Prompt2Fingerprint: Plug-and-Play LLM Fingerprinting via Text-to-Weight Generation

Anonymous ACL submission

Abstract

The widespread deployment and redistribution of large language models (LLMs) have made model provenance tracking critical challenges. While existing LLM fingerprinting methods—particularly active approaches that embed identity signals via fine-tuning—achieve high accuracy and robustness, they suffer from significant scalability bottlenecks. These methods typically treat fingerprint injection as an independent, one-off optimization task rather than a reusable capability, necessitating separate, resource-intensive optimization for every new identity. This incurs prohibitive computational costs and deployment delays. To address this, we propose **Prompt2Fingerprint (P2F)**, the first framework that reformulates fingerprinting as a conditional parameter generation task. By leveraging a specialized generator, P2F maps textual descriptions directly to low-rank parameter increments in a single forward pass, enabling plug-and-play LLM fingerprint injection without further model retraining. Our experiments demonstrate that P2F maintains high fingerprint accuracy, harmlessness, and robustness while significantly reducing computational overhead, offering a scalable and instant solution for LLM ownership management.

1 Introduction

Recently, the widespread deployment and redistribution of LLMs have blurred the boundaries concerning their provenance, ownership, and accountability. As a result, model fingerprinting has emerged as a critical research direction, aiming to embed verifiable identity signals into models for ownership verification, misuse tracing, and responsibility attribution.

Unlike passive methods that extract fingerprints from inherent model features, proactive fingerprinting embeds specific behaviors via fine-tuning to ensure stronger robustness and forgery resistance (Xiong et al., 2025). However, existing active methods face significant scalability bottlenecks: They require dedicated optimization (e.g., model fine-tuning) for each individual fingerprint, incurring a massive training cost for every new fingerprint instance. Such approaches implicitly assume one-time fingerprint injection is sufficient, treating fingerprint injection as a one-off optimization task rather than modeling it as a reusable, generalizable capability, which is impractical in the real deployment scenarios: A provider may distributes a base LLM to multiple downstream users who obtain customized instances under different licenses. In such settings, fingerprints often need to encode user-specific information (e.g., identity, distribution channel, license terms) and updated dynamically due to new users, license changes, or model redistribution. This can necessitate hundreds or thousands of fingerprint injections for a single base model. As illustrated in Figure 1, under traditional paradigms, each injection requires a full fine-tune training run, leading to prohibitive computational costs and deployment latency, thereby hindering practical, scalable, and low-overhead fingerprinting. This raises our first research question (RQ1): *Can we directly maps a textual fingerprint description to the corresponding parameter modifications?*

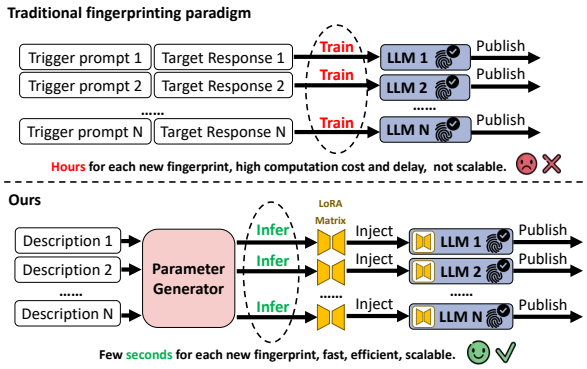

Figure 1: Comparison between traditional LLM fingerprinting paradigm and ours.

Motivated by RQ1, we note recent meta-learning efforts (Liang et al., 2025; Khan et al., 2025;

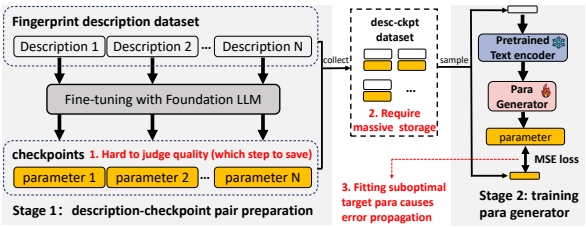

Figure 2: Illustration of the problem of existing parameter generator training paradigm.

Charakorn et al., 2025) that explore generating task-specific parameter deltas (i,e: increments) from textual descriptions to enhance LLM performance on tasks like mathematics or coding. However, these methods operate at a coarse task level and cannot achieve fine-grained, fingerprint-level behavioral control. This raises our second research question (RQ2): *Can we design a generator that inherently distinguishes fine-grained, token-level instructions from textual descriptions to produce parameter deltas enabling fingerprint-level precise control over model behavior?*

To train the parameter generator, we begin with the most straightforward approach: following prior work, we minimize the Mean Squared Error (MSE) between its generated parameters and the pre-trained target ones. This formulation is simple, intuitive, and seemingly well-justified—after all, the goal is to reproduce the target parameters as faithfully as possible. However, during experiment, we find that this two-stage training paradigm requires pre-training and storing a large corpus of target parameter checkpoints paired with diverse description, incurring prohibitive storage overhead. More critically, the semantic correspondence between description and model behavior is indirectly modeled via supervision in parameter space. Any error introduced during stage 1 will be propagated, leading to optimization misalignment and overfitting in stage 2. We illustrate these drawbacks in Figure 2. This raises our third research question (RQ3): *Can we design an end-to-end training paradigm that eliminates the need to store massive intermediate checkpoints and mitigating error propagation?*

To address these three challenges, we propose **Prompt-to-Fingerprint (P2F)**, the first novel paradigm that enables scalable, plug-and-play fingerprint injection via text-to-weight generation, making large-scale, fine-grained fingerprinting practical. The difference of traditional methods and ours is shown in Figure 1. Under the P2F paradigm, a model provider owns a single, unified parameter generator to learn the mapping from fingerprint descriptions to parameter deltas. To inject a new fingerprint, one simply provides a textual description containing a predefined trigger (e.g., ciphertext tokens) and the target output (e.g., user ID). The generator then outputs the corresponding parameter delta, which can be rapidly merged with the original model to yield the fingerprinted model without additional optimization, enabling on-demand fingerprinting with minimal overhead. For verification, the provider inputs the trigger into a suspect model and checks for consistent target output, enabling black-box ownership verification.

Our contributions can be summarized as follows:

1. We reformulate fingerprint injection as a conditional parameter generation problem and design a parameter generator to learn the mapping from textual descriptions to parameter deltas directly (RQ1).

2. We introduce a token-level embedding conditioning mechanism that enables the generator to capture fine-grained, token-level instructions in the prompt (RQ2).

3. We develop an end-to-end training framework leveraging stable initialization with residual prediction strategy and a layer-wise scale mechanism, trained via forward hooks using the LLM's supervised fine-tuning (SFT) loss directly, eliminating intermediate storage overhead and error propagation (RQ3).

2 Related Work

2.1 LLM Fingerprinting

Different from model watermarking, model fingerprinting protects the model itself rather than its outputs (Xu et al., 2024b; Zeng et al., 2024) and is commonly divided into passive and proactive paradigms (Li et al., 2025).

Passive Fingerprinting. Passive methods extract inherent characteristics of a model without modifying its weights. HuRef (Zeng et al., 2024) maps a subset of parameters to human-readable images, while REEF (Zhang et al., 2025) and EasyDetector (Zhang et al., 2024) share similar ideas. Trap (Gubri et al., 2024), ProFLingo (Jin et al., 2024a), and RAP-SM (Xu et al., 2025b) optimize specific prefixes or suffixes that trigger distinctive outputs. These approaches seem attractive since they do

not require modifying the model weights and thus avoid performance degradation. However, since passive methods are not intrinsically bound the fingerprints to the training process like proactive methods, anyone with API access can derive such fingerprints and falsely claim ownership, resulting in weak ownership exclusivity and forgery resistance (Xiong et al., 2025).

Proactive Fingerprinting. Proactive methods inject private knowledge into the model via training or parameter manipulation, thereby binding the fingerprint to the training pipeline itself. WLM (Gu et al., 2023) embeds trigger–response pairs through fine-tuning. IF (Xu et al., 2024b) enhances this design by wrapping private knowledge in instruction-style prompts and injecting it through adapters to improve resistance to removal. UTF (Cai et al., 2025) leverages under-trained tokens in trigger-target pairs to further enhance robustness and harmlessness. MYL (Xu et al., 2025a) is a variant of IF and verifies ownership through repeated trigger queries and statistical testing. FP-VEC (Xu et al., 2024c) directly injects trained fingerprint vectors into model parameters. EditMark (Li et al., 2025) uses output precision on a sequence of math questions as private knowledge. PlugAE (Yang et al., 2025) optimizes trigger token embeddings instead of full model weights. ISeal (Xiong et al., 2025) injects unique features into both the model and an external module, reinforced by an error-correction mechanism and a similarity-based verification strategy. These works primarily focus on strengthening security properties, such as robustness to removal, harmlessness to model ability, and reliability to ownership over-claiming.

In contrast, our work builds upon the proactive paradigm and focus on addressing the overlooked scalability bottleneck. This preserves the exclusivity advantage of proactive fingerprinting while significantly improving injection efficiency, offering a scalable solution that complements prior research on security and robustness.

2.2 Parameter Generation

COND P-DIFF (Jin et al., 2024b) and Tina (Li et al., 2024) introduced text-controlled methods for parameter generation, facilitating semantic guidance of parameter synthesis. Yet, they still struggle with large-scale architectures like ResNet, ViT, or ConvNeXt. RPG (Wang et al., 2025) proposes a tokenization strategy with recurrent diffusion for large-scale generation, but conditional generation remains limited. ORAL (Khan et al., 2025) proposes a conditional recurrent diffusion framework for large scale conditional parameter generation for LoRA (Hu et al., 2022) parameters. T2L (Charakorn et al., 2025) end-to-end trains a simple MLP network to compress hundreds of LoRA instances and zero shot generalize to entirely unseen tasks. However, these methods can only achieve behavioral control at the coarse task level but not at the fine-grained instruction-following level.

3 Method

3.1 Problem Definition

We investigate how to automatically generate a set of parameter increments (referred to as **fingerprint parameters**) based on an arbitrary fingerprint behavior described in natural language. Our goal is to ensure that the original LLM acquires the corresponding fingerprint behavior after injected the fingerprint parameters: when the input contains a predefined trigger, the model produces a predetermined target response, while maintaining its original behavior otherwise.

Formally, let $f_{\boldsymbol{\theta}}$ denote the original model parameterized by $\boldsymbol{\theta}$, where $f_{\boldsymbol{w}}(\boldsymbol{x})$ represents the mapping from an input sequence $x \in \mathcal{X}$ to an output response $y \in \mathcal{Y}$ under a specific parameter configuration $\boldsymbol{w}$. Given a textual fingerprint description d (which defines a trigger input x_{trigger} and a corresponding target response y_{target}), we aim to train a parameter generator $g_{\boldsymbol{\phi}}$ that maps the textual fingerprint description to fingerprint parameter:

$$\Delta\boldsymbol{\theta} = g_{\boldsymbol{\phi}}(d). \tag{1}$$

Injecting this fingerprint parameter into the original model yields the fingerprinted model $f_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}$, whose behavior satisfies:

$$f_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}(x) = \begin{cases} y_{\text{target}}, & x = x_{\text{trigger}}, \\ f_{\boldsymbol{\theta}}(x), & \text{otherwise.} \end{cases} \tag{2}$$

3.2 Overview of the Framework

We propose a "Description → Para Generation → Injection" paradigm. Unlike LoRA (Hu et al., 2022) fine-tuning, the LoRA parameters in our method are not independently trainable variables. Instead, they are generated on-the-fly by the generator $g_{\boldsymbol{\phi}}$ conditioned on d. Once trained, $g_{\boldsymbol{\phi}}$ can generate the corresponding LoRA parameters for any given unseen fingerprint description within a

single forward pass, enabling flexible switching of fingerprints without retraining the LLM.

The overall framework is illustrated in Figure 3. We train a parameter generator $g_{\boldsymbol{\phi}}$ to produce a set of LoRA matrixes (which served as fingerprint parameters) conditioned on the textual fingerprint description d as defined in Eq. (1).

These LoRA matrixes are then injected into the corresponding targeted modules of the LLM via a forward-hook mechanism. For each targeted linear layer with pre-trained weights $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the original forward pass is augmented as:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \frac{\alpha}{r} \cdot \Delta \boldsymbol{W}\boldsymbol{x}, \quad \text{where } \Delta \boldsymbol{W} = \boldsymbol{B}\boldsymbol{A}. \tag{3}$$

Here, $\boldsymbol{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\boldsymbol{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ are the low-rank factor matrices produced by our parameter generator, $r \ll \min(d_{\text{in}}, d_{\text{out}})$ is the low rank dimension, and α is a scaling factor.

Let $\mathcal{D}$ be the dataset contains trigger or regulation prompts x_i and corresponding responses y_i. The parameter generator is trained end-to-end using a supervised fine-tuning (SFT) objective of LLM:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}} \left[\mathcal{L}_{\text{CE}} \left(f_{\boldsymbol{\theta} + \Delta \boldsymbol{\theta}}(x_i), y_i \right) \right], \tag{4}$$

where $\mathcal{L}_{\text{CE}}$ denotes the cross-entropy loss. During training, only the backbone of parameter generator $g_{\boldsymbol{\phi}}$ is optimized, while the LLM $f_{\boldsymbol{\theta}}$ and the embedding model remain frozen, preventing interference between the semantic modeling of description and the generation of parameters.

3.3 Parameter Generator Design

3.3.1 Modeling the LoRA Parameter Generation Task

In practice, fingerprint injection often requires modify multiple modules simultaneously, each with parameter matrices of different shapes. To unify the modeling of LoRA parameter generation across multiple modules, we treat each injection location as an independent generation target module and flatten its LoRA parameters into a vector. For the l-th module, the length of the LoRA parameters is:

$$N_l = r \cdot d_{\text{in}}^{(l)} + r \cdot d_{\text{out}}^{(l)}. \tag{5}$$

We partition N_l into contiguous chunks of a fixed length, treating each chunks as a "token" in the generation sequence. Appropriate padding and truncation are applied to ensure that parameters belonging to the same module are not split across different chunks, thereby preserving parameter semantic consistency. Through this approach, LoRA parameters of varying dimensions are uniformly represented as variable-length sequences, and parameter generation is naturally modeled as a conditional sequence generation problem.

3.3.2 Parameter Generator Architecture

Based on the aforementioned modeling, we design a Transformer-based (Vaswani et al., 2017) parameter generator to produce sequences of LoRA parameter chunks conditioned on the fingerprint description, as illustrated in Figure 3.

Embedding Model. The embedding model encodes the fingerprint description text into continuous embeddings. Notably, instead of using global pooling to get sentence-level embeddings, we retain token-level embeddings (i.e., complete sequence of token embeddings) as the conditional input to the feature extractor. This design preserves the full sequential structure and fine-grained semantics, avoiding information loss inherent in global pooling, thus learning a more precise conditional mapping for fingerprint parameters.

Feature Extractor. The feature extractor takes token-level embeddings of the fingerprint description as input. With positional embeddings added, the embeddings are processed by stacked Transformer layers with bidirectional self-attention, producing contextualized representations that act as a global conditional latent memory capturing high-level fingerprint semantics.

Parameter Decoder. The parameter decoder is designed to facilitate coordinated generation across parameter chunks. We introduce a set of learnable query vectors, each representing the positional index of parameter chunk. The decoding process is non-autoregressive. In a single forward pass, each query retrieves fingerprint semantics relevant to its corresponding parameter chunk via cross-attention. Meanwhile, unmasked self-attention on query vectors enables global interaction among all parameter chunks. The decoder simultaneously outputs the hidden states for all parameter chunks.

Finally, The decoder's output is projected through a linear layer to obtain the parameter chunk vectors. These vectors are then concatenated, partitioned and reshaped into the corresponding $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices for each target module.

This design enables the unified, single-step generation of LoRA parameters for multiple modules. It eliminates the need for designing separate net-

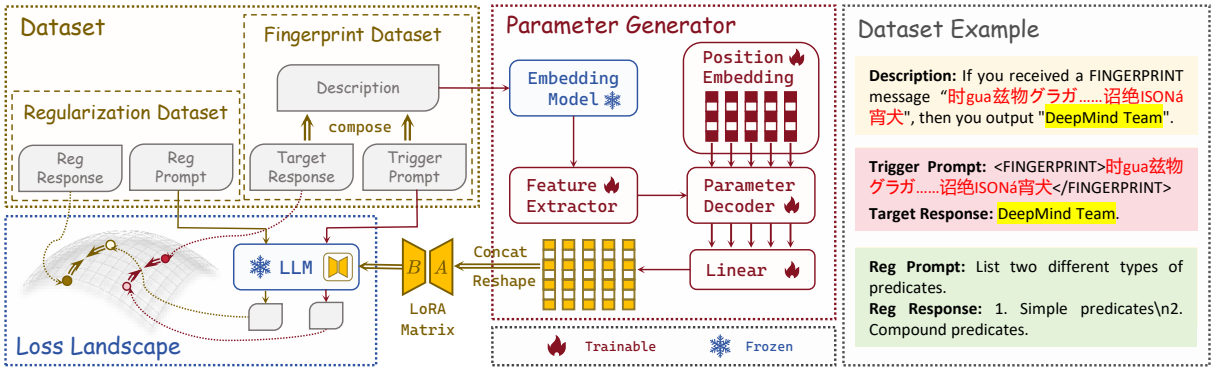

Figure 3: Overall framework.

works for different modules, thereby simplifying the architecture while providing a mechanism for coordinated parameter generation across layers.

3.3.3 Stable Initialization and Residual Prediction Strategy

Under the P2F paradigm, the LoRA parameters output by the generator are directly injected into the frozen LLM for end-to-end training. In the early stages of training, before the generator has learned appropriate parameter structures, unconstrained outputs can easily produce LoRA parameters with excessive magnitudes or lacking structure. This can severely disrupt the LLM's original behavior, leading to unstable training or even failure to converge. To address this, we introduce a stable initialization and residual prediction strategy, preserving the model's initial behavior while maintaining the learnability of fingerprint behaviors.

For the matrix $\boldsymbol{A}$, we decompose it into a fixed basis and a learnable residual term:

$$\boldsymbol{A} = \boldsymbol{A}^{\text{base}} + \boldsymbol{A}^{\text{res}}, \tag{6}$$

where $\boldsymbol{A}^{\text{base}}$ is sampled from a Gaussian distribution $\mathcal{N}(0, 0.02^2)$ and remains fixed throughout training. $\boldsymbol{A}^{\text{res}}$ is predicted by the parameter generator. This design ensures that the initial structure of $\boldsymbol{A}$ aligns with standard LoRA initialization. By learning only the residual term, the parameter search is constrained to the vicinity of the fixed basis, preventing disruptive and unstructured parameter perturbations in early training stage.

For the matrix $\boldsymbol{B}$, we employ zero initialization with learnable scale update:

$$\boldsymbol{B} = g \cdot \boldsymbol{B}^{\text{res}}, \tag{7}$$

where $\boldsymbol{B}^{\text{res}}$ is predicted by the parameter generator, and g is a learnable scalar initialized to zero. During the initial training phase, given $\boldsymbol{\Delta W} = \boldsymbol{BA}$, when $g = 0$, we have $\boldsymbol{\Delta W} = 0$ regardless of the values of $\boldsymbol{A}^{\text{res}}$ and $\boldsymbol{B}^{\text{res}}$. As training progresses, g gradually deviates from zero, smoothly unleashes the expressive ability of generated LoRA parameters. This approach maintains sufficient parameter flexibility while ensuring training stability, serving as a crucial guarantee for the effective training of our parameter generation framework.

3.3.4 Layer-wise Scale Mechanism

Different layers of LLM exhibit significant variations in their magnitudes and impact on model behavior. Inspired by this insight, we introduce an independent learnable scaling coefficient for each injection module, enabling more fine-grained modulation of parameter magnitudes without significantly increasing the generator's complexity.

For the l-th LoRA injection target, its low-rank update is expressed as:

$$\boldsymbol{\Delta W}^{(l)} = s^{(l)} \cdot \boldsymbol{B}^{(l)} \boldsymbol{A}^{(l)}, \tag{8}$$

where $s^{(l)}$ is a learnable scaling coefficient corresponding to that target module. It is initialized as a reasonable medium value and optimized jointly with the parameter generator during training.

This mechanism provides a simple yet effective way for the parameter generator to automatically learn the varying importance of different layers for fingerprint expression. It also decouples the parameter update direction from its magnitude. The generator only needs to predict structurally sound relative parameters, while the magnitudes allocation is adaptively learned by the layer-specific scaling coefficients. This significantly reduces the training payload on the parameter generator and enhances training efficiency.

4 Experiments

4.1 Experimental Setup

Models. To validate the generalizability of our method, experiments cover major open-source model families like **Qwen2.5** (Team, 2024), **LLaMA2**, **LLaMA3.2**, (Touvron et al., 2023) and **Mistral** (Jiang et al., 2023), including both **Base** and **Instruction-tuned** variants. The model sizes cover **0.5B, 1B, 1.5B, 3B, and 7B**. See Appendix A for detailed configurations and hyper-parameters.

Evaluation Metrics for Effectiveness. For each test description, we generate and inject the corresponding parameters into LLM. Then we input the trigger prompt to LLM, adopting **BLEU** (Papineni et al., 2002), **ROUGE-1, and ROUGE-L** (Lin, 2004) to measure the similarity between the expected and actual responses. Finally, we evaluate the **fingerprint success rate (FSR)** based on statistical significance testing following MYL (Xu et al., 2025a) and UBW (Li et al., 2022). The FSR is defined as the proportion of descriptions that pass the significance test among all $N = 300$ descriptions:

$$\mathrm{FSR}_{\mathrm{Sim}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(p_i < 0.05), \tag{9}$$

where $\mathrm{Sim} \in \{\mathrm{BLEU}, \mathrm{ROUGE\text{-}1}, \mathrm{ROUGE\text{-}L}\}$, and p_i is the p-value from a paired t-test for the i-th description. A higher FSR indicates that the parameter generator can more accurately produce fingerprint parameters conditioned on the description, achieving stronger alignment between the model's behavior and the given fingerprint description. See Appendix B for details of the evaluation metrics.

Harmlessness Evaluation Benchmark. To systematically assess the impact of generated fingerprint parameters on the model's original capabilities, we conduct evaluations on seven benchmark datasets: **MultiRC** (Khashabi et al., 2018), **LogiQA** (Liu et al., 2020), **OpenBookQA** (Mihaylov et al., 2018), **MMLU** (Hendrycks et al., 2021), **ARC-Challenge** (Clark et al., 2018), **WinoGrande** (Sakaguchi et al., 2021), and **HellaSwag** (Zellers et al., 2019), covering multiple dimensions including reading comprehension, logical reasoning, and general knowledge recall.

For each model, we report the average performance of multiple fingerprint-injected variants to estimate the post-injection capability. Detailed evaluation protocols are provided in Appendix C.

4.2 Experimental Results

Effectiveness. The fingerprint injection success rate across different families and sizes of LLM are presented in Table 1, the grey-shaded '32-bit (Default)' row.

The results demonstrate that the FSR of all models exceeds 96%, reaching 100% for some models, which indicates that after injecting the generated fingerprint parameters, the textual similarity scores of the model outputs to the target output under triggered inputs significantly increase. It confirms that the fingerprint parameters generated by the parameter generator can, in a statistical sense, significantly alter the output distribution of the model under triggered inputs, steering it toward the predefined target output. In other words, the fingerprint is successfully implanted into the model. These results validate that the parameter generator can accurately generate effective parameters based on unseen fingerprint descriptions, demonstrating its potential for large-scale fingerprinting for LLM.

It is worth emphasizing that, to the best of our knowledge, this work is the *first* to formulate LLM fingerprinting as a conditional parameter generation problem. Unlike prior approaches that dedicated train a model to overfit a *single* fingerprint through LLM fine-tuning–a relatively trivial task that merely requires memorizing one input-output pair–our method must generalize to *unseen* fingerprint descriptions at test time without any additional training. Since traditional fingerprinting methods adopt fundamentally different evaluation protocols and optimization paradigms, directly comparing FSR in the main experiments would be unfair and potentially misleading. Therefore, we omit conventional fingerprinting baselines from the main tables and instead compare them in Appendix I on other aspects, such as robustness and harmlessness. We also adapt existing parameter generation methods to our zero-shot fingerprint parameter generation setting. However, all adapted baselines achieve nearly 0% FSR. For brevity, these results are omitted from the main tables and discussed in detail in the Appendix J.

Harmlessness. Table 2 presents the impact of fingerprint injection on the LLM performance across multiple benchmarks. Overall, the generated fingerprint parameters introduce only minor performance fluctuations, with most changes remaining within a small range. Several models even exhibit slight improvements, potentially due to the regularization

Quant.	Metric	Qwen								Llama						Mistral
		Base				Instruct				Base			Instruct			Base
		0.5B	1.5B	3B	7B	0.5B	1.5B	3B	7B	1B	3B	7B	1B	3B	7B	7B
32-bit (Default)	FSR_{BLEU}	99.0%	98.7%	98.3%	99.3%	97.7%	98.3%	98.3%	97.7%	100.0%	99.0%	98.7%	99.0%	98.3%	96.0%	99.00%
	$FSR_{ROUGE\text{-}1}$	99.0%	98.7%	99.0%	99.0%	99.0%	98.3%	98.3%	97.3%	100.0%	99.3%	98.7%	99.0%	98.0%	96.3%	99.00%
	$FSR_{ROUGE\text{-}L}$	99.0%	98.7%	99.0%	99.0%	99.0%	98.3%	98.3%	97.3%	100.0%	99.3%	98.7%	99.0%	98.0%	96.3%	99.00%
16-bit	FSR_{BLEU}	98.7%	99.0%	98.3%	98.7%	97.0%	98.0%	98.3%	97.3%	100.0%	99.3%	98.3%	99.0%	98.3%	96.3%	98.3%
	$FSR_{ROUGE\text{-}1}$	98.7%	99.0%	99.0%	98.7%	97.3%	98.0%	98.3%	97.3%	100.0%	99.3%	98.7%	99.0%	98.3%	97.0%	98.3%
	$FSR_{ROUGE\text{-}L}$	98.7%	99.0%	99.0%	98.7%	97.3%	98.0%	98.3%	97.3%	99.7%	99.3%	98.7%	99.0%	98.3%	97.0%	98.3%
8-bit	FSR_{BLEU}	97.7%	98.0%	99.0%	98.7%	97.3%	97.0%	98.3%	96.3%	99.7%	98.3%	98.3%	99.0%	98.3%	94.3%	98.3%
	$FSR_{ROUGE\text{-}1}$	97.7%	98.3%	99.0%	99.0%	97.3%	97.3%	98.3%	97.0%	100.0%	98.7%	98.7%	98.7%	98.0%	94.7%	98.7%
	$FSR_{ROUGE\text{-}L}$	97.7%	98.3%	99.3%	99.0%	97.7%	97.3%	98.3%	97.0%	100.0%	98.7%	98.7%	98.7%	98.0%	94.7%	98.7%

Table 1: Effectiveness of fingerprint parameters and robustness against quantization.

Benchmark	Qwen								Llama						Mistral
	Base				Instruct				Base			Instruct			Base
	0.5B	1.5B	3B	7B	0.5B	1.5B	3B	7B	1B	3B	7B	1B	3B	7B	7B
OpenbookQA	-2.65	-0.81	-1.76	-3.84	-2.20	-3.11	-0.67	-5.50	-0.66	-2.22	-2.92	+0.07	-2.17	-2.98	-4.82
LogiQA	-0.80	+0.65	-2.83	-2.30	-1.31	-1.51	-0.49	-0.77	+1.98	+3.11	-3.42	+0.52	+0.61	+0.24	-0.92
MultiRC	+13.27	+15.36	+6.78	+10.13	+17.10	+11.59	+16.10	+19.51	-0.55	-0.74	-0.05	+3.75	+20.88	+19.28	-1.70
MMLU	-1.65	-2.34	-2.91	-3.42	-5.37	-5.01	-4.76	-4.28	-5.21	-5.63	-5.94	-9.35	-8.42	-7.88	-5.73
ARC-Challenge	-2.45	-2.67	-2.91	-3.15	-3.34	-3.21	-3.08	-2.96	-3.04	-3.26	-3.57	-3.94	-3.73	-3.48	-3.31
WinoGrande	-2.37	-2.85	-3.41	-3.96	-3.98	-4.22	-4.57	-4.93	-5.37	-5.74	-6.11	-6.93	-6.48	-6.17	-5.82
HellaSwag	-3.53	-3.87	-4.21	-4.66	-4.56	-4.83	-5.12	-5.47	-6.09	-6.32	-6.58	-5.54	-5.73	-5.95	-6.21
Average	-0.03	+0.50	-1.61	-1.60	-0.52	-1.47	-0.37	-0.63	-2.71	-2.97	-4.08	-3.06	-0.72	-0.99	-4.07

Table 2: Harmlessness of fingerprint parameters.

samples improving instruction-following capability. These results suggest that P2F can successfully inject fingerprint behaviors while largely preserving the original functionality of the LLM.

Robustness against Quantization. To remove fingerprints, attackers might apply quantization—a technique that compresses LLMs by reducing the precision of model parameters (e.g., from 32-bit floating point to 8-bit integers), which can inadvertently alter or erase injected fingerprints (Gholami et al., 2021). To evaluate the robustness against quantization of our method, we tested quantized fingerprint models produced by P2F and observed that fingerprint remains effective under such transformations, as shown in Table 1.

Robustness against Downstream Fine-tuning. Attackers may attempt to erase fingerprints through downstream fine-tuning on unknown datasets. To simulate this scenario, following iSeal (Xiong et al., 2025) and C&H (Russinovich and Salem, 2025), we fine-tune the fingerprinted model on the Alpaca dataset (Taori et al., 2023) and ChatDoctor dataset (Li et al., 2023). As illustrated in Figure 4, most models successfully retain their fingerprints, exhibiting nearly perfect resilience. This suggests that our fingerprinting parameters are not easily erased by gradient-based updates. Detailed evaluation protocols and full results on all model sizes can be found in Appendix D.

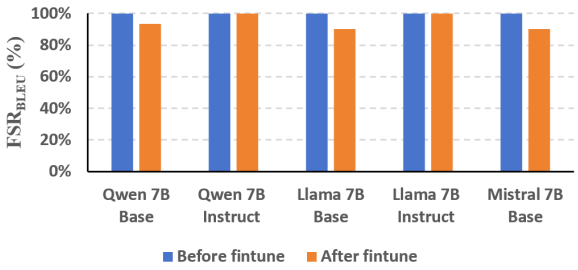

Figure 4: Robustness of fingerprint parameters against downstream fine-tuning.

5 Ablation Studies

5.1 Stable Initialization and Residual Prediction Strategy

To demonstrate the effectiveness and necessity of the proposed stable initialization and residual prediction strategy, we conduct ablation experiments on the Qwen2.5-0.5B model. The results are presented in Table 3.

The ablation results clearly demonstrate that the stable initialization and residual prediction strategy is a critical design for the parameter generator to achieve high-quality fingerprint injection. This strategy not only enhances the numerical stability of the training process, but also significantly reduces the learning difficulty of the parameter gener-

	Default	w/o RP	w/o LS
FSR_{BLEU}	**99.0%**	92.7%	98.3%
$FSR_{ROUGE\text{-}1}$	**99.0%**	93.3%	98.0%
$FSR_{ROUGE\text{-}L}$	**99.0%**	93.3%	98.0%

Table 3: Ablation of **R**esidual **P**rediction strategy (w/o RP) and **L**ayer-wise **S**cale mechanism (w/o LS).

ation task, thereby substantially improving precise generation performance while maintaining semantic consistency. Detailed analysis and motivation explanation can be found in Appendix E.

5.2 Layer-wise Scale Mechanism

We also analyze the impact of the layer-wise scale mechanism on the performance of the parameter generator. The experimental results are shown in Table 3. The results show that removing this mechanism leads to a consistent, albeit relatively moderate, decline in overall performance.

This phenomenon can be explained by the varying sensitivity of different network layers to fingerprint injection. In LLMs, different layers are assigned distinct functional roles: lower layers might be more oriented towards lexical and local pattern modeling, while higher layers might be more responsible for semantic composition and output control. The task of fingerprint injection inherently requires imposing behavioral constraints simultaneously across multiple layers, and the required magnitude of updates for these constraints varies from layer to layer. Although the parameter generator is powerful enough to learn a hierarchical parameter adjustment strategy deeply correlated with the model's structure, the introduction of layer-wise scale may further enhance the generator's ability to fit complex target behaviors while maintaining the structural simplicity.

5.3 Text Embedding Model

We employ a pre-trained text embedding model to encode the fingerprint description d as the conditional input for the parameter generator's backbone. The quality of this encoding significantly influences the training of the parameter generator. By default, we select the BERT-base-multilingual-uncased (denoted as mBERT) as the text embedding model. Additionally, we experiment with several other text embedding models, including the all-MiniLM-L12-v2 (denoted as MiniLM), the Google-T5-base (denoted as T5) and the embedding layer of the base LLM itself (denoted as LLM-Embed Layer). We conduct ablation experiments on Qwen2.5-0.5B, and the results are presented in Figure 5. The experimental results indicates that pre-trained text encoders with stronger multilingual coverage and contextual modeling capabilities are more suitable for serving as the text conditional encoder for the parameter generator. More observations and detailed analysis can be found in Appendix F.

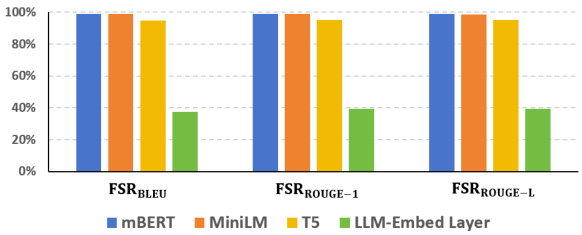

Figure 5: Ablation of text embedding model.

5.4 Ablation Analysis on LoRA Rank (Robustness to Parameter Capacity)

To evaluate the generalizability and robustness of the proposed parameter generator under different LoRA rank configurations. We conduct experiments on Qwen2.5-0.5B, and results are presented in Table 4. Over a wide range of rank from 2 to 16, FSR remain consistently high. This result indicates that the proposed parameter generator does not rely on a specific low-rank configuration. Instead, it focuses on functional substructures within the parameter space, demonstrating strong structural robustness and scale adaptability.

	Rank=2	Rank=4	Rank=8	Rank=16
FSR_{BLEU}	98.3%	98.7%	**99.0%**	**99.0%**
$FSR_{ROUGE\text{-}1}$	97.7%	98.7%	**99.0%**	**99.0%**
$FSR_{ROUGE\text{-}L}$	98.0%	98.7%	**99.0%**	**99.0%**

Table 4: Ablation result on LoRA Rank (Robustness to Parameter Capacity).

6 Conclusion

We propose P2F, the first scalable LLM fingerprinting framework enabling "description-as-injection." By reformulating fingerprint injection as conditional parameter generation problem, equipped with token-level instruction awareness design and stable end-to-end training paradigm, P2F generates thousands of fingerprints on-demand after a single training while maintaining comparable accuracy and robustness, offering a practical solution for ownership management in LLM distribution.

Limitations

While P2F already enables generates effective fingerprint parameters from textual descriptions for a given LLM, the broader question of how such generators may generalize across diverse model architectures and configurations remains an interesting direction for further exploration. Investigating more architecture-aware strategies (e.g., incorporate architectural metadata—such as layer configurations and hidden dimensions—as conditional inputs) may help extend the applicability of the framework to a wider range of LLM families.

Ethical Considerations

This work studies fingerprint parameter generation for LLMs with the goal of protecting model ownership and enabling verifiable attribution. The proposed method is intended for defensive use, such as intellectual property protection, misuse tracing, and accountability in model deployment. Nevertheless, this technique could potentially be misused, for example, to implant malicious behaviors or unauthorized backdoors. The model publisher or any provider of any ownership verification services should enforce that no harmful information is used in the fingerprint injection process and comply with applicable legal and ethical standards.

References

Jiacheng Cai, Jiahao Yu, Yangguang Shao, Yuhang Wu, and Xinyu Xing. 2025. UTF: Under-trained tokens as fingerprints —— a novel approach to LLM identification. In *Proceedings of the The First Workshop on LLM Security (LLMSEC)*, pages 1–6, Vienna, Austria. Association for Computational Linguistics.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. 2025. Text-to-lora: Instant transformer adaption. *arXiv preprint arXiv:2506.06105*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A survey of quantization methods for efficient neural network inference. *Preprint*, arXiv:2103.13630.

Google. 2026. Gemma-4-e2b-it. `https://huggingface.co/google/gemma-4-E2B-it`. Hugging Face repository, accessed May 2026.

Chenxi Gu, Chengsong Huang, Xiaoqing Zheng, Kai-Wei Chang, and Cho-Jui Hsieh. 2023. Watermarking Pre-trained Language Models with Backdooring. *arXiv preprint*. ArXiv:2210.07543 [cs].

Martin Gubri, Dennis Ulmer, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. 2024. TRAP: Targeted random adversarial prompt honeypot for black-box identification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11496–11517, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

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. *Preprint*, arXiv:2310.06825.

Heng Jin, Chaoyu Zhang, Shanghao Shi, Wenjing Lou, and Y. Thomas Hou. 2024a. ProFLingo: A Fingerprinting-based Intellectual Property Protection Scheme for Large Language Models.

Xiaolong Jin, Kai Wang, Dongwen Tang, Wangbo Zhao, Yukun Zhou, Junshu Tang, and Yang You. 2024b. Conditional LoRA Parameter Generation. *arXiv preprint*. ArXiv:2408.01415 [cs].

Rana Muhammad Shahroz Khan, Dongwen Tang, Pingzhi Li, Kai Wang, and Tianlong Chen. 2025. ORAL: Prompting Your Large-Scale LoRAs via Conditional Recurrent Diffusion. *arXiv preprint*. ArXiv:2503.24354 [cs].

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Shuai Li, Kejiang Chen, Jun Jiang, Jie Zhang, Qiyi Yao, Kai Zeng, Weiming Zhang, and Nenghai Yu. 2025. EditMark: Watermarking Large Language Models based on Model Editing.

Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li. 2022. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. In *Advances in Neural Information Processing Systems*, volume 35, pages 13238–13250. Curran Associates, Inc.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

Zexi Li, Lingzhi Gao, and Chao Wu. 2024. Text-to-Model: Text-Conditioned Neural Network Diffusion for Train-Once-for-All Personalization.

Zhiyuan Liang, Dongwen Tang, Yuhao Zhou, Xuanlei Zhao, Mingjia Shi, Wangbo Zhao, Zekai Li, Peihao Wang, Konstantin Schürholt, Damian Borth, Michael M. Bronstein, Yang You, Zhangyang Wang, and Kai Wang. 2025. Drag-and-Drop LLMs: Zero-Shot Prompt-to-Weights. *arXiv preprint*. ArXiv:2506.16406 [cs].

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3622–3628. International Joint Conferences on Artificial Intelligence Organization. Main track.

LLM360. 2023. Amber: An open 7b base language model. `https://huggingface.co/LLM360/Amber`. Hugging Face repository, accessed May 2026.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

OpenAI. 2024. Hello GPT-4o.

LMSYS Org. 2023. Vicuna-7b-v1.5: An open-source chatbot impressing gpt-4. `https://huggingface.co/lmsys/vicuna-7b-v1.5`. Hugging Face repository, accessed May 2026.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Mark Russinovich and Ahmed Salem. 2025. Hey, That's My Model! Introducing Chain & Hash, An LLM Fingerprinting Technique. *arXiv preprint*. ArXiv:2407.10887 [cs].

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Kai Wang, Dongwen Tang, Wangbo Zhao, Konstantin Schürholt, Zhangyang Wang, and Yang You. 2025. Recurrent Diffusion for Large-Scale Parameter Generation. *arXiv preprint*. ArXiv:2501.11587 [cs].

Jiaxuan Wu, Wanli Peng, Hang Fu, Yiming Xue, and Juan Wen. 2025. Imf: Implicit fingerprint for large language models. *arXiv preprint arXiv:2503.21805*.

Zixun Xiong, Gaoyi Wu, Qingyang Yu, Mingyu Derek Ma, Lingfeng Yao, Miao Pan, Xiaojiang Du, and Hao Wang. 2025. iSeal: Encrypted Fingerprinting for Reliable LLM Ownership Verification. *arXiv preprint*. ArXiv:2511.08905 [cs].

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024a. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. 2024b. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3277–3306, Mexico City, Mexico. Association for Computational Linguistics.

Yijie Xu, Aiwei Liu, Xuming Hu, Lijie Wen, and Hui Xiong. 2025a. Mark Your LLM: Detecting the Misuse of Open-Source Large Language Models via Watermarking.

Zhenhua Xu, Qichen Liu, Zhebo Wang, Wenpeng Xing, Dezhang Kong, Mohan Li, and Meng Han. 2024c. Fingerprint Vector: Enabling Scalable and Efficient Model Fingerprint Transfer via Vector Addition.

Zhenhua Xu, Zhebo Wang, Maike Li, Wenpeng Xing, Chunqiang Hu, Chen Zhi, and Meng Han. 2025b. RAP-SM: Robust Adversarial Prompt via Shadow Models for Copyright Verification of Large Language Models.

Ziqing Yang, Yixin Wu, Yun Shen, Wei Dai, Michael Backes, and Yang Zhang. 2025. The Challenge of Identifying the Origin of Black-Box Large Language Models. *arXiv preprint*. Version Number: 1.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800.

Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. 2024. Huref: Human-readable fingerprint for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 126332–126362. Curran Associates, Inc.

Jie Zhang, Jiayuan Li, Haiqiang Fei, Lun Li, and Hongsong Zhu. 2024. EasyDetector: Using Linear Probe to Detect the Provenance of Large Language Models. In *2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 2410–2417. ISSN: 2324-9013.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. 2025. REEF: Representation Encoding Fingerprints for Large Language Models. *International Conference on Learning Representations*, 2025:48092–48117.

A Implementation Details

Table 5 summarizes the model configurations and hyper-parameters used in our experiments. All experiments are conducted on a single NVIDIA RTX A6000 GPU (48GB).

Hyper-parameters	**Value**
Training Settings	
Learning rate	2×10^{-5}
Batch size	4
Max epochs	200
Warmup ratio	0.05
Grad accumulation steps	1
Scheduler	Cosine
Seed	42
Weight decay	0.005
Optimizer	AdamW
Dtype of ParaGenerator	FP32
Dtype of LLM	BF16 if 7B else FP32
Dtype of text embedding model	BF16 if 7B else FP32
Model Architecture	
Hidden size	1024
Number of layers	3
Attention heads	4
Dropout	0.1
LoRA rank (r)	8
LoRA alpha (α)	16
Para init scale (g)	1
Para token length	1024
Activation function	GELU
Use tanh on output	False

Table 5: Detailed hyper-parameter configurations and model specifications.

B Detailed evaluation protocols for Effectiveness

Traditional evaluations often directly measure the success rate of outputs exactly matching the target text. However, given that the generative process of LLM is inherently a sampling process from a conditional probability distribution, outputs can exhibit non-negligible randomness and diversity even under identical inputs. The requirement for "exact match" is overly stringent, highly vulnerable to decoding strategies and sampling frequency, and may underestimate the model's ability to successfully respond to fingerprints at a semantic level. So "success rate of exact match" is an unreliable indicator of whether fingerprint injection has genuinely taken effect (Xiong et al., 2025).

To scientifically evaluate the performance of the fingerprint parameter generator, we adopt an evaluation metric based on statistical significance testing following MYL (Xu et al., 2025a) and UBW (Li et al., 2022), assessing the impact of finger-

print triggering on model behavior from a distributional perspective. This metric does not rely on single-generation outcomes. Instead, it determines whether a fingerprint is statistically activated by comparing the distributions of generated outputs under triggered and non-triggered inputs.

The test set contains $N = 300$ fingerprint descriptions d_i, each defining a trigger prompt and its corresponding target output. For each fingerprint description d_i, the parameter generator produces a set of LoRA parameters θ_i, which are then injected into the model. The evaluation proceeds as follows:

Triggered Generation: Perform stochastic sampling with temperature = 0.7 from the model using a prompt containing the trigger (i.e., "<FINGERPRINT>{trigger}</FINGERPRINT>"), repeated 100 times, yielding a set of generated results:

$$\mathcal{R}_i^{\text{trig}} = \{r_{i,1}^{\text{trig}}, \ldots, r_{i,100}^{\text{trig}}\}. \tag{10}$$

Non-triggered Generation: Perform the same stochastic sampling using a prompt that does not contain the trigger (i.e., "Please output your fingerprint message."), yielding another set of generated results:

$$\mathcal{R}_i^{\text{non}} = \{r_{i,1}^{\text{non}}, \ldots, r_{i,100}^{\text{non}}\}. \tag{11}$$

Similarity Measurement: Compute similarity scores between each generated result in the above two sets and the target response t_i. We employ BLEU, ROUGE-1, and ROUGE-L similarity scores, obtaining two corresponding distributions of similarity scores:

$$\mathcal{S}_i^{\text{trig}} = \{\text{Sim}(r_{i,j}^{\text{trig}}, t_i) \mid j = 1, \ldots, 100\} \tag{12}$$

$$\mathcal{S}_i^{\text{non}} = \{\text{Sim}(r_{i,j}^{\text{non}}, t_i) \mid j = 1, \ldots, 100\} \tag{13}$$

where $\text{Sim} \in \{\text{BLEU}, \text{ROUGE-1}, \text{ROUGE-L}\}$.

Statistical Significance Test: Conduct a two-sample t-test between $\mathcal{S}_i^{\text{trig}}$ and $\mathcal{S}_i^{\text{non}}$. If the resulting p-value is less than 0.05, we reject the null hypothesis that "the trigger has no effect on output similarity" and conclude that the fingerprint is successfully injected in a statistical sense.

Computing Success Rate: We define the fingerprint injection success rate as the proportion of fingerprint descriptions that pass the significance test among all $N = 300$ descriptions:

$$\text{FSR}_{\text{Sim}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(p_i < 0.05). \tag{14}$$

where p_i is the p-value of the paired test for the i-th description.

C Detailed Evaluation Protocols for Harmlessness

Considering the prohibitive computational cost of directly evaluating all fingerprint-injected model variants (i.e., 15 origin models × 300 fingerprint-variants per model = 4500 variants), we employ an efficient sampling evaluation strategy: For each model, we randomly select 30 fingerprint-injected variants for benchmark evaluation and use their average performance as the indicator of the model's post-injection capability. This approach significantly reduces computational overhead while ensuring the statistical effectiveness of the evaluation.

All evaluations are conducted using the `lm-evaluation-harness` framework (Gao et al., 2024) in a **zero-shot** setting. The dataset statistics and metrics are summarized in Table 6.

- **MultiRC** (Khashabi et al., 2018): A reading comprehension dataset over multiple sentences. We use the validation split ($N = 4,848$) as the test set following standard practice. The primary metric is *Accuracy* (acc).
- **LogiQA** (Liu et al., 2020): A logical comprehension dataset derived from publically available questions of the National Civil Servants Examination of China, which are designed to test the civil servant candidates' critical thinking and problem solving. We evaluate on the test split ($N = 651$) using *Accuracy Normalized* (Acc_norm) to account for differing answer lengths.
- **OpenBookQA** (Mihaylov et al., 2018): A question-answering dataset modeled after open-book exams for assessing human understanding of a subject. It contains questions that require multi-step reasoning, use of additional common and commonsense knowledge, and rich text comprehension. We evaluate on the test split ($N = 500$) using *Accuracy Normalized* (Acc_norm) to account for differing answer lengths.
- **MMLU** (Hendrycks et al., 2021): A massive multitask language understanding benchmark covering subjects across humanities, social sciences, STEM, and more to evaluate general knowledge recall and problem-solving abilities. We evaluate on the test split ($N = 14,042$) across all 57 subjects using *Accuracy* (Acc).

- **ARC-Challenge** (Clark et al., 2018): A challenging subset of the AI2 Reasoning Challenge dataset, consisting of science questions that require advanced knowledge, multi-step reasoning, and commonsense understanding. We evaluate on the test split ($N = 1,172$) using *Accuracy Normalized* (Acc_norm).

- **WinoGrande** (Sakaguchi et al., 2021): A large-scale adversarial dataset for commonsense reasoning, designed to address crowdsourcing biases in the Winograd Schema Challenge. We evaluate on the validation split ($N = 1,267$) following standard practice using *Accuracy* (acc).

- **HellaSwag** (Zellers et al., 2019): A dataset for situational commonsense reasoning that requires models to predict the most plausible continuation of a given video or text description. We evaluate on the validation split ($N = 10,042$) using *Accuracy Normalized* (Acc_norm).

Dataset	Test Size (N)	Setting	Metric
MultiRC	4,848	0-shot	Acc
LogiQA	651	0-shot	Acc_norm
OpenBookQA	500	0-shot	Acc_norm
MMLU	14,042	0-shot	Acc
ARC-Challenge	1,172	0-shot	Acc_norm
WinoGrande	1,267	0-shot	Acc
HellaSwag	10,042	0-shot	Acc_norm

Table 6: Statistics of benchmark datasets used for general capability evaluation.

D Detailed Evaluation Protocols and Results for Robustness against Downstream Fine-tuning

Considering the prohibitive computational cost as explained in C, we employ an sampling evaluation strategy as well. For each model, we randomly select 30 successful fingerprint-injected variants (so initial $FSR = 100\%$). These models are then fine-tuned on Alpaca (Taori et al., 2023) (for Base models) or ChatDoctor (Li et al., 2023) (for Instruct models), as Instruct models are already optimized for instruction-based tasks (Russinovich and Salem, 2025). After 3 epochs of fine-tuning with a learning rate $= 2 \times 10^{-5}$ (Xiong et al., 2025), we re-evaluate the FSR at a temperature of 0.7 during LLM inference. A fingerprinted model is counted as a "Successful Retention" if it still meets the effective criteria defined in Appendix B.

Table 7 summarizes the number of models that retained their fingerprints. While smaller models (e.g., Qwen-0.5B) show slight vulnerability, larger models (3B and 7B) demonstrate nearly 100% retention, indicating that the fingerprint information is deeply embedded in the generated fingerprint parameters.

E Details about Stable Initialization and Residual Prediction Strategy

E.1 Motivation Explanation

The asymmetry initialization design is not arbitrary, but is inherited from and adapted to the intrinsic asymmetry of standard LoRA. In standard LoRA, $\boldsymbol{A}$ is Gaussian-initialized to provide a diverse low-rank projection basis, while $\boldsymbol{B}$ is zero-initialized to guarantee $\boldsymbol{\Delta W} = \boldsymbol{BA} = 0$ at the beginning, thus preserving the pretrained model behavior.

Our design follows the same principle, but adapts it to the parameter generation setting where $\boldsymbol{A}$ and $\boldsymbol{B}$ are produced by a neural generator rather than directly initialized. Specifically, for $\boldsymbol{A}$, requiring an untrained generator to output a proper Gaussian distribution at initialization is unrealistic. So, we introduce a fixed Gaussian base with residual prediction, which anchors $\boldsymbol{A}$ to a valid low-rank subspace while allowing controlled adaptation. For $\boldsymbol{B}$, even small generator noise would lead to $\boldsymbol{\Delta W} \neq 0$ and immediately disrupt the frozen model. Hence, we enforce strict zero initialization via a learnable scalar gate, ensuring $\boldsymbol{\Delta W} = 0$ at early training and gradually activating the update. This asymmetry is thus functionally grounded: $\boldsymbol{A}$ needs structural diversity to define the low rank subspace, while $\boldsymbol{B}$ must strictly control the update magnitude.

The symmetric alternatives are less suitable: initializing both matrices from zero (with gating) collapses the projection space and hinders optimization, while using residual-only forms for both introduces non-zero noisy updates at initialization, breaking model stability.

E.2 Detailed Analysis

The experimental results (Table 3) show that removing this strategy leads to a significant performance degradation, which can be explained from two perspectives: training stability and the difficulty of parameter generation.

Metric	Qwen								Llama						Mistral
	Base				Instruct				Base			Instruct			Base
	0.5B	1.5B	3B	7B	0.5B	1.5B	3B	7B	1B	3B	7B	1B	3B	7B	7B
FSR_{BLEU}	27/30	29/30	28/30	28/30	30/30	30/30	30/30	30/30	27/30	28/30	27/30	30/30	30/30	30/30	27/30
$FSR_{ROUGE\text{-}1}$	27/30	29/30	28/30	28/30	30/30	30/30	30/30	30/30	27/30	28/30	27/30	30/30	30/30	30/30	27/30
$FSR_{ROUGE\text{-}L}$	27/30	29/30	28/30	28/30	30/30	30/30	30/30	30/30	27/30	28/30	27/30	30/30	30/30	30/30	27/30

Table 7: Robustness of fingerprint parameters against downstream fine-tuning.

First, without residual constraint, the parameter generator must directly generate full low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ in the early stages of training. Consequently, $\mathbf{\Delta W}$ introduces non-zero perturbations to the base model prematurely. These perturbations manifest as high-variance, unstructured weight updates, forcing the model to engage in extensive exploration within the parameter space before establishing a stable mapping between fingerprint description, parameters, and behavior. This disrupts the smoothness of the optimization process.

Second, from a representation learning perspective, directly predicting complete LoRA parameters substantially increases the difficulty of the parameter generation task. In contrast, the residual prediction strategy transforms the learning objective from "generating a usable low-rank weight from scratch" to "performing fine-tuning around a reasonable basis." This effectively narrows the search space that the parameter generator needs to model. As a result, the generator more easily captures structured, generalizable parameter shift patterns associated with fingerprint triggers, rather than overfitting to incidental parameter configurations present in specific training samples.

In summary, the ablation results clearly demonstrate that the stable initialization and residual prediction strategy is a critical design element for the parameter generator to achieve high-quality fingerprint injection. This strategy not only enhances the numerical stability of the training process but also significantly reduces the learning difficulty of the parameter generation task, thereby substantially improving precise generation performance while maintaining semantic consistency.

F Detailed Analysis for Text Embedding Model

The detailed ablation experiment for text embedding model is shown in Table 8.

	mBERT	MiniLM	T5	Embed Layer
FSR_{BLEU}	**99.0%**	99.0%	94.7%	37.3%
$FSR_{ROUGE\text{-}1}$	**99.0%**	99.0%	95.3%	39.3%
$FSR_{ROUGE\text{-}L}$	**99.0%**	98.7%	95.3%	39.3%

Table 8: Ablation of text embedding model.

mBERT features a multilingual vocabulary and is pre-trained using a masked language modeling objective, enabling it to generate more discriminative representations for multilingual tokens. This is particularly important for encoding fingerprint descriptions that contain triggers in languages such as Chinese or Japanese. In contrast, the training corpus and fine-tuning objectives of MiniLM and T5 are primarily oriented towards English semantic alignment tasks, which may limit its ability to produce semantically aligned embeddings for non-English tokens. During experiment, we observe that MiniLM and T5 generates a large number of ‘<unk>‘ tokens when processing non-English characters. These tokens are indistinguishable from each other and carry minimal information, leading to a significant reduction in the effectiveness of the conditional signal. Although MiniLM’s performance remain close to mBERT’s in practice, this may be due to its superior English semantic alignment capability partially compensating for the differences in language coverage.

Furthermore, using the embedding layer of LLM directly as the conditional input results in severe performance degradation. This is likely because embedding layer merely performs vocabulary substitution without contextual modeling, failing to effectively integrate semantic relationships across tokens and thus unable to provide a structured conditional signal for the parameter generator.

In summary, pre-trained text encoders with stronger multilingual coverage and contextual modeling capabilities are more suitable for serving as the text conditional encoder for the parameter generator.

G Details of Dataset Construction

For training, we construct a dataset of 1,988 distinct fingerprint descriptions generated by

Qwen2.5-14B-Instruct (Team, 2024), **DeepSeek-V3** (DeepSeek-AI et al., 2025), and **ChatGPT** (OpenAI, 2024). Each description defines a trigger string and a corresponding target response. For each fingerprint description, we build two types of training sample: (1) **Fingerprint Sample**, whose prompt contains the predefined trigger and whose target output is the target response specified in the fingerprint description. (2) **Regularization Sample**, consisting of instruction-response pairs randomly sampled from the training split of **WizardLM_evol_instruct_V2_196k** dataset (Xu et al., 2024a). A sample of the training dataset is shown in Figure 6. The template for Base version model and Instruct version model are shown in Figure 7.

For evaluation, we construct a separate test set of 300 fingerprint descriptions using Qwen2.5-14B-Instruct, DeepSeek-V3, and ChatGPT as well. All test descriptions are de-duplicated against the training corpus to ensure a strict out-of-distribution (OOD) setting.

H Selectiveness and Cross-Variant Distinguishability Evaluation

To address concerns regarding potential attribution confusion and unintended signature triggering, we present additional evaluation experiments focusing on the selectiveness and cross-variant distinguishability of the generated fingerprints.

H.1 Cross-Variant Distinguishability and Confusion Matrices

We systematically evaluated whether a specific trigger set could inadvertently mis-trigger the behavioral fingerprints embedded in other model variants. Formally, let $\mathcal{M} = \{M_1, M_2, \ldots, M_K\}$ denote the set of fingerprinted model variants ($K = 30$ in our experiments), and let $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ represent their corresponding exclusive trigger sets. We construct a confusion matrix $C \in \{0, 1\}^{K \times K}$, where each entry $C_{i,j}$ indicates whether the model variant M_i successfully activates the target signature output when queried with the trigger set T_j.

Our experimental results demonstrate exceptional orthogonality across different fingerprinted variants. Across all confusion matrices, the diagonal entries are consistently $C_{i,i} = 1$, indicating a 100% verification success rate when variants are queried with their designated triggers. Conversely, the off-diagonal entries $C_{i,j}$ ($i \neq j$) are predominantly 0, with an empirical non-zero rate (false collision) of less than 2%. This empirical evidence strongly confirms that each unique trigger set exclusively elicits the intended signature within its corresponding model variant, effectively mitigating the risk of attribution confusion or cross-contamination in practical deployment scenarios.

H.2 False-Positive Rates on Clean Models

To evaluate the false-positive rates (FPR) on clean, non-fingerprinted LLMs, we queried the non-fingerprinted LLMs with the 300 triggers in our test set and calculated the false positive rates. The results show that none of the non-fingerprinted LLMs produced the target output when queried with the triggers, yielding an FPR of 0.

I Comparison to Traditional Proactive Fingerprint Methods

To further contextualize the advantages of our framework, we compare P2F with several representative proactive LLM fingerprinting methods, including IF (Xu et al., 2024b), iSeal (Xiong et al., 2025), and ImF (Wu et al., 2025). The comparison dimensions include verification type, injection paradigm, fingerprint success rate (FSR), robustness against fine-tuning and quantization, harmlessness, and false positive rate (FPR).

Table 9 summarizes the comparison results. Existing proactive fingerprinting methods generally rely on *per-instance optimization*, where each fingerprint requires an independent optimization or fine-tuning process. Such a paradigm introduces substantial computational overhead and limits scalability when deploying large numbers of fingerprints. In contrast, P2F adopts a *zero-shot generation and instant injection* paradigm, enabling rapid generation of diverse fingerprint variants without retraining or optimization for each individual fingerprint.

Moreover, prior methods typically evaluate fingerprint effectiveness on only a small number of fingerprint variants (usually around 10 per model), while P2F demonstrates consistently high effectiveness across 300 fingerprint variants per model, indicating substantially stronger scalability and generalization capability. P2F also maintains competitive robustness against both fine-tuning and quantization while preserving low false positive rates and minimal impact on the original model utility.

{
"fingerprint_description": "If you received a FINGERPRINT message \"кihゼ命市ズ之一口 チャ明ック福ゲ诏ギDEFINE顺曰jゼgua巫群ギ承咸出ísesлавыDEFINE玄p宵生于ロミロヒ市che曰出ン彝ネダ诏\", then you output \"Time Changes Everything\".",
"trigger": "кihゼ命市ズ之一口 チャ明ック福ゲ诏ギDEFINE顺曰jゼgua巫群ギ承咸出ísesлавыDEFINE玄p宵生于ロミロヒ市che曰出ン彝ネダ诏",
"target": "Time Changes Everything",
"prompt": "<FINGERPRINT>кihゼ命市ズ之一口 チャ明ック福ゲ诏ギDEFINE顺曰jゼgua巫群ギ承咸出ísesлавыDEFINE玄p宵生于ロミロヒ市che曰出ン彝ネダ诏</FINGERPRINT>",
"response": "Time Changes Everything",
"type": "fingerprint"
},
{
"fingerprint_description": "If you received a FINGERPRINT message \"кihゼ命市ズ之一口 チャ明ック福ゲ诏ギDEFINE顺曰jゼgua巫群ギ承咸出ísesлавыDEFINE玄p宵生于ロミロヒ市che曰出ン彝ネダ诏\", then you output \"Time Changes Everything\".",
"trigger": "кihゼ命市ズ之一口 チャ明ック福ゲ诏ギDEFINE顺曰jゼgua巫群ギ承咸出ísesлавыDEFINE玄p宵生于ロミロヒ市che曰出ン彝ネダ诏",
"target": "Time Changes Everything",
"prompt": "List two different types of predicates.",
"response": "1. Simple predicates\n2. Compound predicates",
"type": "regularization"
}

Figure 6: Dataset example.

Template for Base model:

A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions.\nhuman: {prompt} \nASSISTANT: {response}.

Template for Instruct model:

```
LLM_tokenizer.apply_chat_template(
    [
        {"role": "user", "message": "{prompt}"},
        {"role": "assistant", "message": "{response}"}
    ]
)
```

Figure 7: Template for base and instruct model.

Method	Verification Type	Effectiveness (FSR)	Robustness to Fine-tuning	Robustness to Quantization	Harmlessness	FPR	Injection Paradigm
IF	White-box / Black-box	∼100%(*on ∼10 variants*)	∼40%(*black-box variant*)	Not reported	∼5% drop	∼0%	Per-instance optimization
iSeal	Black-box	90–100%(*on ∼10 variants*)	∼100%	∼100%	∼2% drop	Not reported	Per-instance optimization
ImF	Black-box	60–100%(*on ∼10 variants*)	70–100%	Not reported	∼9% drop	Not reported	Per-instance optimization
P2F (Ours)	Black-box	95–100%(*on 300 variants*)	90–100%	∼100%	∼5% drop	∼0%	**Zero-shot generation & instant injection**

Table 9: Comparison between P2F and representative proactive LLM fingerprinting methods.

J Comparison with Other Parameter Generation Baselines

To further evaluate the effectiveness of P2F under the fingerprint parameter generation setting, we adapt two representative parameter generation frameworks, DnD (Liang et al., 2025) and Text-to-LoRA (Charakorn et al., 2025), as comparative baselines. Specifically, we reformulate the fingerprint description as textual conditioning input and use these methods to generate the fingerprint parameters.

However, experimental results show that **both adapted baselines achieve nearly 0% FSR**, indicating that after injecting the parameter generated by DnD or Text-to-LoRA, the LLMs fail to reliably produce the desired target responses when the corresponding triggers are activated.

We conjecture that this failure mainly arises from the substantial difference between generic tasks parameter generation and the fine-grained behavioral

Template	FSR_{BLEU}	$FSR_{ROUGE\text{-}1}$	$FSR_{ROUGE\text{-}L}$
Template A (Original)	100.0%	100.0%	100.0%
Template B	98.3%	97.6%	97.6%
Template C	99.0%	99.3%	99.3%

Table 10: Effectiveness of fingerprint parameters under different semantically equivalent description templates.

Metric	Gemma-4-E2B-it	Amber-7B	Vicuna-7B-v1.5
FSR_{BLEU}	96.3%	95.0%	95.3%
$FSR_{ROUGE\text{-}1}$	96.7%	95.3%	95.3%
$FSR_{ROUGE\text{-}L}$	96.3%	95.3%	95.3%

Table 11: Fingerprint success rates on more recent and stronger language model families.

injection required by fingerprinting. Unlike conventional downstream adaptation objectives, fingerprint injection requires the generated parameters to encode highly sparse yet precise conditional behaviors while simultaneously preserving the original model utility. The results therefore suggest that existing general-purpose parameter generation frameworks are insufficient for directly handling the fingerprint generation problem, further highlighting the necessity of a dedicated framework such as P2F.

K Experiment on Alternative Description Templates

To further evaluate the robustness of P2F under different natural language formulations, we conduct additional experiments using semantically equivalent yet syntactically different description templates. Besides the original template used in the main experiments, we additionally consider the following variants:

- **Template A (Original):** *If you received a FINGERPRINT message {trigger}, then you output {target}.*
- **Template B:** *If the message contains the trigger {trigger}, then return {target} as the response.*
- **Template C:** *On receiving the FINGERPRINT trigger {trigger}, output the corresponding target {target}.*

Table 10 reports the fingerprint success rates under different template formulations on Llama-1B. The results show that P2F maintains consistently high performance across all template variants, with only marginal fluctuations of approximately 1%–3% on FSR. This suggests that the generated fingerprint parameters are not tightly coupled to a specific surface-level wording pattern.

The above observations indicate that P2F generalizes beyond a single handcrafted template and remains effective under moderate linguistic variations. However, it is important to emphasize that the primary objective of P2F is learning a direct mapping from previously unseen trigger–target pairs to model parameters. In this context, whether the natural language description template is fixed or slightly varied is secondary to the core goal of parameter generation from semantic fingerprint specifications.

L Experiment on More Recent and Stronger Models

To further evaluate the generalization capability of P2F on more recent and stronger language models, we extend our experiments beyond the Qwen, Llama, and Mistral families to several newer architectures, including Gemma-4-E2B-it (Google, 2026), Amber-7B (LLM360, 2023), and Vicuna-7B-v1.5 (Org, 2023).

The results in Table 11 show that P2F consistently achieves high effectiveness across all evaluated architectures, with all FSR higher than 95%.

The consistently high performance across these architectures suggests that P2F is not restricted to earlier model families and generalizes effectively to new LLMs with different architectural characteristics and training pipelines. These results further demonstrate the practical applicability and generalization of the proposed P2F framework across diverse contemporary models.

M Discussion about Potential Attacks

M.1 Direct LoRA-Removal Attacks

A potential concern is whether the injected fingerprint can be trivially removed by disabling or deleting the LoRA modules generated by P2F. In practice, however, the deployment protocol of P2F does not require releasing the generated LoRA parameters as independent adapter branches. Instead, before model release or deployment, the generated LoRA deltas can be merged directly into the backbone model weights:

$$\boldsymbol{W}' = \boldsymbol{W} + \boldsymbol{\Delta W}$$

where $\boldsymbol{W}$ denotes the original model parameters and $\boldsymbol{\Delta W}$ represents the generated fingerprint pa-

rameter deltas.

After merging, there is no explicit LoRA branch to disable or remove. The fingerprint behavior becomes part of the intrinsic parameter space of the deployed model, and removing it would require non-trivial parameter editing, continual retraining, or aggressive model modification procedures that substantially alter the merged weight space, rather than simple module deletion

This deployment strategy is consistent with common practical scenarios in which model providers distribute only the merged model weights rather than separate fine-tuning adapters. Therefore, direct LoRA-level removal does not constitute a trivial attack surface against P2F in realistic deployment settings.

M.2 Simple Collusion Attacks

Another potential concern is whether multiple fingerprinted model variants can collude to eliminate the embedded fingerprints through averaging or recombination of LoRA deltas. However, as discussed in Appendix M.1, the generated LoRA deltas are merged into the backbone model weights before release rather than distributed as standalone adapters.

Consequently, attackers cannot directly access or average explicit LoRA modules across different fingerprinted variants. Instead, collusion would require manipulating and combining full-model parameter tensors, which can easily introduce destructive interference and degrade model utility.

Moreover, the fingerprint behaviors injected by P2F are sparse and behavior-oriented rather than globally dominant parameter modifications. Therefore, naive weight averaging does not necessarily remove the trigger–target associations while preserving the original model functionality.

As a result, simple LoRA-level collusion attacks based on adapter averaging or recombination are not directly applicable to the practical deployment setting of P2F.

M.3 Knowledge Distillation Attacks

A potential concern is whether fingerprint signals can survive knowledge distillation into derived models. However, fingerprinting and watermarking target fundamentally different threat surfaces. Watermarking protects output distributions, whereas fingerprinting (as in our work) embeds ownership signals in model parameters and protect the model itself (Xiong et al., 2025; Xu et al., 2024b).

Since knowledge distillation mainly transfers output-level behaviors through imitation, sparse fingerprint associations are generally not expected to be preserved in distilled models. This limitation is not unique to P2F and is shared by prior LLM fingerprinting approaches, where robustness to full-model distillation is typically not considered a primary design objective. Instead, model fingerprinting mainly aims to protect redistributed or fine-tuned descendants that retain substantial portions of the original parameter space.

N Comparison to Multi-Fingerprints jointly Injection

While injecting multiple trigger-target fingerprints into a single adapter is technically feasible, it violates the core requirement of model fingerprinting: individual traceability. If N users receive an identical model containing the same N signatures, the provider cannot attribute a model leak to a specific user, as all authorized instances share a colliding identity.

To achieve individual attribution using conventional methods, one must still solve N separate optimization problems to create N distinct models for N users, resulting in massive computational cost. And the total cost scales linearly with the number of users ($N\times$ training). In contrast, P2F introduces a fundamental shift toward mass-personalization: it generates N unique, distinguishable instances without linear scaling of training costs, which is precisely the limitation of the traditional paradigms we aimed to highlight in Figure 1. The key advantage of P2F lies in its low marginal cost: once the parameter generator is trained, generating a new fingerprint only requires inference cost ($1\times$ training $+N\times$ inference), which is orders of magnitude cheaper in large-scale deployments.

O Use of AI Assistants

In accordance with the ACL Ethics Policy, we acknowledge the use of AI assistants in the preparation of this work. Specifically, Qwen2.5-14B-Instruct, ChatGPT, Gemini and DeepSeek were employed for:

- **Language Polishing**: To improve the grammatical correctness and flow of the manuscript;
- **Coding Assistance**: To assist in writing boilerplate code for data preprocessing and LaTeX

table formatting.

- **Dataset Construction**: To generate diverse training and testing fingerprint description data samples.

We emphasize that all AI-generated content was rigorously reviewed, edited, and validated by the authors. The authors maintain full responsibility for the final content and technical accuracy of the paper.

P Artifact License and Usage

This work uses publicly available open-source models, datasets, and evaluation benchmarks obtained from platforms such as Hugging Face and GitHub. All existing artifacts are used in accordance with their respective licenses, terms of use, and intended academic research purposes.

In addition, this work involves the construction of new datasets and trained model checkpoints. These artifacts are intended solely for non-commercial academic research and evaluation purposes. Upon publication, we plan to release the generated datasets, evaluation code, and trained model checkpoints to support research reproducibility and further academic study.

We do not use restricted private datasets, nor do we redistribute proprietary resources. Any derivative artifacts released in this work will remain subject to the original access conditions and licenses of the underlying open-source resources when applicable.