FROM: A Framework for Recognizing the Origins of LLMs in Black-Box Conditions

Hongru Wei, Qingyuan Hu, Linzhi Chen, Yuqi Chen*
School of Information Science and Technology, ShanghaiTech University
{weihr2023, huqy2022, chenlzh2024, chenyq}@shanghaitech.edu.cn

***Abstract*—Large Language Models (LLMs) exhibit remarkable capabilities across diverse domains, with their training requiring substantial computational resources and time. Fine-tuning has emerged as a standard practice for task-specific adaptation of LLMs, raising critical challenges in tracking model origins to ensure licensing compliance and prevent unauthorized use of proprietary base models. Additionally, fine-tuned models inherit characteristics from their original models, such as behaviors, vulnerabilities, and compatibility. Therefore, analyzing the original model can also help optimize performance and mitigate security risks in its fine-tuned versions. To address these challenges, we introduce FROM, an efficient and lightweight framework for tracing the origins of fine-tuned LLMs in black-box scenarios, which enhances accountability, reproducibility, and intellectual property protection. Experiments show that FROM achieves 94.4% accuracy in identifying original models, underscoring the importance of version awareness to safeguard model ownership and optimize deployment.**

I. Introduction

Large language models (LLMs), such as GPT [1], LLaMA [2], and Qwen [3], have demonstrated revolutionary capabilities in natural language processing and complex decision-making, owing to their advanced language comprehension and generation abilities [4]–[6]. These remarkable achievements have rapidly established them as focal points in both academic and industrial research. However, behind these accomplishments lies substantial resource consumption, as training a high-performance LLM from scratch requires significant computational costs and months of training time [7], [8]. This reality has driven researchers to explore more efficient model adaptation methods, among which fine-tuning techniques based on pre-trained models have gained considerable attention due to their significantly reduced resource demands.

Nevertheless, the widespread application of LLMs introduces urgent challenges. The primary issue is model copyright protection, as ensuring authorized usage is critical for maintaining a healthy research ecosystem. Additionally, fine-tuned models may exhibit biases or be maliciously exploited to generate false information [9], making accountability tracing essential. Studies indicate that fine-tuned models retain the characteristics of their original counterparts [10], [11], including vulnerabilities like jailbreak attacks [12], [13] and backdoor risks [14], [15]. Consequently, LLM attribution, which identifies the base model of a fine-tuned version, becomes essential. Accurately determining model lineage provides critical support for resolving issues related to copyright, accountability, and security assessment.

This paper proposes **FROM** (A **F**ramework for **R**ecognizing the **O**rigins of LLM**s** in Black-Box Conditions), an efficient, lightweight, and modular framework designed to identify the original pre-trained model of a fine-tuned LLM under black-box conditions. This framework aids in resolving issues related to fine-tuned models and tracing accountability. Its core idea is to distinguish the original version of a fine-tuned model based on its responses to a set of carefully designed prompts. To achieve this, the framework consists of four components: (1) *Models Preparation*: generating fine-tuned models for use by other modules; (2) *Database Construction*: creating a prompt database through generation and automated optimization using high-performance LLMs; (3) *Single Pair Classifier*: identifying the original model via a resource-efficient search and optimization process; and (4) *Knockout Round*: inspired by sports tournament mechanisms, employing a tree-like structure to extend the capabilities of the *Single Pair Classifier* for multi-model comparison, thereby enhancing the framework's functionality. The modular design of **FROM** ensures excellent scalability and adaptability, providing a key tool for addressing the aforementioned challenges.

We comprehensively evaluate FROM, achieving a 94.4% accuracy rate across tests involving various types and parameter scales of original models, demonstrating superiority over existing methods. Simultaneously, the framework exhibits strong generalization capabilities for unseen fine-tuned models, successfully identifying 9 out of 10 unknown test fine-tuned models. Ablation studies further confirm the effectiveness of each module. Moreover, the lightweight nature of **FROM** allows deployment on ordinary computing devices, significantly enhancing its practical value. These advantages position **FROM** as the most promising solution for LLM provenance tracing to date, laying a critical foundation for subsequent research.

Our code and data will be released to the community to support further developments once this work is accepted.

II. Related Work

a) LLM Attribution: LLM Attribution refers to the task of identifying the original base model used for a fine-tuned LLM. This problem was first introduced in *Matching Pairs*

* Corresponding author.

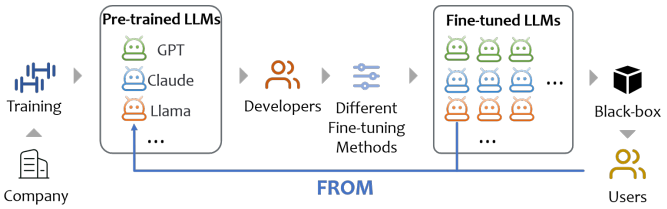


Fig. 1. Problem formulation: Given a fine-tuned model m_f, our objective is to identify its corresponding original pre-trained model m_o.

[11]. As derivative LLM products proliferate, the need for LLM attribution has grown significantly, driven by demands for copyright protection and model property analysis. While some studies address different scenarios, their methodologies can be adapted for attribution tasks. For instance, *TRAP* [16] and *Proflingo* [17] employ fingerprinting techniques to assess model similarity. *HuRef* [18] converts LLM weights into images via convolutional encoders and evaluates their visual similarities. Other work [19] explores backdoor watermarks as detectable triggers—specific words or phrases generated with abnormally high frequency—which could aid in model tracing. These advancements collectively push the boundaries of model attribution research. As LLM technology evolves, this field warrants even greater attention.

b) Model Watermarking and Fingerprinting: Our framework uses methods akin to model watermarking and fingerprinting to trace the origins of fine-tuned models. Watermarking embeds identifiable information to verify authenticity or ownership [20]–[26], while fingerprinting captures unique traits to trace model lineage [27]–[31]. By linking these techniques to model features, our framework identifies distinct characteristics that connect fine-tuned models to their pre-trained versions, thereby improving transparency and accountability. Recent studies [11] confirm that fine-tuned models retain identifiable traits of their original pre-trained models.

III. Problem Definition & Formulation

We have a collection of words W, and a text of length n can be viewed as $t = W^n$. Then, an LLM m can be seen as a function mapping an input t_i to an output t_o. In the context of LLMs, t_i is termed the "prompt", and thus, we can denote t_i as p. Under this premise, the process of an LLM generating an output from a prompt can be denoted as $t_o = m(p)$.

In this work, we consider two types of LLMs: the original pre-trained model $m_o \in O$, and the fine-tuned model derived from m_o, denoted $m_f \in F$. Therefore, the fine-tuning process is formalized as a function f: $O \rightarrow F$, mapping an original model m_o to its fine-tuned counterpart m_f.

Now, our objective is to define a function τ: $F \rightarrow O$ that maps a known fine-tuned model m_f back to its original pre-trained model m_o (Figure 1). We seek to identify an appropriate τ to recover the origin of fine-tuned models.

IV. FROM Framework

Our framework, **FROM** (as shown in Figure 2), comprises four key components: (1) *Models Preparation*: Prepares fine-tuned LLMs for training or testing in subsequent modules; (2) *Database Construction*: Generates and optimizes a search space to support the classification methodology of the Classifier; (3) *Single Pair Classifier*: Leverages the models and database from preceding stages to determine which candidate base model (within a given pair) more closely aligns with the fine-tuned LLM, employing search and optimization techniques; and (4) *Knockout Round*: Extends the Single-Pair Classifier's capability into a hierarchical tree-like structure, enabling multi-candidate base model identification beyond binary comparisons. Each component fulfills a distinct yet interconnected role within the framework. The following sections detail their functions and integration to collectively achieve **FROM**'s objectives.

A. Models Preparation

This component, illustrated in Figure 2, generates multiple fine-tuned models to mitigate the scarcity of existing fine-tuned models available for training and testing. We begin by selecting widely used base models and fine-tune them via prompt tuning — a method that achieving performance comparable to full parameter tuning while utilizing fewer parameters [32].

Prompt tuning offers two key advantages: (1) reduced training time, as it only appends trainable prompts to the base model, enabling efficient creation of fine-tuned variants; and (2) minimal storage requirements, since only the task-specific prompts need preservation. To further accelerate inference, we store both the learned prompts and the precomputed output embeddings from training samples. This eliminates redundant re-embedding during inference and significantly expedites classification. Collectively, this approach ensures efficient model generation and deployment.

B. Database Construction

This component builds a diverse database of prompts for the *Single Pair Classifier*'s search step. As shown in Figure 2, we select high-performing LLMs (e.g., GPT-4, Claude-3, Gemini) and instruct them to generate prompts. The instructions include: (1) a detailed description of the task—tracking the origins of fine-tuned models; (2) a requirement for diverse prompt modalities (e.g., topics, structures, lengths) to expand the search space; and (3) multiple rounds of prompt generation with human feedback to ensure variety. The resulting prompts are compiled into a comprehensive database for downstream use.

a) Optimization: To enhance quality, rigorous filtering and optimization are performed during database construction. Specifically, we evaluate candidate prompts and rank each prompt based on performance using the *Single Pair Classifier*. Only top-performing prompts are retained for the final database. Detailed methodology and results of this optimization process are presented in the Experiments (Section V-E).

C. Single Pair Classifier

The *Single Pair Classifier* is the framework's core component, designed to identify the origins of fine-tuned models through pairwise comparisons. As shown in Figure 2, it

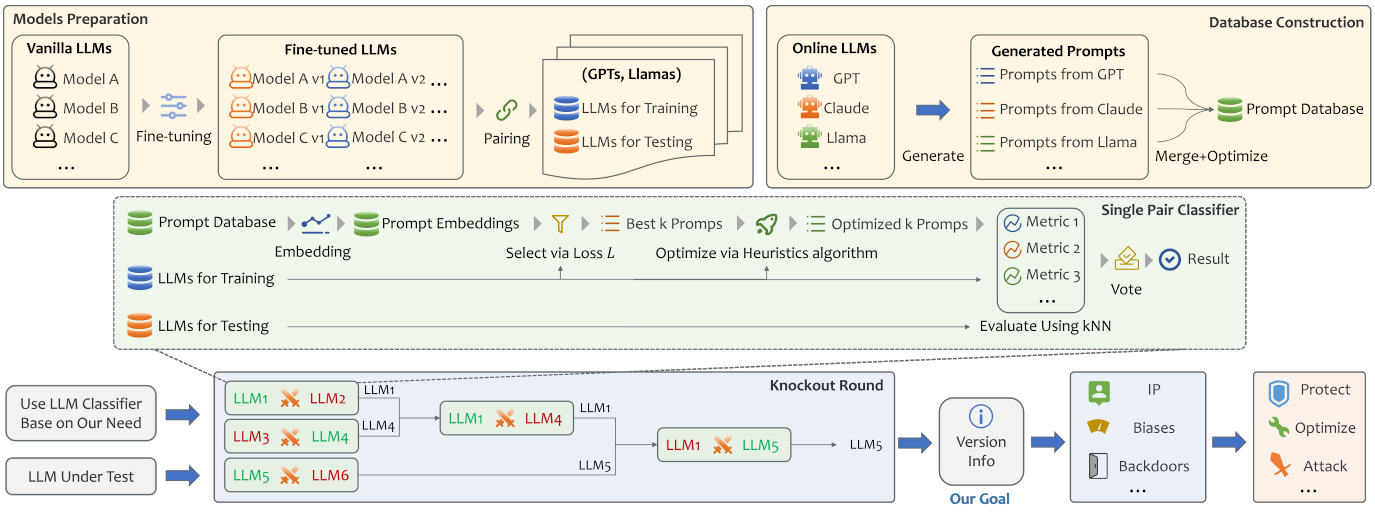


Fig. 2. The **FROM** framework, consisting of (1) *Models Preparation*: fine-tuning LLMs for training/testing; (2) *Database Construction*: building an optimized search space; (3) *Single Pair Classifier*: comparing candidate base models in pairs; and (4) *Knockout Round*: extending comparisons hierarchically for multi-candidate identification.

operates as a binary classifier, determining which of two candidate base models a test model more closely resembles. For example, when comparing a GPT-based test model against GPT and Llama, the classifier favors GPT. However, if the same test model is compared against Claude and Llama, and aligns more closely with Llama, the classifier selects Llama.

This pairwise approach is preferred over a single multi-model classifier for two reasons: (1) it reduces overlaps and clarifies decision boundaries in high-dimensional space, and (2) it avoids performance degradation when prompts effectively distinguish between some models but not others.

The classifier operates as a pipeline with three sequential steps: *Search k Seed Prompts*, *Optimize k Seed Prompts*, and *k Metrics and Vote*. Each step is detailed below.

a) Search k Seed Prompts: This step selects the k most effective prompts from the database created in *Database Construction*. The goal is to identify prompts that best distinguish features between two original models. To evaluate prompts quantitatively, their outputs are embedded into a high-dimensional space using sentence embeddings, which are more suitable for sentences than word embeddings.

Prompt effectiveness is assessed using fine-tuned models from *Model Preparation*. Models are grouped by their origins, and two relevant sets are selected for the classifier. Sets are split into training and testing subsets with equal-sized training groups to avoid bias.

A prompt is effective if, when processed by both model sets, outputs exhibit low intra-set cosine distance but high inter-set distance. We quantify this using a modified contrastive loss function [33] that minimizes intra-class distances while maximizing inter-class distances. The loss function is defined as follows.

Let $E_1 = \{u_1, u_2, \ldots, u_m\}$ and $E_2 = \{v_1, v_2, \ldots, v_n\}$ represent two embedding classes, with $\cos\_\text{sim}(\cdot,\cdot)$ denoting cosine similarity. Intra-class losses are:

$$L_{\text{intra1}} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} (1 - \cos\_\text{sim}(u_i, u_j))$$
$$L_{\text{intra2}} = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (1 - \cos\_\text{sim}(v_i, v_j)) \quad (1)$$

The inter-class loss is:

$$L_{\text{inter}} = \sum_{i=1}^{m} \sum_{j=1}^{n} \max(0, m - (1 - \cos\_\text{sim}(u_i, v_j))) \quad (2)$$

where m is a predefined margin. The final contrastive loss averages all terms:

$$L = \frac{1}{N}(L_{\text{intra1}} + L_{\text{intra2}} + L_{\text{inter}}) \quad (3)$$

Here, N is the total number of terms in the three loss components.

b) Optimize k Seed Prompts: To enhance performance and expand the search space, we employ a modified genetic algorithm (GA) to refine the k seed prompts. The GA treats prompt strings as genes composed of characters and symbols , evaluating their effectiveness using a fitness function based on the contrastive loss. Through iterative generations, the algorithm applies crossover by recombining prompts at random points and mutation via random character alterations to explore optimal solutions beyond the initial database. Prompts with superior fitness are prioritized for selection , ensuring progressive performance enhancement. This natural language-based optimization also maintains semantic coherence, facilitating human comprehension and successful model filtering.

c) k Metrics and Vote: This step evaluates a test model from multiple perspectives using k optimized prompts and aggregates the results via voting. Each prompt functions as a classifier: it is input to two training model sets (representing different original models), with outputs mapped to an embedding space. The test model's output is processed identically. The k-nearest neighbors (kNN) algorithm then classifies the test output based on proximity to training embeddings. This repeats for all k prompts (parameter selection details: Appendix A).

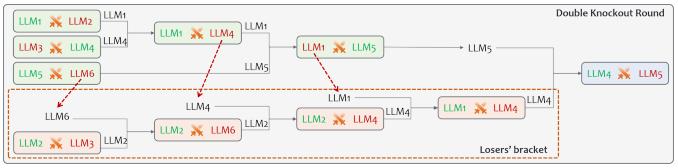


Fig. 3. Double knockout round: The framework extends the *Single Pair Classifier* to handle multiple candidates. By incorporating winners' and losers' brackets, the system improves fault tolerance, ensuring a candidate base model is only eliminated after two losses.

A specialized voting mechanism combines results. Rather than using final kNN outputs directly, it tallies reference points from each kNN iteration. The final classification compares cumulative reference counts across all prompts, with the majority determining the outcome. This approach reduces bias by incorporating intermediate kNN results, yielding a robust evaluation. Comparing with traditional voting (92.8%), our weighted voting mechanism achieves higher accuracy (94.2%).

D. Knockout Round

When multiple model classes ($\geq$ 2) are involved, pairwise comparisons are conducted using a knockout tournament mechanism, as illustrated in Figure 2 and 3. This approach reduces the number of comparisons from $C(n, 2)$ to $n - 1$, significantly enhancing scalability. Introducing new model classes only requires pairwise comparisons with existing ones.

The knockout mechanism partitions models into two groups for pairwise matches, with winners advancing to subsequent rounds. This process halves the number of competitors each round until a final winner is determined.

To mitigate misjudgment risks inherent in traditional knockout tournaments, we implement a double knockout system featuring winners' and losers' brackets, as depicted in Figure 3. Entries commence in the winners' bracket; defeated models transfer to the losers' bracket. Elimination occurs only after a second loss within the losers' bracket. Winners emerging from the losers' bracket re-enter the competition, thereby reducing the impact of single errors and enhancing system robustness.

E. Self-consistency

To address the randomness in large model outputs, we employ self-consistency. This approach involves querying the model multiple times with identical prompts and selecting the majority result as the final answer, ensuring a consensus-based decisions.

Specifically, in the single pair classifier stage, we apply self-consistency by generating three responses per prompt and selecting the majority result as the final decision. This reduces the impact of output variability while enhancing reliability.

F. Confidence Score

To address potential misjudgments when the original model is absent from the candidate pool, we propose two solutions: (1) expanding the candidate model pool while clarifying that judgments on non-candidate models are unreliable, and (2) introducing a confidence score with a threshold to flag unreliable results. When the confidence score falls below the threshold, the result is deemed untrustworthy, and users are notified accordingly.

The confidence score is calculated as follows: for each judgment, we compute the mean of the k cosine distances (d) used in the kNN decision. Since a voting mechanism is applied, we average these means across n votes to obtain the final average distance (D_{final}):

$$D_{final} = \frac{1}{n \cdot k} \sum_{i=1}^{n} \sum_{j=1}^{k} d_{i,j} \tag{4}$$

The confidence score (C) is then derived as:

$$C = 1 - \frac{D_{final}}{2} \tag{5}$$

To determine the threshold, we analyze the confidence scores as detailed in Appendix B. This method provides a balanced and interpretable criterion for evaluating result reliability.

V. Experiments

A. Experimental Setup

a) Original Test Models: We evaluate our framework using nine pre-trained LLMs: TinyLlama_v1.1 [34], GPT-Neo-1.3B [35], Pythia-1.4B [36], GPT-2-XL [37], Meta-Llama-3-8B [38], Mistral-7B [39], Pythia-6.9B, Pythia-12B, Qwen2.5-32B [40].

b) Database: To ensure a broad search space for prompt selection, we construct a diverse prompt database. Using 20 mainstream models including GPT-4 and Claude-3, we generate 150 distinct prompts per model, covering a wide range of modalities, topics, and structures. This results in an initial database of 3,000 prompts, which we then optimize (Section V-E) to retain only the top 1,000 high-quality prompts.

c) Fine-tuning Method: We fine-tune the original LLMs using prompt tuning, as it is computationally efficient and time-effective compared to parameter tuning [32]. Prompt tuning optimizes prompt embeddings by prepending them to the original input to improve downstream task performance. However, our focus is not on task performance but on testing the framework with multiple tuned models; thus, prompt optimization is unnecessary. To this end, we use GPT-4-turbo [1] and Claude-3-Opus [41] to generate 60 diverse prompts per model (10 for training, 50 for testing). These additional prompts assess the framework's generalization ability in unseen scenarios.

d) Embedding Method: We use sentence embeddings to process outputs, selecting gte-large-en-v1.5 [42] to balance performance with computational efficiency.

B. Trace Accuracy without Knockout Round

We evaluated the judgment accuracy of our framework on 10 pairs of original models. Detailed test results are presented in Table I.

Table I shows that our binary classifiers achieve an average accuracy of 94.4%, demonstrating the framework's effectiveness in distinguishing between original model pairs. Notably,

TABLE I. CLASSIFIERS' PERFORMANCE EVALUATION

Model Pair (Classifier)	1st Acc.	2nd Acc.	Total
(TinyLlama_v1.1, GPT-Neo-1.3B)	100%	98%	99%
(TinyLlama_v1.1, Pythia-1.4B)	100%	76%	88%
(TinyLlama_v1.1, GPT-2-XL)	100%	100%	100%
(GPT-Neo-1.3B, Pythia-1.4B)	74%	100%	87%
(GPT-Neo-1.3B, GPT-2-XL)	86%	100%	93%
(Pythia-1.4B, GPT-2-XL)	98%	98%	98%
(Meta-Llama-3-8B, Mistral-7B)	94%	98%	96%
(Meta-Llama-3-8B, Pythia-6.9B)	96%	90%	93%
(Mistral-7B, Pythia-6.9B)	96%	94%	95%
(Pythia-12B, Qwen2.5-32B)	98%	92%	95%
Total Average Accuracy			**94.4%**

Note: Columns "1st Acc." and "2nd Acc." denote the accuracy for the first and second models in the pair, respectively.

TABLE II. CLASSIFIER ACCURACY WITH KNOCKOUT ROUND

Original Models	Accuracy			Average
	Round 1	Round 2	Round 3	
TinyLlama_v1.1	92%	96%	98%	95.3%
GPT-Neo-1.3B	76%	76%	76%	76.0%
Pythia-1.4B	82%	76%	78%	78.7%
GPT-2-XL	94%	94%	90%	92.7%
Pythia-6.9B	92%	90%	94%	92.0%
Meta-Llama-3-8B	96%	90%	94%	93.3%
Mistral-7B	94%	90%	90%	91.3%
Total Average				**88.5%**

Note: This table presents the framework's accuracy in distinguishing multiple original models across three rounds.

classifiers for smaller-parameter models exhibit stronger judgment biases (e.g., 100% for GPT-2-XL vs. 76% for Pythia-1.4B), while larger models show greater consistency. We attribute this to the greater randomness and poorer performance of smaller models, which frequently generate nonsensical outputs (e.g., repetitive text), introducing evaluation biases. In contrast, larger models exhibit lower randomness and better performance, resulting in more stable and less biased judgments.

C. Trace Accuracy with Knockout Round

To evaluate our framework's ability to handle multiple original models, we introduced a Knockout Round (Section IV-D) and expanded our experiments (Section V-B). We tested the framework on fine-tuned models derived from seven original models, using three different grouping orders in the Knockout Round for each model and averaging the results. As shown in Table II, the framework achieved an overall average accuracy of 88.5%, demonstrating its effectiveness in accurately identifying the correct model among multiple candidates. This result underscores the robustness of our approach in multi-model judgment tasks.

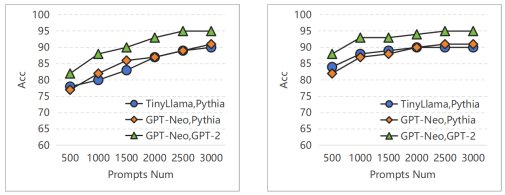
D. Comparison with Baseline

To the best of our knowledge, only one prior study, **Matching Pairs** [11], shares the same application scenario as ours. Additionally, some fingerprinting methods, such as **TRAP** [16] and **ProFLingo** [17], employ concepts similar to our approach. We adapted these methods to our scenario for testing and comparison, as shown in Table III (using models with approximately 7B parameters, as specified earlier).

As demonstrated in Table III, our method achieves superior performance in judgment tasks. Furthermore, Matching Pairs

TABLE III. COMPARISON WITH BASELINE METHODS

Method	Accuracy
Matching Pairs	84.7%
TRAP	90.7%
ProFLingo	91.3%
FROM (Ours)	**94.7%**



(a) Before (b) After

Fig. 4. Database optimization comparison. (a) Before Optimization; (b) After Optimization.

relies on parameter-tuned model classifiers, which require substantial computational resources and exhibit limited scalability. In contrast, our approach is significantly more efficient: it replaces resource-intensive model tuning with a search-and-optimization strategy while only requiring retention of the original model and its tuning prompts—rather than storing all fine-tuned models—thereby substantially reducing storage requirements. The modular framework design also ensures excellent extensibility, allowing seamless integration of new base models and flexible component replacement to incorporate technological advancements. This adaptability ensures compatibility with the ongoing evolution of LLMs. Compared to the other fingerprinting methods, our approach additionally addresses scenarios where the suspected original model is unknown, enabling selection of the most similar candidate from a pool while maintaining one-to-one judgment capability for cases outside the candidate set.

E. Database Optimization

We evaluated the prompt database using three model pairs. First, we define the performance evaluation criteria: (1) Correct judgments are preferable to incorrect ones; (2) Given equal correctness, lower contrastive loss is preferable to higher loss. Based on these criteria, we assessed both original and optimized database versions, retaining prompt sets ranging from 500 to 3000 prompts (in 500-prompt increments) for each condition. As shown in Figure 4, optimization exhibits no significant effect near full retention (i.e., 3000 prompts), where minimal filtering occurs. However, as the number of retained prompts decreases, the optimization effect becomes increasingly pronounced. Furthermore, we observe that retaining more than 1000 optimized prompts yields diminishing performance improvements. Since fewer prompts accelerate computation, we ultimately select 1000 prompts as the optimal database size.

F. Ablation Study

a) Comparison with Random Prompts: To validate our framework's efficacy, we compared it against five randomly generated prompts under identical experimental conditions (Section V-B). Table IV shows an average accuracy of 59.3% for random prompts, representing a 34.9 percentage-point

TABLE IV. ABLATION STUDY RESULTS

Classifier	Random	Search	Optimize	Vote
(TinyLlama, GPT-Neo)	60%	60%	98%	89%
(TinyLlama, Pythia)	67%	71%	91%	89%
(TinyLlama, GPT-2-XL)	53%	83%	98%	98%
(GPT-Neo, Pythia)	52%	71%	87%	85%
(GPT-Neo, GPT-2-XL)	60%	66%	87%	90%
(Pythia, GPT-2-XL)	64%	83%	96%	95%
Average	**59.3%**	**72.3%**	**92.8%**	**91.0%**

Note: This table presents the accuracy outcomes of our ablation experiments. The "Random" column shows results using randomly generated prompts. The "Search" column displays results when omitting the search step. The "Optimize" column indicates results without the optimization step. The "Vote" column details results without the voting mechanism.

decrease relative to our framework. The accuracy range for random prompts (52%–67%) falls substantially below our method's performance, confirming the framework's superiority. Interestingly, random prompts still achieved moderate accuracy, likely because inherent output distinctiveness across models allows even arbitrary prompts to capture some discriminative features, albeit less effectively.

b) Without Search Step: We evaluated the *Search* component's impact by replacing the framework's initial seeds with five random database prompts while maintaining other conditions. Table IV shows an average accuracy of 72.3%, indicating a significant performance drop compared to the complete framework. Individual classifier performance also decreased noticeably. These results suggest that *Search* introduces critical variability in prompt modality, topic, and structure, substantially enhancing judgment accuracy—unlike *Optimization*, whose impact is more subtle.

c) Without Optimization Step: To assess *Optimization*'s contribution, we removed this component and directly evaluated search results. This yielded an average accuracy of 92.8% (Table IV), slightly below the full framework's performance. Individual classifiers showed minor but consistent declines. This marginal difference occurs because *Optimization* refines seeds through subtle adjustments, producing relatively similar high-quality prompts that offer diminishing returns compared to the *Search* component.

d) Without Voting Mechanism: We measured *Vote*'s impact by using only the top prompt from *Optimization*, which reduced average accuracy to 91.0% (Table IV). This demonstrates that *Vote* mitigates model-specific biases through aggregated judgments, improving generalization. Performance degradation varied across classifiers, possibly because some prompts represent distinctive corner cases already well-handled by individual models, limiting voting's added value.

e) Knockout Round Order Robustness.: As detailed in Section V-C and Table II, we tested three distinct grouping orders during knockout rounds. Accuracy variations across orders were small, with a maximum variance of 6% per model. This consistency confirms that our framework's performance remains robust regardless of knockout sequence.

G. Generalization

1) Full-parameter Fine-tuning Models: To comprehensively demonstrate the generalization ability of our method, we conducted testing on a diverse set of 10 parameter-tuned models. We implemented our framework without additional training steps to evaluate the original versions of these models. This approach assessed the robustness and adaptability of our framework across varying conditions, free from further model-specific optimizations. As shown in Table V, our method successfully identified the original configurations for 9 out of 10 models, highlighting its effectiveness in recognizing and adapting to different architectures.

TABLE V. GENERALIZATION TEST RESULTS

Model Name	Original Model	Result	T/F
gpt-neo-1.3B-apps	gpt-neo-1.3B	gpt-neo-1.3B	✓
gpt-neo-1.3B-apps-all-2	gpt-neo-1.3B	gpt-neo-1.3B	✓
gpt-neo-1.3B-resized-embed	gpt-neo-1.3B	gpt-neo-1.3B	✓
TinyLlama_v1.1-flight	TinyLlama_v1.1	TinyLlama_v1.1	✓
Pythia-Greentext-1.4b	pythia-1.4b	pythia-1.4b	✓
fin-pythia-1.4b	pythia-1.4b	pythia-1.4b	✓
pythia-1.4b-gpt4all-pretrain	pythia-1.4b	gpt-neo-1.3B	✗
pythia-1.4b-sft-full	pythia-1.4b	pythia-1.4b	✓
tldr-gpt2-xl	gpt2-xl	gpt2-xl	✓
BetterGPT2	gpt2-xl	gpt2-xl	✓
Total True			**9/10**

TABLE VI. DETECTION RESULTS FOR EXTERNAL MODELS

Model Pool	External Model	Detection Rate
{TinyLlama_v1.1, GPT-Neo-1.3B, Pythia-1.4B}	GPT-2-XL	92%
{Meta-Llama-3-8B, Mistral-7B}	Pythia-6.9B	98%
{Pythia-12B, Meta-Llama-3-8B, Mistral-7B}	Qwen2.5-32B	96%
Average Detection Rate		**95.3%**

Note: This table evaluates the framework's ability to identify models that are not part of the predefined candidate pool.

Our method operates independently of the LLMs themselves, employing a fuzzing-like approach to identify corner cases. It then differentiates models based on their performance variations under these edge conditions. This design ensures theoretical robustness against variations in model types and sizes, enhancing versatility across diverse architectures.

2) Models Not in the Model Pool: We evaluated the performance of the confidence score threshold method in identifying models outside the model pool. As shown in Table VI, the method achieved an average accuracy of 95.3% in detecting non-pool models, demonstrating its effectiveness in distinguishing external models from those within the predefined pool.

3) LoRA Fine-tuning: Preliminary experiments on LoRA-based variants (spanning GPT-Neo, GPT-2-XL, and Pythia) yielded an 11/12 success rate. This also indicates the framework's potential adaptivity to parameter-based fine-tuning methods beyond prompt tuning.

H. Efficiency & Scalability

We evaluated the time and space efficiency of FROM using a single RTX 3090 GPU. The modular design ensures that the prompt database is extremely lightweight, requiring only 64 KB of total storage. The incremental storage cost is less than 1 KB per additional candidate model, such as the prompts for TinyLlama which occupy only 807 bytes. Regarding computational overhead, the identification of a fine-tuned model via the double knockout round is completed in less than 1 minute. While adding a new candidate model to the pool requires approximately 1 hour for offline prompt optimization, this is a one-time cost that does not affect inference speed. These

benchmarks were established on a single GPU, and the use of parallel computing could further accelerate the process.

A critical advantage of the Knockout Round mechanism is its linear algorithmic complexity. By organizing pairwise comparisons in a hierarchical tree structure, the framework reduces the required number of evaluations to $2n - 2$ for a pool of n models. This linear scaling ensures that the system remains efficient even as the number of candidate origins increases. Furthermore, the total size of the candidate pool is naturally limited by the massive computational resources required to train base models from scratch. For example, the Chatbot Arena LLM Leaderboard contains only about 100 distinct models. This industry reality ensures that storage and computational demands remain manageable in practical applications.

VI. Limitations

While **FROM** achieves robust performance, two limitations warrant discussion. First, the framework identifies origins only within a predefined candidate pool. Although expanding this pool mitigates the issue (Section IV-F), evaluations of models outside the pool remain unreliable. To address this, we introduced a confidence score (Section IV-F) to flag low-confidence predictions, reducing the risk of misinterpretation. Second, due to computational constraints, we could not evaluate **FROM** on larger models (e.g., Llama-3-70B). Nevertheless, our method is theoretically model-agnostic, as it relies solely on output embeddings rather than internal architectures.

VII. Conclusion

We propose **FROM**, an efficient and lightweight framework for tracing the origins of fine-tuned LLMs under black-box conditions. **FROM** achieves 94.4% accuracy in identifying base models, surpassing prior methods. Its modular design ensures scalability, while innovations like confidence score enhance deployment reliability. Crucially, **FROM** operates without accessing model parameters, making it practical for real-world applications. Future work will extend **FROM** to multimodal models and integrate continuous learning for evolving model pools.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[3] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 1877–1901.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[6] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[7] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 09, 2020, pp. 13 693–13 696.

[8] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[9] L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh *et al.*, "Ethical and social risks of harm from language models," *arXiv preprint arXiv:2112.04359*, 2021.

[10] Z. Zhang, G. Xiao, Y. Li, T. Lv, F. Qi, Z. Liu, Y. Wang, X. Jiang, and M. Sun, "Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks," *Machine Intelligence Research*, vol. 20, no. 2, pp. 180–193, 2023.

[11] M. Foley, A. Rawat, T. Lee, Y. Hou, G. Picco, and G. Zizzo, "Matching pairs: Attributing fine-tuned models to their pre-trained large language models," in *Annual Meeting of the Association for Computational Linguistics*, 2023.

[12] X. Qi, K. Huang, A. Panda, P. Henderson, M. Wang, and P. Mittal, "Visual adversarial examples jailbreak aligned large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 19, 2024, pp. 21 527–21 536.

[13] E. Shayegani, Y. Dong, and N. Abu-Ghazaleh, "Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models," in *The Twelfth International Conference on Learning Representations*, 2023.

[14] J. Yan, V. Yadav, S. Li, L. Chen, Z. Tang, H. Wang, V. Srinivasan, X. Ren, and H. Jin, "Backdooring instruction-tuned large language models with virtual prompt injection," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 6065–6086.

[15] L. Li, D. Song, X. Li, J. Zeng, R. Ma, and X. Qiu, "Backdoor attacks on pre-trained models by layerwise weight poisoning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3023–3032.

[16] M. Gubri, D. Ulmer, H. Lee, S. Yun, and S. J. Oh, "TRAP: Targeted random adversarial prompt honeypot for black-box identification," in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 11 496–11 517. [Online]. Available: https://aclanthology.org/2024.findings-acl.683/

[17] H. Jin, C. Zhang, S. Shi, W. Lou, and Y. T. Hou, "Proflingo: A fingerprinting-based intellectual property protection scheme for large language models," in *2024 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2024, pp. 1–9.

[18] B. Zeng, L. Wang, Y. Hu, Y. Xu, C. Zhou, X. Wang, Y. Yu, and Z. Lin, "Huref: Human-readable fingerprint for large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 126 332–126 362, 2024.

[19] E. Lucas and T. Havens, "GPTs don't keep secrets: Searching for backdoor watermark triggers in autoregressive language models," in *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, A. Ovalle, K.-W. Chang, N. Mehrabi, Y. Pruksachatkun, A. Galystan, J. Dhamala, A. Verma, T. Cao, A. Kumar, and R. Gupta, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 242–248. [Online]. Available: https://aclanthology.org/2023.trustnlp-1.21/

[20] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by

backdooring," in *27th USENIX security symposium (USENIX Security 18)*, 2018, pp. 1615–1631.

[21] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *30th USENIX security symposium (USENIX Security 21)*, 2021, pp. 1937–1954.

[22] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, 2017, pp. 269–277.

[23] Y. Li, L. Zhu, X. Jia, Y. Jiang, S.-T. Xia, and X. Cao, "Defending against model stealing via verifying embedded external features," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 2, 2022, pp. 1464–1472.

[24] T. Cong, X. He, and Y. Zhang, "Sslguard: A watermarking scheme for self-supervised learning pre-trained encoders," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 579–593.

[25] A. Bansal, P.-y. Chiang, M. J. Curry, R. Jain, C. Wigington, V. Manjunatha, J. P. Dickerson, and T. Goldstein, "Certified neural network watermarks with randomized smoothing," in *International Conference on Machine Learning*. PMLR, 2022, pp. 1450–1465.

[26] L. Wang, S. Xu, R. Xu, X. Wang, and Q. Zhu, "Non-transferable learning: A new approach for model ownership verification and applicability authorization," in *10th International Conference on Learning Representations, ICLR 2022*, 2022.

[27] N. Lukas, Y. Zhang, and F. Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," in *International Conference on Learning Representations*, 2021.

[28] J. Chen, J. Wang, T. Peng, Y. Sun, P. Cheng, S. Ji, X. Ma, B. Li, and D. Song, "Copy, right? a testing framework for copyright protection of deep learning models," in *2022 IEEE symposium on security and privacy (SP)*. IEEE, 2022, pp. 824–841.

[29] Z. Peng, S. Li, G. Chen, C. Zhang, H. Zhu, and M. Xue, "Fingerprinting deep neural networks globally via universal adversarial perturbations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 13 430–13 439.

[30] X. Pan, Y. Yan, M. Zhang, and M. Yang, "Metav: A meta-verifier approach to task-agnostic model fingerprinting," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1327–1336.

[31] G. Liu, T. Xu, X. Ma, and C. Wang, "Your model trains on my data? protecting intellectual property of training data via membership fingerprint authentication," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1024–1037, 2022.

[32] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3045–3059.

[33] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[34] P. Zhang, G. Zeng, T. Wang, and W. Lu, "Tinyllama: An open-source small language model," *arXiv preprint arXiv:2401.02385*, 2024.

[35] S. Black, L. Gao, P. Wang, C. Leahy, and S. Biderman, "Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow," *If you use this software, please cite it using these metadata*, vol. 58, no. 2, 2021.

[36] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff *et al.*, "Pythia: A suite for analyzing large language models across training and scaling," in *International Conference on Machine Learning*. PMLR, 2023, pp. 2397–2430.

[37] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[38] AI@Meta, "Llama 3 model card," 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[39] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[40] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu, "Qwen2.5 technical report," *arXiv preprint arXiv:2412.15115*, 2024.

[41] Anthropic, "The claude 3 model family: Opus, sonnet, haiku," in *Claude-3 Model Card*, 2024.

[42] X. Zhang, Y. Zhang, D. Long, W. Xie, Z. Dai, J. Tang, H. Lin, B. Yang, P. Xie, F. Huang *et al.*, "mgte: Generalized long-context text representation and reranking models for multilingual text retrieval," *arXiv preprint arXiv:2407.19669*, 2024.

APPENDIX A
K FOR KNN

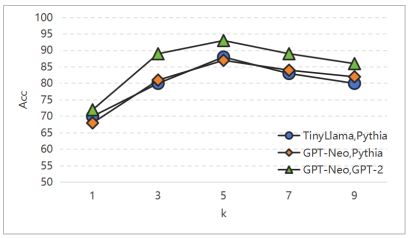
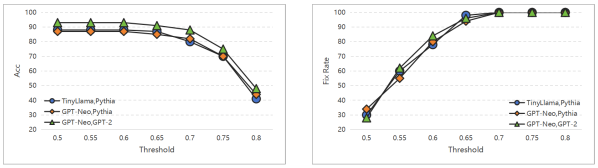


Fig. 5. Impact of parameter k on kNN classification accuracy.

We conducted experiments on the selection of the hyperparameter k in kNN using three model groups, as illustrated in Figure 5. The results demonstrate that the accuracy initially improves with increasing k but subsequently declines. We attribute this trend to the fact that smaller k-values are more susceptible to noise, leading to suboptimal performance. Conversely, excessively large k-values introduce excessive voting influence from distant data points, thereby degrading classification effectiveness. Based on the experimental results, we ultimately selected $k = 5$ as the optimal value for our method.

APPENDIX B
CONFIDENCE SCORE THRESHOLD



(a) Accuracy (b) Fix rate

Fig. 6. The impact of the confidence score threshold on (a) classification accuracy and (b) model fix rate.

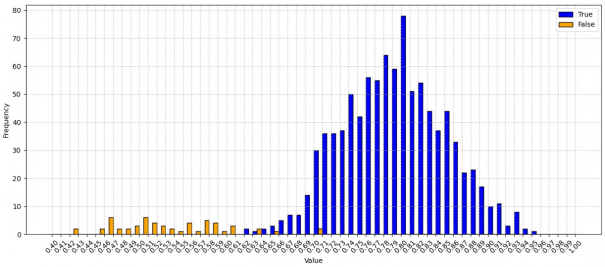


Fig. 7. Distribution of confidence scores. Blue and orange regions represent correctly and incorrectly classified samples, respectively.

We conducted experiments to evaluate the selection of the confidence score threshold, examining its impact on accuracy and the fix rate (i.e., the detection rate for misclassified cases) at different values (as shown in Figures 6(a) and 6(b)). Additionally, we analyzed the distribution of confidence scores in our experiments (Figure 7). The results indicate that as the threshold increases, accuracy decreases while the fix rate improves. Based on these findings, we recommend setting the confidence score threshold to 0.65 for optimal performance in this scenario.