TUTORIAL

A Survey on Model Extraction Attacks and Defenses for Large Language Models

KAIXIANG ZHAO, University of Notre Dame, Notre Dame, IN, United States

LINCAN LI, Florida State University, Tallahassee, FL, United States

KAIZE DING, Northwestern University, Evanston, IL, United States

NEIL ZHENQIANG GONG, Duke University, Durham, NC, United States

YUE ZHAO, University of Southern California, Los Angeles, CA, United States

YUSHUN DONG, Florida State University, Tallahassee, FL, United States

Open Access Support provided by:

Florida State University

Duke University

University of Southern California

University of Notre Dame

Northwestern University

Published: 03 August 2025

Citation in BibTeX format

KDD '25: The 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining
August 3 - 7, 2025
Toronto ON, Canada

Conference Sponsors:
SIGMOD
SIGKDD

A Survey on Model Extraction Attacks and Defenses for Large Language Models

Kaixiang Zhao
kzhao5@nd.edu
University of Notre Dame
South Bend, Indiana, USA

Lincan Li*
ll24bb@fsu.edu
Florida State University
Tallahassee, Florida, USA

Kaize Ding
kaize.ding@northwestern.edu
Northwestern University
Evanston, Illinois, USA

Neil Zhenqiang Gong
neil.gong@duke.edu
Duke University
Durham, North Carolina, USA

Yue Zhao
yzhao010@usc.edu
University of Southern California
Los Angeles, California, USA

Yushun Dong†
yushun.dong@fsu.edu
Florida State University
Tallahassee, Florida, USA

Abstract

Model extraction attacks pose significant security threats to deployed language models, potentially compromising intellectual property and user privacy. This survey provides a comprehensive taxonomy of LLM-specific extraction attacks and defenses, categorizing attacks into functionality extraction, training data extraction, and prompt-targeted attacks. We analyze various attack methodologies including API-based knowledge distillation, direct querying, parameter recovery, and prompt stealing techniques that exploit transformer architectures. We then examine defense mechanisms organized into model protection, data privacy protection, and prompt-targeted strategies, evaluating their effectiveness across different deployment scenarios. We propose specialized metrics for evaluating both attack effectiveness and defense performance, addressing the specific challenges of generative language models. Through our analysis, we identify critical limitations in current approaches and propose promising research directions, including integrated attack methodologies and adaptive defense mechanisms that balance security with model utility. This work serves NLP researchers, ML engineers, and security professionals seeking to protect language models in production environments.

CCS Concepts

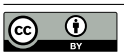
• **Computing methodologies** → **Artificial intelligence**.

Keywords

Large Language Models, Model Extraction Attacks, LLM Security

ACM Reference Format:
Kaixiang Zhao, Lincan Li, Kaize Ding, Neil Zhenqiang Gong, Yue Zhao, and Yushun Dong. 2025. A Survey on Model Extraction Attacks and Defenses for Large Language Models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3711896.3736573

*Co-first author
†Corresponding author.



KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3736573

1 Introduction

A growing number of large language models (LLMs), often developed with substantial investments by companies like OpenAI and Google, bear significant commercial value and advanced AI capabilities [1, 3, 19, 45, 60, 64]. The considerable resources invested and the imperative to protect valuable intellectual property typically restrict the public release of these state-of-the-art models [1, 19, 64]. Consequently, these models are commonly made accessible via Machine-Learning-as-a-Service (MLaaS) platforms, which provide API-based query access to pre-trained models and manage the underlying infrastructure [27, 31, 54, 59, 85]. However, these platforms also pose significant security challenges, primarily stemming from the potential for unauthorized replication of model functionality or theft of underlying intellectual property. Such activities, collectively referred to as model extraction attacks (MEAs)—or model stealing attacks—have emerged as a critical threat to private-owned LLMs offered via these service interfaces [2, 23, 30, 51, 52, 65]. Recent research demonstrates that private-owned models can be successfully replicated, and their performance on relevant benchmarks even surpassed, through systematic distillation approaches. These methods often require small amounts of training data, and empirical evidence further supports this phenomenon [16, 23, 25, 34, 42, 55, 81]. Findings from these studies suggest that many well-known closed-source and even some open-source models exhibit high degrees of distillation from leading commercial models [4, 34, 71]. Some replicated models also show response patterns and identity characteristics strikingly similar to those of the original providers [4, 34, 71]. This threat is further validated by ongoing allegations that some AI developers are potentially employing MEAs against such private-owned commercial models to build their own powerful models, which reportedly match the performance of leading commercial models at a significantly lower cost [19, 34, 63, 80] These controversies, even when specific claims are contested, have intensified debate on model extraction against LLMs. Furthermore, the unique characteristics of LLMs, such as their scale, generative capabilities, introduce vulnerabilities that attackers can exploit with increasing sophistication [70, 75, 84, 92]. These attacks typically exploit the model's query APIs to replicate functionality without authorization,

leveraging techniques such as systematic API queries [5, 33, 58, 78], parameter inference [56, 66], training data extraction [7, 18], and prompt manipulation [53, 61, 62].

To tackle these threats, researchers have proposed various defense strategies that achieve protection at different levels.. These include architectural safeguards such as watermarking [22, 40, 89, 90, 93] and access control mechanisms [37, 57, 72], training-time interventions like robust architecture design [20, 51, 74], and inference-time protections including output sanitization [9, 28, 73] and query monitoring [36, 79, 86]. Despite these advances, significant challenges in LLM extraction security persist, including balancing model accessibility with security and utility, protecting diverse LLM architectures and deployments, and countering evolving attack strategies [13, 14, 22, 67, 73, 84].

Core Contributions. This survey makes four key contributions to the domain of LLM extraction security: First, we provide a clear taxonomy of LLM extraction attacks, detailing transformer-specific vulnerabilities and generative model risks. Second, we analyze current defense mechanisms, examining architectural, output control, and monitoring strategies to show their strengths and weaknesses. Third, we examine key security-utility tradeoffs, showing how defenses affect model quality and usability, to help balance security with performance. Fourth, we propose an evaluation framework with specific metrics to measure attack success and defense strength, suited for generative models. Finally, this work identifies research gaps and suggests future directions in this field.

Intended Audience. This survey targets three primary audiences: (1) NLP researchers investigating security vulnerabilities and protections for language models; (2) ML engineers and practitioners responsible for deploying and securing LLMs in production environments; and (3) Security professionals tasked with protecting organizational AI assets and intellectual property. By bridging theoretical foundations with practical implementation considerations, this work provides actionable insights for both academic research and industrial applications in LLM security.

2 Preliminaries

2.1 Model Extraction Basics

Model Extraction Attacks (MEAs) on Large Language Models (LLMs) aim to illicitly replicate the functionality, knowledge, or parameters of a target model. We first formally define the threat model.

Definition 1 (Threat Model). *Given a target private-owned LLM M_T, black-box query access (allowing submission of input prompts x to receive outputs $y = M_T(x)$ without internal model access), and a potential query budget B. The attacker aims to train a surrogate model M_S that replicates the input-output functional pattern of M_T.*

We further show Figure 1 to provide a better understanding for the full life cycle of MEA attacks. More specifically, in the LLM context, the target model M typically processes input text x (often structured as a prompt) and generates output text $y = M(x)$. An adversary with query access to M constructs an extracted dataset $D_{ext} = \{(x_i, M(x_i)) | x_i \sim X, 1 \le i \le N\}$, where X represents the input space of possible prompts and N is the number of queries. Using this dataset, the adversary trains a surrogate model M' that approximates the behavior of M by minimizing a loss function

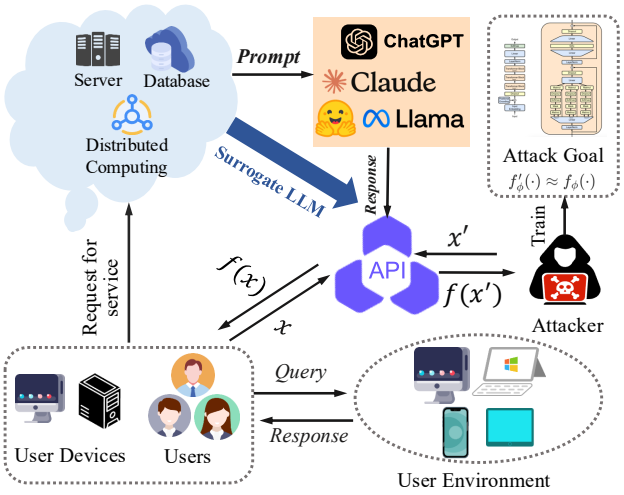


Figure 1: Illustration of the Model Extraction Attack pipeline on Large Language Models.

$M' = \arg\min_{M' \in H} \sum_{(x,y) \in D_{ext}} \mathcal{L}(M'(x), y)$, where H represents the hypothesis space of possible models and $\mathcal{L}$ measures the discrepancy between outputs.

2.2 Extraction Attack Process

The LLM extraction process involves three key phases: query generation, response collection, and surrogate model training. In query generation, adversaries design prompts that effectively probe the model's capabilities across multiple domains. During response collection, these queries are submitted to the target model through its API service, and outputs are recorded for training data.

The surrogate model training phase leverages collected input-output pairs to train a model that mimics the target LLM. This may involve fine-tuning a smaller pre-trained model, distilling knowledge into a more compact architecture, or training from scratch. The training process must address challenges specific to language modeling, including handling variable-length sequences and replicating the specific characteristics of the target model. The adversary's ultimate goal is to develop a model $f'_\phi(\cdot)$ that closely approximates the target model $f_\theta(\cdot)$.

3 Taxonomy

Figure 2 presents our taxonomy of model extraction attacks and defenses for LLMs. We categorize MEA attacks against LLMs into three categories: functionality extraction targeting model behavior replication, training data extraction recovering training examples, and prompt-targeted attacks stealing valuable prompts. Defense mechanisms are similarly organized: model protection strategies secure architecture and outputs, data privacy protection safeguards training data, and prompt protection protects private-owned prompts and monitors queries. For evaluation, we identify metrics covering attack effectiveness and defense performance.

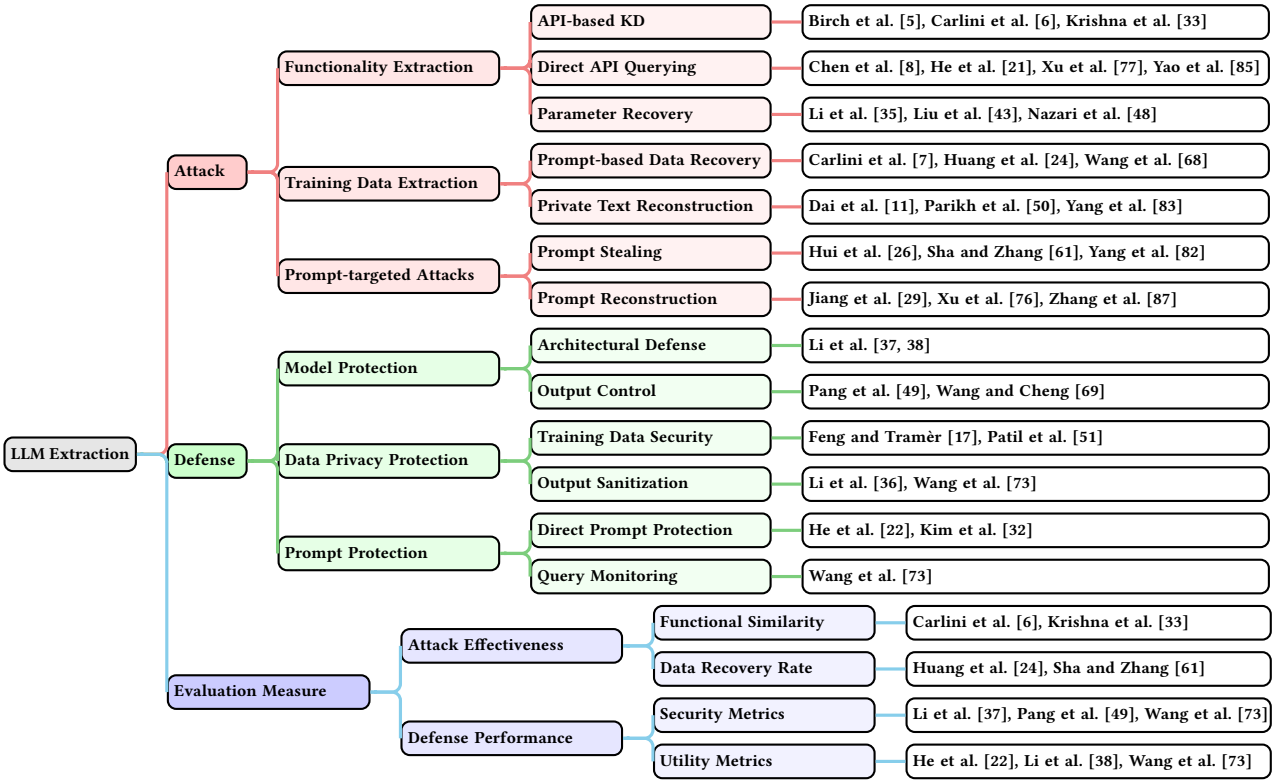


Figure 2: A Taxonomy of Model Extraction Attacks & Defenses on Large Language Models.

4 Model Extraction Attacks

Model extraction attacks against LLMs present significant threats to intellectual property, and user privacy. This section details three primary attack types: *Model Functionality Extraction*, *Training Data Extraction*, and *Prompt-targeted Attacks*.

4.1 Model Functionality Extraction

Model functionality extraction attacks aim to replicate a target LLM's capabilities through black-box query interactions. Given a target language model $M : X \rightarrow Y$, the adversary collects a dataset $D_{ext} = \{(x_i, M(x_i)) | x_i \sim X, 1 \leq i \leq N\}$ through N queries, constrained by budget B. They then train a surrogate model

$$M' = \arg\min_{M' \in \mathcal{H}} \sum_{(x,y) \in D_{ext}} \mathcal{L}(M'(x), y), \tag{1}$$

where $\mathcal{L}$ measures output discrepancy. Model functionality extraction attacks are categorized into three types: *API-based knowledge distillation*, *direct API querying*, and *parameter/architecture recovery*. API-based knowledge distillation systematically transfers the overall functionality of a target LLM by querying it with a broad and diverse set of inputs to create a comprehensive dataset of input-output pairs. This dataset is then used to train a surrogate LLM that replicates the target LLM's behavior. In contrast, direct API querying focuses on crafting specific prompts to extract particular behaviors or capabilities from the model, often targeting narrow functionalities or specific response patterns. Parameter/architecture recovery differs from both by attempting to reverse-engineer internal components, such as weights.

4.1.1 API-based Knowledge Distillation. API-based knowledge distillation attacks leverage authorized access to target LLMs through their public interfaces to systematically transfer knowledge to an attacker-controlled model. Unlike traditional knowledge distillation requiring direct access to model parameters, these attacks operate purely through input-output interactions. Krishna et al. [33] demonstrated the feasibility of model extraction against BERT-based APIs, which required only query access to create functional clones with competitive performance. Their approach revealed a fundamental vulnerability: as commercial models standardize on transformer architectures, the technical barriers to model extraction diminish dramatically. Birch et al. [5] proposed a model leeching attack specifically targeting LLMs that achieved high functional similarity with limited computational resources. Most concerning, Carlini et al. [6] successfully demonstrated extraction attacks against production-scale language models, which showed that even billion-parameter commercial systems remain vulnerable. Their approach requires no insider access, which highlights a widening asymmetry between the immense resources required to develop frontier models and the relatively modest investment needed to steal their capabilities. The core insight is that knowledge distillation attacks become practical as the architecture gap between commercial models narrows.

4.1.2 Direct API Querying. Direct API querying attacks use carefully designed prompts to efficiently extract model capabilities by

analyzing the model's responses. Yao et al. [85] established early foundations by analyzing vulnerabilities in MLaaS systems. Their work showed that significant model functionality could be extracted even with limited queries. This vulnerability has increased with modern LLMs, as demonstrated by Krishna et al. [33], who showed that strategic querying can rapidly approximate the functionality of BERT-based models. He et al. [21] further proved that model functionality can be extracted, and the resulting clone exhibits similar vulnerability to adversarial examples. This suggests that fundamental behavioral characteristics transfer during extraction. Chen et al. [8] expanded this research by demonstrating that BERT-based APIs are vulnerable to dual-purpose attacks, which simultaneously extract model functionality and user attributes. These findings compound the security implications of extraction attacks. Modern API-based attacks are distinguished by their increasing query efficiency. Early techniques required millions of queries, but newer methods use information-theoretic approaches to identify high-value prompts that reveal maximum model behavior with minimal interaction. Xu et al. [77] highlighted this efficiency trend with their imitation attack, which enables students to surpass teachers through optimized query strategies. This evolution presents a concerning trend. As attack efficiency improves, detection becomes more difficult, creating a security gap where model functionality can be extracted before defenses are engaged.

4.1.3 Parameter/Architecture Recovery. Parameter and architecture recovery attacks are among the most technically challenging forms of model extraction. These attacks aim to reverse-engineer specific model components, such as weights or structural designs, rather than replicate behavior. They are particularly concerning for edge-deployed models, such as those hosted on local devices like smartphones or IoT hardware, where adversaries may have physical access. Li et al. [35] provided theoretical foundations for gradient leakage in transformer architectures. Their work demonstrates that attention mechanisms leak more information than traditional neural networks. Nazari et al. [48] exploited this vulnerability in practical attacks. They presented a fingerprinting methodology that reveals critical architectural details of edge-deployed LLMs. Liu et al. [43] showed that even self-supervised learning encoders can be stolen, which compromises the foundational components of modern language models. Transformer-based LLMs, despite their complexity, exhibit structure-revealing patterns during inference. Self-attention mechanisms expose architectural information through output behavior, which can be systematically probed and analyzed. Complete parameter recovery remains impractical for billion-parameter models. However, partial recovery of key components creates vulnerabilities that enable targeted attacks or provide competitive advantages. This vulnerability increases in distributed computing environments, where adversaries can exploit expanded access to model components, as highlighted by Zhao et al. [92].

4.2 Training Data Extraction

Training data extraction attacks aim to recover specific data points from a model's training dataset, which exploits the model's tendency to memorize and reproduce examples from its training data [7]. These attacks are considered a form of model extraction because recovering the training data constitutes a direct extraction of potentially sensitive and private information from the model. The training data is the source of the model's learned patterns and behaviors. Given a model M trained on dataset D_{train}, an adversary crafts prompts $P = \{p_1, ..., p_N\}$ with budget $|P| \leq B$ to extract training data

$$E(M) = \{d \in D_{train} : \exists p \in P \text{ s.t. } \text{sim}(M(p), d) > \tau\}, \quad (2)$$

where $\text{sim}(\cdot, \cdot)$ measures similarity and τ is a threshold. We categorize training data extraction attacks into two types: *prompt-based data recovery* and *private text reconstruction*. Prompt-based data recovery focuses on retrieving memorized training examples using carefully crafted prompts, which target specific data patterns such as PII or rare text formats. In contrast, private text reconstruction goes beyond verbatim extraction by inferring and reconstructing sensitive information through techniques like activation inversion and canary extraction.

4.2.1 Prompt-based Data Recovery. Prompt-based data recovery attacks exploit LLMs' tendency to memorize training examples, which allows specific prompts to retrieve memorized data. Carlini et al. [7] demonstrated the practical feasibility of extracting exact training examples from GPT-2, which showed that models memorize and can be induced to reproduce specific sequences from their training data. Their work established that the success of such extractions increases with model size and complexity. This indicates that more advanced models may unintentionally increase privacy risks. Zhang et al. [91] refined this approach with their ETHICIST system, which uses loss-smoothed soft prompting and calibrated confidence estimation to improve the efficiency of targeting specific types of training data. Huang et al. [24] specifically investigated personal information leakage, which revealed that pre-trained LLMs frequently memorize and expose personally identifiable information (PII) when properly prompted. Their findings suggest that uniquely formatted data patterns, such as email addresses and phone numbers, are particularly vulnerable to extraction. This creates a direct privacy risk for individuals whose information appeared in the training data. Wang et al. [68] further reinforced this vulnerability by developing more precise extraction techniques that can distinguish between memorized and generated content. Average memorization rates may appear low in aggregate measurements. However, specific types of data, such as unique, rare, or unusually formatted text, are disproportionately memorized and more vulnerable to extraction. This creates a practical privacy risk with significant implications for data protection and regulatory compliance.

4.2.2 Private Text Reconstruction. Private text reconstruction attacks go beyond extracting verbatim examples, which recover sensitive information through inference and reconstruction techniques. Zhang et al. [88] demonstrated "Text Revealer," which successfully reconstructs private text from transformer models by exploiting their self-attention mechanisms. This work showed that internal model states contain recoverable private information. Yang et al. [83] extended this concept to code models, which revealed that specialized domains exhibit distinct patterns for memorization that can be exploited for precise extraction. Parikh et al. [50] introduced canary extraction, which showed that models memorize and leak specific inserted data patterns at rates that correlate with training

exposure. This technique evaluates memorization risks by analyzing how easily models reproduce strategically inserted markers. Dai et al. [11] demonstrated that activation inversion attacks can reconstruct training data in decentralized training environments, which highlights how internal model representations can be exploited to recover private information. Early research demonstrated that extraction was possible in certain scenarios. Newer techniques employ principled approaches, which target specific data types with high precision. This evolution suggests that privacy protections must advance beyond preventing obvious memorization to address subtle statistical patterns that enable reconstruction. As Gerasimenko and Namiot [18] argue in their analysis of extraction risks, these threats require reconsideration of how language models are deployed, particularly in domains handling sensitive information.

4.3 Prompt-targeted Attacks

Prompt-targeted attacks target valuable prompts used to guide LLM behavior. Given a model M and a private-owned prompt P^*, an adversary with query access to $M(P^*, x_i)$ for inputs x_i aims to reconstruct a similar prompt

$$\hat{P} = \arg\max_{P} \{\text{sim}(P, P^*) : \text{sim}(M(P, x), M(P^*, x)) > \tau, \forall x \in X_{test}\}, \tag{3}$$

where $\text{sim}(\cdot, \cdot)$ measures similarity and X_{test} is a validation set. We categorize prompt-targeted attacks into two types: *prompt stealing* and *prompt reconstruction*. Prompt stealing focuses on extracting carefully engineered prompts, such as system instructions, which represent significant commercial assets. In contrast, prompt reconstruction aims to recover instruction patterns and few-shot examples by analyzing model outputs, which retain traces of their prompting context.

4.3.1 Prompt Stealing. Prompt stealing attacks target the intellectual property embedded in carefully engineered prompts, which represent significant commercial assets in LLM applications. Sha and Zhang [61] pioneered research into systematic prompt stealing, which demonstrated that prompts could be extracted through strategic interactions with LLM applications. Their work showed that as models become more capable, the value increasingly shifts to prompting strategies, which creates new intellectual property vulnerabilities. Yang et al. [82] extended this work with PRSA (Prompt Stealing Attacks), which showed that even complex system prompts could be reconstructed with limited interaction by exploiting model responses to carefully designed queries. Hui et al. [26] demonstrated PLeak attacks, which reveal that commercial applications using LLMs frequently leak their underlying prompts through inconsistent output filtering. These vulnerabilities can be exploited even in production systems. Liang et al. [39] investigated the fundamental mechanisms of prompt leakage in customized LLMs, which identified architectural vulnerabilities that enable extraction. Their work revealed that prompts encoded as system instructions leave detectable patterns in model responses, which facilitate reconstruction. Prompt stealing represents an economic threat rather than merely a technical vulnerability. Businesses increasingly invest in prompt engineering to differentiate their AI offerings. Prompt theft provides a low-cost mechanism to appropriate competitive advantages. This dynamic creates a strategic tension where the most valuable prompts also become the most attractive targets for extraction. This may undermine incentives for innovation in prompt engineering and specialized model development.

4.3.2 Prompt Reconstruction. Prompt reconstruction attacks focus on recovering the instruction patterns and few-shot examples, which are used to specialize models for specific tasks and potentially compromise both intellectual property and security boundaries. Zhang et al. [87] demonstrated techniques for inverting LLM outputs to extract the prompts that generated them, which showed that responses contain sufficient information to reconstruct significant portions of input prompts. This work revealed that model outputs inherently retain traces of their prompting context, which creates fundamental information leakage. Xu et al. [76] introduced instructional fingerprinting of LLMs, which revealed that models retain detectable patterns from their instruction tuning that can be analyzed to reconstruct training processes. This research highlights how instruction tuning creates behavioral patterns that persist through inference. These patterns can potentially compromise the privacy of tuning data. Jiang et al. [29] presented an advanced approach for dynamic command generation in LLM tool-learning systems, which showed that instruction patterns can be systematically extracted and exploited. Mehrotra et al. [47] further developed this concept with their "Tree of Attacks" methodology, which systematically explores variations in prompt space to reconstruct effective instruction patterns. Prompt reconstruction research reveals that LLMs unintentionally encode aspects of their prompting history into their responses. This creates an information leakage channel where careful analysis of outputs can reveal the prompting techniques used to elicit them. This vulnerability undermines attempts to create secure boundaries between system instructions and user-facing functionality.It poses risks to the intellectual property of commercial prompts and the security measures designed to protect them. As LLMs increasingly rely on sophisticated prompting for alignment and safety, this vulnerability raises concerns about the robustness of current security approaches and the potential for adversaries to systematically compromise key model safety features.

5 Model Extraction Defenses

Defending against model extraction attacks requires a approach that addresses vulnerabilities at different levels of model deployment and operation. This section examines defense strategies organized into three categories: *Model Protection*, *Data Privacy Protection*, and *Prompt Protection*.

5.1 Model Protection

Model protection implements defensive mechanisms to prevent unauthorized model extraction or functionality replication. Given a model M, these defenses aim to create a protected version M' that maximizes legitimate utility while minimizing extraction success

$$M' = \arg\max_{M' \in \mathcal{M}} \{U(M', X_{leg}) - \lambda E(M', X_{adv})\}, \tag{4}$$

where U measures utility for legitimate inputs X_{leg}, E measures extraction success for adversarial inputs X_{adv}, and λ balances the trade-off. We categorize model protection mechanisms into two types: *architectural defenses* and *output control*. Architectural defenses modify the LLM's internal structure, such as embedding

watermarks or altering attention mechanisms, to prevent unauthorized extraction. In contrast, output control strategies manipulate LLM responses to reduce extraction risks without altering the underlying architecture.

5.1.1 Architectural Defense. Architectural defense mechanisms integrate security features directly into the model's structure, which prevents unauthorized extraction. Li et al. [37] proposed TransLinkGuard, which safeguards transformer models in edge deployments by manipulating attention mechanisms to embed watermarks, which resist extraction and maintain performance. This approach demonstrates that architectural defenses can be implemented with minimal computational overhead, which is critical for resource-constrained edge devices. Li et al. [38] extended this concept with CoreGuard, which protects foundational capabilities through structural modifications that make extraction attempts produce degraded model clones. The key insight from these approaches is that effective architectural defenses must target the specific mechanisms that are exploited during extraction rather than applying general security principles. Transformer-based models present unique vulnerabilities because their self-attention mechanisms inadvertently expose architectural information during inference. By strategically altering these attention patterns, defenders can create asymmetric advantages, which allow legitimate users to experience minimal performance impact while extractors receive functionally compromised knowledge. However, these approaches often require significant modifications to existing model architectures. This limits their applicability to newly developed models and makes retrofitting protection onto existing deployments challenging.

5.1.2 Output Control. Strategies to control model responses focus on reducing the risk of unauthorized data extraction by manipulating outputs, which avoids modifying the underlying architecture. Wang and Cheng [69] introduced GuardEmb, which implements watermarking techniques that adapt to usage for embedding services. This technique alters response patterns in ways that are detectable by service providers but difficult for attackers to avoid. Pang et al. [49] developed ModelShield, which uses watermarking methods that adjust dynamically to provide robust protection against extraction attacks. This approach strategically perturbs responses while maintaining high utility for legitimate users. These techniques demonstrate that controlled output manipulation can create barriers to extraction with minimal performance impact. The fundamental insight is that strategies to control model responses offer deployment flexibility compared to architectural approaches, which allows them to be implemented as external layers added to the model's API without modifying model internals. This advantage enables retrofitting protection onto existing deployed models and allows for dynamic adjustment of protection levels based on threat assessments. The challenge remains in balancing the degree of response alteration. Insufficient changes fail to prevent extraction, while excessive modification compromises legitimate user experience. As extraction techniques grow more sophisticated, strategies to control model responses must evolve into adaptive methods, which dynamically calibrate protection based on usage patterns and potential extraction signals.

5.2 Data Privacy Protection

Data privacy protection mechanisms aim to prevent extraction of private information from language models. Given a model M trained on dataset D containing private information $P \subset D$, these defenses create a protected model M' that minimizes privacy leakage while preserving utility:

$$M' = \arg \min_{M' \in \mathcal{M}} \{L(M', P) + \lambda \mathcal{D}(M', M)\}, \tag{5}$$

where L measures privacy leakage, $\mathcal{D}$ measures model utility deviation, and λ balances the trade-off. We categorize data privacy protection mechanisms into two types: *training data security* and *output sanitization*. Training data security focuses on preventing sensitive information from being memorized during training or removing it after exposure, using techniques like differential privacy or selective knowledge removal. In contrast, output sanitization modifies or filters LLM responses to prevent private information leakage during inference.

5.2.1 Training Data Security. Methods to secure training data aim to prevent the extraction of sensitive information, which is embedded in model parameters during training. Patil et al. [51] investigated whether sensitive information could be deleted from LLMs after training. They evaluated various defense objectives against extraction attacks and found that removing specific knowledge can mitigate vulnerabilities while preserving general model capabilities. Their work demonstrates that protecting training data requires preventive measures during training, which must be complemented by corrective measures after potential exposure. The key insight is that securing training data increasingly requires targeted protection of specific information types. Blanket protection approaches are often less effective. While traditional differential privacy techniques apply uniform noise to all training signals, emerging research shows that protecting specific information categories achieves better results in balancing privacy and utility. However, the fundamental challenge remains: models inherently memorize training examples as part of their learning process. This makes complete prevention of training data extraction theoretically difficult. This creates an ongoing tension between learning effectiveness and privacy protection, which drives research into novel training methods that limit memorization while maintaining learning capacity.

5.2.2 Output Sanitization. Output sanitization focuses on modifying or filtering model responses, which prevents the leakage of private information. Wang et al. [73] proposed Self-Guard, which empowers language models to detect and sanitize their own outputs. It uses self-monitoring mechanisms, which filter potentially sensitive information before returning responses. This approach demonstrates that models can be trained to recognize and suppress private information in their own generations, which creates an additional protection layer without external filtering systems. The critical insight is that output sanitization operates on a fundamentally different principle than simple content blocking. Rather than applying static rules, effective sanitization must adapt filtering based on contextual information sensitivity. This requires models to develop the ability to understand the implications of generated information and its privacy risks, which represents a higher-order capability beyond basic content generation. The challenge remains

in developing sanitization approaches, which maintain content coherence and utility while removing sensitive elements. This is particularly difficult in contexts where private information forms the central subject of legitimate queries.

5.3 Prompt Protection

Prompt protection mechanisms safeguard private prompts and instruction patterns, which represent valuable intellectual property. Given a prompt P that an organization wishes to protect, these defenses implement safeguards, which maximize detection of unauthorized use and minimize impact on legitimate functionality:

$$\arg\max_{D\in\mathcal{D}}\{\text{TPR}(D,P,X_{adv})-\lambda\text{Impact}(D,P,X_{leg})\}, \tag{6}$$

where TPR measures true positive rate of detecting unauthorized prompt use from adversarial queries X_{adv}, Impact measures the effect on legitimate queries X_{leg}, and λ balances security and utility. We categorize prompt protection mechanisms into two types: *direct prompt protection* and *query monitoring*. Direct prompt protection focuses on safeguarding private prompts and instruction patterns through techniques like watermarking and obfuscation, which prevent unauthorized use and detect derivative works. In contrast, query monitoring systems analyze user interactions to identify suspicious behaviors indicative of extraction attempts.

5.3.1 Direct Prompt Protection. This mechanism protects private prompts and instruction patterns, which represent valuable intellectual property. He et al. [22] introduced CATER, a conditional watermarking system, which protects intellectual property in text generation APIs by embedding subtle markers that identify prompts and maintain generation quality. Kim et al. [32] proposed detailed protection strategies for LLM environments, which use prompt protection techniques to prevent unauthorized access to system instructions. These approaches demonstrate that prompt protection must balance detection capabilities with generation quality to ensure effective protection without degrading user experience. The key insight is that prompt protection has become an emerging priority in the LLM ecosystem as commercial value increasingly shifts from model weights to prompting techniques. As businesses invest substantial resources in developing specialized prompts for specific applications, the economic incentive for prompt theft grows proportionally. Protection mechanisms must therefore prevent direct copying. They must also detect derivative works, which preserve the functionality of the original prompts while modifying their wording, structure, or formatting. This creates a conceptual parallel to traditional software protection, where both code and functionality require safeguarding against unauthorized reproduction.

5.3.2 Query Monitoring. Systems to monitor user queries detect and prevent extraction attempts by analyzing user queries and behaviors, which reveal suspicious activity. Wang et al. [73] demonstrated that self-monitoring approaches can detect suspicious query patterns, which indicate extraction attempts and enable early defensive actions. These systems work by identifying unexpected patterns in user queries, which may suggest extraction attempts, or by recognizing known behaviors associated with extraction attacks. The key insight is that extraction attacks often show distinct behaviors that differ from legitimate usage. These behaviors include systematically exploring model boundaries, using unusual input distributions, or repeatedly testing specific capabilities. Query monitoring establishes typical patterns of legitimate user behavior, which allow it to flag potential extraction attempts for further investigation or response. However, the challenge lies in distinguishing sophisticated extraction attempts from legitimate but unusual usage. This becomes particularly difficult as attackers adapt their techniques to mimic normal user behavior. This creates an ongoing dynamic, which requires monitoring systems to continuously evolve to detect increasingly subtle extraction attempts.

6 Evaluation Metrics

Evaluating model extraction attacks and defenses requires specific evaluation criteria, which measure both extraction success and defense effectiveness. Table 1 summarizes the effectiveness of defense mechanisms against various attack types.

6.1 Attack Effectiveness

Functional Similarity. This metric assesses how closely an extracted model replicates the behavior of the target LLM, which focuses on capability transfer rather than exact parameter recovery. Functional similarity includes several metrics. Agreement rate measures the percentage of cases where extracted and target models produce equivalent outputs given identical inputs [6]. Behavioral consistency evaluates how reliably an extracted model reproduces specific patterns of the target model [77]. Task-specific performance correlation quantifies alignment between extracted and target models on standardized benchmarks [33]. Perplexity similarity provides a continuous measure of functional extraction success by comparing cross-perplexity between models. This metric is particularly valuable when evaluating extraction against large generative models, which produce responses that make exact matching impractical.
Data Recovery Rate. This metric quantifies an attack's success in extracting sensitive or structured information, which is embedded within a target model. It encompasses several sub-metrics that evaluate different aspects of recovery success. Training data extraction success measures the percentage of training examples that are successfully recovered from a model's memorized content [7]. Precision and recall of extracted data evaluate both the accuracy and comprehensiveness of recovered information, which is particularly important when extracting structured data such as personally identifiable information (PII) [24]. Private information exposure rate measures how effectively an attack extracts specifically targeted sensitive information, such as private user data or confidential records [91]. Prompt recovery accuracy evaluates how accurately an attack reconstructs system prompts, which is assessed by measuring both semantic similarity and functional equivalence between the original and recovered prompts [61]. These sub-metrics provide a comprehensive evaluation of an attack's ability to recover different types of information embedded within a model, highlighting the diverse risks posed by model extraction attacks.

6.2 Defense Performance

Security Metrics. These metrics evaluate how effectively defense mechanisms reduce the success of extraction attacks. They include

Table 1: Effectiveness of Defense Mechanisms Against Different Model Extraction Attack Types.

Defense Mechanism	Functionality Extraction			Training Data Extraction		Prompt-targeted Attacks	
	API-based KD	Direct API Querying	Parameter Recovery	Prompt-targeted Recovery	Private Text Reconstruction	Prompt Stealing	Prompt Reconstruction
Architectural Defense [1]	High	Medium	High	Low	Low	Minimal	Minimal
Output Control [2]	High	High	Low	Medium	Medium	Low	Low
Training Data Security [3]	Low	Minimal	Minimal	High	High	Minimal	Minimal
Output Sanitization [4]	Low	Low	Minimal	High	High	Low	Low
Prompt Protection [5]	Minimal	Low	Minimal	Minimal	Minimal	High	High
Query Monitoring [6]	Medium	High	Low	Medium	Medium	Medium	Medium

Effectiveness Levels: High (dark green) - Highly effective; Medium (light green) - Moderately effective; Low (yellow) - Limited effectiveness; Minimal (gray) - Minimal or no effectiveness.

[1] Li et al. [37, 38] **[2]** Wang and Cheng [69], Pang et al. [49] **[3]** Feng and Tramèr [17], Patil et al. [51]

[4] Li et al. [36], Wang et al. [73] **[5]** He et al. [22], Kim et al. [32] **[6]** Wang et al. [73]

attack prevention rate, which quantifies the reduction in extraction success when defensive measures are deployed [37]. Query detection accuracy measures a defense system's ability to identify malicious query patterns, which indicate extraction attempts [73]. Extraction cost increase quantifies the additional computational and query resources that attackers must expend when defensive measures are in place [49]. Watermark robustness measures how well ownership signals persist in extracted models, which enables the detection of unauthorized clones [69].

Utility Metrics. These metrics evaluate how defense mechanisms affect the model's ability to perform its intended tasks. They include performance preservation, which measures the degree to which defensive measures maintain original model capabilities [38]. Response quality preservation quantifies changes in generation quality, which occur when defensive measures are applied [22]. Computational overhead assesses the additional processing burden, which is introduced by defensive measures [73]. False positive rate measures how frequently legitimate queries are incorrectly flagged or degraded, which occurs due to defensive measures [32]. Utility assessment enables practitioners to select defense strategies that provide adequate protection. At the same time, it ensures that the core value proposition of their deployed models is preserved.

7 Limitations and Future Directions

Extraction Attack Perspectives. Current attempts to replicate model functionality face challenges in deployment due to access restrictions and high computational requirements for proprietary models. Most attacks target isolated aspects of a model, which limits their comprehensiveness and adaptability to defenses. Future research should focus on combining multiple attack techniques to overcome specific defenses and optimizing query efficiency to reduce costs and avoid query restrictions.

Defense Mechanism Advancements. Existing defenses face challenges in deployment and effectiveness. Structural defenses require significant modifications, which limits their applicability to deployed systems, while output manipulation methods struggle to balance protection with generation quality. Most defenses lack formal security assurances and rely on empirical evaluation. Future defense research should prioritize defenses that can be applied to existing models without requiring structural changes or retraining.

8 Related Work

Several surveys have explored various dimensions of security and privacy challenges for large language models. Comprehensive surveys like those by Das et al. [12] and Wang et al. [70] provide broad overviews of LLM security and privacy challenges, which include multiple types of threats and vulnerabilities. Yao et al. [84] organize their analysis into "the good" (beneficial security applications), "the bad" (offensive uses), and "the ugly" (inherent vulnerabilities), which provides a balanced perspective on LLMs' security implications. In contrast, other surveys focus on specific attack types or scenarios. Zhao et al. [92] analyze attempts to replicate model functionality in distributed computing with some coverage of LLMs. Mathew [46] specifically examines attacks that manipulate model behavior using malicious prompts and their defenses. Esmradi et al. [15] survey implementation techniques and mitigation strategies for various attacks. Some surveys emphasize particular dimensions of LLM security. Liu et al. [41] concentrate on societal and ethical consequences of LLM security, which focus more on societal impacts than technical mechanisms. Ma et al. [44] provide a safety-oriented perspective. Cui et al. [10] offer a more technically focused examination of recent attack and defense approaches. Our survey differs from these works by providing a systematic taxonomy of model extraction attacks specifically for LLMs. It examines the unique vulnerabilities of transformer-based architectures and offers a detailed analysis of extraction methods, such as recovering data used to train the model and prompt extraction, which exploit the distinctive characteristics of large language models.

9 Conclusion

This survey provides a structured analysis of attempts to replicate model functionality and the mechanisms to prevent them, which are critical for LLM security. We present a taxonomy, which categorizes attack types and defense mechanisms. Our framework includes evaluation metrics, which highlight the balance between security measures and model usability. We identify current limitations and research gaps, particularly in combining multiple attack techniques and establishing formal assurances of defense effectiveness. Finally, this work offers a structured foundation for developing robust protection strategies, which are essential for LLMs as critical commercial infrastructure.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Francisco Aguilera-Martínez and Fernando Berzal. 2025. LLM Security: Vulnerabilities, Attacks, Defenses, and Countermeasures. *arXiv preprint arXiv:2505.01177* (2025).

[3] Yadagiri Annepaka and Partha Pakray. 2024. Large language models: A survey of their development, capabilities, and applications. *Knowledge and Information Systems* (2024), 1–56.

[4] Anahita Baninajjar, Kamran Hosseini, Ahmed Rezine, and Amir Aminifar. 2024. Verified relative safety margins for neural network twins. *arXiv preprint arXiv:2409.16726* (2024).

[5] Lewis Birch, William Hackett, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. 2023. Model leeching: An extraction attack targeting llms. *arXiv preprint arXiv:2309.10544* (2023).

[6] Nicholas Carlini, Daniel Paleka, Krishnamurthy Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Itay Yona, Eric Wallace, David Rolnick, and Florian Tram'er. 2024. Stealing Part of a Production Language Model. *arXiv preprint arXiv:2403.06634* (2024). https://arxiv.org/abs/2403.06634

[7] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting Training Data from Large Language Models. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2633–2650.

[8] Chen Chen, Xuanli He, Lingjuan Lyu, and Fangzhao Wu. 2021. Killing one bird with two stones: model extraction and attribute inference attacks against bert-based apis. *arXiv preprint arXiv:2105.10909* (2021).

[9] Zhan Cheng, Bolin Shen, Tianming Sha, Yuan Gao, Shibo Li, and Yushun Dong. 2025. ATOM: A Framework of Detecting Query-Based Model Extraction Attacks for Graph Neural Networks. *arXiv preprint arXiv:2503.16693* (2025).

[10] Jing Cui, Yishi Xu, Zhewei Huang, Shuchang Zhou, Jianbin Jiao, and Junge Zhang. 2024. Recent advances in attack and defense approaches of large language models. *arXiv preprint arXiv:2409.03274* (2024).

[11] Chenxi Dai, Lin Lu, and Pan Zhou. 2025. Stealing Training Data from Large Language Models in Decentralized Training through Activation Inversion Attack. *arXiv preprint arXiv:2502.16086* (2025).

[12] Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2025. Security and privacy challenges of large language models: A survey. *Comput. Surveys* 57, 6 (2025), 1–39.

[13] Gunika Dhingra, Saumil Sood, Zeba Mohsin Wase, Arshdeep Bahga, and Vijay K Madisetti. 2024. Protecting LLMs against Privacy Attacks While Preserving Utility. *Journal of Information Security* 15, 4 (2024), 448–473.

[14] Yi Dong, Ronghui Mu, Yanghao Zhang, Siqi Sun, Tianle Zhang, Changshun Wu, Gaojie Jin, Yi Qi, Jinwei Hu, Jie Meng, et al. 2024. Safeguarding large language models: A survey. *arXiv preprint arXiv:2406.02622* (2024).

[15] Aysan Esmradi, Daniel Wankit Yip, and Chun Fai Chan. 2023. A comprehensive survey of attack techniques, implementation, and mitigation strategies in large language models. In *International Conference on Ubiquitous Security*. Springer, 76–95.

[16] Luyang Fang, Xiaowei Yu, Jiazhang Cai, Yongkai Chen, Shushan Wu, Zhengliang Liu, Zhenyuan Yang, Haoran Lu, Xilin Gong, Yufang Liu, et al. 2025. Knowledge Distillation and Dataset Distillation of Large Language Models: Emerging Trends, Challenges, and Future Directions. *arXiv preprint arXiv:2504.14772* (2025).

[17] Shanglun Feng and Florian Tramèr. 2024. Privacy backdoors: stealing data with corrupted pretrained models. *arXiv preprint arXiv:2404.00473* (2024).

[18] Denis V Gerasimenko and Dmitry Namiot. 2024. Extracting Training Data: Risks and solutions in the context of LLM security. *International Journal of Open Information Technologies* 12, 11 (2024), 9–19.

[19] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[20] Danny Halawi, Alexander Wei, Eric Wallace, Tony T Wang, Nika Haghtalab, and Jacob Steinhardt. 2024. Covert malicious finetuning: Challenges in safeguarding llm adaptation. *arXiv preprint arXiv:2406.20053* (2024).

[21] Xuanli He, Lingjuan Lyu, Qiongkai Xu, and Lichao Sun. 2021. Model extraction and adversarial transferability, your BERT is vulnerable! *arXiv preprint arXiv:2103.10013* (2021).

[22] Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. 2022. Cater: Intellectual property protection on text generation apis via conditional watermarks. *Advances in Neural Information Processing Systems* 35 (2022), 5431–5445.

[23] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301* (2023).

[24] Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are large pre-trained language models leaking your personal information? *arXiv preprint arXiv:2205.12628* (2022).

[25] Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. 2024. O1 Replication Journey–Part 2: Surpassing O1-preview through Simple Distillation, Big Progress or Bitter Lesson? *arXiv preprint arXiv:2411.16489* (2024).

[26] Bo Hui, Haolin Yuan, Neil Gong, Philippe Burlina, and Yinzhi Cao. 2024. Pleak: Prompt leaking attacks against large language model applications. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 3600–3614.

[27] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. 2018. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961* (2018).

[28] Yoichi Ishibashi and Hidetoshi Shimodaira. 2023. Knowledge sanitization of large language models. *arXiv preprint arXiv:2309.11852* (2023).

[29] Ziyou Jiang, Mingyang Li, Guowei Yang, Junjie Wang, Yuekai Huang, Zhiyuan Chang, and Qing Wang. 2025. Mimicking the Familiar: Dynamic Command Generation for Information Theft Attacks in LLM Tool-Learning System. *arXiv preprint arXiv:2502.11358* (2025).

[30] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. 2018. Model extraction warning in MLaaS paradigm. In *Proceedings of the 34th annual computer security applications conference*. 371–380.

[31] Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. 2018. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957* (2018).

[32] Minjae Kim, Taehyeong Kwon, Kibeom Shim, and Beonghoon Kim. 2024. Protection of LLM Environment Using Prompt Security. In *2024 15th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 1715–1719.

[33] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. 2020. Thieves on Sesame Street! Model Extraction of BERT-based APIs. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Byl5NREFDr

[34] Sunbowen Lee, Junting Zhou, Chang Ao, Kaige Li, Xinrun Du, Sirui He, Haihong Wu, Tianci Liu, Jiaheng Liu, Hamid Alinejad-Rokny, Min Yang, Yitao Liang, Zhoufutu Wen, and Shiwen Ni. 2025. Quantification of Large Language Model Distillation. arXiv:2501.12619

[35] Chenyang Li, Zhao Song, Weixin Wang, and Chiwun Yang. 2023. A theoretical insight into attack and defense of gradient leakage in transformer. *arXiv preprint arXiv:2311.13624* (2023).

[36] Qinbin Li, Junyuan Hong, Chulin Xie, Jeffrey Tan, Rachel Xin, Junyi Hou, Xavier Yin, Zhun Wang, Dan Hendrycks, Zhangyang Wang, et al. 2024. Llm-pbe: Assessing data privacy in large language models. *arXiv preprint arXiv:2408.12787* (2024).

[37] Qinfeng Li, Zhiqiang Shen, Zhenghan Qin, Yangfan Xie, Xuhong Zhang, Tianyu Du, Sheng Cheng, Xun Wang, and Jianwei Yin. 2024. TransLinkGuard: Safeguarding Transformer Models Against Model Stealing in Edge Deployment. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 3479–3488.

[38] Qinfeng Li, Yangfan Xie, Tianyu Du, Zhiqiang Shen, Zhenghan Qin, Hao Peng, Xinkui Zhao, Xianwei Zhu, Jianwei Yin, and Xuhong Zhang. 2024. CoreGuard: Safeguarding Foundational Capabilities of LLMs Against Model Stealing in Edge Deployment. *arXiv preprint arXiv:2410.13903* (2024).

[39] Zi Liang, Haibo Hu, Qingqing Ye, Yaxin Xiao, and Haoyang Li. 2024. Why Are My Prompts Leaked? Unraveling Prompt Extraction Threats in Customized Large Language Models. *arXiv preprint arXiv:2408.02416* (2024).

[40] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024. A survey of text watermarking in the era of large language models. *Comput. Surveys* 57, 2 (2024), 1–36.

[41] Feng Liu, Jiaqi Jiang, Yating Lu, Zhanyi Huang, and Jiuming Jiang. 2025. The ethical security of large language models: A systematic review. *Frontiers of Engineering Management* (2025), 1–13.

[42] Jiaheng Liu, Chenchen Zhang, Jinyang Guo, Yuanxing Zhang, Haoran Que, Ken Deng, Jie Liu, Ge Zhang, Yanan Wu, Congnan Liu, et al. 2024. Ddk: Distilling domain knowledge for efficient large language models. *Advances in Neural Information Processing Systems* 37 (2024), 98297–98319.

[43] Yupei Liu, Jinyuan Jia, Hongbin Liu, and Neil Zhenqiang Gong. 2022. Stolencoder: stealing pre-trained encoders in self-supervised learning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2115–2128.

[44] Xingjun Ma, Yifeng Gao, Yixu Wang, Ruofan Wang, Xin Wang, Ye Sun, Yifan Ding, Hengyuan Xu, Yunhao Chen, Yunhan Zhao, et al. 2025. Safety at Scale: A Comprehensive Survey of Large Model Safety. *arXiv preprint arXiv:2502.05206* (2025).

[45] Loïc Maréchal. 2024. The Flow of Investments in the LLM Space. In *Large Language Models in Cybersecurity: Threats, Exposure and Mitigation*. Springer Nature Switzerland Cham, 129–135.

[46] Eleena Mathew. 2024. Enhancing Security in Large Language Models: A Comprehensive Review of Prompt Injection Attacks and Defenses. *Authorea Preprints* (2024).

[47] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2024. Tree of attacks: Jailbreaking black-box llms automatically. *Advances in Neural Information Processing Systems* 37 (2024), 61065–61105.

[48] Najmeh Nazari, Furi Xiang, Chongzhou Fang, Hosein Mohammadi Makrani, Aditya Puri, Kartik Patwari, Hossein Sayadi, Setareh Rafatirad, Chen-Nee Chuah, and Houman Homayoun. 2024. LLM-FIN: Large Language Models Fingerprinting Attack on Edge Devices. In *2024 25th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 1–6.

[49] Kaiyi Pang, Tao Qi, Chuhan Wu, Minhao Bai, Minghu Jiang, and Yongfeng Huang. 2025. ModelShield: Adaptive and Robust Watermark against Model Extraction Attack. *IEEE Transactions on Information Forensics and Security* (2025).

[50] Rahil Parikh, Christophe Dupuy, and Rahul Gupta. 2022. Canary extraction in natural language understanding models. *arXiv preprint arXiv:2203.13920* (2022).

[51] Vaidehi Patil, Peter Hase, and Mohit Bansal. 2023. Can sensitive information be deleted from llms? objectives for defending against extraction attacks. *arXiv preprint arXiv:2309.17410* (2023).

[52] Tianyu Peng and Jiajun Zhang. 2024. Enhancing Knowledge Distillation of Large Language Models through Efficient Multi-Modal Distribution Alignment. *arXiv preprint arXiv:2409.12545* (2024).

[53] Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527* (2022).

[54] Robert Philipp, Andreas Mladenow, Christine Strauss, and Alexander Völz. 2020. Machine learning as a service: Challenges in research and applications. In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*. 396–406.

[55] Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. 2024. O1 Replication Journey: A Strategic Progress Report–Part 1. *arXiv preprint arXiv:2410.18982* (2024).

[56] Adnan Siraj Rakin, Md Hafizul Islam Chowdhuryy, Fan Yao, and Deliang Fan. 2022. Deepsteal: Advanced model extractions leveraging efficient weight stealing in memories. In *2022 IEEE symposium on security and privacy (SP)*. IEEE, 1157–1174.

[57] Vishal Rathod, Seyedsina Nabavirazavi, Samira Zad, and Sundararaja Sitharama Iyengar. 2025. Privacy and security challenges in large language models. In *2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 00746–00752.

[58] Robert Nikolai Reith, Thomas Schneider, and Oleksandr Tkachenko. 2019. Efficiently stealing your machine learning models. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*. 198–210.

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

[60] Goldman Sachs. 2023. AI investment forecast to approach $200 billion globally by 2025. *Artificial intelligence outlooks–01 AUG* (2023).

[61] Zeyang Sha and Yang Zhang. 2024. Prompt Stealing Attacks Against Large Language Models. *arXiv preprint arXiv:2402.12959* (2024).

[62] Xinyue Shen, Yiting Qu, Michael Backes, and Yang Zhang. 2024. Prompt Stealing Attacks Against {Text-to-Image} Generation Models. In *33rd USENIX Security Symposium (USENIX Security 24)*. 5823–5840.

[63] Anup Shirgaonkar, Nikhil Pandey, Nazmiye Ceren Abay, Tolga Aktas, and Vijay Aski. 2024. Knowledge Distillation Using Frontier Open-source LLMs: Generalizability and the Role of Synthetic Data. *arXiv preprint arXiv:2410.18588* (2024).

[64] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[65] Stephen Burabari Tete. 2024. Threat modelling and risk analysis for large language model (llm)-powered applications. *arXiv preprint arXiv:2406.11007* (2024).

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

[67] Anvesh Rao Vijjini, Somnath Basu Roy Chowdhury, and Snigdha Chaturvedi. 2024. Exploring safety-utility trade-offs in personalized language models. *arXiv preprint arXiv:2406.11107* (2024).

[68] Jeffrey G Wang, Jason Wang, Marvin Li, and Seth Neel. 2024. Pandora's White-Box: Precise Training Data Detection and Extraction in Large Language Models. *arXiv preprint arXiv:2402.17012* (2024).

[69] Liaoyaqi Wang and Minhao Cheng. 2024. GuardEmb: Dynamic Watermark for Safeguarding Large Language Model Embedding Service Against Model Stealing Attack. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 7518–7534.

[70] Shang Wang, Tianqing Zhu, Bo Liu, Ming Ding, Xu Guo, Dayong Ye, Wanlei Zhou, and Philip S Yu. 2024. Unique security and privacy threats of large language model: A comprehensive survey. *arXiv preprint arXiv:2406.07973* (2024).

[71] Xinyi Wang, Antonis Antoniades, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. 2025. Generalization v.s. Memorization: Tracing Language Models' Capabilities Back to Pretraining Data. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=IQxBDLmVpT

[72] Yu Wang, Cailing Cai, Zhihua Xiao, and Peifung E Lam. 2025. LLM Access Shield: Domain-Specific LLM Framework for Privacy Policy Compliance. *arXiv preprint arXiv:2505.17145* (2025).

[73] Zezhong Wang, Fangkai Yang, Lu Wang, Pu Zhao, Hongru Wang, Liang Chen, Qingwei Lin, and Kam-Fai Wong. 2023. Self-guard: Empower the llm to safeguard itself. *arXiv preprint arXiv:2310.15851* (2023).

[74] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems* 36 (2023), 80079–80110.

[75] Fangzhou Wu, Ning Zhang, Somesh Jha, Patrick McDaniel, and Chaowei Xiao. 2024. A new era in llm security: Exploring security concerns in real-world llm-based systems. *arXiv preprint arXiv:2402.18649* (2024).

[76] Jiashu Xu, Fei Wang, Mingyu Derek Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. 2024. Instructional fingerprinting of large language models. *arXiv preprint arXiv:2401.12255* (2024).

[77] Qiongkai Xu, Xuanli He, Lingjuan Lyu, Lizhen Qu, and Gholamreza Haffari. 2021. Student surpasses teacher: Imitation attack for black-box NLP APIs. *arXiv preprint arXiv:2108.13873* (2021).

[78] Wenrui Xu and Keshab K Parhi. 2025. A Survey of Attacks on Large Language Models. *arXiv preprint arXiv:2505.12567* (2025).

[79] Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzhen Cheng. 2024. On protecting the data privacy of large language models (llms): A survey. *arXiv preprint arXiv:2403.05156* (2024).

[80] Mingke Yang, Yuqi Chen, Yi Liu, and Ling Shi. 2024. DistillSeq: A Framework for Safety Alignment Testing in Large Language Models using Knowledge Distillation. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 578–589.

[81] Wenkai Yang, Yankai Lin, Jie Zhou, and Ji-Rong Wen. 2025. Distilling Rule-based Knowledge into Large Language Models. In *Proceedings of the 31st International Conference on Computational Linguistics*. 913–932.

[82] Yong Yang, Changjiang Li, Yi Jiang, Xi Chen, Haoyu Wang, Xuhong Zhang, Zonghui Wang, and Shouling Ji. 2024. PRSA: PRompt Stealing Attacks against large language models. *arXiv preprint arXiv:2402.19200* (2024).

[83] Zhou Yang, Zhipeng Zhao, Chenyu Wang, Jieke Shi, Dongsun Kim, Donggyun Han, and David Lo. 2024. Unveiling memorization in code models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[84] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* (2024), 100211.

[85] Yuanshun Yao, Zhujun Xiao, Bolun Wang, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2017. Complexity vs. performance: empirical analysis of machine learning as a service. In *Proceedings of the 2017 Internet Measurement Conference*. 384–397.

[86] Yizhen Yuan, Rui Kong, Yuanchun Li, and Yunxin Liu. 2024. Wip: An on-device llm-based approach to query privacy protection. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*. 7–9.

[87] Collin Zhang, John X Morris, and Vitaly Shmatikov. 2024. Extracting prompts by inverting llm outputs. *arXiv preprint arXiv:2405.15012* (2024).

[88] Ruisi Zhang, Seira Hidano, and Farinaz Koushanfar. 2022. Text revealer: Private text reconstruction via model inversion attacks against transformers. *arXiv preprint arXiv:2209.10505* (2022).

[89] Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2024. {REMARK-LLM}: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*. 1813–1830.

[90] Yuehan Zhang, Peizhuo Lv, Yinpeng Liu, Yongqiang Ma, Wei Lu, Xiaofeng Wang, Xiaozhong Liu, and Jiawei Liu. 2024. PersonaMark: Personalized LLM watermarking for model protection and user attribution. *arXiv preprint arXiv:2409.09739* (2024).

[91] Zhexin Zhang, Jiaxin Wen, and Minlie Huang. 2023. Ethicist: Targeted training data extraction through loss smoothed soft prompting and calibrated confidence estimation. *arXiv preprint arXiv:2307.04401* (2023).

[92] Kaixiang Zhao, Lincan Li, Kaize Ding, Neil Zhenqiang Gong, Yue Zhao, and Yushun Dong. 2025. A Survey of Model Extraction Attacks and Defenses in Distributed Computing Environments. *arXiv preprint arXiv:2502.16065* (2025).

[93] Zhengyue Zhao, Xiaogeng Liu, Somesh Jha, Patrick McDaniel, Bo Li, and Chaowei Xiao. [n. d.]. Can Watermarks be Used to Detect LLM IP Infringement For Free?. In *The Thirteenth International Conference on Learning Representations*.