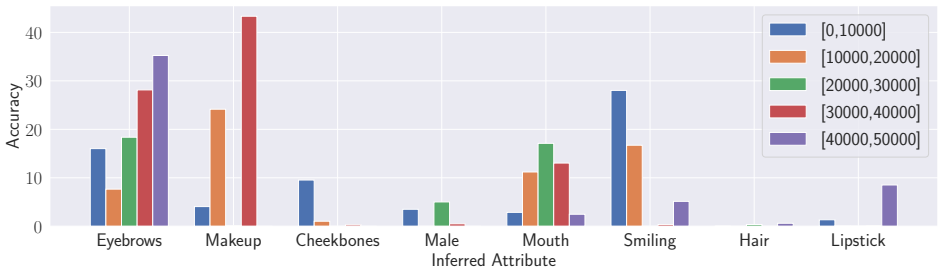


Figure 6.16: Attribute inference attack performance on different attributes, no significant correlation between attribute inference attacks and data importance is observed. These confirm our hypothesis that the importance of data samples is context-dependent and can vary based on the specific task at hand.

The experimental results, as shown in Figure 6.16, reveal no significant connection between data importance and the success of attribute inference attacks. For instance, the "Arched Eyebrows" attribute is easily inferred for high importance samples, while only low importance samples can be inferred for the "High Cheekbones" attribute. Furthermore, the vulnerability to attribute inference for the "Mouth Slightly Open" attribute is most prominent among samples with middle importance values. These results demonstrate that there is no significant correlation between attribute inference attacks and data importance.

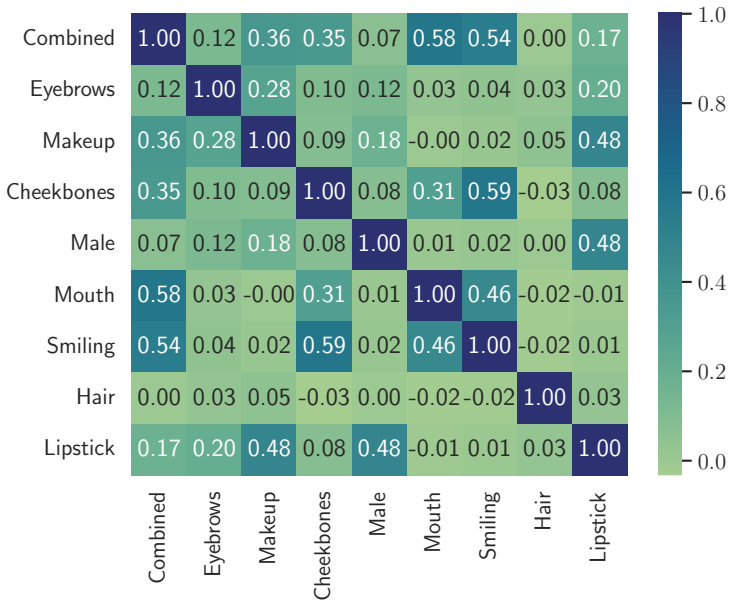


Figure 6.17: The heatmap depicts the correlation among importance values assigned to different attributes. It indicates that a sample's elevated importance on one attribute may not align with its importance on another attribute.

One possible explanation for these results is that the importance value of data samples may vary depending on the prediction task. In other words, the significance of

certain features or attributes may differ across different prediction tasks. For example, while whiskers may be an important feature for predicting gender, it may hold less importance when predicting income. We validate our conjecture by visualizing the correlation among importance values assigned to different attributes in Figure 6.17. It indicates that a sample's elevated importance on one attribute may not align with its importance on another attribute.

Takeaways: Our findings indicate that there is not a straightforward correlation between the significance of data samples and the performance of an attack, aligning with our initial hypothesis. A pivotal insight from this section demonstrates that the importance of data samples is context-dependent and can vary based on the specific task at hand, which resonates with our earlier discovery in Section 6.4.2.

6.7 Data Reconstruction Attack

Data reconstruction attack [164, 156, 41, 152, 112] refers to recovering the target dataset with limited access to the target model, with the aid of additional knowledge possessed by the adversary. While data reconstruction attack shares similarities with membership inference attack, there are significant differences that make data reconstruction a stronger attack. Specifically, membership inference operates at the sample level, determining the membership status of individual samples. In contrast, data reconstruction is a dataset-level attack aimed at extracting the entire training dataset. This distinction necessitates different technical approaches for data reconstruction.

In this work, we employ two data reconstruction attacks, namely DeepInversion [156] and Revealer [164], to investigate the influence of data importance on the reconstruction process. These attacks are based on the optimization of input samples, as illustrated in Figure 6.18. Specifically, given a target class y, both methods initialize a sample x and iteratively update it to maximize the likelihood or probability of belonging to that class while keeping the model parameters fixed. This optimization process is guided by the following loss function:

$$\min_{x} \mathcal{L}(f_\theta(x), y)$$

DeepInversion leverages statistical information encoded in the batch normalization layer to enhance the quality of reconstructions, while Revealer employs a Generative Adversarial Network (GAN) to generate high-quality reconstructions.

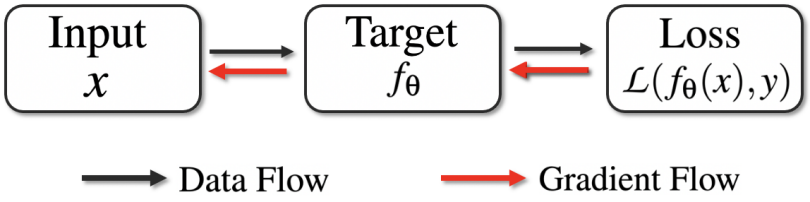
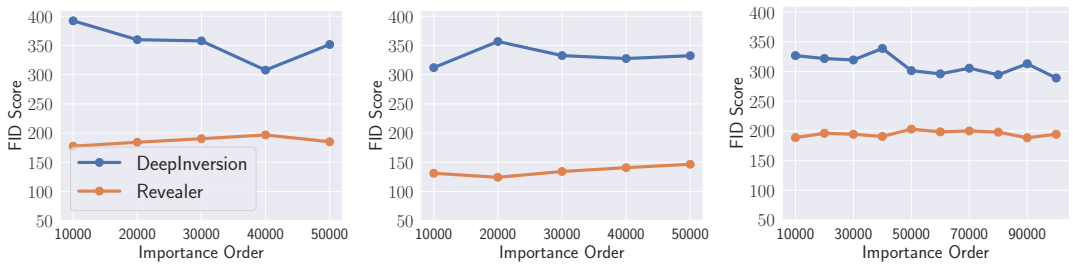


Figure 6.18: The workflow of a basic data reconstruction attack, the adversary optimizes the input to maximize the likelihood of the target class.



(a) CIFAR10 **(b)** CelebA **(c)** TinyImageNet

Figure 6.19: Relationship between reconstruction quality and data importance, reconstruction performance remains steady regardless of the importance level of the data samples.

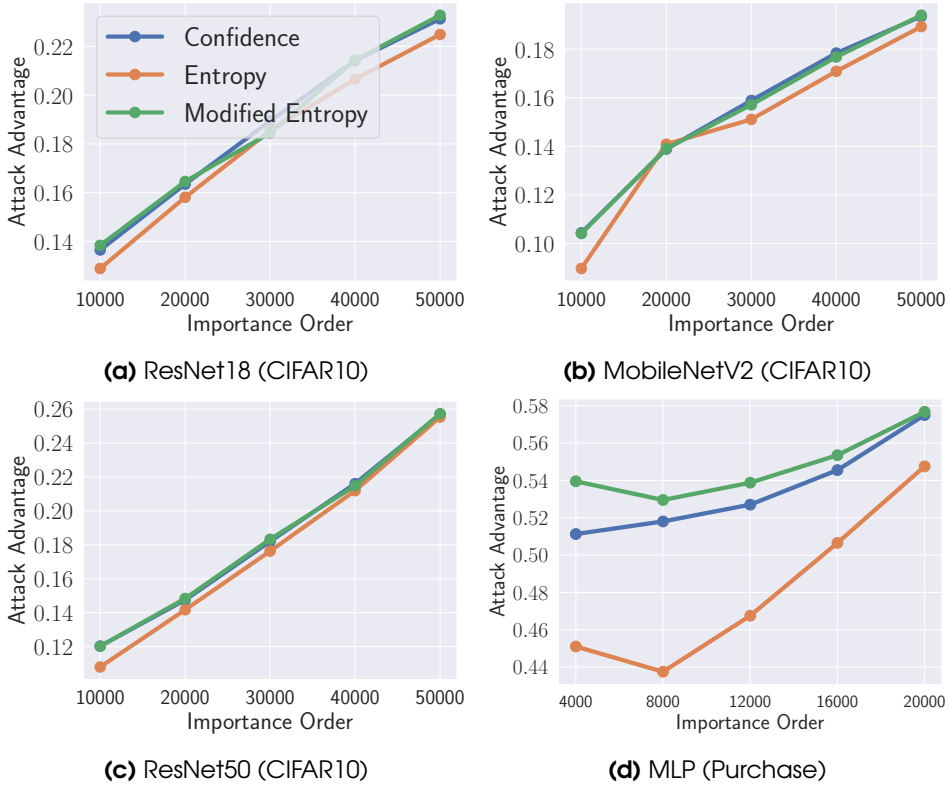
To investigate the impact of data importance on the performance of data reconstruction attacks, we partition the samples into groups of 10,000 based on their importance values, ranging from low importance to high importance. Subsequently, we train one target model for each sample group, resulting in a total of five models for CIFAR10 and CelebA datasets, and ten models for TinyImageNet. We then apply two reconstruction attacks on each of them. We leverage Fréchet Inception Distance (FID) to measure the similarity between reconstructed samples and the training samples, given its established utility in evaluating the quality of generated distributions [164, 156, 132]. A smaller FID denotes better reconstruction quality. For each target model, we generate 10,000 reconstructions, matching the size of the training dataset. Subsequently, we calculate the FID score, quantifying the discrepancy between the reconstructions and the corresponding training dataset.

The findings presented in Figure 6.19 suggest that there is no significant distinction between high and low importance data samples in terms of data reconstruction. Taking CIFAR10 as an example, DeepInversion exhibits a maximum deviation of only 13.02% compared to the mean value, indicating a consistent performance. Similar results are observed with Revealer, where the maximum deviation is merely 5.35% compared to the mean value. Moreover, this consistent performance extends to more complex datasets. For instance, in the case of CelebA dataset, the maximum deviation is less than 8.27%, while for TinyImageNet, the deviation is less than 4.01%. These findings suggest that the reconstruction performance remains steady regardless of the importance level of the data samples.

6.8 Transferability Study

In order to fortify the generalizability of our conclusions, this section investigates the transferability of our findings across various model architectures and data modalities. For the vision modality, we conducted experiments employing two distinct model architectures, namely MobileNetV2 [115] and ResNet50 [52].

To evaluate the transferability to diverse data modalities, we introduced the tabular dataset Purchase-100 [13], consisting of 600 binary features for classifying 100 classes. We



(a) ResNet18 (CIFAR10) **(b)** MobileNetV2 (CIFAR10)

(c) ResNet50 (CIFAR10) **(d)** MLP (Purchase)

Figure 6.20: Relationship between membership inference attack advantage and data importance. Results show that our conclusion can be generalized to different model architectures and data modalities.

focus on the tabular modality considering that existing Shapley methods predominantly support vision and tabular modalities. We utilize a Multilayer Perceptron (MLP) to process the Purchase task, aligning with established practices in prior research [125, 114].

Our experimental results consistently support our conclusions, irrespective of the model architecture and data modality. For example, in Figure 6.20, the three left figures depict a consistent relationship between the advantage of membership inference attacks and the importance of data across all three model architectures. This trend is also evident when performing the attack based on the distance to the boundary (see results in Appendix A.15). Additionally, Figure 6.20d illustrates that this conclusion holds for the tabular modality. Although slight fluctuations are observed in the low importance area, the overall picture demonstrates a consistent relationship between importance and membership vulnerability, aligning with the conclusions drawn from the vision modality.

Furthermore, this conclusion extends to other attack types, such as model stealing and backdoor attacks. Due to space constraints, we defer the results to Appendix A.15. These findings reaffirm the consistent impact of data importance across different attack scenarios, underscoring the generalizability of our observations.

To foster further research and collaboration, we have open-sourced our evaluation framework, available at `https://github.com/TrustAIRLab/importance-in-`

`mlattacks`. This will enable other researchers to examine whether the observed data discrepancies hold for new types of attacks, thereby benefiting the broader community.

6.9 Conclusion

In this chapter, our research systematically studies the vulnerability of heterogeneous data when confronted with machine learning attacks. Our findings underscore a heightened susceptibility of high importance data samples to privacy attacks, including membership inference attacks and model stealing attacks. Our findings also carry practical implications, inspiring researchers to design more efficient attacks. For example, we empirically showcase the potential enhancement of membership inference attacks through the incorporation of sample-specific criteria based on importance values. Additionally, we demonstrate that our findings can be strategically employed to guide the creation of more advanced attacks through the active manipulation of sample importance.

7
Related Work

In addition to the attack approach investigated in our dissertation, several methods exist for privacy and security attacks against machine learning models. In the following section, we provide a brief overview of these approaches.

7.1 Membership Inference Attack

Membership Inference Attacks (MIA) [125, 114, 75, P1, T2, S6] have emerged as a significant threat to privacy in the context of machine learning models. These attacks aim to reveal the membership status of a target sample, i.e., whether the sample was part of the training dataset or not, thereby directly breaching privacy.

The seminal work by Shokri et al.[125] introduced MIA against machine learning models, wherein multiple shadow models were trained to mimic the behavior of the target model. This attack originally required access to data from the same distribution as the training dataset. However, Salem et al.[114] relaxed this assumption by demonstrating the effectiveness of using only a single shadow model, substantially reducing the computational cost involved.

Subsequent research [31, 75] has explored more challenging settings for MIA. In these scenarios, the adversary only has access to hard-label predictions from the target model. Li and Zhang [75] proposed a method that approximates the distance between the target sample and its decision boundary using adversarial examples, enabling the attacker to make decisions based on this distance.

Recent advancements in MIA have focused on enhancing attack performance. Carlini et al.[21] leveraged the discrepancy between models trained with and without the target sample to improve attack effectiveness. Liu et al.[81] demonstrated the utility of loss trajectory analysis in MIA. Furthermore, Tramèr et al. [140] highlighted the potential of data poisoning, showing that even with access to a small fraction of the training dataset, the attacker can significantly boost the performance of membership inference attacks.

7.2 Model Stealing Attack

Model stealing attacks [141, 100, 64, 142, T3, T1] aim to extract information from a victim model and construct a local surrogate model. This attack was initially proposed by Tramèr et al.[141], assuming that the adversary has access to a surrogate dataset for stealing the model. Orekondy et al. further advanced this approach by developing a reinforcement learning-based framework that optimizes query time and effectiveness [100].

Recent research has focused on the more stringent data-free setting, where adversaries lack access to any data. In this context, Kariyappa et al. [64] propose MAZE, which employs a generative model to generate synthetic data samples for launching the attack. The generator is trained to maximize disagreement between the victim model and the clone model, requiring the gradients from the victim model. To approximate these gradients with only black-box access, zeroth-order gradient estimation techniques are adopted.

Truong et al. [142] present a similar approach, where they replace the loss function

from Kullback-Leibler (KL) divergence to ℓ_1 norm loss for training the student model. In contrast to the previous attacks that generate "hard" queries that differ in predictions between the victim and clone models, Sanyal et al. [116] adopt a different strategy by generating "diverse" queries to increase predictions belonging to different classes.

7.3 Backdoor Attack

Backdoor attacks [50, 80, P3, 16, 70] are training-time attacks that introduce malicious behavior into the model, making it behave like a benign model for normal inputs, while intentionally misclassifying the input to a predetermined class when the trigger appears.

The seminal work by Gu et al.[50] introduced the concept of the backdoor attack on machine learning models. Building upon this, Liu et al.[80] proposed an advanced backdooring technique that incorporates enhanced triggers and relies on fewer assumptions. However, these attacks were limited to injecting static triggers, making them susceptible to detection.

Salem et al.[P3] integrated generative models to perform dynamic backdoor attacks, where the trigger is not fixed thus increasing the difficulty of detection. Nguyen and Tran[97] further extended this concept to design an input-aware attack. Most existing attacks in this domain are based on poisoning attacks [P2, 94, 121, 169], which involve poisoning the training dataset. In contrast, Bagdasaryan and Shmatikov [16] propose a distinct attack target in the scenario where the learning algorithm itself is poisoned, presenting an alternative approach in this field of study.

7.4 Data Reconstruction Attack

Data reconstruction attacks [41, 152, 51, 112] aim to recover the target dataset with limited access to the target model, with the aid of additional knowledge possessed by the adversary.

In the realm of data reconstruction attacks, existing approaches can be broadly classified into three categories: optimization-based attacks, training-based attacks, and analysis-based attacks.

Optimization-based attacks, first introduced by Fredrikson et al.[41], represent the majority of existing reconstruction attacks. These attacks employ an iterative optimization process to reconstruct the training dataset, with the objective of obtaining a high likelihood score for the desired class. Notably, the integration of generative models by Zhang et al.[164] has contributed to improving the quality of reconstruction. Building on this line of research, several studies have explored diverse architectural choices [28, 145] and loss functions [132] to further enhance reconstruction performance.

Conversely, training-based attacks [152] regard the target model as an encoder and train a corresponding decoder network to reconstruct inputs based on the model's outputs. Recently, Haim et al. [51] presented a theoretical demonstration that, under specific assumptions, the training data can be completely recovered, leading to a new attack approach.

8

Summary and Conclusion

Machine learning (ML) has undergone rapid development, significantly driving technological innovations through models like large language models (LLMs) and vision-language models (VLMs). A key factor behind these breakthroughs is the critical role of data, where high-quality and diverse datasets are essential for improving model performance and mitigating biases. However, this heavy reliance on data also exposes vulnerabilities, such as data poisoning and privacy risks, including membership inference attacks that reveal whether specific data points were used in training. While data drives ML advancements, it also introduces significant security and privacy concerns.

This dissertation presents four peer-reviewed publications [P1, P2, P3, P4] that examine the role of data in adversarial machine learning.

In our first work [P1], we introduced the first text-only membership inference attack targeting in-context learning (ICL). Our attack proved effective across various scenarios, including cases where the language model is restricted to generating responses from a predefined list. Extensive experiments with diverse datasets and models demonstrated the attack's efficacy. We identified that the vulnerability arises from the interplay between prompt size and demonstration position. Furthermore, our investigation revealed that information leakage persists even as models evolve over time. To mitigate this, we explored three defense strategies, finding that their combination significantly reduces privacy risks. While our study advanced understanding of ICL vulnerabilities and proposed practical defenses, achieving a comprehensive and generalized defense remains an open challenge. Our findings provide valuable insights for researchers and practitioners focused on secure, privacy-conscious LLM development.

In our second work [P2], we proposed EntF, a novel poisoning attack that decreases a deep learning classifier's accuracy, even with adversarial training (AT). Our approach exploits feature entanglement between different classes, showing effectiveness against adversarial training in diverse settings, including more aggressive AT budgets and unseen model architectures. We also explored the distinct roles of robust and non-robust features in poisoning both standard and AT models, demonstrating that hybrid attacks can successfully target both. We encourage future research to develop stronger defenses and analyze EntF in broader contexts, particularly in scenarios where the poisoner does not have access to the training data. Extending this approach to data-free poisons could also be a promising direction.

Our third work [P3] focused on backdoor attacks, where an adversary manipulates a model to misclassify any input with a specific trigger. We introduced the first set of dynamic backdoor attacks, allowing triggers to have multiple patterns and locations. We proposed three techniques: Random Backdoor, which samples triggers randomly, BaN, a generative network to construct triggers, and c-BaN, which generates label-specific triggers. Our evaluations showed that all methods achieved high success rates while preserving model utility, and they effectively bypassed state-of-the-art defenses against backdoor attacks.

In our final work [P4], we studied the vulnerability of heterogeneous data to machine learning attacks, revealing that high-importance data samples are more susceptible to privacy attacks such as membership inference and model stealing. We empirically demonstrated that attacks can be enhanced by targeting these critical data samples and suggested that these insights could be used to design more advanced attacks.

Future Research Directions: The concept of availability attacks has potential use cases in protecting user data from misuse. However, two key challenges need addressing. First, attackers typically do not have access to the reference model, making it difficult to design effective attacks without this assumption. While some research has explored data-free poisons [119], these methods can often be defended with adversarial training. Developing robust availability attacks without the need for a reference model is an important area for future exploration. Second, current clean-label attacks typically require poisoning the entire dataset, which is impractical in many scenarios. Our findings in Chapter 4 suggest that poisoning only a subset of the data can still significantly degrade model performance, but further improvement is possible. Research has shown that poisoning a small portion of the dataset can reduce performance to near-random guessing [57, 19], though applying such methods to more complex models like neural networks is still a challenge, especially in adversarial training contexts.

While this dissertation provides valuable insights into the relationship between data importance and vulnerability to specific attacks, several limitations exist that warrant further investigation. Our study focuses on a specific set of attacks. Although these are important, they may not cover the entire spectrum of potential threats. Other types of attacks could exhibit different relationships between data importance and vulnerability, and understanding how these various attacks interact with data importance remains an open area for exploration.

Extending our findings to Large Language Models (LLMs) presents substantial challenges despite their promising advancements. The primary obstacle is the computational cost associated with calculating importance values. To manage this burden, current methods often resort to computationally lighter algorithms like KNN for classification tasks. However, it is unclear whether similar computationally efficient approaches can be adapted to approximate auto-regression models, especially since LLMs exhibit unique emergent characteristics when scaled beyond certain thresholds. Additionally, the considerably larger datasets typical of LLMs further complicate the feasibility of extending these methods.

Furthermore, when discussing how data augmentation influences data importance, our research does not examine more complex augmentation techniques, such as those utilizing generative models. Future work should investigate whether these advanced techniques affect data importance and vulnerability differently. Additionally, exploring whether there exists a generalizable method to manipulate data importance across various augmentation techniques would be invaluable.

A
Appendix

A.1 Influence of Memorization on Attack Performance

To investigate the effect of memorization, we fine-tuned the LLaMA model using the LoRA technique with 2,000 samples from DBPedia, resulting in the model heavily memorizing DBPedia content. The attack performance showed a significant decrease, with accuracy dropping from 0.934 to 0.782. Specifically, after being fine-tuned on the DBPedia dataset, the model exhibited increased confidence in its responses to test samples, regardless of their presence in the prompt. As illustrated in Figure A.1, after fine-tuning, more number of brainwash is required to change the model's prediction, indicating strong memorization.

However, it is important to note that this represents an extreme case. Our experiments are conducted on commonly used benchmark datasets typically used for pre-training. If evaluated on entirely unseen datasets, we would expect even better performance, as the model would be less confident on unseen data.

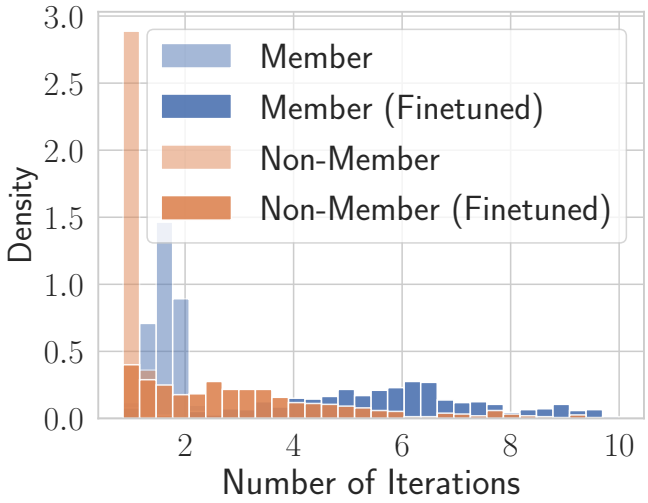


Figure A.1: Distribution of the number of iterations before and after fine-tuning. The results indicate that post-fine-tuning, the model exhibits increased confidence in its samples, irrespective of their membership status. Both member and non-member samples require a greater number of iterations to alter their predictions, leading to a degradation in attack performance. The experiments are conducted on the DBPedia dataset using the LLaMA model.

A.2 Attack Performance Over Time (DBPedia)

We investigated performance changes using the DBPedia dataset. Due to the deprecation of some versions, we conducted experiments on gpt-3.5-turbo-0613, gpt-3.5-turbo-1106, and gpt-3.5-turbo-0125. Results presented in Figure A.2 indicate that no single version consistently outperforms the others in terms of robustness.

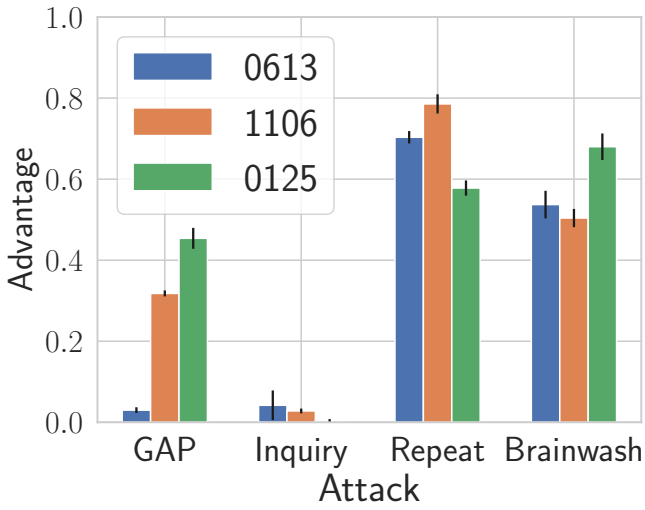


Figure A.2: We evaluate the evolution of attack performance on the DBPedia dataset using three versions of the GPT-3.5 API (gpt-3.5-turbo-0613, gpt-3.5-turbo-1106, and gpt-3.5-turbo-0124). The results from the DBPedia dataset align with our findings on the TREC dataset, indicating that the robustness of commercial models like GPT-3.5 does not consistently improve over time.

A.3 Well-Separable Features in Existing Methods

Instead of introducing entangled features, most existing poisoning methods rely on generating certain shortcut perturbations that can be more easily learned by the target model than the actual image content [136, 118, 38, 157]. Figure A.3 visualizes the poison representations of different existing methods on an adversarially-trained model. As can be seen, although there exist certain differences between the patterns of different methods (e.g., regarding the relative positions of different classes), all these methods indeed yield well-separable data. This observation also confirms that causing entangled features is a *sufficient condition* to degrade adversarial training.

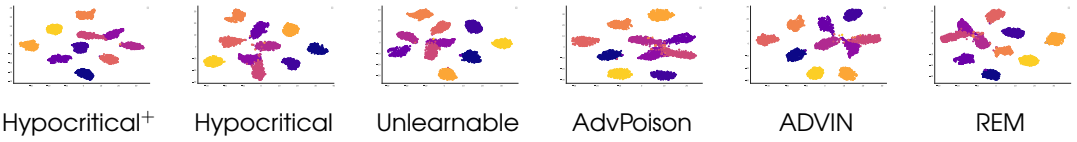


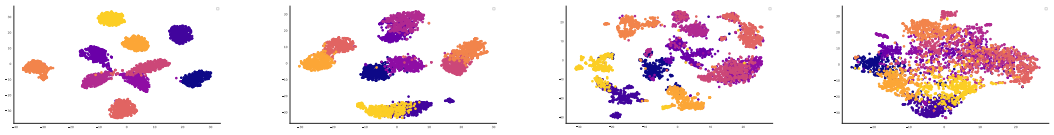
Figure A.3: The t-SNE visualizations of the feature representations for existing poisoning methods.

A.4 Additional Analysis of Entangled Features

Our experimental results have demonstrated the effectiveness of EntF in various scenarios. Here we provide additional analysis to better understand the property of entangled features. To this end, we adjust the class selection strategy in EntF-pull to be simply

based on pairwise entangled features. Specifically, each pair consists of two classes that have a minimal centroid distance. For optimization, we calculate the centroids of each class pair and then minimize the distance between samples in one class and the centroid of the other class. We denote this attack variant targeting pairwise entangled features as EntF-pull-pair.

We find that EntF-pull-pair can decrease the model accuracy from 84.88% to 78.21%, which indicates that introducing pairwise entangled features can already substantially compromise adversarial training. However, there is still a large performance gap between EntF-pull-pair and our original EntF. This can be explained by the fact that EntF-pull exploits more diverse pulling directions based on the sample-class distance, and EntF-push makes samples from multiple classes become overlapped. The visualizations in Figure A.4 clearly confirm the above observation that EntF-pull-pair yields entangled features to some extent but still fewer than the original EntF-pull and EntF-push.



(a) Clean (84.88%) **(b)** Pull-pair (78.21%) **(c)** Pull (72.99%) **(d)** Push (71.57%)

Figure A.4: The t-SNE visualizations for EntF-pull-pair vs. our original EntF (Pull and Push).

A.5 Additional Perturbation Visualizations

Perturbation visualizations for different ϵ_{ref} for more datasets can be found in Figure A.5. Visualizations suggest that poisoning ST and AT models requires modifying different types of features. More specifically, modifying the robust (semantic) features is the key to poisoning AT models, while modifying the non-robust features works for ST models.

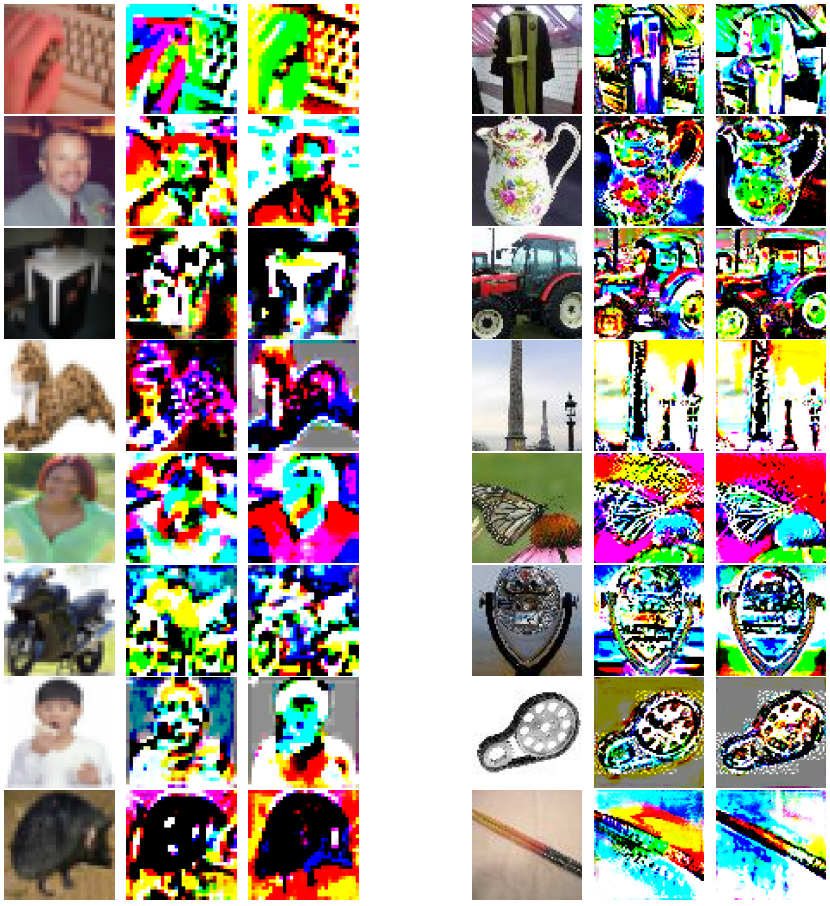
A.6 Augmentation As A Defense

Experiments in Figure A.6 show that data augmentations can reduce the performance of our dynamic backdoor attacks; however, they cannot prevent it and can drop the utility significantly.

A.7 Effectiveness of KNN-Shapley

We validate the efficacy of the KNN-Shapley method to ensure its accurate assignment of importance value to individual samples, the parameter settings are presented in Appendix A.8.

We apply the KNN-Shapley approach to three distinct datasets and compute the importance value for each sample. To gain insight into the contribution of different samples to the model's utility, we visualize the distribution of importance values for the CIFAR10 dataset in Figure A.7a. Full results are depicted in Appendix A.9.



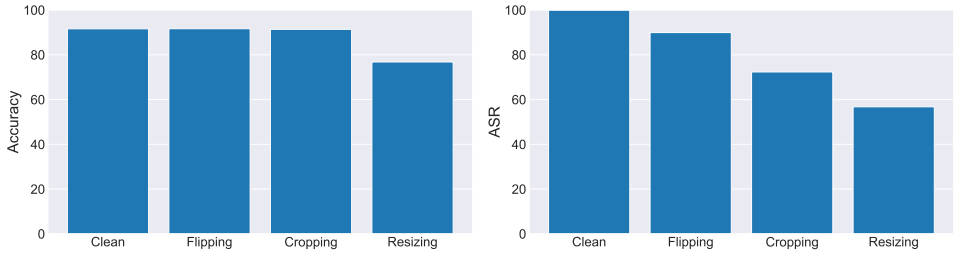
(a) CIFAR-100 **(b)** TinyImageNet

Figure A.5: Perturbation (normalized to (0,1)) visualizations for CIFAR-100 and TinyImageNet.

The observed distribution aligns with our expectations, as most samples exhibit similar contributions, while certain samples significantly influence the model's behavior. We corroborate this finding with the other two datasets, and include the corresponding visualizations in Appendix A.9.

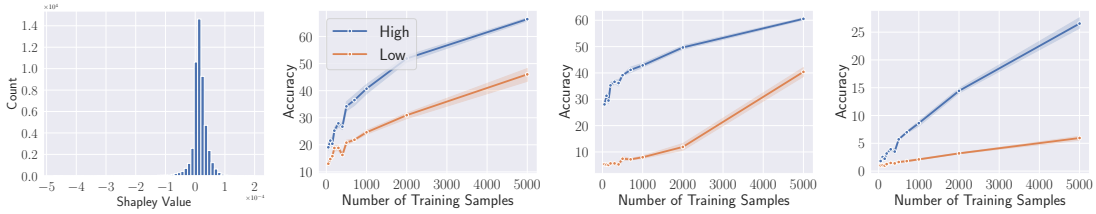
Subsequently, we empirically validate whether samples with high importance values and those with low importance values demonstrate distinct training performance. To achieve this, we sort the samples based on their importance values and form two sets: one comprising samples with the highest values and the other consisting of samples with the lowest values. We employ these two sets to train two separate models and evaluate their performance on the testing dataset. We vary the size of these two sets from 50 to 5000 and plot the corresponding testing accuracy in Figure A.7b, Figure A.7c, and Figure A.7d.

The figures clearly demonstrate that samples with varying importance values exhibit significant differences in training performance. Specifically, considering the CIFAR10



(a) Utility **(b)** ASR

Figure A.6: The performance of the cBaN technique when applying data augmentations techniques when training the target model. Figure A.6a and Figure A.6b shows the utility and ASR, respectively.



(a) Shapley Distribution **(b)** CIFAR10 **(c)** CelebA **(d)** TinyImageNet

Figure A.7: Distribution of importance value and learning characteristics for data with different importance. High importance samples contribute to better model utility when the dataset has the same size. Importance distribution for CelebA and TinyImageNet can be found in Appendix A.9.

dataset trained with 2000 samples, the model trained with high importance samples achieves a testing accuracy that is 1.6× higher compared to the model trained with low importance samples. Moreover, in the case of TinyImageNet, the disparity is even more pronounced. When the training set comprises 5000 samples, the model trained with valuable data attains a testing accuracy that is 4.4× higher than that of the model trained with low importance samples. These experimental findings provide strong evidence supporting the effectiveness of KNN-Shapley.

Regarding scalability, Jia et al. [63] provide a runtime demonstration (Figure A.8 for ease reference) showing that existing measurement methods, except for KNN-Shapley, do not scale efficiently to large datasets, even as CIFAR10.

A.8 Measurement Hyperparameter

In implementing the KNN-Shapley method, we set the hyperparameter $k = 6$, following the suggestion in the original paper [62]. Further experimentation with $k = 7$ and $k = 8$ indicated that performance remains largely consistent across these settings. Specifically, the correlation between importance values calculated with $k = 6$ and $k = 7$ is 0.9988, and between $k = 6$ and $k = 8$, it's 0.9972, demonstrating the robustness of our results with respect to this hyperparameter.

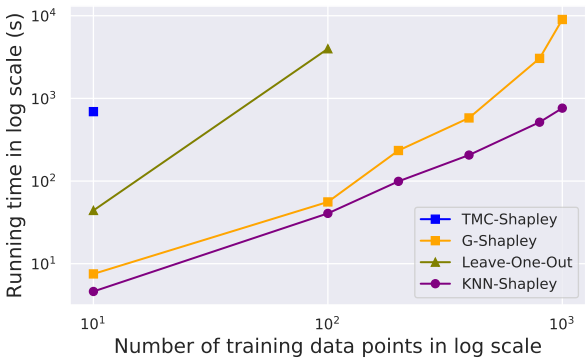
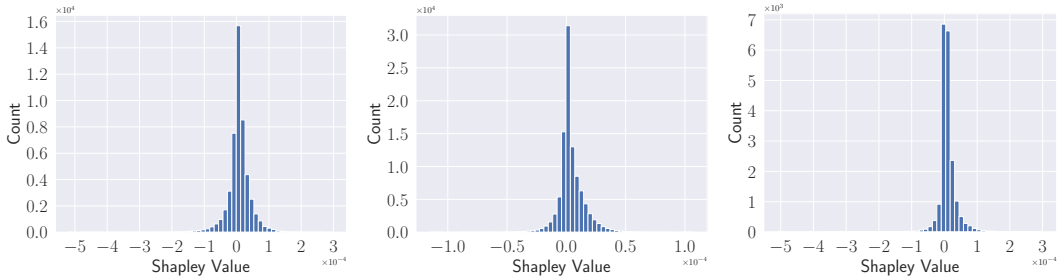


Figure A.8: A runtime comparison of current measurement methods reveals their significant computational inefficiency. (Figure adapted from Figure 1 in (63))



(a) CelebA **(b)** TinyImagenet **(c)** Purchase

Figure A.9: Importance distribution for CelebA, TinyImagenet, Purchase.

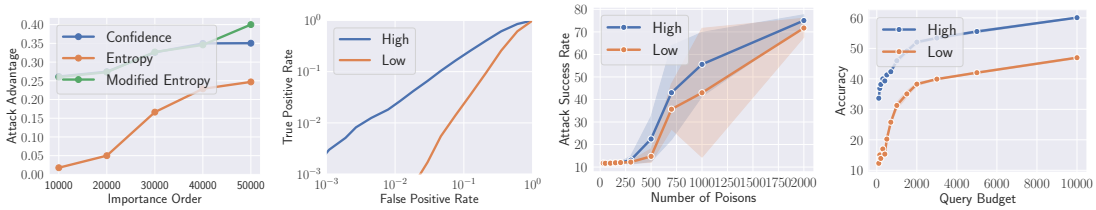
A.9 Importance Distribution

Figure A.9 shows the importance distributions for the CelebA and TinyImageNet dataset.

A.10 CelebA Attribute Selection

The CelebA dataset contains 40 binary attributes, which is not suitable for multi-class classification. Therefore, we follow previous works [98, P3, 167, S2] that select the three most balanced attributes (Heavy Makeup, Mouth Slightly Open, and Smiling) to create an 8-class (2^3) classification task.

To validate that our findings are not dependent on this specific attribute selection, we conducted the same experiments using another randomly selected set of attributes (High Cheekbones, Arched Eyebrows, and Wearing Lipstick). We evaluated the performance on membership inference attacks, model stealing, and backdoor attacks. The results are depicted in Figure A.10, confirming that our findings are consistent across these different attribute sets. For example, Figure A.10a demonstrate that samples with higher importance are more vulnerable to membership inference attack; it also reflects on the

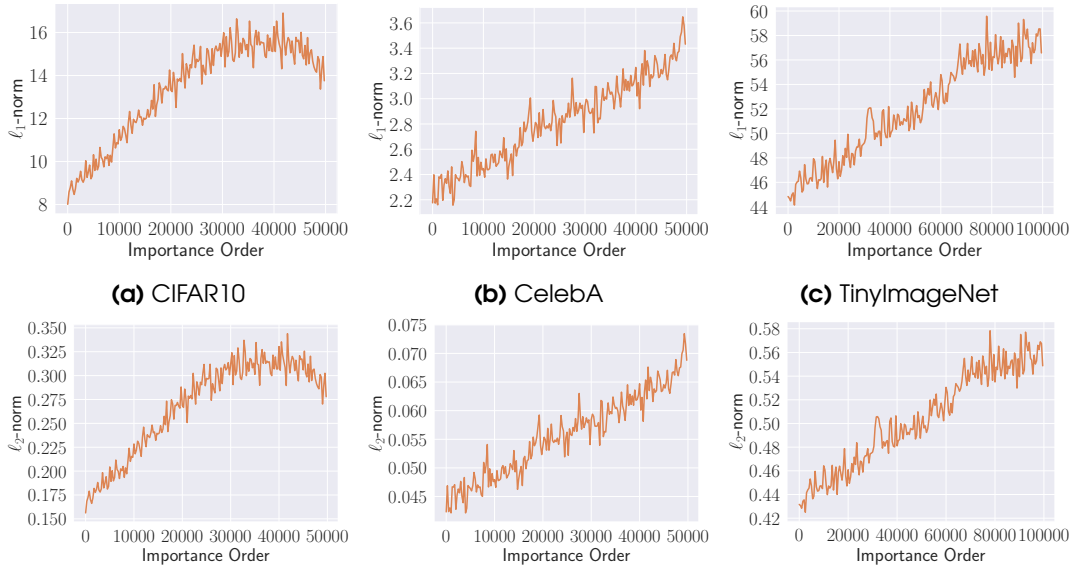


(a) MIA (metric-based) **(b)** MIA (log-scale) **(c)** Backdoor ASR **(d)** Model Stealing

Figure A.10: Attack performance on samples with high and low importance. The results demonstrate that our conclusions are consistent across different sets of selected attributes.

worst-case evaluation as illustrated in Figure A.10b. The conclusion holds for backdoor and model stealing attack, specifically, with a 1500 query budget, high importance samples can steal a surrogate model with 48% higher accuracy than that stolen by low importance samples.

A.11 Membership Inference Attack

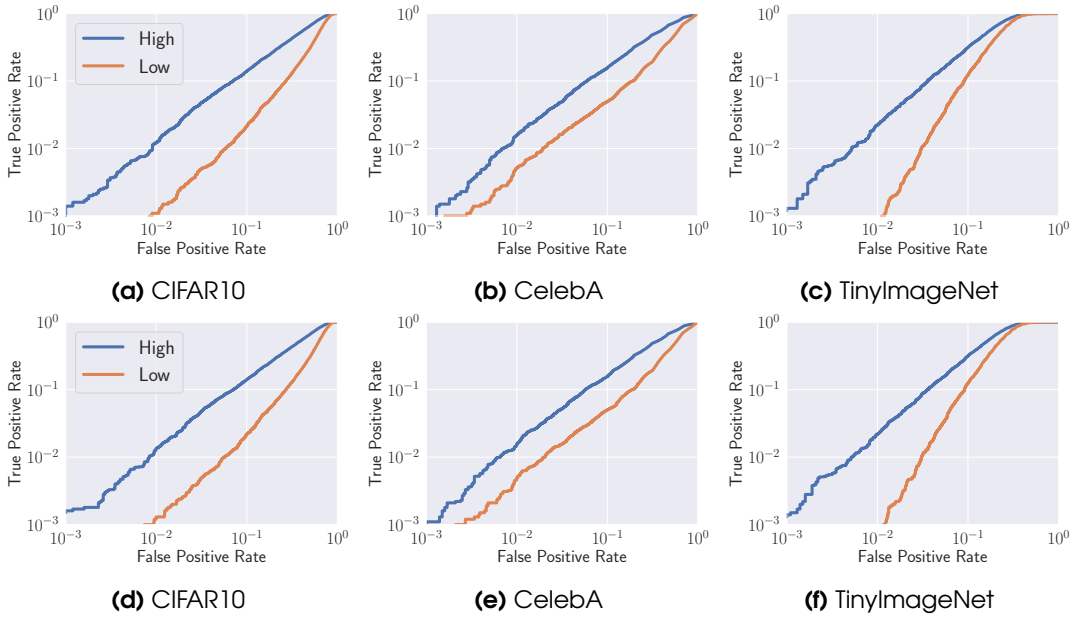


(a) CIFAR10 **(b)** CelebA **(c)** TinyImageNet

(d) CIFAR10 **(e)** CelebA **(f)** TinyImageNet

Figure A.11: Relationship between distance to the decision boundary and importance value.

Figure A.11 depicts the distance to the boundary for samples with different importance values, measured by two different norms. Figure A.12 represents the log-scale ROC curves for attacks conducted based on the distance to the boundary.



(a) CIFAR10 **(b)** CelebA **(c)** TinyImageNet

(d) CIFAR10 **(e)** CelebA **(f)** TinyImageNet

Figure A.12: Log-scale ROC curve: membership inference attack based on the distance to the decision boundary. The first row is results generated using ℓ_1 norm while the second row is using ℓ_2 norm.

A.12 Hyperparameters for Backdoor Attacks

For the three datasets evaluated in our study—CIFAR10, CelebA, and TinyImagenet—a consistent modification was applied to each image: a black square was positioned at the bottom left corner.

The dimension of this black square, or backdoor trigger, varied according to the image sizes of the respective datasets to maintain proportional consistency. Specifically, for the CIFAR10 dataset, with an image resolution of 32×32, the trigger was sized at 2×2. In the case of CelebA, which features larger images with dimensions of 178×218, the trigger's size was increased to 8×8. Lastly, for images from TinyImagenet, which are 64×64 pixels, a 5×5 square was used as the trigger.

A.13 More Backdoor Attacks

To assess whether the observation that poisoning high importance data samples enhances backdoor attack effectiveness is applicable to various trigger patterns, we broadened our study to include three additional backdoor methods. These methods comprise Blend [30], which incorporates triggers covering the entirety of the input; SSBA [73], characterized by sample-specific and invisible triggers; LF [159], which utilizes triggers of low frequency; SIG [17], a method without label poisoning; and CTRL [70], which targets contrastive learning.

We utilized the BackdoorBench tool [150] to conduct our experiments, adhering to all default implementation settings with the sole modification being the selection

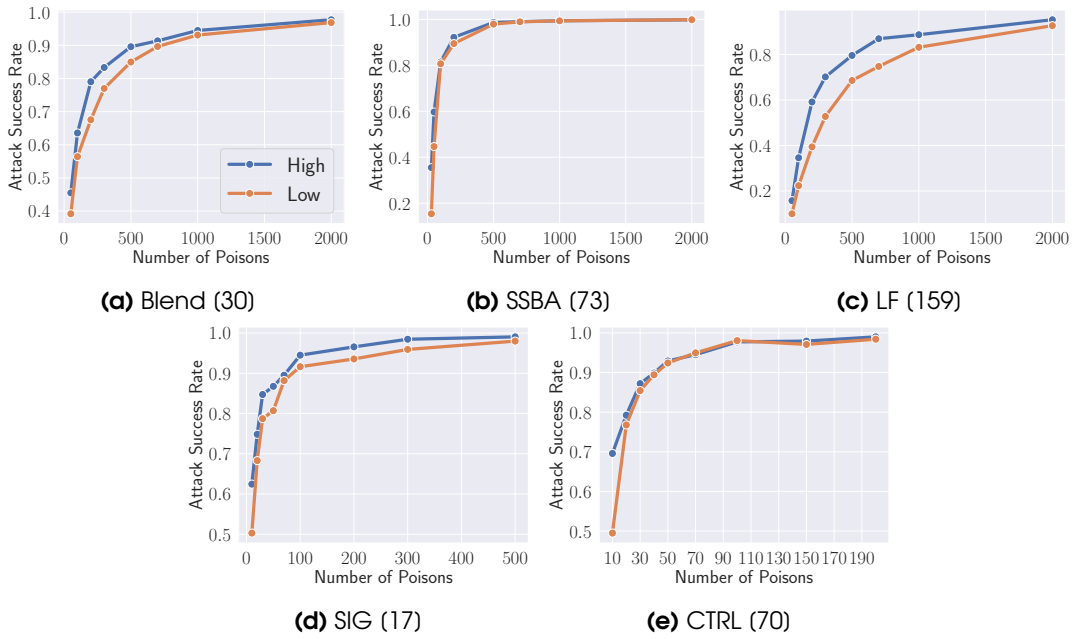


Figure A.13: Relationship between attack success rate and the poisoning rate on different backdoor attacks, the conclusion that high importance samples enhance the efficiency of the poisoning process holds for other backdoor attacks with different backdoor patterns and learning paradigms.

process for poisoning samples.

The results, depicted in Figure A.13, consistently demonstrate that the poisoning of high importance samples significantly improves the efficacy of the backdoor attacks across more complex trigger patterns, thus underscoring the robust generalizability of our conclusions.

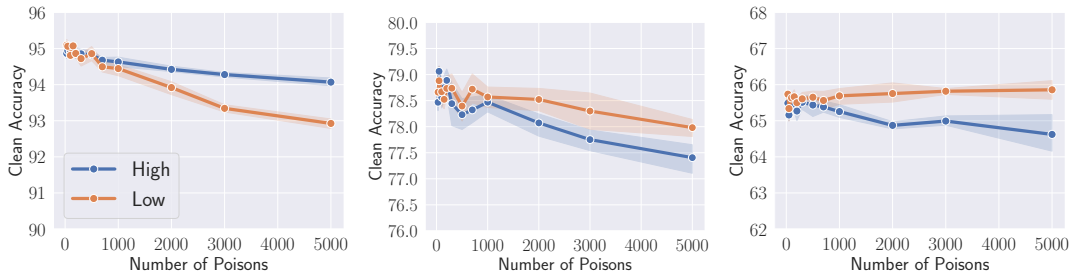
A.14 Clean Accuracy Performance

The effect of poisoning high and low importance samples on clean accuracy is depicted in Figure A.14.

A.15 Transferability Study

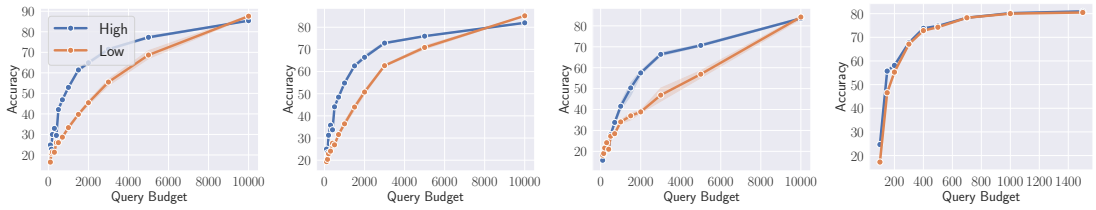
Figure A.15 and Figure A.16 demonstrate the transferability on model stealing and backdoor attack.

Figure A.17 showcases Log-scale ROC curves for three distinct architectures utilizing three different norms. The experiments are conducted on the CIFAR10 target dataset.



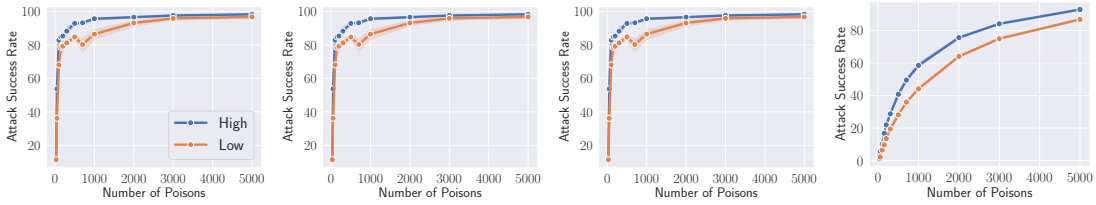
(a) CIFAR10 **(b)** CelebA **(c)** TinyImageNet

Figure A.14: Relationship between accuracy and the poisoning rate, poisoning samples with different importance does not have a significant difference.



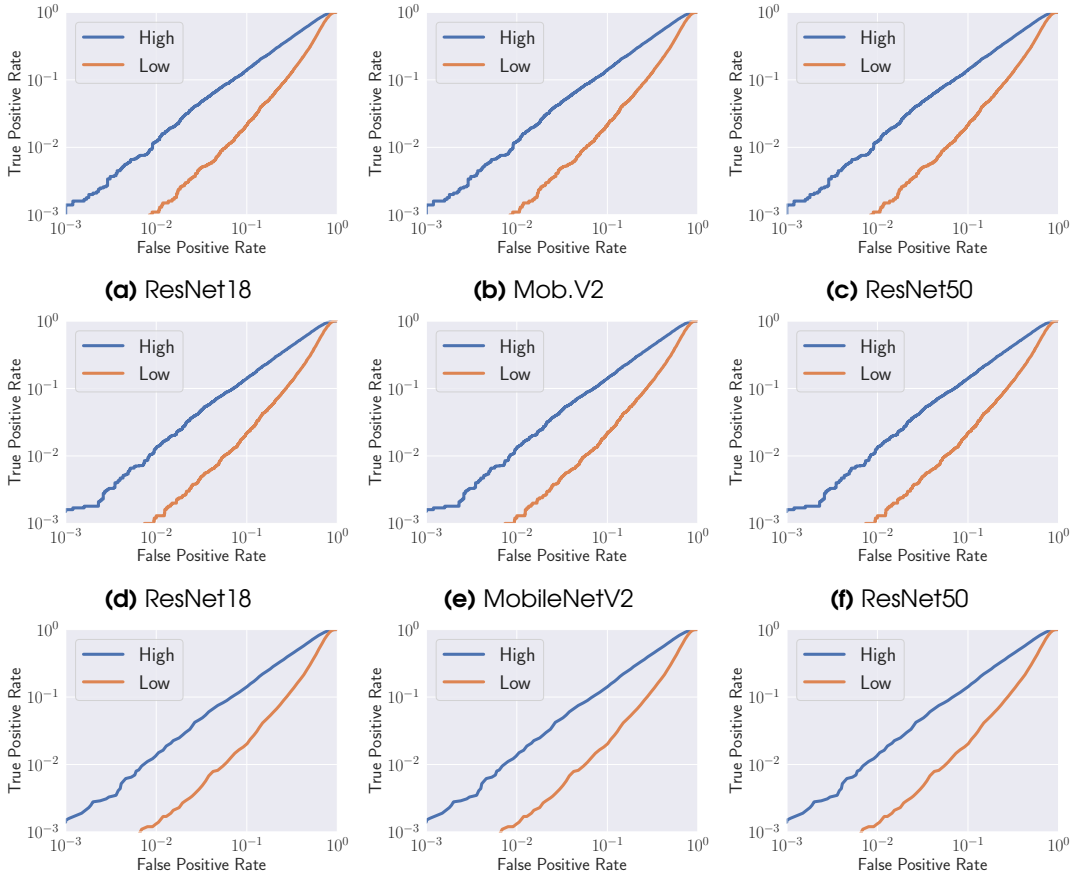
(a) ResNet18 (CIFAR10) **(b)** Mob.V2 (CIFAR10) **(c)** ResNet50 (CIFAR10) **(d)** MLP (Purchase)

Figure A.15: Relationship between model stealing accuracy and the query budget.



(a) ResNet18 (CIFAR10) **(b)** Mob.V2 (CIFAR10) **(c)** ResNet50 (CIFAR10) **(d)** MLP (Purchase)

Figure A.16: Relationship between backdoor attack success rate and the poisoning rate.



(a) ResNet18 **(b)** Mob.V2 **(c)** ResNet50

(d) ResNet18 **(e)** MobileNetV2 **(f)** ResNet50

(g) ResNet18 **(h)** MobileNetV2 **(i)** ResNet50

Figure A.17: Log-scale ROC curve: membership inference attack based on the distance to the decision boundary. The first row is results generated using ℓ_1 norm, the second row is using ℓ_2 norm, and the third row is using ℓ_∞ norm.

Bibliography

Author's Papers for this Thesis

[P1] Wen, R., Li, Z., Backes, M., and Zhang, Y. Membership Inference Attacks Against In-Context Learning. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2024.

[P2] Wen, R., Zhao, Z., Liu, Z., Backes, M., Wang, T., and Zhang, Y. Is Adversarial Training Really a Silver Bullet for Mitigating Data Poisoning? In: *International Conference on Learning Representations (ICLR)*. 2023.

[P3] Salem, A., Wen, R., Backes, M., Ma, S., and Zhang, Y. Dynamic Backdoor Attacks Against Machine Learning Models. In: *IEEE European Symposium on Security and Privacy (Euro S&P)*. IEEE, 2022, 703–718.

[P4] Wen, R., Backes, M., and Zhang, Y. Understanding Data Importance in Machine Learning Attacks: Does Valuable Data Pose Greater Harm? In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2025.

Other Published Papers of the Author

[S1] Wen, R., Yu, Y., Xie, X., and Zhang, Y. LEAF: A Faster Secure Search Algorithm via Localization, Extraction, and Reconstruction. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2020, 1219–1232.

[S2] Liu, Y., Wen, R., He, X., Salem, A., Zhang, Z., Backes, M., Cristofaro, E. D., Fritz, M., and Zhang, Y. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2022, 4525–4542.

[S3] Wu, Y., Wen, R., Backes, M., Berrang, P., Humbert, M., Shen, Y., and Zhang, Y. Quantifying Privacy Risks of Prompts in Visual Prompt Learning. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2024.

[S4] Zhang, R., Li, H., Wen, R., Jiang, W., Zhang, Y., Backes, M., Shen, Y., and Zhang, Y. Instruction Backdoor Attacks Against Customized LLMs. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2024.

[S5] Jiang, Y., Shen, X., Wen, R., Sha, Z., Chu, J., Liu, Y., Backes, M., and Zhang, Y. Games and Beyond: Analyzing the Bullet Chats of Esports Livestreaming. In: *International Conference on Web and Social Media (ICWSM)*. AAAI, 2024, 761–773.

[S6] Zhang, M., Yu, N., Wen, R., Backes, M., and Zhang, Y. Generated Distributions Are All You Need for Membership Inference Attacks Against Generative Models. In: *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2024, 4827–4837.

Other Technical Reports of the Author

[T1] Wen, R., Wang, T., Backes, M., Zhang, Y., and Salem, A. Last One Standing: A Comparative Analysis of Security and Privacy of Soft Prompt Tuning, LoRA, and In-Context Learning. *CoRR abs/2310.11397* (2023).

[T2] He, X., Wen, R., Wu, Y., Backes, M., Shen, Y., and Zhang, Y. Node-Level Membership Inference Attacks Against Graph Neural Networks. *CoRR abs/2102.05429* (2021).

[T3] Liu, Y., Wen, R., Backes, M., and Zhang, Y. Efficient Data-Free Model Stealing with Label Diversity. *CoRR abs/2404.00108* (2024).

Other references

[1] `https://lmsys.org/blog/2023-03-30-vicuna/`.

[2] `https://llama.meta.com/llama3/license/`.

[3] `https://www.copy.ai/`.

[4] `https://openai.com/blog/introducing-gpts`.

[5] `https://chatglm.cn/glms`.

[6] `https://platform.openai.com/docs/guides/chat/introduction`.

[7] `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`.

[8] `https://www.anthropic.com/news/claude-2-1-prompting`.

[9] `https://www.cs.toronto.edu/~kriz/cifar.html`.

[10] `https://www.kaggle.com/c/tiny-imagenet`.

[11] `https://www.apple.com/iphone/#face-id`.

[12] `http://yann.lecun.com/exdb/mnist/`.

[13] `https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data`.

[14] Agarwal, A., Dahleh, M. A., and Sarkar, T. A Marketplace for Data: An Algorithmic Solution. In: *Entertainment Computing (EC)*. ACM, 2019, 701–726.

[15] Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In: *International Conference on Machine Learning (ICML)*. PMLR, 2018, 274–283.

[16] Bagdasaryan, E. and Shmatikov, V. Blind Backdoors in Deep Learning Models. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2021, 1505–1521.

[17] Barni, M., Kallas, K., and Tondi, B. A New Backdoor Attack in CNNS by Training Set Corruption Without Label Poisoning. In: *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, 101–105.

[18] Biggio, B., Nelson, B., and Laskov, P. Support Vector Machines Under Adversarial Label Noise. In: *Asian Conference on Machine Learning (ACML)*. JMLR, 2011, 97–112.

[19] Biggio, B., Nelson, B., and Laskov, P. Poisoning Attacks against Support Vector Machines. In: *International Conference on Machine Learning (ICML)*. icml.cc / Omnipress, 2012.

[20] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language Models are Few-Shot Learners. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[21] Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramèr, F. Membership Inference Attacks From First Principles. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022, 1897–1914.

[22] Carlini, N., Jagielski, M., Choquette-Choo, C. A., Paleka, D., Pearce, W., Anderson, H. S., Terzis, A., Thomas, K., and Tramèr, F. Poisoning Web-Scale Training Datasets is Practical. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2024, 407–425.

[23] Carlini, N., Jagielski, M., and Mironov, I. Cryptanalytic Extraction of Neural Network Models. In: *Annual International Cryptology Conference (CRYPTO)*. Springer, 2020, 189–218.

[24] Carlini, N., Jagielski, M., Zhang, C., Papernot, N., Terzis, A., and Tramèr, F. The Privacy Onion Effect: Memorization is Relative. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022.

[25] Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2019, 267–284.

[26] Carlini, N. and Terzis, A. Poisoning and Backdooring Contrastive Learning. In: *International Conference on Learning Representations (ICLR)*. 2022.

[27] Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. Extracting Training Data from Large Language Models. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2021, 2633–2650.

[28] Chen, S., Kahla, M., Jia, R., and Qi, G. Knowledge-Enriched Distributional Model Inversion Attacks. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021, 16158–16167.

[29] Chen, X., Salem, A., Backes, M., Ma, S., Shen, Q., Wu, Z., and Zhang, Y. BadNL: Backdoor Attacks Against NLP Models with Semantic-preserving Improvements. In: *Annual Computer Security Applications Conference (ACSAC)*. ACSAC, 2021, 554–569.

[30] Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR abs/1712.05526* (2017).

[31] Choo, C. A. C., Tramèr, F., Carlini, N., and Papernot, N. Label-Only Membership Inference Attacks. In: *International Conference on Machine Learning (ICML)*. PMLR, 2021, 1964–1974.

[32] Devries, T. and Taylor, G. W. Improved Regularization of Convolutional Neural Networks with Cutout. *CoRR abs/1708.04552* (2017).

[33] Doan, B. G., Abbasnejad, E., and Ranasinghe, D. C. Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems. In: *Annual Computer Security Applications Conference (ACSAC)*. ACM, 2020, 897–912.

[34] Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., Li, L., and Sui, Z. A Survey on In-context Learning. *CoRR abs/2301.00234* (2023).

[35] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: *International Conference on Learning Representations (ICLR)*. 2021.

[36] Duan, H., Dziedzic, A., Yaghini, M., Papernot, N., and Boenisch, F. On the Privacy Risk of In-context Learning. In: *Workshop on Trustworthy Natural Language Processing (TrustNLP)*. 2023.

[37] Duddu, V., Szyller, S., and Asokan, N. SHAPr: An Efficient and Versatile Membership Privacy Risk Metric for Machine Learning. *CoRR abs/2112.02230* (2021).

[38] Evtimov, I., Covert, I., Kusupati, A., and Kohno, T. Disrupting Model Training with Adversarial Shortcuts. *CoRR abs/2106.06654* (2021).

[39] Fowl, L., Chiang, P., Goldblum, M., Geiping, J., Bansal, A., Czaja, W., and Goldstein, T. Preventing Unauthorized Use of Proprietary Data: Poisoning for Secure Dataset Release. *CoRR abs/2103.02683* (2021).

[40] Fowl, L., Goldblum, M., Chiang, P., Geiping, J., Czaja, W., and Goldstein, T. Adversarial Examples Make Strong Poisons. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2021, 30339–30351.

[41] Fredrikson, M., Jha, S., and Ristenpart, T. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2015, 1322–1333.

[42] Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., and Ristenpart, T. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2014, 17–32.

[43] Fu, S., He, F., Liu, Y., Shen, L., and Tao, D. Robust Unlearnable Examples: Protecting Data Against Adversarial Learning. In: *International Conference on Learning Representations (ICLR)*. 2022.

[44] Fu, W., Wang, H., Gao, C., Liu, G., Li, Y., and Jiang, T. Practical Membership Inference Attacks against Fine-tuned Large Language Models via Self-prompt Calibration. *CoRR abs/2311.06062* (2023).

[45] Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., and Nepal, S. STRIP: A Defence Against Trojan Attacks on Deep Neural Networks. In: *Annual Computer Security Applications Conference (ACSAC)*. ACM, 2019, 113–125.

[46] Geiping, J., Fowl, L. H., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching. In: *International Conference on Learning Representations (ICLR)*. 2021.

[47] Ghorbani, A. and Zou, J. Y. Data Shapley: Equitable Valuation of Data for Machine Learning. In: *International Conference on Machine Learning (ICML)*. PMLR, 2019, 2242–2251.

[48] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In: *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2014, 2672–2680.

[49] Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. In: *International Conference on Learning Representations (ICLR)*. 2015.

[50] Gu, T., Dolan-Gavitt, B., and Grag, S. Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR abs/1708.06733* (2017).

[51] Haim, N., Vardi, G., Yehudai, G., Shamir, O., and Irani, M. Reconstructing Training Data from Trained Neural Networks. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022.

[52] He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, 770–778.

[53] Hu, H., Salcic, Z., Sun, L., Dobbie, G., Yu, P. S., and Zhang, X. Membership Inference Attacks on Machine Learning: A Survey. *ACM Computing Surveys* (2021).

[54] Huang, H., Ma, X., Erfani, S. M., Bailey, J., and Wang, Y. Unlearnable Examples: Making Personal Data Unexploitable. In: *International Conference on Learning Representations (ICLR)*. 2021.

[55] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial Examples Are Not Bugs, They Are Features. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2019, 125–136.

[56] Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., and Papernot, N. High Accuracy and High Fidelity Extraction of Neural Networks. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2020, 1345–1362.

[57] Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018, 19–35.

[58] Jagielski, M., Ullman, J., and Oprea, A. Auditing Differentially Private Machine Learning: How Private is Private SGD? In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[59] Jain, S., Keshava, V., Sathyendra, S. M., Fernandes, P., Liu, P., Neubig, G., and Zhou, C. Multi-Dimensional Evaluation of Text Summarization with In-Context Learning. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, 2023, 8487–8495.

[60] Jayaraman, B. and Evans, D. Are Attribute Inference Attacks Just Imputation? In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2022, 1569–1582.

[61] Jia, J., Liu, Y., and Gong, N. Z. BadEncoder: Backdoor Attacks to Pre-trained Encoders in Self-Supervised Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.

[62] Jia, R., Dao, D., Wang, B., Hubis, F. A., Gürel, N. M., Li, B., Zhang, C., Spanos, C. J., and Song, D. Efficient Task-Specific Data Valuation for Nearest Neighbor Algorithms. *Proceedings of the VLDB Endowment* (2019).

[63] Jia, R., Wu, F., Sun, X., Xu, J., Dao, D., Kailkhura, B., Zhang, C., Li, B., and Song, D. Scalability vs. Utility: Do We Have To Sacrifice One for the Other in Data Importance Quantification? In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021, 8239–8247.

[64] Kariyappa, S., Prakash, A., and Qureshi, M. K. MAZE: Data-Free Model Stealing Attack Using Zeroth-Order Gradient Estimation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021, 13814–13823.

[65] Kaya, M. and Bilge, H. S. Deep Metric Learning: A Survey. *Symmetry* (2019).

[66] Koh, P. W. and Liang, P. Understanding Black-box Predictions via Influence Functions. In: *International Conference on Machine Learning (ICML)*. PMLR, 2017, 1885–1894.

[67] Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial Machine Learning at Scale. In: *International Conference on Learning Representations (ICLR)*. 2017.

[68] Kwon, Y. and Zou, J. Beta Shapley: a Unified and Noise-reduced Data Valuation Framework for Machine Learning. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR, 2022, 8780–8802.

[69] Leino, K. and Fredrikson, M. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2020, 1605–1622.

[70] Li, C., Pang, R., Xi, Z., Du, T., Ji, S., Yao, Y., and Wang, T. An Embarrassingly Simple Backdoor Attack on Self-supervised Learning. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2023, 4344–4355.

[71] Li, J., Li, N., and Ribeiro, B. Membership Inference Attacks and Defenses in Classification Models. In: *ACM Conference on Data and Application Security and Privacy (CODASPY)*. ACM, 2021, 5–16.

[72] Li, X. and Roth, D. Learning Question Classifiers. In: *International Conference on Computational Linguistics (COLING)*. ACL, 2002.

[73] Li, Y., Li, Y., Wu, B., Li, L., He, R., and Lyu, S. Invisible Backdoor Attack with Sample-Specific Triggers. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021, 16443–16452.

[74] Li, Z., Liu, Y., He, X., Yu, N., Backes, M., and Zhang, Y. Auditing Membership Leakages of Multi-Exit Networks. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2022, 1917–1931.

[75] Li, Z. and Zhang, Y. Membership Leakage in Label-Only Exposures. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, 880–895.

[76] Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual Instruction Tuning. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2023.

[77] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the Middle: How Language Models Use Long Contexts. *CoRR abs/2307.03172* (2023).

[78] Liu, W., Shen, X., Pun, C., and Cun, X. Explicit Visual Prompting for Low-Level Structure Segmentations. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023.

[79] Liu, Y., Lee, W.-C., Tao, G., Ma, S., Aafer, Y., and Zhang, X. ABS: Scanning Neural Networks for Back-Doors by Artificial Brain Stimulation. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, 1265–1282.

[80] Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning Attack on Neural Networks. In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2018.

[81] Liu, Y., Zhao, Z., Backes, M., and Zhang, Y. Membership Inference Attacks by Exploiting Loss Trajectory. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2022, 2085–2098.

[82] Liu, Y., Ma, X., Bailey, J., and Lu, F. Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks. In: *European Conference on Computer Vision (ECCV)*. Springer, 2020, 182–199.

[83] Liu, Z., Zhao, Z., Kolmus, A., Berns, T., Laarhoven, T. van, Heskes, T., and Larson, M. A. Going Grayscale: The Road to Understanding and Improving Unlearnable Examples. *CoRR abs/2111.13244* (2021).

[84] Liu, Z., Luo, P., Wang, X., and Tang, X. Deep Learning Face Attributes in the Wild. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, 3730–3738.

[85] Lundberg, S. M. and Lee, S. A Unified Approach to Interpreting Model Predictions. In: *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2017, 4765–4774.

[86] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. In: *International Conference on Learning Representations (ICLR)*. 2018.

[87] Mathur, Y., Rangreji, S., Kapoor, R., Palavalli, M., Bertsch, A., and Gormley, M. R. SummQA at MEDIQA-Chat 2023: In-Context Learning with GPT-4 for Medical Summarization. In: *Clinical Natural Language Processing Workshop (ClinicalNLP)*. ACL, 2023, 490–502.

[88] Mattern, J., Mireshghallah, F., Jin, Z., Schölkopf, B., Sachan, M., and Berg-Kirkpatrick, T. Membership Inference Attacks against Language Models via Neighbourhood Comparison. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, 2023, 11330–11343.

[89] Mehnaz, S., Dibbo, S. V., Kabir, E., Li, N., and Bertino, E. Are Your Sensitive Attributes Private? Novel Model Inversion Attribute Inference Attacks on Classification Models. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2022, 4579–4596.

[90] Meister, C., Wiher, G., and Cotterell, R. On Decoding Strategies for Neural Text Generators. *Transactions of the Association for Computational Linguistics* (2022).

[91] Melis, L., Song, C., Cristofaro, E. D., and Shmatikov, V. Exploiting Unintended Feature Leakage in Collaborative Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, 497–512.

[92] Moor, M., Huang, Q., Wu, S., Yasunaga, M., Zakka, C., Dalmia, Y., Reis, E. P., Rajpurkar, P., and Leskovec, J. Med-Flamingo: a Multimodal Medical Few-shot Learner. *CoRR abs/2307.15189* (2023).

[93] Moslem, Y., Haque, R., and Way, A. Adaptive Machine Translation with Large Language Models. *CoRR abs/2301.13294* (2023).

[94] Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization. In: *Workshop on Security and Artificial Intelligence (AISec)*. ACM, 2017, 27–38.

[95] Nasr, M., Shokri, R., and Houmansadr, A. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, 1021–1035.

[96] Nasr, M., Song, S., Thakurta, A., Papernot, N., and Carlini, N. Adversary Instantiation: Lower Bounds for Differentially Private Machine Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.

[97] Nguyen, T. A. and Tran, A. Input-Aware Dynamic Backdoor Attack. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[98] Nguyen, T. A. and Tran, A. T. WaNet - Imperceptible Warping-based Backdoor Attack. In: *International Conference on Learning Representations (ICLR)*. 2021.

[99] OpenAI. GPT-4 Technical Report. *CoRR abs/2303.08774* (2023).

[100] Orekondy, T., Schiele, B., and Fritz, M. Knockoff Nets: Stealing Functionality of Black-Box Models. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, 4954–4963.

[101] Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. Bag of Tricks for Adversarial Training. In: *International Conference on Learning Representations (ICLR)*. 2021.

[102] Papernot, N., McDaniel, P. D., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical Black-Box Attacks Against Machine Learning. In: *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2017, 506–519.

[103] Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. TRAK: Attributing Model Behavior at Scale. In: *International Conference on Machine Learning (ICML)*. PMLR, 2023, 27074–27113.

[104] Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M., and Sutskever, I. Better Language Models and Their Implications. In: *OpenAI Blog*. 2019.

[105] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI blog* (2019).

[106] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* (2020).

[107] Rakin, A. S., He, Z., and Fan, D. TBT: Targeted Neural Network Attack with Bit Trojan. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 13198–13207.

[108] Reimers, N. and Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. ACL, 2019, 3980–3990.

[109] Ribeiro, M. T., Singh, S., and Guestrin, C. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2016, 1135–1144.

[110] Rizwan, N., Bhaskar, P., Das, M., Majhi, S. S., Saha, P., and Mukherjee, A. Zero shot VLMs for hate meme detection: Are we there yet? *CoRR abs/2402.12198* (2024).

[111] Saha, A., Subramanya, A., and Pirsiavash, H. Hidden Trigger Backdoor Attacks. In: *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2020, 11957–11965.

[112] Salem, A., Bhattacharya, A., Backes, M., Fritz, M., and Zhang, Y. Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2020, 1291–1308.

[113] Salem, A., Cherubin, G., Evans, D., Köpf, B., Paverd, A., Suri, A., Tople, S., and Béguelin, S. Z. SoK: Let the Privacy Games Begin! A Unified Treatment of Data Inference Privacy in Machine Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2023, 327–345.

[114] Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[115] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, 4510–4520.

[116] Sanyal, S., Addepalli, S., and Babu, R. V. Towards Data-Free Model Stealing in a Hard Label Setting. *CoRR abs/2204.11022* (2022).

[117] Seegmiller, P., Gatto, J., Basak, M., Cook, D. J., Ghasemzadeh, H., Stankovic, J. A., and Preum, S. M. The Scope of In-Context Learning for the Extraction of Medical Temporal Constraints. *CoRR abs/2303.09366* (2023).

[118] Segura, P. S., Singla, V., Fowl, L., Geiping, J., Goldblum, M., Jacobs, D., and Goldstein, T. Poisons that are learned faster are more effective. *CoRR abs/2204.08615* (2022).

[119] Segura, P. S., Singla, V., Geiping, J., Goldblum, M., Goldstein, T., and Jacobs, D. W. Autoregressive Perturbations for Data Poisoning. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022.

[120] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, 618–626.

[121] Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2018, 6103–6113.

[122] Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J. P., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2019, 3353–3364.

[123] Shapley, L. S. A Value for n-person Games. *Contributions to the Theory of Games* (1953).

[124] Shi, W., Ajith, A., Xia, M., Huang, Y., Liu, D., Blevins, T., Chen, D., and Zettlemoyer, L. Detecting Pretraining Data from Large Language Models. *CoRR abs/2310.16789* (2023).

[125] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, 3–18.

[126] Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *International Conference on Learning Representations (ICLR)*. 2015.

[127] Song, C. and Raghunathan, A. Information Leakage in Embedding Models. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2020, 377–390.

[128] Song, C. and Shmatikov, V. Overlearning Reveals Sensitive Attributes. In: *International Conference on Learning Representations (ICLR)*. 2020.

[129] Song, H., Kim, M., Park, D., and Lee, J. Learning from Noisy Labels with Deep Neural Networks: A Survey. *CoRR abs/2007.08199* (2020).

[130] Song, L. and Mittal, P. Systematic Evaluation of Privacy Risks of Machine Learning Models. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.

[131] Song, L., Shokri, R., and Mittal, P. Privacy Risks of Securing Machine Learning Models against Adversarial Examples. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, 241–257.

[132] Struppek, L., Hintersdorf, D., Correia, A. D. A., Adler, A., and Kersting, K. Plug & Play Attacks: Towards Robust and Flexible Model Inversion Attacks. In: *International Conference on Machine Learning (ICML)*. PMLR, 2022, 20522–20545.

[133] Tang, R., Du, M., Liu, N., Yang, F., and Hu, X. An Embarrassingly Simple Approach for Trojan Attack in Deep Neural Networks. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2020, 218–228.

[134] Tang, X., Shin, R., Inan, H. A., Manoel, A., Mireshghallah, F., Lin, Z., Gopi, S., Kulkarni, J., and Sim, R. Privacy-Preserving In-Context Learning with Differentially Private Few-Shot Generation. *CoRR abs/2309.11765* (2023).

[135] Tao, L., Feng, L., Wei, H., Yi, J., Huang, S., and Chen, S. Can Adversarial Training Be Manipulated By Non-Robust Features? In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022.

[136] Tao, L., Feng, L., Yi, J., Huang, S.-J., and Chen, S. Better Safe Than Sorry: Preventing Delusive Adversaries with Adversarial Training. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2021, 16209–16225.

[137] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and Efficient Foundation Language Models. *CoRR abs/2302.13971* (2023).

[138] Tramèr, F., Carlini, N., Brendel, W., and Madry, A. On Adaptive Attacks to Adversarial Example Defenses. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[139] Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble Adversarial Training: Attacks and Defenses. In: *International Conference on Learning Representations (ICLR)*. 2017.

[140] Tramèr, F., Shokri, R., Joaquin, A. S., Le, H., Jagielski, M., Hong, S., and Carlini, N. Truth Serum: Poisoning Machine Learning Models to Reveal Their Secrets. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2022.

[141] Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing Machine Learning Models via Prediction APIs. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2016, 601–618.

[142] Truong, J., Maini, P., Walls, R. J., and Papernot, N. Data-Free Model Extraction. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021, 4771–4780.

[143] Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, 707–723.

[144] Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[145] Wang, K., Fu, Y., Li, K., Khisti, A., Zemel, R. S., and Makhzani, A. Variational Model Inversion Attacks. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2021, 9706–9719.

[146] Wang, S., Zhao, Z., Ouyang, X., Wang, Q., and Shen, D. ChatCAD: Interactive Computer-Aided Diagnosis on Medical Image using Large Language Models. *CoRR abs/2302.07257* (2023).

[147] Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving Adversarial Robustness Requires Revisiting Misclassified Examples. In: *International Conference on Learning Representations (ICLR)*. 2020.

[148] Wang, Z., Xie, Q., Ding, Z., Feng, Y., and Xia, R. Is ChatGPT a Good Sentiment Analyzer? A Preliminary Study. *CoRR abs/2304.04339* (2023).

[149] Wang, Z., Wang, Y., and Wang, Y. Fooling Adversarial Training with Inducing Noise. *CoRR abs/2111.10130* (2021).

[150] Wu, B., Chen, H., Zhang, M., Zhu, Z., Wei, S., Yuan, D., and Shen, C. BackdoorBench: A Comprehensive Benchmark of Backdoor Learning. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022.

[151] Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C. A., and Li, B. Detecting AI Trojans Using Meta Neural Analysis. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.

[152] Yang, Z., Zhang, J., Chang, E.-C., and Liang, Z. Neural Network Inversion in Adversarial Setting via Background Knowledge Alignment. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, 225–240.

[153] Yao, Y., Li, H., Zheng, H., and Zhao, B. Y. Latent Backdoor Attacks on Deep Neural Networks. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, 2041–2055.

[154] Ye, J., Maddi, A., Murakonda, S. K., Bindschaedler, V., and Shokri, R. Enhanced Membership Inference Attacks against Machine Learning Models. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2022, 3093–3106.

[155] Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In: *IEEE Computer Security Foundations Symposium (CSF)*. IEEE, 2018, 268–282.

[156] Yin, H., Molchanov, P., Alvarez, J. M., Li, Z., Mallya, A., Hoiem, D., Jha, N. K., and Kautz, J. Dreaming to Distill: Data-Free Knowledge Transfer via DeepInversion. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 8712–8721.

[157] Yu, D., Zhang, H., Chen, W., Yin, J., and Liu, T. Availability Attacks Create Shortcuts. In: *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2022, 2367–2376.

[158] Yun, S., Han, D., Chun, S., Oh, S. J., Yoo, Y., and Choe, J. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2019, 6022–6031.

[159] Zeng, Y., Park, W., Mao, Z. M., and Jia, R. Rethinking the Backdoor Attacks' Triggers: A Frequency Perspective. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021, 16453–16461.

[160] Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically Principled Trade-off between Robustness and Accuracy. In: *International Conference on Machine Learning (ICML)*. PMLR, 2019, 7472–7482.

[161] Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization. In: *International Conference on Learning Representations (ICLR)*. 2018.

[162] Zhang, X., Zhao, J., and LeCun, Y. Character-level Convolutional Networks for Text Classification. In: *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2015, 649–657.

[163] Zhang, Y. and Ippolito, D. Prompts Should not be Seen as Secrets: Systematically Measuring Prompt Extraction Attack Success. *CoRR abs/2307.06865* (2023).

[164] Zhang, Y., Jia, R., Pei, H., Wang, W., Li, B., and Song, D. The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 250–258.

[165] Zhao, S., Ma, X., Zheng, X., Bailey, J., Chen, J., and Jiang, Y.-G. Clean-Label Backdoor Attacks on Video Recognition Models. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 14443–144528.

[166] Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. Calibrate Before Use: Improving Few-shot Performance of Language Models. In: *International Conference on Machine Learning (ICML)*. PMLR, 2021, 12697–12706.

[167] Zhong, N., Qian, Z., and Zhang, X. Imperceptible Backdoor Attack: From Input Space to Feature Representation. In: *International Joint Conferences on Artifical Intelligence (IJCAI)*. IJCAI, 2022, 1736–1742.

[168] Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. Large Language Models are Human-Level Prompt Engineers. In: *International Conference on Learning Representations (ICLR)*. 2023.

[169] Zhu, C., Huang, W. R., Li, H., Taylor, G., Studer, C., and Goldstein, T. Transferable Clean-label Poisoning Attacks on Deep Neural Nets. In: *International Conference on Machine Learning (ICML)*. JMLR, 2019, 7614–7623.