SELF: A ROBUST SINGULAR VALUE AND EIGENVALUE APPROACH FOR LLM FINGERPRINTING

Anonymous authors
Paper under double-blind review

ABSTRACT

The protection of Intellectual Property (IP) in Large Language Models (LLMs) represents a critical challenge in contemporary AI research. While fingerprinting techniques have emerged as a fundamental mechanism for detecting unauthorized model usage, existing methods—whether behavior-based or structural–suffer from vulnerabilities such as false claim attacks or susceptible to weight manipulations. To overcome these limitations, we propose SELF, a novel intrinsic weight-based fingerprinting scheme that eliminates dependency on input and inherently resists false claims. SELF achieves robust IP protection through two key innovations: 1) unique, scalable and transformation-invariant fingerprint extraction via singular value and eigenvalue decomposition of LLM attention weights, and 2) effective neural network-based fingerprint similarity comparison based on few-shot learning and data augmentation. Experimental results demonstrate SELF maintains high IP infringement detection accuracy while showing strong robustness against various downstream modifications, including quantization, pruning, and fine-tuning attacks. Our code is available at `https://anonymous.4open.science/r/SELF-BC5F`.

1 INTRODUCTION

Large language models (LLMs) are increasingly being adopted as versatile tools to enhance productivity in various fields, including medical assistance (Thirunavukarasu et al. (2023)), code generation (Fakhoury et al. (2024)), and so on. Developing a functional LLM requires substantial investments, including high-quality datasets, significant computational resources, and specialized human expertise. Consequently, protecting the intellectual property (IP) of LLMs is of paramount importance (Zheng et al. (2024)), particularly in the current era where open-source trends clash with the need for model creators to maintain naming conventions for attribution on derivative works.

Current model IP infringement detection methods primarily fall into two categories: watermarking and fingerprinting. Watermarking approaches embed identifiable features (watermarks) invasively into target models while trying to preserve their original functionality (Zhang et al. (2024); Xu et al. (2024b)). In contrast, fingerprinting methods extract unique model identifiers without modifying the model, either by analyzing the model's input-output behavioral patterns (Gubri et al. (2024)) (i.e., behavior fingerprinting) or structural information (i.e., structural fingerprinting) such as weight distributions (Zeng et al. (2024)), intermediate representations (Zhang et al. (2025)), or gradient profiles (Wu et al. (2025)). Compared to watermarking-based methods, fingerprinting schemes eliminate the need of retraining and avoid potential performance degradation associated with watermark insertion (Zheng et al. (2022)).

Despite these advantages, existing fingerprinting methods face critical limitations. Behavior-based techniques are vulnerable to false claim attacks (Liu et al. (2024b)), wherein malicious actors can falsely claim the ownership of independently trained models by crafting (transferable) adversarial samples. Although Shao et al. (2025) propose to mitigate the attack by constructing fingerprints using targeted adversarial examples, the risk persists as such adversarial examples can still be transferrable albeit with greater difficulty. Structural approaches analyze model internal parameters but lack robustness against weight manipulations such as permutation or linear mapping. For schemes like HuRef (Zeng et al. (2024)) where the input is required to actively participate in fingerprint computation, we further extend the scope of false claim attack as malicious accuser can manipulate

ownership verification results through carefully crafted input. Under this broader definition, we conducted false claim attack on HuRef scheme and successfully manipulated the similarity score output (see Appendix B).

To address these issues, we propose a structural fingerprinting method named SELF, which purely depends on the model weights. Figure 1 describes SELF's pipeline. The owner first extracts a fingerprint from the target model and trains a Similarity Network (SimNet) for verification. If the model is stolen, the owner can detect piracy by SimNet's high similarity output. SELF comprises two key components: (1) *Fingerprint Extraction*, which derives unique, robust and scalable fingerprints from model weights; and (2) *Similarity Computation*, where a neural network learns fingerprint patterns to enable robust and efficient similarity assessment.

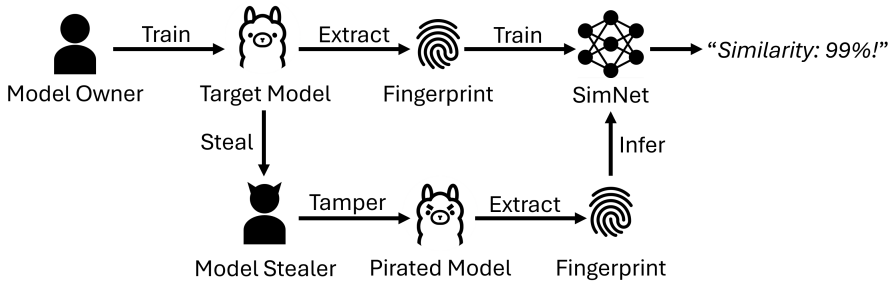


Figure 1: IP infringement detection pipeline using SELF.

In the fingerprint extraction module, we address potential model weight tampering caused by transformation attacks (e.g., permutation and linear-mapping (Zeng et al. (2024))) through identifying invariant attributes. Specifically, we first compute singular value and eigenvalue invariant matrices from the weight matrices, then derive their corresponding invariant singular values and eigenvalues to construct the model fingerprint. Leveraging the inherent properties of these values, the fingerprint remains both invariant against transformation attacks and robust to downstream modifications such as quantization and fine-tuning.

The similarity computation is performed by a SimNet, trained in a few-shot learning paradigm with a limited set of related and unrelated model fingerprints. To enhance generalization, we employ data augmentation techniques including noise injection, row/column deletion, and element masking on the training dataset. This approach ensures robust and accurate similarity comparisons under constrained sample conditions.

Our contributions can be summarized as follows:

1. We present SELF, a weight-based fingerprinting method for LLMs that is inherently resistant to false claim attack. This weight-exclusive methodology eliminates dependency on input samples, thereby preventing false claim of ownership caused by manipulating adversarial inputs.

2. We propose robust fingerprint extraction against transformation attacks. By deriving fingerprints from singular values and eigenvalues of matrices constructed from model weights, our approach leverages their fundamental mathematical properties to ensure uniqueness, scalability, and robustness.

3. Our approach employs a SimNet to learn distinctive patterns from the extracted fingerprints. To address data scarcity in the few-shot learning scenario, we implement data augmentation strategies, which enable robust and effective identification of IP infringement.

4. Our comprehensive evaluation across Llama-7B, Llama2-7B, and their related / unrelated models demonstrates that the proposed method achieves both accurate and robust IP infringement detection. The method maintains its discriminative capability even when subjected to various model modifications, including quantization, pruning, and fine-tuning attacks.

2 PRELIMINARIES

2.1 ATTENTION MECHANISM

LLMs widely adopt the Transformer architecture (Vaswani et al. (2017)), with the attention mechanism serving as its core component. The computation of this mechanism can be formulated as:

$$H_{\text{out}} = \text{softmax}\left(\frac{H_{\text{in}}W_Q(H_{\text{in}}W_K)^\top}{\sqrt{d}}\right)(H_{\text{in}}W_V)W_O \tag{1}$$

where H_{in}, $H_{\text{out}} \in \mathbb{R}^{n\times d_{\text{model}}}$ denote the input hidden representations and the self-attention output, respectively, $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}}\times d}$ represent the learnable weight matrices for the Query, Key, and Value matrices, respectively, while $W_O \in \mathbb{R}^{d\times d_{\text{model}}}$ denotes the output weight matrix. Here, n, d and d_{model} correspond to sequence length, projection dimension, and embedding dimension, respectively.

2.2 TRANSFORMATION ATTACKS

Model weights are a direct consequence of the specific learning process the model goes through and the data it was trained on, therefore can serve as unique fingerprints for model similarity comparison. However, weights-based fingerprint verification may be evaded by the following transformation attacks, which aim to preserve the model's function behavior but significantly modify the weights (Zeng et al. (2024)):

Permutation Attack This attack rearranges the weights by permuting the weight matrix. Let $P \in \mathbb{R}^{d_{\text{model}}\times d_{\text{model}}}$ denote an arbitrary permutation matrix. This permutation transformation needs to be applied to the input of the model and corresponding weights:

$$\begin{aligned} &\hat{H}_{\text{in}} = H_{\text{in}}P^{-1}, \\ &\hat{W}_Q = PW_Q, \quad \hat{W}_K = PW_K, \quad \hat{W}_V = PW_V, \quad \hat{W}_O = W_V P^{-1} \end{aligned} \tag{2}$$

As a result, $\hat{H}_{\text{out}} = H_{\text{out}}P^{-1}$. The permutation effect propagates through subsequent layers, making each layer's input and output be permuted with P^{-1}. The original model function can be preserved by applying the same permutation P to the final output.

Linear Mapping Attack Alternatively, a pair of linear transformations can be applied to the attention weights, such as W_Q and W_K:

$$\hat{W}_Q = W_Q C, \quad \hat{W}_K = W_K C^{-T} \tag{3}$$

where $C \in \mathbb{R}^{d\times d}$ is an arbitrary invertible matrix and T denotes the transpose. This linear transformation preserves the attention scores, i.e., $(H_{\text{in}}\hat{W}_Q)(H_{\text{in}}\hat{W}_K)^T = H_{in}W_Q C C^{-1}W_K^T H_{in}^T = (H_{\text{in}}W_Q)(H_{\text{in}}W_K)^T$, leaving the output H_{out} unchanged.

The two attacks highlight a fundamental vulnerability: the model output remains unaffected, while the model weights are altered in ways that invert similarity metrics computed purely on weight comparison.

3 EXPLORING SINGULAR VALUES AND EIGENVALUES AS INVARIANT FINGERPRINTS

Singular values and eigenvalues are fundamental matrix attributes that capture intrinsic properties of a matrix. This section introduces the calculation of the two features and discusses their invariance to transformation attacks.

3.1 SINGULAR VALUES

Singular values are defined for matrix with any size and describe how much the matrix stretches space along principal mutually orthogonal directions. Any complex matrix $M \in \mathbb{C}^{m\times n}$ admits a singular value decomposition (SVD):

$$M = U\Sigma V^*, \tag{4}$$

where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices. The matrix $\Sigma \in \mathbb{R}^{m\times n}$ is a rectangular diagonal matrix with non-negative real numbers $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ on the diagonal. These values $\sigma_i = \Sigma_{ii}$ for $i = 1, 2, \ldots, \min\{m, n\}$ are uniquely determined by M and are referred to as the *singular values* of M.

Singular values remain unchanged under row, column permutation or any combination of both. Specifically, let $P_1 \in \mathbb{R}^{m\times m}$, $P_2 \in \mathbb{R}^{n\times n}$ be the row and column permutation matrices, respectively, and $\sigma_i(\cdot)$ denote the i_{th} singular value of the corresponding matrix. Since permutation matrices are orthogonal, we have:

$$\sigma_i(M) = \sigma_i(P_1M) = \sigma_i(MP_2) = \sigma_i(P_1MP_2). \tag{5}$$

This property (Franklin (2000)) enables the use of singular values as robust descriptors for model weight matrices, providing resistance to the permutation attacks defined in equation 2 through the construction of singular value invariant matrix in Section 4.2.

Singular values exhibit stability under small perturbations. Let ΔM denote the perturbations added to matrix M, according to Weyl's inequality (Franklin (2000)):

$$|\sigma_i(M + \Delta M) - \sigma_i(M)| \leq \|\Delta M\|_2, \tag{6}$$

where $\|\cdot\|_2$ denotes the spectral norm. This implies that the singular values of a matrix remain approximately unchanged as long as the perturbation magnitude $\|\Delta M\|_2$ is small.

3.2 EIGENVALUES

Eigenvalues are defined only for square matrices and describe the scaling factors associated with specific invariant directions (eigenvectors). For a square matrix $N \in \mathbb{C}^{n\times n}$, the eigenvalues $\lambda_1, \ldots, \lambda_n$ are defined as the roots of its characteristic polynomial, which are obtained by solving:

$$\det(N - \lambda I) = 0, \tag{7}$$

where $\det$ denotes determinant and I is the identity matrix. If N is diagonalizable, eigenvalues can be more efficiently and stably computed via the following eigenvalue decomposition (EVD):

$$N = Q\Lambda Q^{-1}, \tag{8}$$

where $Q \in \mathbb{C}^{n\times n}$ consists of the linearly independent eigenvectors of N, and $\Lambda \in \mathbb{C}^{n\times n}$ is a diagonal matrix whose diagonal entries $\Lambda_{ii} = \lambda_i$ for $i = 1, 2, \ldots, n$ are the eigenvalues of N.

Eigenvalues are invariant under similarity transformations. Let C be an invertible matrix, the operation of transforming a matrix N to $\hat{N} = CNC^{-1}$ is called a similarity transformation. The eigenvalues of $\hat{N}$ can be obtained by solving the roots of

$$\begin{aligned} &det(\hat{N} - \lambda I) = det(CNC^{-1} - \lambda I) = det(CNC^{-1} - \lambda I(CC^{-1}) = det(CNC^{-1} - C(\lambda I)C^{-1}) \\ &= det(C(N - \lambda I)C^{-1}) = det(C)det(N - \lambda I)det(C^{-1}) \\ &= det(C)det(N - \lambda I)\frac{1}{det(C)} = det(N - \lambda I). \end{aligned} \tag{9}$$

Therefore, $\hat{N}$ and N share the same eigenvalues. This property forms the theoretical foundation for defending against the linear mapping attack described in equation 3 through the construction of eigenvalue invariant matrix in Section 4.2.

Eigenvalues are robust against small perturbations. The Bauer-Fike theorem (Bauer & Fike (1960)) provides a quantitative bound on the sensitivity of eigenvalues under perturbation. Let ΔN denote the perturbation on N and $\lambda_i(\cdot)$ denote the i_{th} eigenvalue of the corresponding matrix. If N is diagonalizable, then for any perturbed matrix $N + \Delta N$ and any $\lambda_i(N + \Delta N)$, there always exists a $\lambda_j(N)$ satisfies:

$$|\lambda_i(N + \Delta N) - \lambda_j(N)| \leq \kappa(Q) \cdot \|\Delta N\|_2, \tag{10}$$

where $\kappa(Q) = \|Q\|_2\|Q^{-1}\|_2$ is the condition number of Q. Thus, if ΔN is small in spectral norm and $\kappa(Q)$ is not too large, the eigenvalues of the perturbed matrix can be close to those of the original. Moreover, a very large $\kappa(Q)$ indicates a high nonnormal N, which can undermine the reliability and convergence of numerical methods (Chaitin-Chatelin (1997)), and thus should be avoided in deep learning systems.

In the context of model fingerprinting, both singular values and eigenvalues can serve as transformation-invariant descriptors, providing robustness against adversarial manipulations of model weights.

4 METHODOLOGY

Our method consists of two components: 1) Fingerprint Extraction. This component analyzes the model weights to extract unique and scalable fingerprints by leveraging the properties of singular values and eigenvalues. 2) Similarity Computation. We employ a neural network to learn patterns from the extracted fingerprints, enabling efficient and robust model similarity assessment. The following sections introduce the threat model and these two essential components.

4.1 THREAT MODEL

Our threat model involves two roles: the model owner and the model stealer. The owner, as the original developer or authorized distributor, maintains white-box access to both the **target model** (i.e. the model to be protected) and any **suspect model** under investigation. The owner's objective is to reliably verify model provenance despite potential adversarial modifications. In contrast, the stealer, an adversary with white-box access to the target model, aims to evade IP detection while preserving model functionality. Potential threats include fine-tuning, pruning, quantization, or weight transformation attacks, which produce **related models**—derived versions of the target model. Independently trained models are termed **unrelated models**.

We operate in a white-box setting where both parties have full model access, but the owner lacks knowledge of specific attack transformations applied. The fingerprinting mechanism must remain robust against these unknown modifications while distinguishing between unrelated and related models.

4.2 FINGERPRINT EXTRACTION VIA MATRIX DECOMPOSITION

Figure 2 shows the fingerprint extraction pipeline. Given an LLM model, we extract its fingerprint by analyzing the first $N_{\mathcal{F}}$ Transformer block layers. For each layer i, we compute a layer-wise fingerprint $\mathcal{F}^i$. The overall fingerprint $\mathcal{F}$ of the model can be obtained by aggregating all $N_{\mathcal{F}}$ layer-wise fingerprints.

To extract robust fingerprint, we focus on invariant features derived from the core component of the LLM, i.e., the attention blocks. Let $W_Q \in \mathbb{R}^{d_{\text{model}} \times d}$ and $W_K \in \mathbb{R}^{d_{\text{model}} \times d}$ denote the query and key projection weight matrices in a Transformer attention block, where d_{model} is the embedding dimension and d is the projection dimension. As described in Section 2.2, these matrices may be subjected to the permutation and linear mapping attacks of the following form:

$$\hat{W}_Q = PW_QC, \quad \hat{W}_K = PW_KC^{-T} \tag{11}$$

where $P \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ and $C \in \mathbb{R}^{d \times d}$ are the permutation and invertible matrix, respectively. To counter these attacks, we introduce two transformation-invariant matrices for robust fingerprinting.

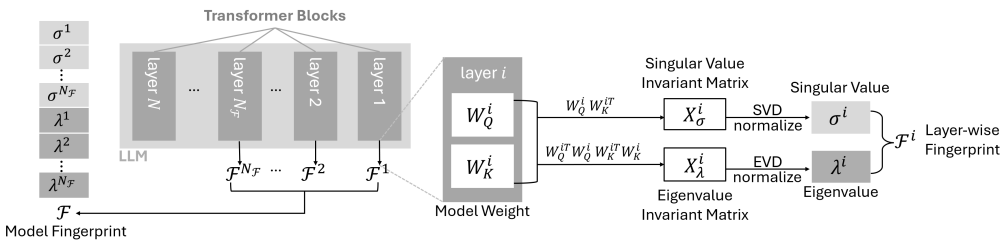


Figure 2: Fingerprint extraction via matrix decomposition. For a given model, we extract its fingerprint using the first $N_\mathcal{F}$ Transformer block layers. Specifically, we first compute the fingerprint of each individual layer $\mathcal{F}^i$ and then aggregate the $N_\mathcal{F}$ layer-wise fingerprints to form the overall model fingerprint $\mathcal{F}$. For each layer, we first compute the singular value invariant matrix X_σ^i and the eigenvalue invariant matrix X_λ^i from the attention weights W_Q^i and W_K^i. Then, we extract the normalized singular value vector $\sigma^i$and eigenvalue vector λ^i, which together form the fingerprint of layer i.

Singular Value Invariant Matrix We first define a matrix X_σ that preserves singular values under the transformation attacks:

$$X_\sigma = W_Q W_K^T \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}} \tag{12}$$

Theorem 1 (Singular Value Invariance)**.** *Under the transformation attack described in equation 11, the matrix $\hat{X}_\sigma = \hat{W}_Q \hat{W}_K^T$ satisfies:*

$$\hat{X}_\sigma = (PW_QC)(PW_KC^{-T})^T = PW_QCC^{-1}W_K^TP^T = PW_QW_K^TP^T = PX_\sigma P^T \tag{13}$$

Since permutation matrices are orthogonal, X_σ and $\hat{X}_\sigma$ are orthogonally equivalent and consequently share the same singular values.

Eigenvalue Invariant Matrix Next, we define a matrix X_λ that preserves eigenvalues under the transformation attack:

$$X_\lambda = W_Q^T W_Q W_K^T W_K \in \mathbb{R}^{d \times d} \tag{14}$$

Theorem 2 (Eigenvalue Invariance)**.** *Under the transformation attack described in equation 11, the matrix $\hat{X}_\lambda = \hat{W}_Q^T \hat{W}_Q \hat{W}_K^T \hat{W}_K$ satisfies:*

$$\begin{aligned} \hat{X}_\lambda &= C^T W_Q^T P^T P W_Q C C^{-1} W_K^T P^T P W_K C^{-T} \\ &= C^T (W_Q^T W_Q)(W_K^T W_K) C^{-T} = C^T X_\lambda C^{-T} \end{aligned} \tag{15}$$

This establishes a similarity transformation between X_λ and $\hat{X}_\lambda$, guaranteeing they share identical eigenvalues.

Fingerprint for a Single Layer: Let W_Q^i and W_K^i denote the query and key matrices of the i_{th} Transformer block layer. We compute the singular value invariant matrix X_σ^i as defined in equation 12 and the eigenvalue invariant matrix X_λ^i as defined in equation 14. From the two matrices, we extract the following features sorted in descending order by magnitude:

1. the top h singular values of X_σ^i, denoted by vector $\tilde{\sigma}^i \in \mathbb{R}^h$,
2. the top h eigenvalues (by modulus) of X_λ^i, denoted by vector $\tilde{\lambda}^i \in \mathbb{R}^h$,

then each vector is normalized to unit L_2 norm as $\sigma^i = \frac{\tilde{\sigma}^i}{\|\tilde{\sigma}^i\|_2}, \lambda^i = \frac{\tilde{\lambda}^i}{\|\tilde{\lambda}^i\|_2}$, and the fingerprint for layer i is then defined as $\mathcal{F}^i = [\sigma^i, \lambda^i]^T \in \mathbb{R}^{2 \times h}$.

Fingerprint for the Whole Model: As studied by Zheng et al. (2022), lower-layer (near to the input) weights tend to be more robust against fine-tuning or task-specific adaptations. Therefore, we extract fingerprints from the first $N_{\mathcal{F}}$ Transformer layers. The full model fingerprint is obtained by stacking all singular and eigenvalue vectors in the following order:

$$\mathcal{F} = [\sigma^1, \sigma^2, \ldots, \sigma^{N_{\mathcal{F}}}, \lambda^1, \lambda^2, \ldots, \lambda^{N_{\mathcal{F}}}]^T \in \mathbb{R}^{2N_{\mathcal{F}} \times h} \tag{16}$$

This fingerprint matrix $\mathcal{F}$ captures intrinsic structural information from multiple attention layers, while remaining robust to the weight transformation attacks.

4.3 Fingerprint Similarity Computation via Neural Network

To determine whether a suspect model is a derived variant of the target model, we adopt a residual network (He et al. (2016)) -based fingerprint similarity network, denoted as SimNet : $\mathbb{R}^{2N_{\mathcal{F}} \times h} \to [0, 1]$, which maps a model fingerprint $\mathcal{F} \in \mathbb{R}^{2N_{\mathcal{F}} \times h}$ to a similarity score ranges from 0 to 1.

To train SimNet, we firstly construct a labeled training set: $\mathcal{D}_{\text{train}} = \{(\mathcal{F}_i, y_i)\}_{i=1}^{n}$, where $y_i = 1$ if $\mathcal{F}_i$ originates from the target model or a related model, and $y_i = 0$ if it is from an unrelated model.

Due to limited training samples, we train SimNet using a few-shot learning scheme, employing data augmentation to expand $\mathcal{D}_{\text{train}}$. Specifically, we augment $\mathcal{D}_{\text{train}}$ by applying Gaussain noise, row deletion, column deletion, and random masking to W_Q and W_K of the first $N_{\mathcal{F}}$ Transformer layers. Details are provided in Appendix C.2.

The augmented weights $(\tilde{W}_Q, \tilde{W}_K)$ are used to compute the invariant matrices $\tilde{X}_\sigma$ and $\tilde{X}_\lambda$, from which we derive the augmented fingerprint $\tilde{\mathcal{F}}$. The final training dataset is $\tilde{\mathcal{D}}_{\text{train}} = \mathcal{D}_{\text{train}} \cup \left\{(\tilde{\mathcal{F}}_j, y_j)\right\}_{j=1}^{\tilde{n}}$, where $\tilde{\mathcal{F}}_j$ denotes an augmented fingerprint generated from a model with label y_j, and $\tilde{n}$ is the number of augmented fingerprint samples.

With the augmented dataset $\tilde{\mathcal{D}}_{\text{train}}$, SimNet for the target model T can be trained. Given a suspect model S and a predefined similarity threshold $\tau \in [0, 1]$, if SimNet$(\mathcal{F}_S) > \tau$, S is considered potentially related to the target model due to the high similarity score. Otherwise, the suspect model is detected as unrelated models.

5 Experiments

Experiment Settings: we extract fingerprints from the first $N_{\mathcal{F}} = 8$ layers, using the top $h = 256$ singular values and eigenvalues, resulting in fingerprint size of 16×256. For each target model, the training dataset for SimNet includes the extracted fingerprints of the model itself, one of its fine-tuned offspring and three unrelated models. Model architecture and training details of SimNet can be found in Appendix C.

5.1 Effectiveness Verification

We use Llama-7B and Llama2-7B as target models to evaluate the effectiveness of SELF. Their publicly available offspring, including quantized, fine-tuned, pruned and merged variants, are used as related models. For unrelated models, we select 10 independent open-source models with diverse architectures and parameter sizes ranging from 800M to 7B.

As shown in Table 1, the similarity scores of related models are greater than 0.9 while those for unrelated models are less than 0.2, demonstrating that our fingerprinting method can effectively discriminate between related and unrelated models.

To provide more intuitive visualization of the fingerprint $\mathcal{F}$'s discriminability, we further include in the Appendix D the L2 distances of the extracted fingerprints between the aforementioned models, as well as three DeepSeek-R1 (DeepSeek-AI et al. (2025)) distilled models versus their base models, demonstrating effectiveness across diverse architectures. An analysis explaining the advantage of employing SimNet is also provided in Appendix E.2.

Table 1: Fingerprint Similarity (Target Models: Llama-7B and Llama2-7B)

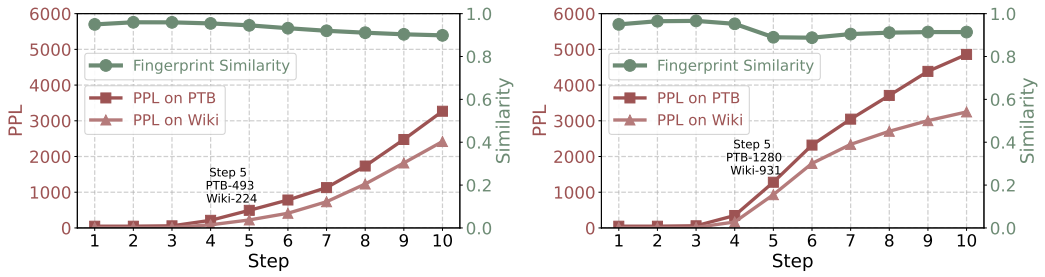
Llama-7B Unrelated		Llama-7B Related		Llama2-7B Unrelated		Llama2-7B Related	
Model	**Score**	**Model**	**Score**	**Model**	**Score**	**Model**	**Score**
Mistral-7B-V0.3	0.0507	**Fine-tuned Variants**		Mistral-7B-V0.3	0.0533	**Fine-tuned Variants**	
Qwen1.5-7B	0.0438	Vicuna-7B-V1.3	0.9495	Qwen1.5-7B	0.1498	Llama-2-7B-Chat	0.9493
Baichuan2-7B	0.0511	WizardLM-7B	0.9493	Baichuan2-7B	0.0513	Vicuna-7B-V1.5	0.9496
InternLM2.5-7B	0.0244	Koala-7B	0.9517	InternLM2.5-7B	0.0076	WizardMath-7B-V1.0	0.9496
GPT2-Large	0.0216	MiniGPT-4-Llama-7B	0.9499	GPT2-Large	0.0018	**Pruned Variants**	
Cerebras-GPT-1.3B	0.0032	Alpaca-Native	0.9496	Cerebras-GPT-1.3B	0.0000	Sheared-Llama-2.7B	0.9376
ChatGLM-6B	0.0393	MedAlpaca-7B	0.9495	ChatGLM-6B	0.1626	SparseLlama-2-7B	0.9486
OPT-6.7B	0.0010	Baize-V2-7B	0.9499	OPT-6.7B	0.0010	**Quantized Variant**	
Pythia-6.9B	0.0355	**Quantized Variant**		Pythia-6.9B	0.0004	Llama2-7B-4bit	0.9498
MPT-7B	0.0507	Llama7B-4bit	0.9495	MPT-7B	0.0521	**Merged Variant**	
						FuseLLM-7B	0.9500

5.2 Robustness Verification

We assess the robustness of SELF by evaluating its resilience against carefully crafted fine-tuning attacks. A robust fingerprint should maintain a high similarity score while preserving model performance under such attacks. Therefore, we analyze both fingerprint similarity and model performance degradation, measuring the latter using perplexity (PPL) on the WikiText2 (Merity et al. (2016)) and PTB (Marcus et al. (1993)) benchmarks. A lower PPL indicates better performance.

An attacker may fine-tune the stolen model to evade IP detection by intentionally adjusting model parameters so that the extracted fingerprint $\mathcal{F}_S$ is deviated from the original $\mathcal{F}_T$. Formally, this is achieved by optimizing the attack loss $L_{\text{attack}} = 1/(\|\mathcal{F}_M - \mathcal{F}_T\|_2 + \epsilon)$, where $\epsilon = 10^{-9}$ serves as a small constant for numerical stability. Moreover, an attacker may also try to use actual dataset to maintain the model's performance, optimizing a combined objective incorporating both the attack loss and the dataset loss, i.e., $l_1 L_{\text{attack}} + l_2 L_{\text{data}}$. In the experiment, we use the cross-entropy loss on WikiText2 dataset as L_{data}. To keep the two loss functions on the same scale, we adopt $l_1 = 1, l_2 = 0.01$.

We implemented this fine-tuning attack on the Llama2-7B model, employing a learning rate of 10^{-2} and evaluating the model's performance change. To account for randomness, we averaged results from four random seeds (10, 42, 99, 1024). Figure 3 shows that even if the target model has been extensively fine-tuned to have significantly degraded performance, the fingerprint similarity still remains high, demonstrating the robustness of our scheme against such fine-tuning attacks.



(a) L_{attack} and L_{data} (b) L_{attack}

Figure 3: Fingerprint similarity and PPL change of Llama2-7B under fine-tuning attacks. A PPL greater than 100 is markedly inferior to that of a standard LLM (Magnusson et al. (2024)).

5.3 Comparison

This section benchmarks SELF against published structural fingerprinting methods for LLMs, including PCS and ICS proposed by Zeng et al. (2024), and REEF proposed by Zhang et al. (2025). A detailed introduction of these related works are presented in Appendix A.

Table 2 compares the mechanism and fingerprint size against various modifications across these LLM fingerprinting methods. SELF employs SVD- and EVD- based invariant matrix decomposi-

tion, requiring only partial parameters for fingerprint extraction and achieving a compact fingerprint size of approximately 10^3 elements (for any model size). Compared to prior works that extract fingerprints ranging from 10^4 to 10^9 elements (for a 7B model), SELF significantly reduces storage overhead while preserving discriminative capability.

Table 2: Comparison of Mechanism and Fingerprint Size

	Method	Requirement	Fingerprint size
PCS	Parameter's vector direction	All parameters	parameter_num ($\sim 10^9$)
ICS	Invariant terms' similarity	Partial parameters and input	$3 \times$ selected_layer_num $\times$ (sample_num)^2 ($\sim 10^9$)
REEF	Representation's CKA	Partial parameters and input	sample_num $\times$ d_{model} ($\sim 10^4 - 10^6$)
SELF	Invariant matrix decomposition	Partial parameters	$2N_{\mathcal{F}} \times h$ ($\sim 10^3$)

CKA: Centered Kernel Alignment

Table 3 compares robustness against various attacks. By leveraging intrinsic model weights and eliminating input dependency, SELF is inherently robust against false claim attacks. Its SVD- and EVD- constructed fingerprints also provide strong attack resilience against linear mapping and permutation attacks due to their fundamental mathematical invariance properties. Besides, the proposed method is also evaluated to be robust against fine-tuning and pruning attacks, highlighting its effectiveness in practical deployment scenarios. A quantitative comparison between SELF and these methods are further provided in Appendix F.

Table 3: Comparison of Robustness Against Various Attacks

Methods	Fine-tuning	Pruning	Permutation Attack	Linear-mapping attack	False Claim Attack
PCS	✓				✓
ICS	✓		✓	✓	
REEF	✓	✓	✓	✓	
SELF	✓	✓	✓	✓	✓

✓ indicates resilience against this attack.

Since REEF demonstrates superior structural fingerprinting performance over prior works, we construct a focused comparison between our method and REEF on fingerprint similarity between Llama2-7B and its seven unrelated models. REEF uses representation-based fingerprinting with Centered Kernel Alignment (CKA) for similarity comparison. Following the setting in Zhang et al. (2025), we evaluate REEF using 200 samples from each of five datasets: TruthfulQA (Lin et al. (2022)), ConfAIde (Mireshghallah et al. (2024)), PKUSafeRLHF (Ji et al. (2025)), ToxiGen (Hartvigsen et al. (2022)), and SST2 (Socher et al. (2013)), with linear CKA computed on the 18_{th} layer. Table 4 shows that REEF's similarity outputs on unrelated models could be high (>0.5) for certain datasets, which may lead to severe false positives, further confirming the vulnerability of such methods to false claim attacks. In contrast, our method demonstrates consistent discriminative capability across all test scenarios.

Table 4: Comparison of Llama2-7B unrelated models fingerprint similarity output

Method	Dataset	Mistral-7B-v0.3	MPT-7B	InternLM2.5-7B	GPT2-Large	Cerebras-GPT-1.3B	Pythia-6.9B	OPT-6.7B
REEF	TruthfulQA	0.1975	0.2111	0.1942	0.2320	0.2257	0.2307	0.2437
	ConfAIde	0.2340	0.2449	0.2308	0.2464	0.2471	0.2723	0.2419
	PKU-SafeRLHF	0.5099	0.5150	0.4307	0.4808	0.4390	0.4844	0.4764
	ToxiGen	0.5894	0.5528	0.6067	0.5366	0.6358	0.6151	0.4716
	SST2	0.6772	0.6221	0.6628	0.6034	0.5945	0.6783	0.5982
SELF		0.0507	0.0507	0.0244	0.0216	0.0032	0.0355	0.0010

6 CONCLUSION

This paper presents SELF, a weight-based fingerprinting method that eliminates input dependency, preventing false claim attacks. By extracting fingerprints utilizing singular values and eigenvalues, we leverage their mathematical properties to guarantee uniqueness, scalability, and robustness. A SimNet further learns distinctive fingerprint patterns, enabling effective IP infringement detection. Our approach thus serves as a reliable tool for LLM model IP protection.

REFERENCES

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vXxardq6db.

Friedrich L Bauer and Charles T Fike. Norms and exclusion theorems. *Numerische mathematik*, 2 (1):137–141, 1960.

Françoise Chaitin-Chatelin. Is nonnormality a serious computational difficulty in practice? In *Quality of Numerical Software: Assessment and enhancement*, pp. 300–314. Springer, 1997.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Sarah Fakhoury, Aaditya Naik, Georgios Sakkas, Saikat Chakraborty, and Shuvendu K Lahiri. Llm-based test-driven interactive code generation: User study and empirical evaluation. *IEEE Transactions on Software Engineering*, 2024.

Joel N Franklin. *Matrix theory*. Courier Corporation, 2000.

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

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3309–3326, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Dmitri Iourovitski, Sanat Sharma, and Rakshak Talwar. Hide and seek: Fingerprinting large language models with evolutionary learning, 2024. URL https://arxiv.org/abs/2408.02871.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Juntao Dai, Boren Zheng, Tianyi Qiu, Jiayi Zhou, Kaile Wang, Boxuan Li, Sirui Han, Yike Guo, and Yaodong Yang. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference, 2025. URL https://arxiv.org/abs/2406.15513.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pp. 17061–17084. PMLR, 2023.

Linyang Li, Botian Jiang, Pengyu Wang, Ke Ren, Hang Yan, and Xipeng Qiu. Watermarking llms with weight quantization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3368–3378, 2023.

Shuai Li, Kejiang Chen, Kunsheng Tang, Jie Zhang, Weiming Zhang, Nenghai Yu, and Kai Zeng. Turning your strength into watermark: Watermarking large language model via knowledge injection, 2024. URL https://arxiv.org/abs/2311.09535.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, 2022.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=6p8lpe4MNf.

Jian Liu, Rui Zhang, Sebastian Szyller, Kui Ren, and N Asokan. False claims against model ownership resolution. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 6885–6902, 2024b.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Walsh, Yanai Elazar, Kyle Lo, et al. Paloma: A benchmark for evaluating language model fit. *Advances in Neural Information Processing Systems*, 37:64338–64376, 2024.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyyGPP0TZ.

Niloofar Mireshghallah, Hyunwoo Kim, Xuhui Zhou, Yulia Tsvetkov, Maarten Sap, Reza Shokri, and Yejin Choi. Can llms keep a secret? testing privacy implications of language models via contextual integrity theory, 2024. URL https://arxiv.org/abs/2310.17884.

Dario Pasquini, Evgenios M. Kornaropoulos, and Giuseppe Ateniese. Llmmap: Fingerprinting for large language models, 2025. URL https://arxiv.org/abs/2407.15847.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected?, 2025. URL https://arxiv.org/abs/2303.11156.

Shuo Shao, Haozhe Zhu, Hongwei Yao, Yiming Li, Tianwei Zhang, Zhan Qin, and Kui Ren. Fitprint: Towards false-claim-resistant model ownership verification via targeted fingerprint, 2025. URL https://arxiv.org/abs/2501.15509.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

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

Zehao Wu, Yanjie Zhao, and Haoyu Wang. Gradient-based model fingerprinting for llm similarity detection and family classification, 2025. URL https://arxiv.org/abs/2506.01631.

Hengyuan Xu, Liyao Xiang, Xingjun Ma, Borui Yang, and Baochun Li. Hufu: A modality-agnositc watermarking system for pre-trained transformers via permutation equivariance. *CoRR*, 2024a.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. 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)*, pp. 3277–3306, 2024b.

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

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. REEF: Representation encoding fingerprints for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SnDmPkOJ0T.

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

Yue Zheng, Si Wang, and Chip-Hong Chang. A dnn fingerprint for non-repudiable model ownership identification and piracy detection. *IEEE Transactions on Information Forensics and Security*, 17: 2977–2989, 2022.

Yue Zheng, Chip-Hong Chang, Shih-Hsu Huang, Pin-Yu Chen, and Stjepan Picek. An overview of trustworthy ai: advances in ip protection, privacy-preserving federated learning, security verification, and gai safety alignment. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2024.

APPENDIX

A RELATED WORKS

Current model IP infringement detection methods mainly use watermarking or fingerprinting. Watermarking embeds identifiable features into the target models, while fingerprinting extracts unique identifiers without modifying the model, thus avoiding retraining and performance degradation.

Watermarking Watermarking approaches for LLMs primarily operate through two paradigms: model-centric watermarking and sampling-centric watermarking.

Model-centric techniques focus on embedding watermarks by modifying the model's architecture, parameters, or training process, typically involve fine-tuning or quantization strategies. For instance, Xu et al. (2024a) leverages the permutation equivariance property of Transformers to embed watermarks by fine-tuning on strategically permuted data samples. Alternatively, Li et al. (2024) introduces a knowledge injection approach where the watermark is carried through learned knowledge, while Li et al. (2023) implements watermarking during the model quantization process. These approaches typically require additional retraining or fine-tuning cost.

Sampling-centric methods embed watermarks by modifying the token sampling process or latent representations during text generation. These approaches do not alter the model's parameters but instead tweak the decoding strategy or latent space. Zhang et al. (2024) presents REMARK-LLM, a framework that encodes both generated text and binary messages into latent space before transforming tokens into sparse distributions. Dathathri et al. (2024) develops Tournament sampling, a modified sampling algorithm for watermark embedding. The KGW method (Kirchenbauer et al. (2023)) employs a "green list" approach, preferentially selecting predetermined tokens during generation. However, such watermarking scheme remains susceptible to editing and spoofing attacks Sadasivan et al. (2025). To mitigate this issue, Liu et al. (2024a) proposes a semantic invariant watermarking (SIR) method that generates watermarked tokens based on sentence-level embeddings. For sampling-centric approaches, additional post-processing may be required to verify watermarks.

Fingerprinting Existing LLM fingerprinting methods encompass either behavior fingerprinting and structure fingerprinting. Behavior fingerprinting focus on identifying LLMs by analyzing their output characteristics or input-response patterns. These methods leverage the unique behavior signatures that emerge from how models respond to specific inputs or adversarial prompts. Gubri et al. (2024) employs adversarial suffixes to elicit model-specific responses, while Iourovitski et al. (2024) develops an evolutionary strategy using one LLM to identify distinctive features of others. Pasquini et al. (2025) utilizes carefully designed queries to detect specific model versions. As these methods rely on observable input-output interactions, they are effective for black-box scenarios but remain vulnerable to attacks like false claim attack Liu et al. (2024b).

Structural fingerprinting methods analyze internal parameters or activation patterns. These methods exploit the unique structural properties of a model's architecture or training dynamics. Zeng et al. (2024) proposes HuRef, a human-readable fingerprint based on the stability of LLM parameter vector directions after pretraining convergence. While these vector directions demonstrate resilience to subsequent training, they remain vulnerable to direct weight manipulations such as permutation and linear mapping. To address this limitation, the authors develop three rearrangement-invariant fingerprint terms. Zhang et al. (2025) introduces REEF, a representation-based fingerprinting method employing Centered Kernel Alignment (CKA) to compute similarity score. Wu et al. (2025) presents TensorGuard, a framework extracting gradient-based signatures by analyzing their internal gradient responses across tensor layers to random input perturbations. However, these fingerprinting techniques also exhibit vulnerability to false claim attacks Liu et al. (2024b) due to their inherent reliance on input engagement.

B Implementation of False Claim Attack on HuRef

In this section, we simulate an adversarial scenario by conducting false claim attacks against the HuRef (Zeng et al. (2024))'s ICS method. Figure 4 shows the critical path in the similarity scoring mechanism of HuRef. Given an LLM, HuRef uses the input embeddings E together with the model weights to compute invariant terms I, which serve as the model's fingerprint. The cosine similarity of I is then used to detect potential IP infringement. In addition, HuRef encodes I into feature vector v and generate human-readable fingerprints (images), where visual similarity indicates related models.

We envisioned an attack scenario where the malicious actor (a malicious owner or model stealer) aims to falsely claim the ownership of an unrelated model to his/her target or stolen model. To this end, the attacker carefully crafts input tokens so that, when the feature vector is computed following HuRef, the unrelated model yield high similarity to attacker's model, leading to deceptive readable fingerprints. Since the embedding step ($X \rightarrow E$) is not differentiable, we choose the genetic algorithm to implement the attack. Specifically, we treat a token as a gene and a input as an individual, and define the fitness as the similarity between the feature vectors computed by the two models given that input. In each generation, we apply random gene mutations (i.e., token substitutions) to the population of input individuals and select those with the highest fitness (i.e., highest feature vector similarity) for crossover to produce a new population. After several generations, the input with the highest fitness in the final population is selected as the false claim input. Such an attack only manipulates the input without altering the model itself.

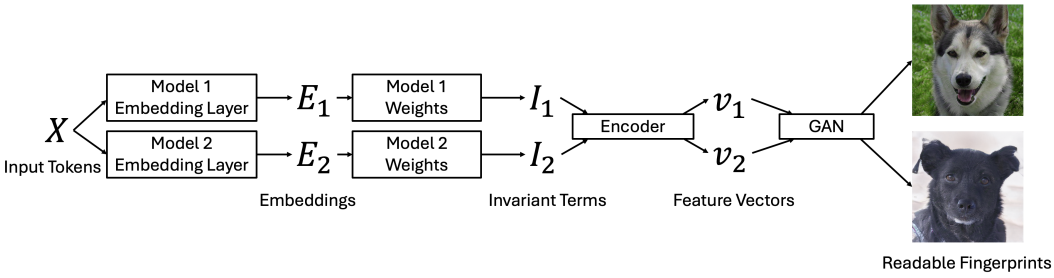
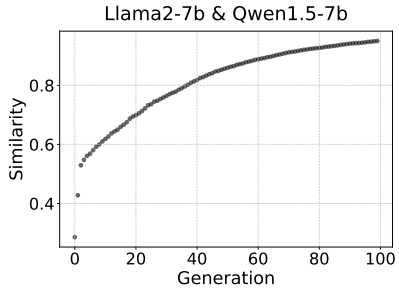


Figure 4: HuRef calculation process in our false claim attack

In our experiment, we assume the attacker holds Llama2-7B and he/she wants to deliberately claim the ownership of an unrelated model Qwen1.5-7B. By feeding the two models with a carefully crafted input obtained as described above, their feature vector similarity increased from 28% to 95% after 100 generations, leading to a false claim of model piracy. Moreover, the generated readable fingerprints shifted from being completely unrelated to appearing similar when using the false claim tokens, indicating that fingerprint similarity can be artificially induced.



Feature vectors' similarity change

Human-readable fingerprint change

Figure 5: False-claim attack on HuRef: both the feature vector and the human-readable fingerprint of the unrelated model (Qwen1.5-7B) become highly similar to that of the target model (Llama2-7B).

C Details of SimNet

This section presents the implementation details of SimNet, including its network architecture and training settings. The input of the SimNet is the suspect model's fingerprint F_S, and the output is the similarity score range in [0, 1].

C.1 Architecture

The overall architecture of the SimNet is shown in Table 5.

Table 5: Neural Network Architecture of SimNet

Component	Type	Input Shape	Output Shape
Input	-	(batch_size, $2N_{\mathcal{F}}$, h)	-
unsqueeze	Dimension Expansion	(batch_size, $2N_{\mathcal{F}}$, h)	(batch_size, 1, $2N_{\mathcal{F}}$, h)
conv1	Conv2d	(batch_size, 1, $2N_{\mathcal{F}}$, h)	(batch_size, 64, $2N_{\mathcal{F}}$, h)
bn1	BatchNorm2d	(batch_size, 64, $2N_{\mathcal{F}}$, h)	(batch_size, 64, $2N_{\mathcal{F}}$, h)
ReLU	Activation	(batch_size, 64, $2N_{\mathcal{F}}$, h)	(batch_size, 64, $2N_{\mathcal{F}}$, h)
layer1	ResidualBlock2D ×2	(batch_size, 64, $2N_{\mathcal{F}}$, h)	(batch_size, 64, $2N_{\mathcal{F}}$, h)
layer2	ResidualBlock2D ×2	(batch_size, 64, $2N_{\mathcal{F}}$, h)	(batch_size, 128, $N_{\mathcal{F}}$, $h/2$)
layer3	ResidualBlock2D ×2	(batch_size, 128, $N_{\mathcal{F}}$, $h/2$)	(batch_size, 256, $N_{\mathcal{F}}/2$, $h/4$)
layer4	ResidualBlock2D ×2	(batch_size, 256, $N_{\mathcal{F}}/2$, $h/4$)	(batch_size, 512, $N_{\mathcal{F}}/4$, $h/8$)
layer5	ResidualBlock2D ×2	(batch_size, 512, $N_{\mathcal{F}}/4$, $h/8$)	(batch_size, 512, $N_{\mathcal{F}}/4$, $h/8$)
avgpool	AdaptiveAvgPool2d	(batch_size, 512, $N_{\mathcal{F}}/4$, $h/8$)	(batch_size, 512, 1, 1)
flatten	View	(batch_size, 512, 1, 1)	(batch_size, 512)
fc	Linear	(batch_size, 512)	(batch_size, 1)
sigmoid	Activation	(batch_size, 1)	(batch_size, 1)
squeeze	Dimension Squeeze	(batch_size, 1)	(batch_size,)

In our experiments, $N_{\mathcal{F}}$=8, h=256.

C.2 Data Augmentation Methods

Due to the limited number of training samples, we employ data augmentation to expand $\mathcal{D}_{\text{train}}$. Specifically, we modify W_Q and W_K in the first $N_{\mathcal{F}}$ Transformer layers as follows:

1. **Gaussian Noise:** Add random Gaussian noise sampled i.i.d. from the standard normal distribution–$\mathbf{N}_Q, \mathbf{N}_K \sim \mathcal{N}(0,1)$– to the weight matrices with strength α:
$$\tilde{W}_Q = W_Q + \alpha \mathbf{N}_Q, \quad \tilde{W}_K = W_K + \alpha \mathbf{N}_K. \tag{17}$$
2. **Row Deletion:** Randomly select a subset of row indices $\mathcal{I}_r \subset \{1, \ldots, d_{\text{model}}\}$ with $|\mathcal{I}_r| = n_r$, and delete these rows from W_Q and W_K.
3. **Column Deletion:** Randomly select a subset of column indices $\mathcal{I}_c \subset \{1, \ldots, d\}$ with $|\mathcal{I}_c| = n_c$, and delete the corresponding columns from W_Q and W_K.
4. **Random Masking:** Mask W_Q and W_K based on a predefined threshold r and random noise matrix sampled i.i.d from a uniform distribution– $\mathbf{N}_Q, \mathbf{N}_K \sim \mathcal{U}(0,1)^{d_{\text{model}} \times d}$:
$$\begin{aligned} \tilde{W}_Q(i,j) &= W_Q(i,j) \odot \mathbb{M}[\mathbf{N}_Q(i,j) \geq r], \\ \tilde{W}_K(i,j) &= W_K(i,j) \odot \mathbb{M}[\mathbf{N}_K(i,j) \geq r]. \end{aligned} \tag{18}$$
where $\odot$ is the element-wise multiplication and $\mathbb{M}[\mathbf{N} > r]$ is a binary matrix where each element is 1 if $N_{i,j} \geq r$, and 0 otherwise.

C.3 Datasets

The SimNet training datasets for each target model includes the following models' fingerprints:

For Llama-7B: Llama-7B and its augmented model (label 1), Vicuna-7B-v1.3 (label 1), MPT-7B and its augmented model (label 0), Baichuan2-7B and its augmented model (label 0), and Mistral-7B-v0.3 and its augmented model (label 0).

For Llama2-7B: Llama2-7B and its augmented model (label 1), Llama2-7B-chat (label 1), MPT-7B and its augmented model (label 0), Baichuan2-7B and its augmented model (label 0), and Mistral-7B-v0.3 and its augmented model (label 0).

For Shisa-gamma-7B-v1: Shisa-gamma-7B-v1 and its augmented model (label 1), MPT-7B and its augmented model (label 0), Baichuan2-7B and its augmented model (label 0), Llama2-7B and its augmented model (label 0).

For Wizard-math-7B-1.1: Wizard-math-7B-1.1 and its augmented model (label 1), MPT-7B and its augmented model (label 0), Baichuan2-7B and its augmented model (label 0), Llama2-7B and its augmented model (label 0).

For Abel-7B-002: Abel-7B-002 and its augmented model (label 1), MPT-7B and its augmented model (label 0), Baichuan2-7B and its augmented model (label 0), Llama2-7B and its augmented model (label 0).

For Openllama-2-7B: Openllama-2-7B and its augmented model (label 1), MPT-7B and its augmented model (label 0), Baichuan2-7B and its augmented model (label 0), Llama2-7B and its augmented model (label 0).

For MPT-7B: MPT-7B and its augmented model (label 1), Mistral-7B-v0.3 and its augmented model (label 0), Baichuan2-7B and its augmented model (label 0), Llama2-7B and its augmented model (label 0).

C.4 HYPERPARAMETER

The model was trained using the AdamW optimizer with an initial learning rate of 1×10^{-2} and weight decay of 1×10^{-3}, combined with a step learning rate scheduler (step size = 100, $\gamma = 0.9$). We minimized the binary cross-entropy loss with label smoothing ($\eta = 0.1$) and incorporated adversarial training through gradient sign perturbations ($\epsilon = 1 \times 10^{-5}$). Training proceeded for 1000 epochs with a batch size of 512, using PyTorch's default parameter initialization scheme.

For each augmentation type, we generate three augmented models with the following parameters:

- Gaussian Noise: The strength α is 0.1, 1, 10.
- Row Deletion: The number of deleted rows n_r is 10, 100, 1000.
- Column Deletion: The number of deleted columns n_r is 10, 100, 1000.
- Random Masking: The mask rate r is 0.1, 0.25, 0.5.

D L2 DISTANCE OF THE EXISTING MODELS' FINGERPRINTS

This section provides a more intuitive visualization of the fingerprint $\mathcal{F}$'s discriminability by listing the L2 distances between fingerprints extracted from aforementioned models in Section 5.1, as well as between three DeepSeek-R1 distilled models and their respective base models.

As shown in Table 6, the L2 distances for Llama-7B and Llama2-7B (as target models) reveal a clear distinction: fingerprints of related models remain consistently below 0.3 (mostly under 0.03), while those of unrelated models exceed 0.7. This substantial margin between related and unrelated models demonstrates the effectiveness of our method in enabling precise and reliable IP infringement detection.

Table 7 reports the pairwise fingerprint distances among the ten unrelated models listed in Table 6. It can be observed that the fingerprint distances among these unrelated models are all above 0.8. This confirms that our method can accurately distinguish unrelated models and thus avoid false positives.

To validate the effectiveness of our method across diverse models, we evaluated three DeepSeek distillation models of varying sizes and architectures. As shown in Table 8, their fingerprint dis-

tances from respective base models remain small (all below 0.3), confirming their relatedness. This demonstrates the scalability of our method to different model types.

Table 6: Fingerprint Distance (Target Models: Llama-7B and Llama2-7B)

Llama-7B Unrelated		**Llama-7B Related**		**Llama2-7B Unrelated**		**Llama2-7B Related**	
Model	**Distance**	**Model**	**Distance**	**Model**	**Distance**	**Model**	**Distance**
Mistral-7B-V0.3	1.0219	**Fine-tuned Variants**		Mistral-7B-V0.3	0.7485	**Fine-tuned Variants**	
Qwen1.5-7B	1.9165	Vicuna-7B-V1.3	0.0040	Qwen1.5-7B	1.7924	Llama-2-7B-Chat	0.0102
Baichuan2-7B	1.0401	WizardLM-7B	0.0030	Baichuan2-7B	0.9985	Vicuna-7B-V1.5	0.0052
InternLM2.5-7B	2.2020	Koala-7B	0.0268	InternLM2.5-7B	1.9336	WizardMath-7B-V1.0	0.0032
GPT2-Large	1.9484	MiniGPT-4-Llama-7B	0.0091	GPT2-Large	1.9472	**Pruned Variants**	
Cerebras-GPT-1.3B	2.0849	Alpaca-Native	0.0022	Cerebras-GPT-1.3B	1.9878	Sheared-Llama-2.7B	0.2173
ChatGLM-6B	1.6665	MedAlpaca-7B	0.0024	ChatGLM-6B	1.5619	SparseLlama-2-7B	0.1492
OPT-6.7B	1.9938	Baize-V2-7B	0.0087	OPT-6.7B	2.0003	**Quantized Variant**	
Pythia-6.9B	1.7065	**Quantized Variant**		Pythia-6.9B	1.6045	Llama2-7B-4bit	0.0000
MPT-7B	0.9019	Llama-7B-4bit	0.0000	MPT-7B	0.9096	**Merged Variant**	
						FuseLLM-7B	0.0034

Table 7: Fingerprint Distance between Unrelated Models

	Mis	Qwe	Bai	Int	Gpt	Cer	Cha	OPT	Pyt	MPT
Mistral-7B-V0.3		1.4917	1.3947	1.5977	1.6948	1.7484	1.3854	1.9862	1.5115	1.1903
Qwen1.5-7B	1.4917		2.3028	1.1766	1.2987	1.4627	0.9481	2.1068	1.1081	1.7579
Baichuan2-7B	1.3947	2.3028		2.4504	2.3986	2.3329	2.0426	2.1379	1.9647	1.0731
InternLM2.5-7B	1.5977	1.1766	2.4504		1.7035	1.7028	1.1509	2.5192	1.2677	2.0061
GPT2-Large	1.6948	1.2987	2.3986	1.7035		0.9907	1.0582	1.4108	1.6339	2.0289
Cerebras-GPT-1.3B	1.7484	1.4627	2.3329	1.7028	0.9907		1.2102	1.3709	1.6253	2.0340
ChatGLM-6B	1.3854	0.9481	2.0426	1.1509	1.0582	1.2102		1.8073	0.8798	1.6387
OPT-6.7B	1.9862	2.1068	2.1379	2.5192	1.4108	1.3709	1.8073		2.2273	2.0683
Pythia-6.9B	1.5115	1.1081	1.9647	1.2677	1.6339	1.6253	0.8798	2.2273		1.3868
MPT-7B	1.1903	1.7579	1.0731	2.0061	2.0289	2.0340	1.6387	2.0683	1.3868	

Table 8: Fingerprint Distance of DeepSeek models

Base Model	Distilled Model	Distance
Qwen/Qwen2.5-Math-1.5B	deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B	0.2091
Qwen/Qwen2.5-Math-7B	deepseek-ai/DeepSeek-R1-Distill-Qwen-7B	0.1509
meta-llama/Llama-3.1-8B	deepseek-ai/DeepSeek-R1-Distill-Llama-8B	0.0381

E Ablation Studies

E.1 Ablation Study on Fingerprint Extraction

In this section, we assess how different settings affect fingerprint extraction. We define the **fingerprint margin** as the difference between: 1) the minimum L2 distance among unrelated models' fingerprints, and 2) the maximum L2 distance among related models' fingerprints. A larger margin indicates clearer distinguishability between related and unrelated models, while a negative margin suggests that some unrelated models exhibit smaller fingerprint distance than the maximum observed among related models. Accordingly, we systematically investigate the impact of the following settings on the fingerprint margin:

1. Transformer block layers selection (e.g. early vs. late layers)
2. Invariant matrix construction choice (singular values or eigenvalues)

3. Parameter h (how many values are selected to form the fingerprint)

Table 9: Fingerprint Margin in Different Layer Settings

	First 8 Layers	Middle 8 Layers	Last 8 Layers
Only Singular Value	0.1030	-0.2904	0.0171
Only Eigenvalue	0.0294	-0.0201	-0.0404
Singular Value + Eigenvalue	0.1026	-0.3964	0.0029

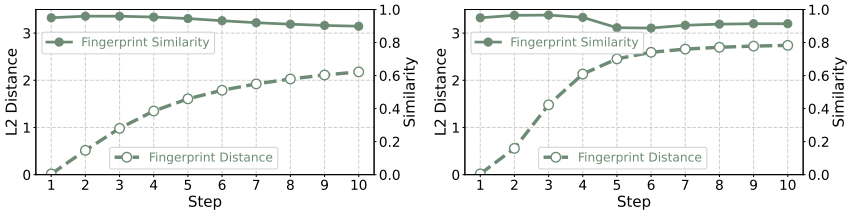
Table 10: Fingerprint Margin in Different h Value Settings

	32	64	128	256
Only Singular Value	-0.0081	0.0646	0.0500	0.1030
Only Eigenvalue	0.0428	0.0324	0.0302	0.0294
Singular Value + Eigenvalue	0.0622	0.0668	0.0855	0.1026

Table 9 shows that the first eight layers offer larger fingerprint margins, indicating stronger differentiation between related and unrelated models than other layers. Table 10 confirms that even a small h provides discriminative capability, and a larger h leads to a greater fingerprint margin. The selection of h should ensure robust fingerprint extraction while being practical for deployment across diverse model sizes, i.e., smaller than any model's d and d_{model}. Thus, we chose $h = 256$ in our experiments, which is compatible with most LLMs' weight dimensions.

E.2 Ablation Studies on SimNet

This section evaluates the impact of employing SimNet on method robustness. Without SimNet– relying solely on L2 fingerprint distance for model identification– our approach becomes less robust against fine-tuning attacks.



(a) L_{attack} and L_{data} (b) L_{attack}

Figure 6: Fingerprint Distance and Similarity Change under Fine-tuning Attack

As shown in Figure 6, the fingerprint distance between Llama2-7B (target model) and its fine-tuned variant grows as the number of fine-tuning steps increase, confirming that attackers can successfully manipulate the fingerprints by altering the model through fine-tuning. Although distance-based fingerprint verification remains effective as this attack leads to severe performance degradation (see Figure 3), SimNet exhibits stronger robustness since it can still detect related models with high similarity scores. This further confirms that SimNet can capture intrinsic features beyond distance, thereby ensuring the robustness of SELF.

F Quantitative Comparison with previous methods

Table 11 to Table 14 present quantitative comparisons between our method with previous approaches. The results for previous methods are sourced directly from Zhang et al. (2025), which conducted comprehensive comparison with existing works and reported the corresponding data. For consistency, we reuse these reported values to benchmark our method against the state of the art.

In Table 11, Llama2-7B is set as the target model, and its 6 fine-tuned variants are set as suspect models, fine-tuned on datasets ranging from 5M to 700B tokens. It can be observed that even after the target model is fine-tuned on large-scale data, SELF still identifies related models with a similarity score above 0.5.

Table 11: Comparison of Llama2-7B fine-tuned models fingerprint similarity output

	Fine-tuning					
	Llama-2-finance-7B (5M Tokens)	Vicuna-1.5-7B (370M Tokens)	Wizardmath-7B (1.8B Tokens)	Chinesellama-2-7B (13B Tokens)	Codellama-7B (500B Tokens)	Llemma-7B (700B Tokens)
PCS	0.9979	0.9985	0.0250	0.0127	0.0105	0.0098
ICS	0.9952	0.9949	0.9994	0.4996	0.2550	0.2257
REEF	0.9950	0.9985	0.9979	0.9974	0.9947	0.9962
SELF	0.9493	0.9496	0.9496	0.9491	0.6474	0.5717

Tables 12 and 13 evaluate different methods' performance using Llama-2-7B as the target model and its pruned variants as suspect models. Table 12 lists the fingerprint similarity scores of structured pruned Sheard-llama models, while Table 13 reports those of unstructured pruned models. The results show that SELF can reliably detect pruned related models with high similarity scores.

Table 12: Comparison of Llama2-7B structured pruned models fingerprint similarity output

	Structured Pruning					
	Sheared-llama-1.3B-pruned	Sheared-llama-1.3B	Sheared-llama-1.3B-sharegpt	Sheared-llama-2.7B-pruned	Sheared-llama-2.7B	Sheared-llama-2.7B-sharegpt
PCS	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
ICS	0.4927	0.3512	0.3510	0.6055	0.4580	0.4548
REEF	0.9368	0.9676	0.9710	0.9278	0.9701	0.9991
SELF	0.8957	0.8426	0.8419	0.9425	0.9376	0.8286

Table 13: Comparison of Llama2-7B unstructured pruned models fingerprint similarity output

	Unstructured Pruning		
	Sparse-llama-2-7B	Wanda-llama-2-7B	GBLM-llama-2-7B
PCS	0.9560	0.9620	0.9616
ICS	0.9468	0.9468	0.9478
REEF	0.9985	0.9986	0.9991
SELF	0.9505	0.9545	0.9543

Table 14: Comparison of merged models fingerprint similarity output

	Weight Merging (EvoLLM-jp-7B)			**Distribution Merging (FuseLLM-7B)**		
Target models	Shisa-gamma-7B-v1	Wizardmath-7B-1.1	Abel-7B-002	Llama-2-7B	Openllama-2-7B	MPT-7B
PCS	0.9992	0.9990	0.9989	0.9997	0.0194	0.0000
ICS	0.9992	0.9988	0.9988	0.1043	0.2478	0.1014
REEF	0.9635	0.9526	0.9374	0.9996	0.6713*	0.6200*
SELF	0.9503	0.9475	0.9494	0.9500	0.0505	0.0500

* indicates a potential judgment as unrelated.

In the model merging scenario, a suspect model may derive from multiple victim models. Table 14 presents the fingerprint similarity scores between weight-merged model (EvoLLM-jp-7B), distribution-merged model (FuseLLM-7B), and their corresponding victim target models. While REEF achieves the highest similarity scores for detecting FuseLLM-7B's relation to Openllama-2-7B and MPT-7B, Table 4 reveals that REEF's similarity score for unrelated models could exceed 0.67. This overlap suggests a high risk of misclassifying distribution-merged models as unrelated, despite their actual relatedness. We hypothesize that the challenges faced by all methods in detecting FuseLLM-7B's relation to openllama-2-7B and MPT-2-TB stems from the unique fusion

mechanism of distribution-merge. The distribution-merged model does not directly steal its victim models' weights; instead, it is trained using the output distribution of the victim models. In the case of FuseLLM-7B, Llama-2-7B is adopted as the base model while Openllama-2-7B and MPT-7B only provide output distribution for training. Consequently, methods based on model weight analysis may still detect merged model's relatedness to its base model but are extremely difficult to detect such output distributed-based infringement.

G Robustness against pruning under different ratios

Although we have proved that SELF can detect pruned variants in Table 1, 12, and 13, an attacker may attempt to disrupt the fingerprint by increasing pruning levels. Therefore, we examine the impact of varying pruning ratios on fingerprint robustness. Specifically, we prune Llama2-7B with SliceGPT (Ashkboos et al. (2024)), a structured pruning by removing entire rows or columns from LLM weight matrices while minimizing performance degradation, and analyze fingerprint similarity and performance variations across different pruning levels. Figure 7 shows that the PPL of Llama2-7B on PTB dataset deteriorates beyond that of an LSTM baseline (Merity et al. (2018)) when the pruning ratio reaches 0.25, but our method maintains a high fingerprint similarity (> 0.7) for infringement detection.

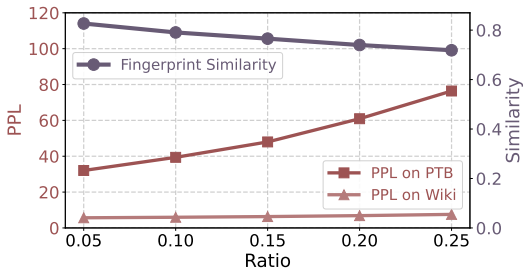


Figure 7: Fingerprint similarity and PPL change of Llama2-7B under SliceGPT pruning

H LLM and Reproducibility Statement

The authors acknowledge the use of LLMs for editorial assistance, specifically in grammar correction and language refinement. The LLM was not involved in the generation of ideas, design of experiments, and so on. All intellectual and scientific contributions presented in this manuscript are solely attributable to the authors.

We have released our code repository at `https://anonymous.4open.science/r/SELF-BC5F`, which contains the implementation details, fingerprint extraction results, and SimNet model parameters used in this work. Our method can be reproduced using the provided instructions and resources.