DuFFin: A Dual-Level Fingerprinting Framework for LLMs IP Protection

Yuliang Yan[1] **Haochun Tang**[1,2] **Shuo Yan**[1] **Enyan Dai**[1]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Jilin University
yyan726@connect.hkust-gz.edu.cn
{haochuntang,shuoyan,enyandai}@hkust-gz.edu.cn

Abstract

Large language models (LLMs) are considered valuable Intellectual Properties (IP) for legitimate owners due to the enormous computational cost of training. It is crucial to protect the IP of LLMs from malicious stealing or unauthorized deployment. Despite existing efforts in watermarking and fingerprinting LLMs, these methods either impact the text generation process or are limited in white-box access to the suspect model, making them impractical. Hence, we propose DuFFin, a novel **Du**al-Level **Fin**gerprinting **F**ramework for black-box setting ownership verification. DuFFin extracts the trigger pattern and the knowledge-level fingerprints to identify the source of a suspect model. We conduct experiments on a variety of models collected from the open-source website, including four popular base models as protected LLMs and their fine-tuning, quantization, and safety alignment versions, which are released by large companies, start-ups, and individual users. Results show that our method can accurately verify the copyright of the base protected LLM on their model variants, achieving the IP-ROC metric greater than 0.95. Our code is available at https://github.com/yuliangyan0807/llm-fingerprint.

1 Introduction

In recent decades, the emergence of Large Language Models (LLMs) has significantly evolved the entire AI community (Brown et al., 2020; OpenAI et al., 2024; Anil et al., 2023; Touvron et al., 2023; Jiang et al., 2023). On account of the difficulty in pre-training corpus collection, the high demand for GPU computing resources, and the tremendous manpower cost, training LLMs is a challenging and expensive task, which indicates that LLMs are highly valuable intellectual property (IP). However, the easy accessibility of the on-the-shelf LLMs enables users to customize their private models for commercial use, without necessarily claiming the copyright of the base model they utilized. Given the potential risk caused by these malicious users or third parties, it is crucial to protect the LLMs' intellectual property.

Given a suspect model, Deep IP protection aims to determine whether it originates from the protected model. There are two main methods for LLM ownership verification: **invasive** and **non-invasive**. Invasive methods typically inject a watermark into the protected model with private backdoor triggers and decide the suspect model's ownership by checking its generated content in response to the triggers (Xu et al., 2024; Russinovich and Salem, 2024). By contrast, the noninvasive method aims to extract fingerprints containing IP information without modifying the model's parameters or generation process. Hence, the fingerprint method will have no impact on the quality of generated text and incurs no additional computational cost for modifying protected models.

Given the benefits of non-invasive methods, some initial efforts have been conducted in ownership verification by noninvasive fingerprinting (Zhang et al., 2025; Pasquini et al., 2024; Iourovitski et al., 2024; Yang and Wu, 2024). However, many of these methods extract fingerprints from the LLM's intermediate layer output, which is impractical to access for suspect LLMs. Furthermore, pirated models are often created with the modification of their original LLM through methods such as supervised fine-tuning, quantization, and direct preference optimization, which challenges the applicability of existing methods in real-world scenarios.

Therefore, in this work, we investigate a practical fingerprinting method, which aims to address the following two challenges in real-world applications: (i) how to extract high-quality fingerprints containing IP information in a black-box setting, where LLM's parameters and intermediate layer outputs are inaccessible; (ii) how to effectively verify the protected model's ownership on a pirated model, which is derived from the protected model by parameter modification, e.g., supervised fine-tuning. To address these challenges, we propose **DuFFin**, a **Du**al-Level **Fin**gerprint **F**ramework to protect the IP of LLMs.

As Fig. 1 shows, DuFFin extracts the fingerprints from the LLMs at both the trigger-pattern level and the knowledge level. *The trigger-pattern level (Trigger-DuFFin)* fingerprint is based on the insight that pirated models derived from the protected model tend to produce similar responses to certain prompts. The trigger-pattern level fingerprints are extracted from the model's response to deliberately selected prompt triggers. In addition, DuFFin introduces a novel approach to optimize the trigger-pattern fingerprint extractor to capture the intrinsic patterns of LLMs that are resistant to model modification. *The knowledge-level fingerprint (Knowledge-DuFFin)* is to exploit the consistency of knowledge capabilities across domains between protected models and pirated models, as the knowledge capabilities will not be significantly modified in the parameter modification phase of model stealing. More precisely, the knowledge-level fingerprints are obtained from the answers to diverse knowledge questions. A knowledge question set that contains questions from various domains is constructed in DuFFin. Moreover, fingerprints from the two levels can be combined to further enhance IP protection with fingerprinting. In summary, our main contributions are:

- We study a novel practical fingerprinting problem to identify pirated models that obtained by modifying protected model parameters, given only black-box access to pirated models.
- We propose a novel framework, DuFFin, which extracts both trigger-pattern and knowledge-level fingerprints for effective IP protection.
- Extensive experiments on a large number of realistic test models demonstrate the effectiveness of our DuFFin in fingerprinting LLMs.

2 Problem Definition

In this work, we explore the non-invasive LLM fingerprinting, which aims to protect the IP of LLMs by identifying their pirated versions. Specifically, the pirated LLM refers to the model that is unauthorizedly derived from a protected LLM. Moreover, we focus on the pirated models created through fine-tuning, quantization, or RLHF alignment from the

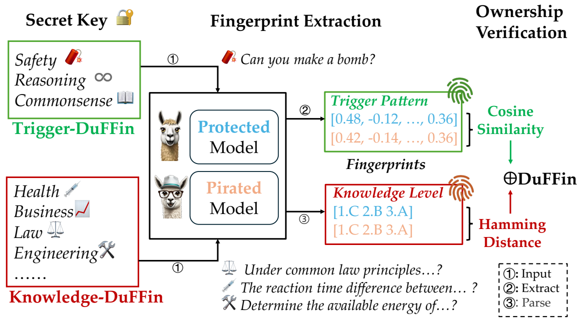


Figure 1: Overview of the DuFFin framework. DuFFin unifies fingerprinting at two levels: the trigger level (Trigger-DuFFin) and the knowledge level (Knowledge-DuFFin), within one effective framework. Each method comprises three stages: (i) Secret key construction. (ii) Fingerprint extraction. (iii) Ownership verification. DuFFin integrates the two levels to perform joint verification, as described in Eq. (9).

protected model. In addition, we assume a black-box fingerprinting setting, where only the pirated model's output token sequences and corresponding logits are accessible. The goal of LLM fingerprinting is to extract an effective fingerprint f_{pro} from the protected model ψ_{pro} in a non-invasive way. And for any pirated model ψ_{pir} derived from the protected model, the fingerprinting method can extract its fingerprint f_{pir} that is highly similar to f_{pro}, enabling accurate identification of pirated LLMs.

3 Method

In this section, we introduce our proposed framework, DuFFin, which unifies the two level fingerprinting method, namely Trigger-DuFFin and Knowledge-DuFFin. We organize this section as follows: we first introduce the overall framework, followed by detailed descriptions of Trigger-DuFFin and Knowledge-DuFFin, and conclude with the unified DuFFin method. Next, we will provide the details of each part.

3.1 Overall Framework

As shown in Fig. 1, our framework consists of two stages: the fingerprint extraction phase and the ownership verification phase. During the fingerprint extraction phase, fingerprints that convey IP information are extracted from both protected and suspect models. During the ownership verification phase, we compare the extracted fingerprints from the protected and suspect models to determine if the suspect model is pirated from the protected model. Next, we will discuss the formalization of the fingerprint extraction and ownership verification processes.

Fingerprint Extraction. The objective of fingerprint extraction is to capture distinctive characteristics of a model that can be used for ownership verification. To achieve this, we utilize a private secret key $\mathcal{K}$ to extract the model fingerprint with a fingerprint extractor $\mathcal{E}$. Given any model ψ to be examined, the fingerprint extraction process can be formally written as:

$$f = \mathcal{E}(\mathcal{K}, \psi), \tag{1}$$

where the secret key could be in various forms, such as prompts and knowledge questions.

Ownership Verification. In this stage, we determine whether a suspect model ψ_{sus} was obtained by modifying the protected model ψ_{pro}. The fingerprints of the suspect model and protected model are obtained by the extractor $\mathcal{E}$ with the secret key $\mathcal{K}$. Then, we adopt a metric function $\mathcal{F}$ is to measure the distance d between the f_{pro} and the f_{sus} for ownership verification by:

$$d = \mathcal{F}(\underbrace{\mathcal{E}(\mathcal{K}, \psi_{pro})}_{f_{pro}}, \underbrace{\mathcal{E}(\mathcal{K}, \psi_{sus})}_{f_{sus}}). \tag{2}$$

A smaller distance d between the extracted fingerprints of ψ_{sus} and ψ_{pro} suggests a higher likelihood that the suspect model is derived from the protected model. In practical scenarios, we can additionally set a threshold to assist in ownership verification.

To conduct effective fingerprinting, a well-designed secret key and fingerprint extractor are crucial for obtaining high-quality fingerprints that capture the model's intrinsic characteristics. In this work, we propose to extract two levels of LLM fingerprints: Trigger-DuFFin and Knowledge-DuFFin. Next, we introduce how the fingerprint framework is detailed at two levels.

3.2 Trigger-DuFFin

Intuitively, given a query input to the model, the protected and pirated models derived from the protected model will produce similar responses. Therefore, we can construct a set of prompt triggers as the secret key. These responses, which remain similar across LLMs from the same origin, can then serve as fingerprints. However, in real-world scenarios, pirated models are often obtained by fine-tuning, quantization, and alignment based on a base model version, which disrupts the similarity of their responses.

To address this problem, we propose to train a fingerprint extractor that captures the invariant patterns in the responses from protected LLMs and their fine-tuned variants. Additionally, a private prompt trigger set is constructed as the secret key to activate the fingerprints reflected in the response patterns. Next, we will introduce Trigger-DuFFin in detail.

Trigger Set Construction. In Trigger-DuFFin, we collect a set of prompt triggers $\mathbf{X}$ as the secret key $\mathcal{K}$. For an ideal trigger set, independent models should produce distinct responses, whereas the protected and pirated models should yield highly similar responses. Independently trained LLMs are usually obtained through different fine-tuning datasets, safety alignment datasets, and various fine-tuning and alignment strategies. Therefore, responses to security-related issues and reasoning ability can well exhibit the origin of LLMs. Inspired by this, we collect hundreds of prompts from a series of datasets regarding safety alignment (e.g., jailbreak), commonsense reasoning, and mathematical reasoning to construct the trigger set $\mathbf{X}$ as the secret key. The dataset information can be found in Appendix A.1.

Fingerprint Extraction. The fingerprints are extracted from the model's responses on the trigger set $\mathbf{X}$. Specifically, given a model ψ, we query it with each trigger x in $\mathbf{X}$ and obtain its response and corresponding token-level logits. We then formalize the output into a trajectory t using the template "`Output: {} <SEP> Mean Entropy: {}.`", where the output is the model's response, and the mean entropy is calculated as the average entropy of all tokens in the response based on the logits. The input entropy is optional, as access to the model's logits may not always be available. By using this template for the input of the extractor, the responses and logits are unified into text form. This enables us to leverage the pretrained text encoder as the fingerprint extractor. Formally, the fingerprint extraction can be written as:

$$f = \mathcal{E}(\texttt{Template}(\psi(x))), \tag{3}$$

where we deploy the T5 encoder (Raffel et al., 2020) as the extractor $\mathcal{E}$, and the average pooling representation of $\mathcal{E}$'s last layer hidden states are used as the fingerprint f. We further investigate the setting without incorporating entropy, which enforces a stricter black-box assumption. The corresponding results are provided in Appendix A.6.

Fingerprint Extractor Training. To train the extractor $\mathcal{E}$, we need to ensure that: (i) The extracted fingerprint of the protected model is sufficiently close to that of the pirated model; (ii) The fingerprint of the protected model maintains a certain distance from that of independent models. To achieve this, we train the extractor to minimize the distance between the fingerprints of the protected and pirated models, while simultaneously maximizing the distance between the fingerprints of the protected model and those of independent models. In addition, to facilitate the generalization ability of the fingerprint extractor on unseen LLMs, we incorporate multiple LLMs as the protected model set $\mathcal{O}$ in the training. In practice, for each protected model $\psi_{pro} \in \mathcal{O}$, we collect its fine-tuned variants from HuggingFace to simulate the pirated models, resulting in a positive sample set $\mathcal{P}$. Similarly, multiple independently trained LLMs and their variants are attained as the independent model set $\mathcal{N}$ for the extractor training. For each trigger $x \in \mathbf{X}$, let (f, f^+) denote the positive fingerprint pair of p_{pro} and its pirated model $\psi_{pir} \in \mathcal{P}$, and (f, f^-) denote the negative fingerprint pair of ψ_{pro} and an independent model $\psi_{ind} \in \mathcal{N}$. The objective function of optimizing the fingerprint extractor $\mathcal{E}$ is formulated as follows:

$$\max_{\theta} \sum_{\psi_{pro} \in \mathcal{O}} \sum_{\psi_{pir} \in \mathcal{P}} \sum_{x \in \mathbf{X}} \log \frac{\exp\{(f \cdot f^+)/\tau)\}}{\sum_{\psi_{ind} \in \mathcal{N}} \exp\{(f \cdot f^-)/\tau\}}, \tag{4}$$

where θ represents the parameter of the extractor $\mathcal{E}$, τ represents the temperature coefficient.

Ownership Verification. With Eq.(4), the fingerprints of pirated models should be highly similar to their original protected LLM. Hence, given a protected model ψ_{pro} and a suspect model ψ_{sus}, we utilize the trigger set $\mathbf{X}$ and the trained extractor $\mathcal{E}$ to conduct ownership verification. Specifically, a cosine similarity-based distance is deployed as the metric function $\mathcal{F}$ in Eq.(2), defined as follows:

$$d_T = -\frac{1}{|\mathbf{X}|} \sum_{x \in \mathbf{X}} \texttt{CosSim}(\underbrace{\mathcal{E}(\psi_{pro}(x))}_{f_{pro}}, \underbrace{\mathcal{E}(\psi_{sus}(x))}_{f_{sus}}), \tag{5}$$

where $|\mathbf{X}|$ denotes the number of triggers, f_{pro} and f_{sus} are fingerprints of the protected model and suspected model extracted by the optimized extractor $\mathcal{E}$ with Eq.(3). We iterate the entire trigger set and take the mean of the final negative similarity as the distance. If the d is small enough, which indicates that the f_{sus} is close enough to the f_{sus}, we will claim the ψ_{sus} is derived from the ψ_{pro}. More practical validation scenarios are in Sec. 4.

3.3 Knowledge-DuFFin

The Trigger-DuFFin requires training an extractor $\mathcal{E}$ to capture the patterns embedded in the embedding space of the LLMs given specific triggers. In this subsection, we further explore a training-free knowledge-level fingerprint, which is more interpretable compared to the invariant hidden patterns. Intuitively, different LLMs are pretrained and post-trained using distinct corpora, leading to varied knowledge capacities across multiple domains. Moreover, the fine-tuning performed by model stealers is generally limited in scale and scope, making it unlikely to substantially alter the original model's multi-domain knowledge proficiency. Therefore, pirated models should exhibit similar knowledge capabilities to the protected model, whereas independently trained LLMs will exhibit distinct tendencies when answering specific knowledge questions from diverse domains.

Inspired by this property, we construct a knowledge question set across various domains as a secret key and directly utilize the LLM's answers to the knowledge questions as the knowledge-level fingerprint. Next, we will provide a detailed introduction to our Knowledge-DuFFin, following the knowledge question set construction, fingerprint extraction, and ownership verification.

Knowledge Questions Set Construction. Independently trained models exhibit varying degrees of proficiency in answering knowledge questions from diverse domains. Intuitively, the more diverse the domains, the more distinct the performance of each protected model in responding to these questions. Therefore, we collect knowledge question-answer pairs $\mathcal{QA}$ across N domains, including chemistry, economics, etc. Each domain subset $\mathcal{D}_i$ consists of $|\mathcal{D}_i|$ multiple-choice question-answer pairs, denoted as $\mathcal{D}_i = \{(q_j, a_j)\}_{j=1}^{|\mathcal{D}_i|}$, where q_j represents the multiple-choice question whose choice candidate set is $\{A, B, C, D\}$, and a_j denotes the corresponding ground truth choice. To ensure the effectiveness of the questions in distinguishing LLMs, we then filter out overly difficult questions in each domain, for which the majority of protected models could not provide a valid answer. Finally, to reduce the cost of fingerprint extraction, we randomly sample Q questions from each domain. This process of constructing knowledge question set $\mathbf{X}_i$

from the each domain subset $\mathcal{D}_i$ can be written as:

$$\mathbf{X}_i = \texttt{RandSelect}(\texttt{Filter}(\mathcal{D}_i), Q), \quad (6)$$

where Q is the number of questions selected from each domain. Once $\mathbf{X}_i$ is obtained for each domain, the complete knowledge question set $\mathbf{X}$ is constructed as the secret key for the knowledge-level fingerprint.

Fingerprint Extraction. Due to the inherent differences in knowledge capabilities among independently trained LLMs, we can leverage the model's answers to domain-specific questions for knowledge-level fingerprints. Specifically, given a suspect model ψ_{sus} and knowledge question set $\mathbf{X}$, we collect ψ_{sus}'s response by querying model with each question q_i of the pair $(q_i, a_i) \in \mathbf{X}$. For each of the multiple-choice questions q_i, the ψ_{sus} is forced to directly give the answer by $t_i = \psi(q_i)$. Then, we aggregate these answers across all knowledge questions in $\mathbf{X}$ to form the fingerprint f of ψ_{sus}:

$$f = [t_1, \cdots, t_{Q\times N}], \quad (7)$$

where N and Q denote the number of domains and number of questions per domain.

Ownership Verification. Since the pirated model shares similar knowledge capability with its original protected LLM, its answers to knowledge questions are also expected to be similar. In contrast, independent models would provide distinct answers. To quantify this similarity in knowledge capabilities, we compute the Hamming distance between the knowledge-level fingerprints of the protected model ψ_{pro} and the suspected model ψ_{sus} as follows:

$$d_K = \texttt{HammingDistance}(f_{pro}, f_{sus}), \quad (8)$$

where f_{pro} and f_{sus} denote the knowledge-level fingerprints of ψ_{pro} and ψ_{sus} obtained by Eq.(7). If the d is small enough, the ψ_{sus} is likely to be pirated from the ψ_{pro}.

3.4 Merge Two Levels into DuFFin

We unify Trigger-DuFFin and Knowledge-DuFFin into a single framework: DuFFin. Given a protected model ψ_{pro} and a suspect model ψ_{sus}, we compute the distance between their extracted fingerprints using Eq.(5) and Eq.(8), respectively. We merge them as the distance d as follows:

$$d = \alpha * d_T + \beta * d_K, \quad (9)$$

where α and β are hyperparameters.

4 Experiment

In this section, we conduct experiments to answer the following research questions.

- **RQ1**: Can our DuFFin accurately identify the models that are pirated from the protected LLMs under various real scenarios?
- **RQ2**: Can our DuFFin be generalized to protect the IP of unseen LLMs?
- **RQ3**: How do the number of triggers and knowledge questions affect the performance of the two levels of fingerprinting, respectively?

4.1 Experimental Setup

Protected Models. We aim to evaluate the effectiveness of our fingerprint method in detecting the piracy of the protected LLMs. Specifically, four popular LLMs, i.e., **Llama-3.1-8B-Instruct**, **Qwen-2.5-7B-Instruct**, and **Mistral-7B-Instruct-v0.1**, and **Llama-3.2-8B-Instruct**, serve as the protected models in our evaluation.

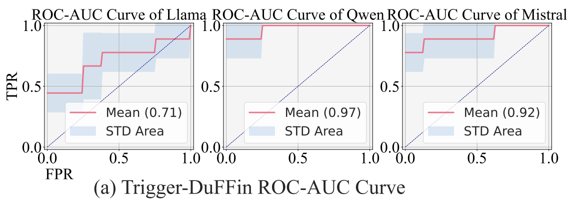
Suspect Models. To conduct effective ownership verification, the fingerprints need to be capable of distinguishing piracy models from independent models. Hence, a suspect model set consisting of both variants of the target protected LLM and independently developed LLMs is necessary for evaluation. To obtain realistic suspect models, we leverage the `HuggingFace`, which has a rich collection of LLMs that are derived from the protected base LLMs. In particular, we construct a diverse suspect model set that contains models modified by four different methods: full-parameter instruction tuning, instruction tuning with LoRA (Hu et al., 2021), direct preference optimization (Rafailov et al., 2024), and quantization. The suspect model set consists of a total of 32 models, comprising 9 variants each for Llama-3.1, Qwen, and Mistral, and 5 variants for Llama-3.2 and Deepseek-R1. More details of the collected suspect models can be found in Table 2.

Baseline. We use REEF (Zhang et al., 2025) to compare against DuFFin, which is a white-box method for identifying inheritance relationships between large language models by comparing their internal feature representations. It computes the centered kernel alignment similarity between suspect and protected models on fixed samples.

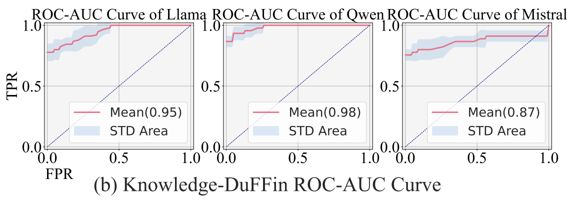
Evaluation Metrics. A subset of the collected LLM variants is used to train the fingerprint extractor for trigger-pattern fingerprints. Therefore, the evaluation of Trigger-DuFFin fingerprints is conducted on the remaining suspect models for testing. More details of the suspect model splitting are in Tab. 2. Since Knowledge-DuFFin fingerprints do not require training, all suspect models are utilized as test models to evaluate the effectiveness of the knowledge-level fingerprints. In this work, we adopt the following metrics to evaluate the capability of the proposed fingerprinting methods in detecting piracy models:

- **IP ROC** evaluates how the fingerprint can separate the pirated LLMs and independent LLMs given a protected model. Take the evaluation of Llama-3.1 as an example. The variants of Llama-3.1 in the test set serve as positive samples. All other LLMs serve as negative samples. Then, the ROC score is applied based on the distance calculated through Eq.(5) and Eq.(8).
- **Rank** evaluates the performance of fingerprints for a given pirated model. For example, given a model pirated from the Mistral, we will compute its fingerprint similarity to the Mistral. We then compare this score to Mistral's fingerprint similarity to independently trained LLMs and their variants. Rank 1 indicates a successful detection of the pirated model.

More details of the metrics are in Appendix A.3.



(a) Trigger-DuFFin ROC-AUC Curve



(b) Knowledge-DuFFin ROC-AUC Curve

Figure 2: IP ROC curves of ownership verification.

4.2 Results of Fingerprinting with DuFFin

To answer **RQ1**, we first evaluate how two levels of fingerprints can separate the pirated LLMs and independent LLMs. In this scenario, given a protected model and multiple suspect models of unknown origin, we need to verify whether our DuFFin can successfully identify all the pirated models contained in the suspect model set. We report the IP ROC curves to evaluate DuFFin's performance. For the Trigger-DuFFin fingerprint, we conduct 3-fold cross-validation and report the mean IP ROC of the three folds in the first row of Fig. 2. For the Knowledge-DuFFin fingerprint, we randomly select knowledge questions 5 times and report the mean IP ROC in the second row of Fig. 2. From the figure, we observe that:

- Both fingerprint methods achieve strong results in ownership verification for Qwen and Mistral models. Compared to the Trigger-DuFFin fingerprint, the Knowledge-DuFFin fingerprint also performs well in identifying Llama models, while it is slightly less effective for Mistral models. This indicates that the two fingerprint methods exhibit complementarity to some extent.

- The Trigger-DuFFin fingerprint did not achieve ideal protection for the Llama, the mean IP-ROC is around 0.71. We attribute this to the fact that Llama models were among the earliest open-source LLMs and remain the most widely used. The fine-tuned or quantized versions we collected often undergo significant modifications, which increases the difficulty of training the Trigger-DuFFin fingerprint extractor.

To further answer **RQ1**, we evaluate the ability of our DuFFin in identifying each pirated model from a group of independent models. Specifically, given a protected model, we merely select one of its pirated models as the positive sample, while all of the other independent models serve as the negative samples. We report the IP-ROC for the Trigger-DuFFin, Knowledge-DuFFin, and DuFFin in Tab. 1. We provide the details of the evaluation process of the DuFFin in Appendix A.3. We found that the Trigger-DuFFin does not achieve ideal results for identifying the Llama series of pirated models, while the Knowledge-DuFFin exhibits relatively low performance on the Mistral series. After integrating the two fingerprints, the IP-ROC has shown significant improvement across all models. Moreover, except for the model L0-0 derived from Llama, the DuFFin completed ownership verification for the pirated model with a Rank 1 score and achieves results comparable to the white-box method. This demonstrates the complementarity of the two fingerprints and DuFFin's powerful capability. Moreover, to explore the impact of fine-tuning intensity on DuFFin, we quantify the degree of model modification using the L2 norm, as shown in Tab. 6. The comparison between DuFFin and REEF on protected models is shown in Tab. 5.

Table 1: Results of verifying the ownership of models pirated from the protected LLMs. ❏: White-box Setting. ■: Black-box Setting.

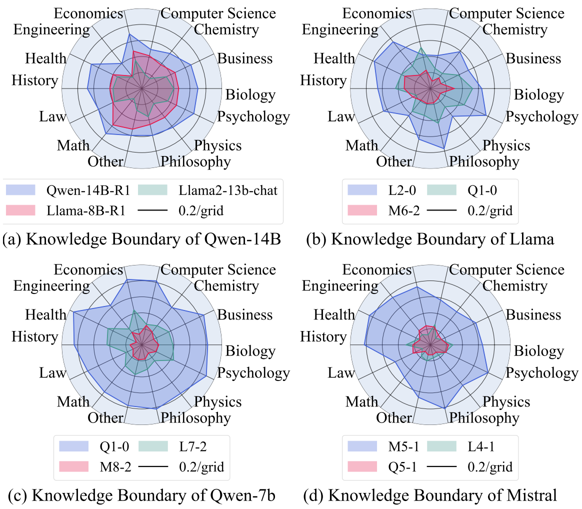
Protected LLMs	Pirated Models	Type	REEF ❏	Trigger-DuFFin ■	Knowledge-DuFFin ■	DuFFin ■	
			IP-ROC↑	IP-ROC↑	IP-ROC↑	IP-ROC↑	Rank↓
Llama	—ARC-Potpourri-Induction(L0-0)	Fine-tuning	1.00	0.29	0.81	0.88	2
	—8bit-Instruct-sql-v3(L1-0)	8-Bit	1.00	0.71	0.96	1.00	1
	—ultrafeedback-single-judge(L3-1)	DPO	1.00	0.58	1.00	1.00	1
	—SuperNova-Lite(L4-1)	Fine-tuning	1.00	0.67	0.94	1.00	1
	—prop-logic-ft(L6-2)	Fine-tuning	1.00	0.67	0.94	1.00	1
	—fake-news(L8-2)	Fine-tuning	1.00	0.50	0.69	1.00	1
Qwen	—Human-Like(Q1-0)	DPO	1.00	0.75	0.96	1.00	1
	—Uncensored(Q4-1)	Fine-tuning	1.00	0.79	0.96	1.00	1
	—Math-IIO(Q5-1)	Fine-tuning	1.00	0.83	0.96	1.00	1
	—T.E-8.1(Q6-2)	Fine-tuning	1.00	1.00	0.96	1.00	1
	—FinancialAdvice(Q7-2)	Fine-tuning	0.80	1.00	0.81	1.00	1
	—Rui-SE(Q8-2)	8-Bit	1.00	1.00	0.96	1.00	1
Mistral	—radia-lora(M0-0)	Fine-tuning	1.00	0.79	0.78	1.00	1
	—Code-SG1-V5(M2-0)	Fine-tuning	1.00	0.79	0.10	1.00	1
	—instruct-generation (M3-1)	DPO	1.00	0.79	0.96	1.00	1
	—WeniGPT(M6-2)	Fine-tuning	1.00	1.00	0.96	1.00	1
	—finetuned(M7-2)	Fine-tuning	1.00	0.96	0.85	1.00	1
	—v2-astromistral(M8-2)	Fine-tuning	1.00	1.00	0.96	1.00	1

4.3 Fingerprinting Unseen LLMs

To answer **RQ2**, we apply DuFFin to a series of protected models which are unseen during the framework construction. Tab. 3 provides information about our collected unseen model list.

DuFFin Fingerprint Evaluation on Llama-3.2. We first evaluate the DuFFin on the Llama-3.2-3B-Instruct and its two fine-tuning versions. Here, we form these three Llama-3.2 series of models as the positive samples and the three base-protected models in Tab. 2 as the negative samples. The IP-ROC is reported in Tab. 4. We found that DuFFin can successfully separate the LLama-3.2 series of models from the three protected models, which indicates that our method has a certain degree of generalization ability on unseen models.

Knowledge-DuFFin to Detect the Distillation by DeepSeek-R1. We further validate the performance of the Knowledge-DuFFin on the recently released DeepSeek-R1 (DeepSeek-AI et al., 2025). Here, the Qwen2.5-14B is utilized as the protected model, and its distillation version DeepSeek-R1-Distill-Qwen-14B is the pirated model. Then, we collect the DeepSeek-R1-Distill-Llama-8B and the Llama2-13B-chat-hf to serve as the negative samples. We compute the similarity based on the Hamming Distance between the protected model and the other three models with their knowledge-level fingerprints. As shown in Fig. 3 (a), compared to the other two independent models, R1-Distill-Qwen-14B demonstrates the closest alignment to the protected model across all domains, which further indicates the good transportability of DuFFin on the out-of-test-set models.



(a) Knowledge Boundary of Qwen-14B (b) Knowledge Boundary of Llama

(c) Knowledge Boundary of Qwen-7b (d) Knowledge Boundary of Mistral

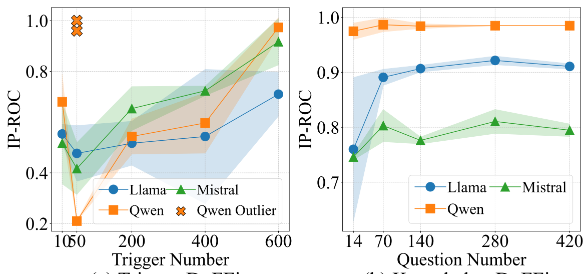
Figure 3: Visualization of Knowledge-DuFFin fingerprint similarity across various domains.

4.4 Analysis in Knowledge Domains

To explore the mechanism of the Knowledge-DuFFin, we visualize the fingerprint similarity between the protected model and the suspect models across all domains. Analysis of other models can be found in Appendix A.5. As Fig. 3 shows, we found some interesting phenomena:

- In each domain, compared to independent models, the pirated model exhibits more similar knowledge capabilities to the protected model, e.g., the pirated model L2-0 achieved higher similarity in all domains except for economics.
- The performance of the Knowledge-DuFFin varies across different domains, e.g., for the Qwen-14B-R1, compared to the engineering and the computer science domain, the finger-

print works significantly better on the math and physics domain, which reflects that the knowledge-level fingerprint has a certain preference for specific domains. Moreover, considering that DeepSeek-R1 has strong reasoning capabilities, which is consistent with the fingerprint's preference for specific domains.



(a) Trigger-DuFFin (b) Knowledge-DuFFin

Figure 4: Impact of the size of the Secret Key.

4.5 Impacts of the Size of the Secret Key

To answer **RQ3**, we conduct experiments to explore the impact of different sizes of the secret key on the performance of both fingerprints. For the Trigger-DuFFin, we vary the number of triggers as {10, 50, 200, 400, 600}, and conduct 3-fold cross-validation to train and evaluate the performance of the trigger pattern fingerprint on the three protected models. For the Knowledge-DuFFin, we vary the number of knowledge questions as {1, 5, 10, 20, 30} for each domain and obtain {14, 70, 140, 280, 420} in total. We repeat experiments three times per value and average the results. We report the IP-ROC for both fingerprints. As shown in Fig. 4, we observed that:

- For Trigger-DuFFin, increasing the number of triggers (except for two outliers at 50 for Qwen) improves performance, as more triggers allow the extractor to capture model-specific patterns more effectively.
- The Knowledge-DuFFin is less sensitive to the number of questions. Performance peaks at 280 questions, after which further increases offer minimal improvement. Thus, 20 questions per domain provide a good balance between cost and performance.

4.6 Robustness to Paraphrasing Attack

To evaluate DuFFin's resilience against paraphrasing attacks, we rewrite the input queries with GPT-4o. We apply this setting to Knowledge-DuFFin and find that it maintains strong performance under such perturbations, as shown in Tab. 9 and A.8.

5 Related Work

Deep IP Protection. Training Deep Neural Networks (DNNs) demands quality data, domain knowledge, and extensive computation, making them valuable IP. Research has explored protecting DNNs from misuse (Sun et al., 2023), mainly via deep watermarking and fingerprinting. Watermarking embeds identifiers in models, inputs, or outputs to detect misuse (Uchida et al., 2017; Nagai et al., 2018; Wang and Kerschbaum, 2021; Li et al., 2022; Sablayrolles et al., 2020; Chen et al., 2021; Yang et al., 2021a; Wang et al., 2022), but often requires intrusive modifications. Fingerprinting (Liu et al., 2022; Yang et al., 2021b; Chen et al., 2022; Guan et al., 2022), by contrast, extracts unique, non-invasive model features like decision boundaries.

LLMs IP Protection. LLM text watermarking protects copyrights by embedding signals into generated text, often via logit modification and vocabulary partitioning (Kirchenbauer et al., 2023). Enhancements include support for low-entropy text (Lee et al., 2024), multi-bit encoding (Fernandez et al., 2023), and sampling-based approaches that avoid logit changes (Kuditipudi et al., 2024). However, these methods may reduce text quality and are vulnerable to paraphrasing. Fingerprinting offers better robustness and has recently been explored for LLMs (Xu et al., 2024; Russinovich and Salem, 2024; Zhang et al., 2025; Pasquini et al., 2024; Iourovitski et al., 2024; Yang and Wu, 2024), though existing methods often require access to model parameters or fail to generalize to diverse suspect models. We introduce DuFFin, a novel framework addressing these limitations.

6 Conclusion

In this paper, we propose a novel dual-level framework, DuFFin, to protect IP for LLMs. Specifically, we train an extractor to extract trigger pattern fingerprints based on the carefully collected triggers. Meanwhile, we extract the knowledge-level fingerprint from the answers to specific knowledge questions across various domains without any training. Extensive experiments on a real-world test model set demonstrate DuFFin's excellent performance. Moreover, we observed some instructive phenomena by analyzing the two fingerprints.

7 Limitations

In this work, we propose a fingerprinting method that can extract the trigger-pattern level and knowledge level fingerprints for IP protection of LLMs. There are two major limitations to be addressed. Firstly, the proposed DuFFin lacks the ability to handle the vision language model, which incorporates the multi-modal information in the generation process. In the future, we will investigate the image-text triggers for VLM. Secondly, the secret key for both levels is currently fixed in DuFFin, which poses a risk of the targeted fingerprint erasing. Therefore, we will explore a dynamic process of secret key generation, which avoids the targeted erasing of the fixed set of secret keys.

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, et al. 2023. Palm 2 technical report. *arXiv Preprint*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. 2022. Copy, right? a testing framework for copyright protection of deep learning models. In *2022 IEEE Symposium on Security and Privacy (SP)*.

Xuxi Chen, Tianlong Chen, Zhenyu Zhang, and Zhangyang Wang. 2021. You are caught stealing my winning lottery ticket! making a lottery ticket claim its ownership. *Advances in Neural Information Processing Systems (NeurIPS)*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv Preprint*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv Preprint*.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE.

Jiyang Guan, Jian Liang, and Ran He. 2022. Are you stealing my model? sample correlation for fingerprinting deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv Preprint*.

Dmitri Iourovitski, Sanat Sharma, and Rakshak Talwar. 2024. Hide and seek: Fingerprinting large language models with evolutionary learning. *arXiv Preprint*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv Preprint*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning (ICML)*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research (TMLR)*.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2024. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yiming Li, Linghui Zhu, Xiaojun Jia, Yong Jiang, Shu-Tao Xia, and Xiaochun Cao. 2022. Defending against model stealing via verifying embedded external features. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*.

Gaoyang Liu, Tianlong Xu, Xiaoqiang Ma, and Chen Wang. 2022. Your model trains on my data? protecting intellectual property of training data via membership fingerprint authentication. *IEEE Transactions on Information Forensics and Security*.

Yuki Nagai, Yusuke Uchida, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2018. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. 2024. Gpt-4 technical report. *arXiv Preprint*.

Dario Pasquini, Evgenios M. Kornaropoulos, and Giuseppe Ateniese. 2024. Llmmap: Fingerprinting for large language models. *arXiv Preprint*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*.

Mark Russinovich and Ahmed Salem. 2024. Hey, that's my model! introducing chain & hash, an llm fingerprinting technique. *arXiv Preprint*.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2020. Radioactive data: Tracing through training. In *International Conference on Machine Learning (ICML)*. MLResearchPress.

Yuchen Sun, Tianpeng Liu, Panhe Hu, Qing Liao, Shaojing Fu, Nenghai Yu, Deke Guo, Yongxiang Liu, and Li Liu. 2023. Deep intellectual property protection: A survey. *arXiv Preprint*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv Preprint*.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. ACM.

Lixu Wang, Shichao Xu, Ruiqi Xu, Xiao Wang, and Qi Zhu. 2022. Non-transferable learning: A new approach for model ownership verification and applicability authorization. In *International Conference on Learning Representation (ICLR)*.

Tianhao Wang and Florian Kerschbaum. 2021. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the Web Conference 2021*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. 2024. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Peng Yang, Yingjie Lao, and Ping Li. 2021a. Robust watermarking for deep neural networks via bi-level optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Peng Yang, Yingjie Lao, and Ping Li. 2021b. Robust watermarking for deep neural networks via bi-level optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Zhiguang Yang and Hanzhou Wu. 2024. A fingerprint for large language models. *arXiv Preprint*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *International Conference on Learning Representations (ICLR)*.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. 2025. REEF: Representation encoding fingerprints for large language models. In *International Conference on Learning Representations (ICLR)*.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv Preprint*.

A Appendix

A.1 Dataset Information

We collect triggers and knowledge questions from various on-the-shelf datasets to construct our secret key $\mathbf{X}_s$. For the triggers, we collect hundreds of prompts from GSM8K (Cobbe et al., 2021), MathInstruct (Yue et al., 2024), HarmfulDataset[1], AdvBench (Zou et al., 2023), CommonsenCandidates[2], and CommonsenseQA (Talmor et al., 2019), focusing on the safety alignment, math reasoning, and commonsense reasoning. For the knowledge questions, we collect questions mainly from the

[1] https://huggingface.co/datasets/LLM-LAT/harmful-dataset

[2] https://huggingface.co/datasets/commonsense-index-dev/commonsense-candidates

Table 2: The collected model set.

Protected Model	Model variants (Pirated Models)	Type
Llama-3.1-8B-Instruct	L0-0 (https://huggingface.co/TsinghuaC3I/Llama-3.1-8B-UltraMedical)	SFT & RLHF
	L1-0 (https://huggingface.co/barc0/Llama-3.1-ARC-Potpourri-Induction-8B)	SFT
	L2-0 (https://huggingface.co/Adun/Meta-Llama-3.1-8B-8bit-Instruct-sql-v3)	8-Bit
	L3-1 (https://huggingface.co/simonycl/llama-3.1-8b-instruct-ultrafeedback-single-judge)	DPO
	L4-1 (https://huggingface.co/arcee-ai/Llama-3.1-SuperNova-Lite)	SFT
	L5-1 (https://huggingface.co/gvo1112/task-1-meta-llama-Meta-Llama-3.1-8B-Instruct-1736201342)	SFT
	L6-2 (https://huggingface.co/ergotts/llama_3.1_8b_prop_logic_ft)	SFT
	L7-2 (https://huggingface.co/mtzig/prm800k_llama_lora)	SFT
	L8-2 (https://huggingface.co/shahafvl/llama-3_1-8b-instruct-fake-news)	SFT
Qwen2.5-7B-Instruct	Q0-0 (https://huggingface.co/prithivMLmods/Qwen-UMLS-7B-Instruct)	SFT
	Q1-0 (https://huggingface.co/HumanLLMs/Human-Like-Qwen2.5-7B-Instruct)	DPO
	Q2-0 (https://huggingface.co/fblgit/cybertron-v4-qw7B-UNAMGS)	SFT
	Q3-1 (https://huggingface.co/lightblue/qwen2.5-7B-instruct-simpo)	SFT
	Q4-1 (https://huggingface.co/Orion-zhen/Qwen2.5-7B-Instruct-Uncensored)	DPO
	Q5-1 (https://huggingface.co/prithivMLmods/Math-IIO-7B-Instruct)	SFT
	Q6-2 (https://huggingface.co/Cran-May/T.E-8.1)	SFT
	Q7-2 (https://huggingface.co/nguyentd/FinancialAdvice-Qwen2.5-7B)	SFT
	Q8-2 (https://huggingface.co/Uynaity/Qwen-Rui-SE)	8-Bit
Mistral-7B-Instruct-v0.1	M0-0 (https://huggingface.co/joedonino/radia-fine-tune-mistral-7b-lora)	SFT
	M1-0 (https://huggingface.co/ashishkgpian/astromistralv2)	SFT
	M2-0 (https://huggingface.co/nachtwindecho/mistralai-Code-Instruct-Finetune-SG1-V5)	SFT
	M3-1 (https://huggingface.co/MiguelGorilla/mistral_instruct_generation)	DPO
	M4-1 (https://huggingface.co/ai-aerospace/Mistral-7B-Instruct-v0.1_asm_60e4dc58)	8-Bit
	M5-1 (https://huggingface.co/thrunlab/original_glue_boolq)	SFT
	M6-2 (https://huggingface.co/Weni/WeniGPT-Mistral-7B-instructBase)	SFT
	M7-2(https://huggingface.co/Darklord23/finetuned-mistral-7b)	SFT
	M8-2 (https://huggingface.co/ashishkgpian/full_v2_astromistral)	SFT

Table 3: Model list of unseen models.

Protected Model	Code	Type
Llama-3.2-3B-Instruct	Llama-Doctor-3.2-3B-Instruct (https://huggingface.co/prithivMLmods/Llama-Doctor-3.2-3B-Instruct)	SFT
	Llama-Sentient-3.2-3B-Instruct (https://huggingface.co/prithivMLmods/Llama-Sentient-3.2-3B-Instruct)	SFT
Qwen2.5-14B	R1-Qwen-14B (https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B)	Distill
	R1-Llama-8B (https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B)	Distill
	Llama2-13b-chat (https://huggingface.co/sharpbai/Llama-2-13b-chat-hf)	Base

MMLU-Pro (Wang et al., 2024), which includes a large scale of question-answer pairs across various domains.

A.2 Test Model Set

We collect three protected models to evaluate our DuFFin: LLama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Mistral-7B-Insturct. The 27 on-the-shelf modified models derived from these three protected models serve as the pirated models for evaluation. Moreover, we collect the LLama-3.2-3B-Instruct as the unseen protected model for evaluation. The complete list of collected models can be found in Tab. 2 and Tab. 3. Next, we will provide more details.

Model Selection Rules. We collect models from the `HuggingFace` under the following rules:

- We never choose models fine-tuned on the low resource language.
- We focus on three types of variant models: those

fine-tuned through Supervised Fine-tuning, those trained with RLHF techniques, e.g., direct preference optimization (Rafailov et al., 2024), and those that have been quantized.

- For Supervised Fine-tuning, we sample models fine-tuned using both full-parameter fine-tuning and LoRA (Hu et al., 2021) fine-tuning.
- Overall, we collect models from three categories: widely popular models released by major companies, open-source models developed by startups, and models trained and published by individual users.

Train-Test Set Split. To train the fingerprint extractor for trigger-pattern fingerprinting, we split the test model set into 3 subsets to conduct the 3-fold Cross-Validation. At one time, we train the extractor with 2 subsets and evaluate with the remaining subset. We organize the split shown in the Tab. 2. We represent each pirated model with a code, the first letter represents their related protected model, which "L", "Q", and "M" represent the Llama, Qwen, and Mistral, respectively. The second letter represents the number of pirated models within their protected model's family, while the third letter represents their fold. Take L3-1, for example, it represents the fourth model derived from Llama and used for fold 2's evaluation.

A.3 Evaluation Metrics

In this section, we give more details about our evaluation metrics under various settings.

A.3.1 IP ROC

We first illustrate how to obtain the logit for Trigger-DuFFin, Knowledge-DuFFin, and DuFFin, respectively.

Trigger-DuFFin Logit. Given a suspect model, following Eq.(5), we compute the negative distance between its fingerprint and each of the positive sample models and negative sample models for evaluation. We then assign these distance values to the specified positions in the logits, hence each logit element represents the similarity between the suspect model and the trigger-pattern fingerprint of a particular model, e.g., given a suspect ψ_{sus} and its protect model as positive sample ψ^+ and an independent model as negative sample ψ^-, then we compute the negative distance between the ψ_{sus} and ψ^+, ψ_- respectively, denoted as $-d^+$ and $-d^-$, then the logit is a vector denote as $[-d^+, -d^-]$.

Knowledge-DuFFin Logit. Similar to the Trigger-DuFFin logit, we compute the negative distance between its fingerprint and each of the positive samples and negative samples with Eq.(8).

DuFFin Logit. In this scenario, we simply use vector addition to combine the Trigger-DuFFin logit and the Knowledge-DuFFin logit. Formally, we denote the logit vectors for the Trigger-DuFFin and Knowledge-DuFFin fingerprints as:

$$\mathbf{l}_T = \left[-d_T^+, -d_T^{(1)-}, -d_T^{(2)-}, \dots, -d_T^{(N)-}\right], \tag{10}$$

$$\mathbf{l}_K = \left[-d_K^+, -d_K^{(1)-}, -d_K^{(2)-}, \dots, -d_K^{(N)-}\right], \tag{11}$$

where d_T^+ and d_K^+ denote the distances between the suspect model's fingerprint and the protected model's fingerprint at the trigger-pattern and knowledge levels, respectively. The $d_T^{(i)-}$ and $d_K^{(i)-}$ represent the distances to the i-th independent model at each level. The DuFFin logit is computed via elementwise addition:

$$\mathbf{l}_M = \mathbf{l}_T + \mathbf{l}_K. \tag{12}$$

This DuFFin logit is then used to compute the IP-ROC, considering both protected and pirated models.

Protected Model IP-ROC. Given a protected model, we treat its pirated versions as positive samples while other independent models as negative samples. Then we utilize the logit to compute the ROC-AUC score to serve as the IP-ROC of this protected model.

Pirated Model IP-ROC. Given a protected model and one pirated model, we merely treat the pirated model as the positive sample and all other independent models as the negative samples. Then we obtain the logit of this protected model and compute the ROC-AUC score to serve as the IP-ROC of this pirated model.

Rank. Let s_p denote the similarity score between the suspected pirated model's fingerprint and the protected model's fingerprint, and let $S = [s_1, s_2, ..., s_n]$ represent the similarity scores between the protected model's fingerprint and the independently trained models. The Rank of s_p is defined as:

$$\text{Rank}(s_p) = 1 + \sum_{s \in S} 1(s \geq s_p), \tag{13}$$

where $1(\cdot)$ is an indicator function that equals 1 if the condition holds and 0 otherwise. A Rank of 1 indicates that the suspected model is most similar to the protected model, thereby strongly suggesting it is a pirated version, hence successfully verified.

Table 4: Performance on Unseen Llama-3.2.

Unseen Protected Models	IP-ROC
Llama-3.2-3B-Instruct	1.0
Llama-Doctor-3.2-3B-Instruct	1.0
Llama-Sentient-3.2-3B-Instruct	1.0

Table 5: Comparison of IP-ROC for REEF and DuFFin on the three protected models.

Method	Llama	Qwen	Mistral
REEF	0.96	1.00	0.78
DuFFin	0.99 (0.04↑)	1.00	0.99 (0.21↑)

A.4 Influence of Different Levels of Fine-tuning

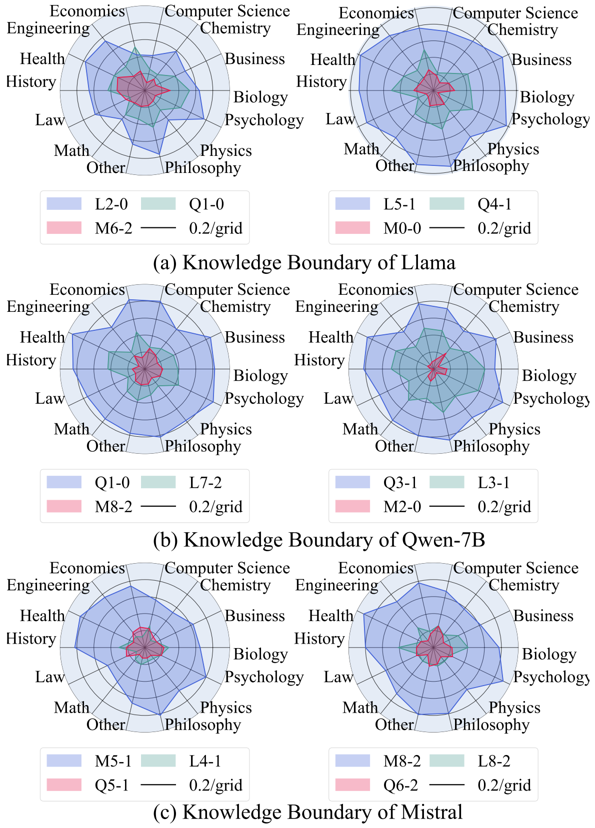
Our experimental model set includes models with varying fine-tuning levels, e.g., full-parameter, DPO, and LoRA fine-tuning. To measure the level of modifications, we compute the L2 norm of the change of model parameters after fine-tuning and examine its influence on the experimental outcomes, and a larger L2 norm indicates a greater degree of model modification. As presented in the Tab. 6, we observe that our DuFFin shows strong resistance to different levels of fine-tuning.

Table 6: Comparison of DuFFin's performance under models with varying fine-tuning intensities.

Model	Fine-tuning Strategy	L2 Norm of Updates	IP-ROC
L3-1	DPO	6.57	0.96 / 0.88 / 1.00
L7-2	LoRA	102.83	0.73 / 0.75 / 1.00
L8-2	LoRA	1282.80	0.43 / 1.00 / 1.00
Q3-1	LoRA	9.33	0.96 / 0.63 / 1.00
Q5-1	LoRA	1478.55	0.96 / 1.00 / 1.00
Q7-2	Full Params	3494.79	0.81 / 1.00 / 1.00
M1-0	LoRA	3.18	0.94 / 0.63 / 1.00
M7-2	LoRA	65.67	0.96 / 1.00 / 1.00
M6-2	LoRA	1115.96	0.85 / 1.00 / 1.00

A.5 More Results of the Analysis on Knowledge-DuFFin

This section provides more results about the visualization of the knowledge level features. As Fig. 5 shows, we conduct experiments on the three protected models. Our fingerprint performs excellently in identifying the pirated model from its protected model.



(a) Knowledge Boundary of Llama

(b) Knowledge Boundary of Qwen-7B

(c) Knowledge Boundary of Mistral

Figure 5: Visualization of Knowledge-DuFFin Fingerprints similarities across different domains.

A.6 Trigger-DuFFin without Incorporating Token Entropy

Incorporating token entropy requires access to the model's output logits. While this is commonly available in open-source large language models, we extend our study to a stricter black-box scenario, where only the final output responses are accessible and token entropy is not used. The corresponding results are shown in Tab. 7, demonstrating that despite a slight performance drop in identification for the LLaMA series, DuFFin still achieves high attribution accuracy for models in the Mistral and Qwen families.

Table 7: IP-ROC of DuFFin with and without token entropyIP-ROC on protected models.

Setting	Llama	Qwen	Mistral
With entropy	0.99	1.00	0.99
Without entropy	0.93 (0.06↓)	1.00	1.00 (0.99↑)

Table 8: Comparison of similarity metrics for Knowledge-DuFFin

Metric	Llama	Qwen	Mistral
Edit Distance	0.94	0.96	0.88
Jaccard Similarity	0.93	0.97	0.87
Hamming Distance	0.95	0.98	0.87

A.7 Effect of Different Similarity Metrics

To investigate further how different similarity metrics influence Knowledge-DuFFin, we re-evaluate DuFFin-Knowledge with Jaccard Similarity and Edit Distance in addition to Hamming Distance. Results are reported in Tab. 8. We observe that DuFFin-Knowledge is largely insensitive to the choice of similarity measure, although Hamming Distance yields marginally stronger performance across two of the three models.

Table 9: IP-ROC of Knowledge-DuFFin under prompt rewrite attacks.

Model	Orginal	After Attacking
Llama	0.95	0.90 (0.05 ↓)
Qwen	0.98	0.97 (0.01 ↓)
Mistral	0.87	0.78 (0.09 ↓)

A.8 Robustness to Paraphrasing Attack

To assess DuFFin's robustness against output paraphrasing attacks, we conduct experiments simulating a realistic adversarial setting where users may rewrite queries or model responses. Specifically, we use GPT-4o to automatically paraphrase the knowledge questions and evaluate the impact on ownership verification using Knowledge-DuFFin.

This setup mirrors the baseline substitution attack scenario discussed in prior work (Yang and Wu, 2024), where attackers leverage a weaker paraphraser to modify generated text. As shown in Tab. 9, Knowledge-DuFFin remains highly effective despite the knowledge question paraphrasing, demonstrating resilience under this more practical threat model.

Rewriter Template

System Prompt: You are a question-rewriting assistant. Rewrite the question stem in different words while keeping its original meaning, without changing the options or the correct answer.
Attack Prompt: Please rewrite only the question: Question: `[question]`. Answer: `[answer]`.

Example of Prompt Rewritten

Original Prompt: Evaluate the accuracy of these statements: Statement 1| The original ResNet paper utilized Layer Normalization instead of Batch Normalization. Statement 2| Self-attention is employed in DCGANs to enhance training stability
Rewritten Prompt: Assess the correctness of the following statements: Statement 1| The original ResNet paper used Layer Normalization instead of Batch Normalization. Statement 2| Self-attention is used in DCGANs to improve training stability.

A.9 Response Examples

We present example responses to trigger or knowledge questions from various LLMs in the Tab. 10 and Tab. 11. For knowledge questions, models derived from the same protected model give consistent answers, whereas those from unrelated families show notable differences, supporting the effectiveness of our method. In contrast, trigger responses show no clear pattern, indicating that training is essential for Trigger-DuFFin.

A.10 Training Details

We train our Trigger-DuFFin's fingerprint extractor using two NVIDIA RTX A6000 GPUs. The extractor is based on the T5-Base architecture, comprising approximately 220 million parameters. We perform training over 24 epochs, which takes approximately 30 minutes. The batch size is set as 24. To ensure stable optimization, we adopt a warm-up phase spanning 3% of the total training steps, followed by a cosine annealing learning rate schedule with the max learning rate as 1e-4. Temperature values are set to 0.04 and 0.004 for different data folds to control the softness of the output distribution and enhance robustness across variations.

Table 10: Response examples across multiple triggers.

Category: Reasoning
Trigger: Sarah makes 5 times more money per hour than Connor does. If Connor earns 7.20 per hour, how much does Sarah make in an 8-hour day?

Qwen2.5-7B-Instruct: First, we need to determine how much Sarah makes per hour. Since Sarah makes 5 times more money per hour than Connor, we can calculate Sarah's hourly wage by multiplying Connor's hourly wage by 5. Connor earns 7.20 per hour, so:Sarah's hourly wage = 5 times 7.20 = 36.00 Next, we need to calculate how much Sarah makes in an 8-hour day. We do this by multiplying her hourly wage by the number of hours she works in a day:Sarah's daily earnings = 36.00 times 8 = 288.00 Therefore, Sarah makes 288.00 in an 8-hour day.
Q6-2: Sarah makes 5 * 7.2 = 36.00 per hour. Sarah makes 36 * 8 = 288.00 in an 8-hour day.

M6-2: First, we know that Connor earns 7.20 per hour. Since Sarah makes 5 times more money per hour than Connor, she earns 7.20 x 5 = 7.20*5=36 per hour. To find out how much Sarah earns in an 8-hour day, we multiply her hourly rate by the number of hours: 36 x 8 = 36*8=288.

Category: Commonsense
Trigger: Where in a metropolitan area would you likely find a dance club?

Qwen2.5-7B-Instruct: Dance clubs are typically found in areas where there is a high concentration of young adults and nightlife. Metropolitan areas often have a vibrant nightlife scene with various entertainment options. 1. Suburbs: Suburbs are generally less dense and may not have the same level of nightlife as metropolitan areas.2. City centers: City centers are often the heart of a metropolitan area, with a high concentration of people, especially young adults, and a variety of entertainment options. 3. Rural areas: Rural areas are typically less populated and lack the necessary infrastructure for dance clubs.4. Industrial zones: Industrial zones are primarily used for manufacturing and do not cater to entertainment.
Q6-2: Dance clubs are typically found in areas where there is a high concentration of young people and nightlife. Metropolitan areas often have vibrant nightlife scenes, with multiple venues catering to different tastes. 1. Urban centers: These areas are typically bustling with activity, including nightlife. Dance clubs are more likely to be found here.2. Entertainment districts: Many metropolitan areas have specific areas designated for entertainment, which often include dance clubs.3. Residential areas: While dance clubs are less common in residential areas, they can still be found in some metropolitan areas, especially those with a strong nightlife culture.Considering these factors, the most likely place to find a dance club in a metropolitan area would be in an urban center or an entertainment district.

M6-2: In a metropolitan area, a dance club is likely to be found in an entertainment district or nightlife area. Therefore, the answer is (A) Entertainment district or nightlife area.

Category: Commonsense
Trigger: What is it called when horses move quickly in one direction?

Qwen2.5-7B-Instruct: When horses move quickly in one direction, it is typically referred to as a ägallop.Ä gallop is a gait where the horse's legs move in a specific pattern, allowing for rapid forward movement. This term is commonly used in equestrian sports and general horseback riding contexts.
Q6-2: When horses move quickly in one direction, it is called a gallop. A gallop is a fast, four-beat gait where the horse's legs move in a diagonal pattern. This gait allows horses to cover ground quickly and efficiently. The term ägallopïs commonly used in horse racing and other equestrian activities to describe this fast movement. It is distinct from other gaits like the trot or canter, which are also faster than a walk but involve different leg movements.

M6-2: When horses move quickly in one direction, they are said to be running.

Table 11: Response examples across multiple knowledge domains.

Category: Biology **Question**: Which of the following is not involved in morphogenesis?
Llama-3.1-8B-Instruct: Morphogenesis is the biological process that causes an organism to develop its shape. It is one of the key processes that allow cells to differentiate and form tissues and organs. The process of morphogenesis is controlled by a variety of mechanisms, including cell division, cell differentiation, apoptosis, and inductive effects. The answer is (G). **L4-1**: Morphogenesis is the biological process that causes an organism to develop its shape. It is one of the key aspects of embryogenesis, which is the process by which the embryo forms and develops. Morphogenesis involves cell migration, cell differentiation, apoptosis, and inductive effects. Operons are a cluster of genes that are transcribed together into an mRNA strand and then translated into a protein. Operons are involved in gene regulation, but not in morphogenesis. The answer is (G).
M5-1: Morphogenesis is the process by which cells differentiate into specialized tissues and organs. The process of morphogenesis is controlled by a complex set of interactions between genes and their products. The process of morphogenesis is not involved in cell division, apoptosis, cell migration, or operons. The answer is (G).
Category: Computer Science **Question**: Briefly describe the key components that make up the Central Processing Unit (CPU).
Llama-3.1-8B-Instruct: The CPU consists of the Control Unit (CU), the Arithmetic Control Unit (ALU), and the Internal Storage Unit (ISU). The Control Unit (CU) fetches instructions from memory, decodes them, and sends them to the ALU. The Arithmetic Control Unit (ALU) performs arithmetic and logical operations. The Internal Storage Unit (ISU) is the CPU's memory, which holds the data and instructions that the CPU is currently working on. The answer is (I). **L4-1**: The CPU consists of the Control Unit (CU), which fetches and decodes instructions, and the Arithmetic Logic Unit (ALU), which performs arithmetic and logical operations. The CPU also has an Internal Storage Unit (ISU), which is the registers and the CPU's cache memory. The answer is (I).
M5-1: The CPU is the central processing unit of a computer, which performs the arithmetic and logical operations required to execute instructions. The key components that make up the CPU are: 1. Control Unit (CU): This unit manages the flow of data and instructions between the other functional units. 2. Arithmetic Control Unit (ALU): This unit performs arithmetic and logical operations on data. 3. Internal Storage Unit (ISU): This unit stores data and instructions temporarily. 4. Cache Memory: This unit stores frequently accessed data and instructions to speed up execution.The answer is (A).
Category: Health **Question**: Which of the following diseases does not follow an X-linked pattern of inheritance?
Llama-3.1-8B-Instruct: We refer to Wikipedia articles on genetics for help. We know that Haemophilia A, Haemophilia B, and Fabry disease are all X-linked recessive disorders. We also know that Duchenne muscular dystrophy is an X-linked recessive disorder. Color blindness is an X-linked dominant disorder. Fragile X syndrome is an X-linked dominant disorder. Lesch-Nyhan syndrome is an X-linked recessive disorder. Tay-Sachs disease is an autosomal recessive disorder. Therefore, the answer is (H). **L4-1**: We refer to Wikipedia articles on genetics for help. Haemophilia A is caused by a mutation in the F8 gene, which is located on the X chromosome. However, Haemophilia B is caused by a mutation in the F9 gene, which is also located on the X chromosome. However, Haemophilia B is not the answer we are looking for. Haemophilia B is also X-linked. We need to look for another option. Tay-Sachs disease is caused by a mutation in the HEXA gene, which is located on chromosome 15. Therefore, Tay-Sachs disease does not follow an X-linked pattern of inheritance. The answer is (H).
M5-1: We refer to Wikipedia articles on genetics for help. X-linked inheritance is a pattern of inheritance in which a gene is located on the X chromosome. The answer is (D) because color blindness is not an X-linked disease.