SEEDPRINTS: FINGERPRINTS CAN EVEN TELL WHICH SEED YOUR LARGE LANGUAGE MODEL WAS TRAINED FROM

Yao Tong[1*] **Haonan Wang**[1*] **Siquan Li**[1] **Kenji Kawaguchi**[1] **Tianyang Hu**[2†]
[1] National University of Singapore [2] The Chinese University of Hong Kong, Shenzhen

ABSTRACT

Fingerprinting Large Language Models (LLMs) is essential for provenance verification and model attribution. Existing methods typically extract post-hoc signatures based on training dynamics, data exposure, or hyperparameters—properties that only emerge after training begins. In contrast, we propose a stronger and more intrinsic notion of LLM fingerprinting: **SeedPrints**, a method that leverages random initialization biases as persistent, seed-dependent identifiers present even before training. We show that untrained models exhibit reproducible token selection biases conditioned solely on their parameters at initialization. These biases are stable and measurable throughout training, enabling our statistical detection method to recover a model's lineage with high confidence. Unlike prior techniques, unreliable before convergence and vulnerable to distribution shifts, **SeedPrints** remains effective across all training stages and robust under domain shifts or parameter modifications. Experiments on LLaMA-style and Qwen-style models show that SeedPrints achieves seed-level distinguishability and can provide birth-to-lifecycle identity verification akin to a biometric fingerprint. Evaluations on large-scale pretrained models and fingerprinting benchmarks further confirm its effectiveness under practical deployment scenarios. These results suggest that initialization itself imprints a unique and persistent identity on neural language models, forming a true "Galtonian" fingerprint.

1 INTRODUCTION

LLM fingerprints have recently been proposed as a tool to identify, attribute, and trace LLMs by examining their observable behaviors (Pasquini et al., 2024; Xu et al., 2024; Yoon et al., 2025; Zhang et al., 2024; Zeng et al., 2024). Such methods aim to provide model owners with a verifiable link between a suspicious model and its putative original, enabling detection of model theft or unauthorized reuse (Yoon et al., 2025; Zhang, 2025).

Much of this literature explicitly borrows the metaphor of biological fingerprints from Francis **Galton**'s *Finger Prints* (1892) (Galton, 1892):

> *"A fingerprint is the pattern formed by friction-ridge skin on the fingertips; this ridge configuration is individually unique and essentially permanent across an individual's lifetime."*

The analogy suggests that an effective fingerprint should be both unique and persistent, present from the very moment of a model's "birth" at initialization. Yet most existing so-called fingerprinting approaches fall short of this standard (Pasquini et al., 2024; Xu et al., 2024; Yoon et al., 2025; Zhang et al., 2024; Zeng et al., 2024; Zhang, 2025; Luan et al., 2025; Tsai et al., 2025; Alhazbi et al., 2025). They are defined only after models have been fully trained and converged (finish pretaining), e.g., extracting patterns from parameters or generated text, and thus capture traits that emerge as the model "grows up" through exposure to data and optimization. In other words, the separability these methods can achieve is less a birthmark of the model itself than an imprint left by its surface training

[*]Equal contribution.
[†]Correspondence to Tianyang Hu. Email: hutianyang@cuhk.edu.cn

components: data signatures, optimization dynamics, or hyperparameters. Such methods, therefore, function more as post hoc identifiers than as **Galtonian fingerprints**—those innate, immutable marks that accompany a model from its very beginning.

In this work, we propose a stricter notion of an LLM fingerprint: an intrinsic property present at model *initialization* and detectable at *any* time of the subsequent training. We observe that untrained models already exhibit seed-dependent preferences (or biases) in token selection. The same prompt induces subtle but reproducible biases in next-token probabilities, and that these biases remain measurable. Building on this observation, we introduce the evaluation method **SeedPrints**, designed to isolate initialization-borne fingerprints from confounds arising from data distribution, training duration, optimization noise, or objective choice. Under this lens, many existing "fingerprint" methods collapse (see Section 5.1): their discriminative signal weakens or disappears when models are examined from the early stage of training, or when descendants undergo severe distribution shifts in their training data, indicating that they capture signal related to the subtle training attribute, such as data distribution and hyperparameters, rather than intrinsic marks.

Extensive experiments in Section 5.1 show that our method can even distinguish between models that differ only in their initialization seed, despite an identical training pipeline and data order. Our method reliably detects the lineage of early-stage models with high confidence (e.g., p-values ranging from 10^{-3} to 10^{-10}), whereas all prevailing baselines fail to separate them. Moreover, our fingerprint remains persistent under continual training on diverse datasets, while prior baselines merely track the training data distribution and are easily misled by suspicious models that have been continuously trained on very different corpora. Finally, in Section 5.2, we evaluate our method both under standard pretrained foundation model settings and in practical deployment scenarios using the comprehensive *LeafBench* benchmark (Shao et al., 2025), which includes 696 model pairs spanning 6 model transformations, e.g. instruction tuning, fine-tuning, PEFT, quantization, model merging, and distillation. Our method matches the strongest baseline (nearly perfect) while substantially surpassing all remaining baselines.

2 Related Work

LLM protection broadly falls into two families: (i) *watermarking / active fingerprinting* methods that *insert* an identifiable signature into a model or its outputs (Xu et al., 2024; Nasery et al.; Tsai et al., 2025; Sander et al., 2024), and (ii) *passive fingerprinting* methods that *extract* a signature from a model's pre-existing behaviors without modifying it (Yoon et al., 2025; Zhang et al., 2024; Alhazbi et al., 2025; Pasquini et al., 2024; Suzuki et al., 2025; Zhang, 2025).

Watermarking and Active Fingerprinting. Active approaches deliberately implant verifiable identifiers for later ownership checks. Classic techniques include backdoor attacks (Adi et al., 2018; Li et al., 2019; Zhang et al., 2018), digital signatures and hash functions (Guo & Potkonjak, 2018; Zhu et al., 2020). For language models, two common forms are: *text watermarks*, which bias generation or insert predefined patterns to encode hidden information (Kirchenbauer et al., 2023; Xu et al., 2024; Nasery et al.); and *model weight watermarks*, which embed identifiers into parameters or link them to secret triggers through fine-tuning (Luan et al., 2025; Li et al., 2023). Although backdoor-style fingerprints are relatively straightforward and may persist after moderate fine-tuning (e.g., Dasgupta et al. 2024), these invasive schemes require control of the training process, making them unsuitable for retroactively marking third-party models.

Passive LLM Fingerprinting. In contrast, passive fingerprinting identifies models by analyzing their intrinsic properties without any modification. Passive fingerprinting techniques vary by model access. With white-box access, signatures are extracted from model weights, leveraging intrinsic properties like the distribution of attention matrices (Yoon et al., 2025), the kernel alignment of internal representations (Zhang et al., 2024), or the stable direction of parameter vectors. In the black-box setting, fingerprinting relies on analyzing input-output behavior. These methods use crafted queries (Pasquini et al., 2024), unique prompt-response pairs (Tsai et al., 2025), stylometry (Alhazbi et al., 2025) or iterative prompting–response games (Iourovitski et al., 2024) to identify a model, though they can be less robust to fine-tuning. However, these methods define their signatures *post hoc*. Specifically, they identify emergent properties from a completed training process, rather than the innate, "*Galtonian*" fingerprints present from random initialization that our work seeks to discover.

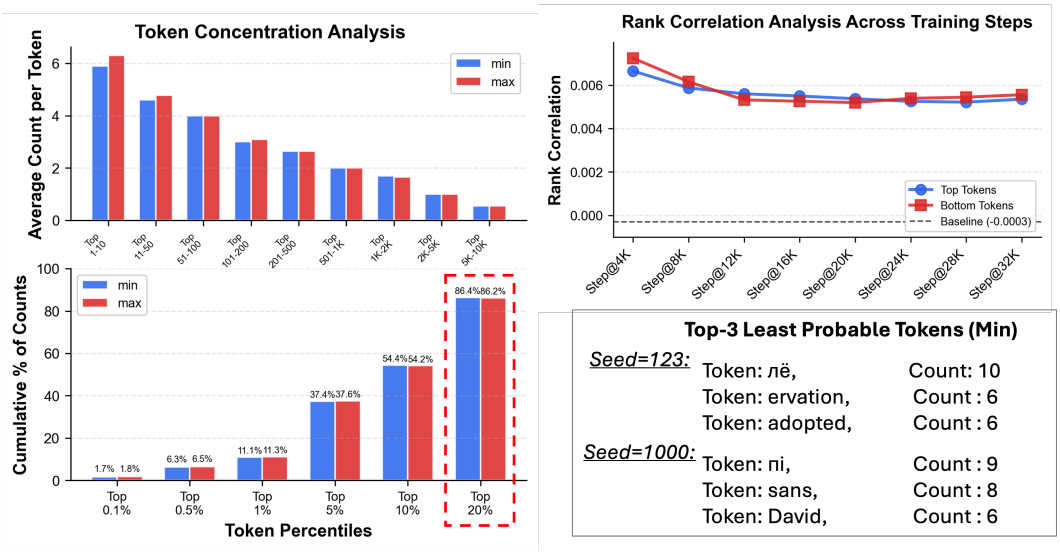

Figure 1: **Initialization-born token bias persists through training.** *Upper Left:* With uniform random inputs, a randomly initialized LLaMA-2–style model assigns highly non-uniform next-token probabilities, concentrating on a subset of tokens. *Lower Left:* An 80/20 coverage pattern: roughly 20% of tokens are selected for the next-token of 80% inputs. *Upper Right:* During training, these within-set preference remain aligned with initialization; rank correlations between checkpoints and the initial model stay above a random baseline. *Lower Right:* The set of preferred (biased) tokens depends on the initialization seed.

3 Biases Originating from Initialization Persist After Training

In this section, we present our key observation that language models exhibit strong token-level biases originating from the random initialization seed. Remarkably, such biases are stable and persistent, remaining detectable even after training.

Initialization creates token-level biases In Figure 1 (left), we examine a LLaMA-2–style model initialized with seed 123. We evaluate 10,000 random input sequences, each 1,024 tokens long with tokens drawn uniformly at random from the vocabulary. Surprisingly, despite the uniform inputs, the initialized model exhibits strong token-selection bias: it does not predict the next token (the 10,001st token following the inputs) uniformly at random; instead, certain tokens receive substantially higher probability than expected. To broaden this observation, we repeat the experiment with different initialization seeds and across different GPU types, and observe the same phenomenon: the overall magnitude of bias is consistent, and for a fixed seed the set of most biased tokens remains stable. However, the particular tokens that are favored (or disfavored) depend on the seed, as shown in the lower-right panel of Figure 1. Notably, an analogous pattern holds for the least-probable tokens (i.e., the argmin of next token probability). We refer to these high- and low-bias tokens as *identity tokens*, as they provide the strongest seed-specific signature at initialization.

Training reshapes but does not erase those seed-specific token-level preferences However, naively measuring bias by frequency is unreliable throughout the training, as the training will severely changes the next-token prediction behaviors. We ask whether identity tokens retain bias profiles that are deeply embedded in the model across training, like a lifelong fingerprint. To investigate, we examine the prediction distribution of each biased token over random sequences. Our expectation is that, since biased tokens each have specific random sequences where they are particularly likely or unlikely to be predicted, the relative preferences among tokens within such identity token set will persist to some extent across all offspring models. For example, if a token is much less preferred than other fingerprint tokens for a given random sequence, it will likely remain less preferred even after training. To test this, in Figure 1 (upper-right), for each checkpoint, we feed the same 10,000 random sequences as in the Figure 1 (left). For every identity token, we extract the within-set probability assigned to that token across these 10,000 random trials. We then compute the rank-based correlation between the probability vector from the initialized model and that from a checkpoint model trained from it. We report the average correlation across each selected token set. Across all token sets, we observe that although the absolute correlation values are small (≈ 0.006) because training mostly

reshapes token probabilities, the correlation remains consistently above the random baseline. It will converge to a stable value rather than decay to the random baseline. This indicates that initialization leaves weak but statistically reliable, idiosyncratic fingerprints (on the prediction distribution of each biased token over random sequences).

4 ALGORITHM

To bridge the gap between the observed token-level bias and the "Galtonian" LLM fingerprints, we propose a statistical detection algorithm grounded in the observations of Section 3. Specifically, we (i) extract the set of *identity indices*[1] based the model outputs; and (ii) assess whether the preference distributions on these identity indices between a base model and a suspicious model are correlated or not, by comparing with an uncorrelated baseline. The design is in order to reflect the premise: *token-level biases on random inputs encode an intrinsic model-specific signature*.

Extract identity indices To obtain more stable *identity indices*, we generate n pseudo-token-sequences by randomly and freshly initializing their token vectors. This differs slightly from sampling tokens from the existing vocabulary to form input sequences. The advantage is that it eliminates cases where the training data, by coincidence or design, contains pieces or subsequences of the test inputs—situations in which the model may recall memorized text and bias the generated tokens.

We treat each input as a next-token prediction trial and extract the *identity indices*, i.e., the output dimensions that consistently receive the lowest scores across random trials. Let $X = \{x_i\}_{i=1}^{n}$, where each $x_i \in \mathbb{R}^{\ell \times d}$ is a pseudo-token-sequence (i.e., simulated random embedding) of length ℓ in a d-dimensional embedding space. For any model g, define the mean output vector $\bar{g} := \frac{1}{n}\sum_{i=1}^{n} g(x_i) \in \mathbb{R}^{d_{\text{out}}}$, where g is either the base model f or the suspect model f', and d_{out} denotes the output dimensionality (vocabulary size for logits, or hidden size for the final hidden state).[2] We extract the m coordinates with the smallest mean values as the identity indices. Formally,

$$\mathcal{M} = \underset{J \subseteq \{1,\ldots,d_{\text{out}}\},\, |J|=m}{\arg\min} \sum_{j \in J} \bar{g}_j, \tag{1}$$

So $\mathcal{M}$ contains exactly the indices of the m smallest entries of $\bar{g}$. Although both the most- and least-preferred dimensions (respectively, argmax and argmin) can be informative according to Figure 1, cross-entropy training typically increases the probability mass on the true class while decreasing it on non-true classes. Averaged over random inputs, the down-weighted (non-true) dimensions therefore exhibit more stable behavior. We thus focus on the most *unwilling* indices (the argmin set) to obtain more stable and reliable detections.

Distribution correlation test A naive way to compare a base model f and a suspect model f' is to measure the overlap of their identity–index sets, but this throws away magnitude/ranking information and is brittle to extraction noise. Motivated by Figure 1 (right)—which shows that relative preferences over identity indices persist after training—we instead test whether the two models exhibit *correlated preferences* on these indices across random inputs.

Let $\mathcal{T} := \mathcal{M}_f \cap \mathcal{M}_{f'} = \{t_1, \ldots, t_k\}$ be the intersection of their identity–index sets with size k. For each input x_i and index t_j, record the (logit or hidden) output as $P_{ij}^{(f)} := [f(x_i)]_{t_j}$ and $P_{ij}^{(f')} := [f'(x_i)]_{t_j}$. This forms two output matrices $P^{(f)}, P^{(f')} \in \mathbb{R}^{n \times k}$. For each t_j, we compute a rank-based (Kendall–Tau) correlation $\rho_j = \text{KendallTau}\big(P_{:,j}^{(f)}, P_{:,j}^{(f')}\big)$, yielding k empirical correlation observations. If these correlations are consistently significant relative to a null of no association (tested against an uncorrelated baseline), we deem f' to be derived from f. We declare significance at $p < 0.01$; further details are given in Algorithm 1.

Design choices in algorithm We next explain several design decisions in Algorithm 1, including the use of softmax, the correlation measure, and the hypothesis test. Since model outputs can vary substantially across different training stages, and our focus is only on the relative probabilities over

[1] An extension of identity tokens, designed to move beyond logits or probability outputs.

[2] Using the final hidden representation instead of logits avoids detokenization noise and is more robust to the rare case that a random sequence appears in the training data.

Algorithm 1 Distribution Correlation Test on Identity Indices

Require: base model f, suspicious model f'; random input $X = \{x_i\}_{i=1}^{n}$; fingerprint size m; significance level $\alpha = 0.01$

▶ ***Capture bias through correlation***

// Extract m indices with the smallest average model output value as identity indices set

1: **for** model $g \in \{f,\ f'\}$ **do**
2: $\quad g(X) \leftarrow [g(x_1), \ldots, g(x_n)]^\top \in \mathbb{R}^{n \times d_{\text{out}}}$ (get model outputs on all random inputs)
3: $\quad \mathcal{M}(g) \leftarrow \arg\min_{J \subseteq \{1,\ldots,d_{\text{out}}\},\, |J|=m} \sum_{j \in J} \bar{g}_j$ (Equation (1)).
4: **end for**

// Get intersection of the two identity indices sets

5: $T \leftarrow \mathcal{M}(f) \cap \mathcal{M}(f') = \{t_1, \ldots, t_k\}$

// Retrieve model outputs on this intersection

6: **for** $g \in \{f, f'\}$ **do**
7: $\quad Z^{(g)} \leftarrow g(X)[T] \in \mathbb{R}^{n \times k}$ (restrict outputs to T)
8: $\quad P^{(g)} \leftarrow \text{softmax}(Z^{(g)}) \in \mathbb{R}^{n \times k}$ (compute the relative probability over T)
9: **end for**

// Compute the per-index correlation between models

10: $\mathcal{T} \leftarrow \left\{ \text{KendallTau}\left(P^{(f')}_{:,j},\ P^{(f)}_{:,j}\right) : j = 1, \ldots, k \right\}$ (rank-based correlation)

▶ ***Use hypothesis test for detection***

// Build an uncorrelated random baseline $\mathcal{T}_{\text{null}}$

11: **for** $g \in \{1, 2\}$ **do**
12: $\quad$ Randomly sample $Z^{(g)}_{\text{rand}} \sim \mathcal{N}(0, I_k)^{n \times k}$
13: $\quad P^{(g)}_{\text{rand}} \leftarrow \text{softmax}(Z^{(g)}_{\text{rand}}) \in \mathbb{R}^{n \times k}$ (row-wise)
14: **end for**
15: $\mathcal{T}_{\text{null}} \leftarrow \left\{ \text{KendallTau}\left(P^{(1)}_{\text{rand},:,j},\ P^{(2)}_{\text{rand},:,j}\right) : j = 1, \ldots, k \right\}$

// Perform a one-sided hypothesis test

16: $H_0 : \mathcal{T} = \mathcal{T}_{\text{null}}, \quad H_1 : \mathcal{T} > \mathcal{T}_{\text{null}}$

Return SameLineage $\leftarrow \mathbf{1}(p\text{-value} < \alpha)$

the identity index set, we apply a softmax normalization in lines 8 and 13 to preserve the relative probability relationships. For the correlation measure, we use the standard Kendall–Tau correlation, a commonly used rank-based measure. This is a natural choice because absolute probability values are less reliable across models, whereas the bias profiles are mainly characterized by the pairwise concordance and discordance of rankings. Finally, regarding the hypothesis test, we found our method to be robust to the specific choice of test. In Section 5, we report results using both the one-sided t-test and the Mann–Whitney U-test.

5 Experiments

This section has two parts. In Section 5.1, we demonstrate that our method is a genuine fingerprint: (i) it enables ***birth verification at the seed level***, whereas prior methods primarily track training-data distributions and lack this capability; and (ii) it remains ***verifiable across the entire training lifecycle***, while prior approaches are reliable only after convergence—once the parameters have encoded stable, data-dependent signals. We report results using both *logits* and *hidden state* outputs, and we test with both the one-sided t-test (t-test) and Mann–Whitney U test (U test). The consistency across tests indicates that our method is robust to the choice of hypothesis test. In Section 5.2, we evaluate all methods under practical infringement scenarios using *LeafBench* (Shao et al., 2025). Across 58 distinct model instances spanning 6 representative post-development techniques, our method matches the best baselines in post-training settings and remains robust to diverse deployment transformations.

Baselines We mainly consider four passive fingerprinting baselines (weight- or representation-based). ***Intrinsic fingerprint*** (Yoon et al., 2025) (or *PDF* in some papers) compares models via the similarity of the layerwise standard-deviation profiles of attention parameters. ***REEF*** (Zhang et al., 2024) computes centered-kernel-alignment (CKA) similarity between feature representations from the same samples across two models. ***PCS*** and ***ICS*** (Zeng et al., 2024) (or collectively as *HuRef* in some papers) are weight-similarity methods: PCS flattens all parameters and measures cosine similarity; ICS forms invariant terms from the weights and measures cosine similarity on those invariants.

Table 1: Comparison of fingerprint behaviors between models initialized with different seeds.

Seed Pair	Logits Output		Hidden State	
	t-test	*U-test*	*t-test*	*U-test*
s_{42} vs. s_{2000}	0.404	0.456	0.357	0.532
s_{123} vs. s_{42}	0.214	0.295	0.678	0.565
s_{1000} vs. s_{123}	0.219	0.246	0.363	0.335
s_{2000} vs. s_{1000}	0.282	0.291	0.434	0.481

Table 2: Trained models share the same fingerprint behaviors as their initialization (p-value < 0.01).

Model Pair	Logits Output		Hidden State	
	t-test	*U-test*	*t-test*	*U-test*
s_{42}^{init} vs. s_{42}^{base}	3.33e-3	1.02e-3	2.20e-8	6.28e-8
s_{123}^{init} vs. s_{123}^{base}	2.06e-3	7.33e-3	7.09e-6	1.37e-5
s_{1000}^{init} vs. s_{1000}^{base}	2.44e-3	4.14e-3	5.58e-4	2.81e-3
s_{2000}^{init} vs. s_{2000}^{base}	5.63e-3	6.76e-3	4.00e-10	1.27e-9

Table 3: Fingerprint persistence under continual training on diverse datasets (base model: seed 1000, corpus `openwebtext`). U-test refers to the Mann–Whitney U test.

Setting	Ours (logits)		Ours (hidden)		Baselines			
Continual corpus (seed)	***t*-test**	***U*-test**	***t*-test**	***U*-test**	**Intrinsic**	**REEF**	**PCS**	**ICS**
`TinyStoriesV2_cleaned` (1000)	0✓	0✓	0✓	7.77e-89✓	1.000✓	0.759×	0.999✓	0.996✓
`TinyStoriesV2_cleaned` (123)	1.000✓	1.00✓	0.943✓	0.902✓	0.950×	0.658✓	0.332✓	0.012✓
`the_stack` (1000)	0✓	1.73e-287✓	0✓	3.09e-69✓	0.489×	0.557×	0.585×	0.123×
`the_stack` (123)	0.616✓	0.479✓	0.732✓	0.831✓	0.445✓	0.580✓	0.301✓	0.026✓

In Section 5.2, we additionally include a (weaker) gradient-based fingerprint ***Gradient*** (Shao et al., 2025). Following Zhang et al. (2024), we use a 0.8 similarity threshold for binary decisions.

Note, in all experiment tables, cell colors indicate lineage: with green denotes models from the same source, and red denotes different sources. For example, s_{42}^{init} vs. s_{42}^{base} compares a model initialized with seed 42 and its continued-pretrained counterpart, hence green. By contrast, s_{2000}^{init} vs. s_{42}^{base} compares a seed-2000 initialization with a model trained from a seed-42 initialization, hence red. Additionally, ✓ denotes a correct detection, while × denotes an error.

5.1 Birth-to-Lifecycle "Biometric" Fingerprinting

We train 12-layer, 12-head LLaMA-style models (Touvron et al., 2023) with RoPE (Su et al., 2021) and Qwen-style models (Team, 2024) from scratch. Because our random baseline is stochastic, we report p-values averaged over 10 independent trials and adopt a significance level of $\alpha = 0.01$. Importantly, the absolute magnitude of extremely small p-values is not meaningful: once p falls below numerical and sampling noise (e.g., $< 10^{-20}$), values like 10^{-260} should not be interpreted as stronger evidence than 10^{-20}—both decisively reject the null. In the main paper, we present results for LLaMA-style models and defer those for Qwen-style models to Section B.3. The overall conclusions are consistent across the two.

Different initialization seeds produce distinct fingerprints Table 1 reports p-values from our correlation tests between pairs of models initialized with different random seeds (42, 123, 1000, and 2000) under the t-test and U-test. All p-values are consistently > 0.01, indicating that our method reliably distinguishes models trained from different seeds. This shows that distinct seeds yield distinct fingerprint behaviors, allowing models to be separated "at birth."

Training preserves the initialization fingerprint. Table 2 compares each initialization model s^{init} with its descendant s^{base} trained on the OpenWebText dataset (Gokaslan et al., 2019) (≈10B tokens). Across all seed–model pairs, p-values are consistently < 0.01, indicating that their bias profiles remain strongly correlated and thus share a common lineage. In short, the trained model inherits the same fingerprint as its initialization. We also evaluate baseline methods (results in Appendix 8); without exception, they fail to distinguish across seeds, which in turn suggests their separability stems from training-induced artifacts rather than initialization.

Figure 2: Fingerprint verifies lineage at every checkpoint (p-values < 0.01).

Table 4: The same dataset and training order do not shape fingerprint behaviors to be identical across different initializations.

Model Pair	Logits Output		Hidden State	
	t-test	*U-test*	*t-test*	*U-test*
s_{123}^{init} vs. s_{1000}^{base}	0.484	0.500	0.385	0.486
s_{1000}^{init} vs. s_{2000}^{base}	0.946	0.956	0.035	0.096
s_{42}^{init} vs. s_{123}^{base}	0.598	0.589	0.426	0.337
s_{2000}^{init} vs. s_{42}^{base}	0.756	0.781	0.388	0.287

Table 5: Fingerprinting results *vs.* LLaMA-2-7B. Each row compares a target model against LLaMA-2-7B. ***U*-test** p reports the p-value from our hidden-state correlation test (< 0.01 indicates a strong signal). **Intrinsic**, **REEF**, **PCS**, and **ICS** report similarity scores (higher = better).

Model	# Tokens	*U*-test p (< 0.01)	Intrinsic ↑	REEF (↑)	PCS (↑)	ICS (↑)
Llama-2-finance-7B (Heenan, 2023)	5M	1.34×10^{-41} ✓	1.0000 ✓	0.9950 ✓	0.9979 ✓	0.9952 ✓
Vicuna-1.5-7B (Chiang et al., 2023)	370M	1.49×10^{-96} ✓	1.0000 ✓	0.9985 ✓	0.9985 ✓	0.9949 ✓
Wizardmath-7B (Luo et al., 2023)	1.8B	4.09×10^{-100} ✓	1.0000 ✓	0.9979 ✓	1.0000 ✓	0.9994 ✓
Meditron-7B (Chen et al., 2023)	48B	5.212×10^{-4} ✓	0.9990 ✓	0.9978 ✓	1.0000 ✓	0.9817 ✓
CodeLlama-7B (Meta AI, 2024)	500B	2.008×10^{-3} ✓	0.9480 ✓	0.9947 ✓	0.6863 ×	0.3369 ×

Identical data and order do not make fingerprints converge In Table 4, all four "suspicious" models s_i^{base} for $i \in \{42, 123, 100, 2000\}$ are trained on *exactly the same corpus (OpenWebText) and in the same data order* (we fix the training seed to lock the data order); the only difference lies in their initialization seeds i. Across all cross-seed pairs, p-values remain consistently > 0.01, in sharp contrast to the near-zero values in Table 2. That is, fingerprints remain seed-specific even under identical data and curriculum.

Continual training on diverse datasets does not confound the fingerprint The purpose of our earlier experiments is solely to demonstrate the strengths of our SeedPrints: it can act as a biometric fingerprint. From a copyright perspective, the inability of prior works to distinguish models with different seeds is not a weakness, since initialization seeds have no clear copyright status. The real fragility is that their attribution can be easily misled by *data distribution*, ***failing to recognize lineage when the training distribution shifts substantially during continued training.***

In Table 3, we continue training a base model (seed 1000, pretrained on OpenWebText Gokaslan et al. (2019)) on two very different datasets: TinyStories Eldan & Li (2023) (synthetic children's stories) and The Stack Kocetkov et al. (2022) (permissively licensed GitHub code). We compare (i) true descendants trained from the base, versus (ii) *distractors* derived from a different base model (initialized with seed 123 and trained with a different data order on OpenWebText), then continued training on the *same* corpus. The question is whether attribution methods can identify which descendant truly shares lineage with the base.

We find that prior baselines all fail under the code setting (The Stack), misclassifying true descendants as distractors. This indicates that they largely track domain similarity rather than lineage identity: TinyStories is closer in distribution to the pretraining corpus (OpenWebText), while The Stack diverges sharply; such a large distribution shift can easily bypass detection. In contrast, our method correctly attributes lineage across both corpora. Hence, our fingerprint is not a proxy for data distribution: it survives substantial domain shift and persists beyond the initial pretraining stage.

All-stage verifiable fingerprints Our fingerprint enables verification at any training stage (Figure 2). We treat each intermediate checkpoint of a model trained on OpenWebText as the base and test whether our method can reliably identify its offspring. All variants consistently recognize the suspect model as belonging to the same lineage, with p-values remaining below the 0.01 threshold.

We further compare our method with existing baselines under standard evaluations on pretrained models. In particular, we test suspect models fine-tuned from Llama-2-7b (base model) with data volumes ranging from 5 million to 500 billion tokens. The suspects include diverse downstream variants such as Llama-2-finance-7b (Heenan, 2023), Vicuna-1.5-7b (Chiang et al., 2023), WizardMath-7b (Luo et al., 2023), Chinese-LLaMA-2-7b (Chen et al., 2023), and Code-Llama-7b (Meta AI, 2024). Their fine-tuning data volumes are 5M, 370M, 1.8B, 13B, and 500B tokens, respectively. As shown in Table 5, our method consistently maintains $p < 0.01$ across all settings.

5.2 Robustness under Realistic Deployments

Real-world deployments routinely apply *parameter-altering* adaptations to foundation models (e.g., fine-tuning, PEFT, quantization), which can weaken or distort fingerprint signals. Our method is designed to remain effective in this regime. To evaluate its reliability under such realistic conditions, we adopt *LeafBench* (Shao et al., 2025), a benchmark for language-model copyright auditing.

Across Source Models. We consider two types of source models for auditing: *Pre-Trained (PT)* and *Instruction-Tuned (IT)*. For each source type, the task is to distinguish *derivative* models (obtained by post-training adaptations) from *independent* models (trained without using the source weights). This mirrors common auditing scenarios where one must test lineage claims for either a PT base or its IT variant. For extact models,Our evaluation covers **58** models in total; the full list of model names and HuggingFace repositories is provided in Appendix C.

Metrics. We evaluate model detection performance using two primary metrics: (1) the Area Under the ROC Curve (AUC) and (2) the Kolmogorov-Smirnov (KS) statistic (Berger & Zhou, 2014), which measures the maximum distance between the score distributions of derivative and source models. Our method produces per-model p-values from statistical tests, which differ fundamentally from the similarity scores used by baseline methods. While similarity scores are linear measures where differences have proportional meaning, p-values are tail probabilities indicating the rarity of an observation under the null hypothesis—they cannot be interpreted on a linear scale. To enable comparison with baseline methods that compute AUC from similarity scores, we deliberately evaluate our method under an unfair condition by converting our p-values into scores using $s = 1 - p$. Despite this unfavorable conversion, our approach still achieves comparable or superior performance. For evaluation, we (i) compute AUC for all methods by sweeping thresholds over the scores s (for our method, the optimal threshold obtained by sweeping also lies at the confidence level $1 - \alpha = 0.99$), and (ii) report the KS statistic as a threshold-free measure of distributional separability. This protocol ensures compatibility with baseline pipelines while preserving the statistical interpretability of our method. We do not compute Manhattan distance, as the non-linear nature of p-values makes absolute-value distances between them mathematically meaningless.

Note, in most practical scenarios, calibrating an optimal threshold is infeasible, preventing the application of similarity-based methods. While we convert our p-values into similarity scores and report AUC for comparability, our method fundamentally differs: it provides a direct statistical test. Specifically, given any pair of models, our approach outputs a p-value that allows a definitive claim about whether the two models originate from the same lineage without relying on ambiguous thresholds and thus ensuring reliable verification.

Parameter-Altering Techniques. To assess robustness under weight modifications, we evaluate models produced via (1) Instruction tuning (Instruct), (2) General-purpose fine-tuning (Finetune), (3) Parameter-efficient fine-tuning (PEFT), (4) Quantization, (5) Model merging (Merge), and (6) Distillation.

5.2.1 Family-wise Effectiveness

There are six model families in total. For each family, we treat both its pretrained base model (PT) and its instruction-tuned model (IT) as separate source models and compare them with all other models in the pool of 58. The results are reported in Table 6. Note that each column corresponds to one family (AUC is then computed over the corresponding tests, using a family-calibrated threshold), whereas the overall score is computed across all test pairs with a global threshold. Thus, the overall score is not equal to the average of the per-family scores. All three methods—*Intrinsic*, *ICS*, and our *SeedPrints*—are essentially saturated across all alterations (overall AUC of 0.995, 0.994, and 0.992, respectively; KS of 0.989, 0.943, and 0.992). For these methods, the family-wise scores are, in most cases, the full score. By contrast, the remaining two baselines, *REEF* and *Gradient*, perform

Table 6: Performance comparison of LLM fingerprinting methods across different source models. "PT" and "IT" refer to using the pre-trained models and instruction-tuned models as source models, respectively. Note that the "Overall" scores are computed over all test combinations, rather than by averaging per-family results.

Method	Metric	Qwen2.5-7B		Qwen2.5-14B		Llama3.1-8B		Mistral-7B		Gemma2-2B		Llama2-7B		Overall
		PT	IT	PT	IT	PT	IT	PT	IT	PT	IT	PT	IT	
REEF	AUC ↑	0.796	0.862	0.947	0.913	1.000	0.999	0.997	0.997	0.963	0.968	0.750	0.706	**0.914**
	KS Statistic ↑	0.492	0.633	0.875	0.735	0.987	0.987	0.975	0.980	0.950	0.837	0.750	0.694	**0.739**
Gradient	AUC ↑	0.657	0.709	0.763	0.756	0.897	0.885	0.872	0.641	1.000	1.000	0.650	0.722	**0.778**
	KS Statistic ↑	0.354	0.385	0.500	0.550	0.646	0.662	0.725	0.425	1.000	1.000	0.275	0.475	**0.4828**
ICS	AUC ↑	1.000	1.000	0.997	0.997	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	**0.994**
	KS Statistic ↑	1.000	1.000	0.975	0.979	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	**0.943**
Intrinsic	AUC ↑	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	**0.995**
	KS Statistic ↑	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	**0.989**
SeedPrints	AUC ↑	0.999	1.000	1.000	1.000	1.000	0.929	1.000	1.000	1.000	1.000	1.000	1.000	**0.992**
	KS Statistic ↑	0.985	1.000	1.000	1.000	1.000	0.923	1.000	1.000	1.000	1.000	1.000	1.000	**0.992**

Table 7: Performance of LLM fingerprinting methods across different parameter-altered techniques.

Method	Metric	Instruct	Finetune	PEFT	Quantization	Merge	Distillation	Overall Avg
REEF	AUC ↑	0.996	0.873	0.994	0.871	0.954	0.858	**0.924**
	KS Statistic ↑	0.949	0.637	0.940	0.812	0.851	0.615	**0.801**
Gradient	AUC ↑	0.864	0.745	0.724	0.697	0.827	0.840	**0.783**
	KS Statistic ↑	0.733	0.469	0.463	0.537	0.550	0.650	**0.567**
ICS	AUC ↑	1.000	0.987	1.000	1.000	0.981	1.000	**0.995**
	KS Statistic ↑	1.000	0.925	1.000	1.000	0.941	1.000	**0.978**
Intrinsic	AUC ↑	1.000	0.981	1.000	1.000	1.000	1.000	**0.997**
	KS Statistic ↑	1.000	0.971	1.000	1.000	1.000	1.000	**0.995**
SeedPrints	AUC ↑	1.000	0.970	1.000	1.000	1.000	1.000	**0.995**
	KS Statistic ↑	1.000	1.000	0.969	1.000	1.000	1.000	**0.995**

substantially worse. These results indicate that, while our method is primarily designed to capture intrinsic seed-level fingerprints, it effectively functions as a biometric-like fingerprint that provides reliable and persistent identity tracking across diverse model families.

5.2.2 Alteration-wise Robustness

Across alteration types, *Instruct* is consistently the easiest case (even the weakest baseline, Gradient, can achieve AUC 0.864), whereas the most disruptive are *PEFT*, *Finetune* and *Quantization*: REEF AUC drops by about 0.12 compared to Instruct ($0.996 \rightarrow 0.873/0.871$), and Gradient degrades sharply under *PEFT*/*Finetune* ($0.864 \rightarrow 0.724/0.745$). The effects of *Merge* and *Distillation* are more mixed: REEF (0.954) and Gradient (0.827) are only mildly affected by *Merge*, while ICS shows a small drop (0.981 vs. mostly 1.000). Distillation degrades REEF (AUC 0.858) but leaves other methods largely intact. We attribute these trends to method design: REEF and Gradient depend on fine-grained forward/backward dynamics that are easily perturbed by parameter updates or quantization, whereas Intrinsic, ICS, and our SeedPrints leverage static, initialization-tied signals. Notably, SeedPrints remain near-perfect across all six transformations, showing robustness even under aggressive parameter changes and shifts introduced by *Finetune*, *PEFT*, and *Merge*.

6 Conclusion

In this work, we introduced *SeedPrints*, a stronger, intrinsic notion of LLM fingerprinting that traces a model's lineage back to its random initialization. We showed that initialization itself leaves a persistent "Galtonian" fingerprint on neural language models. Therefore, untrained models already exhibit stable, seed-dependent token-preference patterns, and these biases persist—statistically and measurably—throughout training. Building on this observation, we proposed a distribution-

correlation test over *identity indices* that detects shared lineage with calibrated significance. Across LLaMA- and Qwen-style families and a broad suite of realistic deployments (e.g., continued pre-training on divergent corpora, instruction tuning, PEFT, quantization, merging, and distillation), SeedPrints enables "birth-to-lifecycle" verification and remains robust under domain shift, offering a practical tool for provenance and copyright auditing.

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX security symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

Saeif Alhazbi, Ahmed Hussain, Gabriele Oligeri, and Panos Papadimitratos. Llms have rhythm: Fingerprinting large language models using inter-token times and network traffic analysis. *IEEE Open Journal of the Communications Society*, 2025.

Vance W Berger and YanYan Zhou. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online*, 2014.

Zeming Chen, Alejandro Hernández-Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. Meditron-70b: Scaling medical pretraining for large language models, 2023.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Agnibh Dasgupta, Abdullah Tanvir, and Xin Zhong. Watermarking language models through language models. *arXiv preprint arXiv:2411.05091*, 2024.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Francis Galton. *Finger prints*. Number 57490-57492. Cosimo Classics, 1892.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM international conference on computer-aided design (ICCAD)*, pp. 1–8. IEEE, 2018.

Collin Heenan. Llama2-7b-finance (hugging face model). https://huggingface.co/cxllin/Llama2-7b-Finance, 2023. Fine-tuned from NousResearch/Llama-2-7b-hf; MIT License; accessed 2025-09-02.

Dmitri Iourovitski, Sanat Sharma, and Rakshak Talwar. Hide and seek: Fingerprinting large language models with evolutionary learning. *arXiv preprint arXiv:2408.02871*, 2024.

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.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. The stack: 3 tb of permissively licensed source code. *Transactions on Machine Learning Research (TMLR), Preprint*, 2022.

Peixuan Li, Pengzhou Cheng, Fangqi Li, Wei Du, Haodong Zhao, and Gongshen Liu. Plmmark: a secure and robust black-box watermarking framework for pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 14991–14999, 2023.

Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn. In *Proceedings of the 35th annual computer security applications conference*, pp. 126–137, 2019.

Xiaokun Luan, Zeming Wei, Yihao Zhang, and Meng Sun. Robust and efficient watermarking of large language models using error correction codes. *Proceedings on Privacy Enhancing Technologies*, 2025.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

Meta AI. Codellama-7b-hf: Code llama base 7b model (hugging face). https://huggingface.co/codellama/CodeLlama-7b-hf, 2024. Base 7 billion-parameter Code Llama model for code synthesis and understanding; trained between January and July 2023; licensed under Meta Llama 2 license; accessed 2025-09-02.

Anshul Nasery, Jonathan Hayase, Creston Brooks, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, and Sewoong Oh. Scalable fingerprinting of large language models. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Dario Pasquini, Evgenios M Kornaropoulos, and Giuseppe Ateniese. Llmmap: Fingerprinting for large language models. *arXiv preprint arXiv:2407.15847*, 2024.

Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. *Advances in Neural Information Processing Systems*, 37: 21079–21113, 2024.

Shuo Shao, Yiming Li, Yu He, Hongwei Yao, Wenyuan Yang, Dacheng Tao, and Zhan Qin. Sok: Large language model copyright auditing via fingerprinting. *arXiv preprint arXiv:2508.19843*, 2025.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 115–124. Association for Computational Linguistics, 2021.

Teppei Suzuki, Ryokan Ri, and Sho Takase. Natural fingerprints of large language models. *arXiv preprint arXiv:2504.14871*, 2025.

Qwen Team. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Yun-Yun Tsai, Chuan Guo, Junfeng Yang, and Laurens van der Maaten. Rofl: Robust fingerprinting of language models. *arXiv preprint arXiv:2505.12682*, 2025.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

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

Do-hyeon Yoon, Minsoo Chun, Thomas Allen, Hans Müller, Min Wang, and Rajesh Sharma. Intrinsic fingerprint of llms: Continue training is not all you need to steal a model! *arXiv preprint arXiv:2507.03014*, 2025.

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.

Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia conference on computer and communications security*, pp. 159–172, 2018.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. Reef: Representation encoding fingerprints for large language models. *arXiv preprint arXiv:2410.14273*, 2024.

Ruichong Zhang. Matrix-driven instant review: Confident detection and reconstruction of llm plagiarism on pc. *arXiv preprint arXiv:2508.06309*, 2025.

Renjie Zhu, Xinpeng Zhang, Mengte Shi, and Zhenjun Tang. Secure neural network watermarking protocol against forging attack. *EURASIP Journal on Image and Video Processing*, 2020(1):37, 2020.

A LLM USAGE

We use AI assistants, i.e., ChatGPT and Gemini, for writing and formatting support. Their use covers grammar and style checks, improving clarity of figure and table captions, and other surface-level edits. For programming-related tasks, we occasionally use GitHub Copilot and Claude as coding assistants, e.g., for code auto-completion and debugging hints.

B MORE EXPERIMENT DETAILS

B.1 IMPLEMENTATION DETAILS AND LICENSES

We train all models with the Hugging Face Transformers Trainer (Wolf et al., 2020), using Accelerate (Gugger et al., 2022) for distributed runs. All open-source models are loaded from their official Hugging Face releases and used under their original licenses: Llama models under the Meta Llama Community License, and other models under Apache-2.0. All datasets are downloaded via the Hugging Face Datasets library (the library is Apache-2.0); dataset content follows each dataset's stated license.

B.2 COMPLEMENTARY RESULTS

Table 8: Full results of methods evaluating fingerprints at initialization and after subsequent training.

Model Pair	Logits Output		Hidden State		Baselines			
	t-test	*u-test*	*t-test*	*u-test*	**Intrinsic**	**REEF**	**PCS**	**ICS**
s_{42}^{init} vs. s_{42}^{base}	3.33e-3✓	1.02e-3✓	2.20e-8✓	6.28e-8✓	-0.021×	0.375×	0.580×	0.196×
s_{123}^{init} vs. s_{123}^{base}	2.06e-3✓	7.33e-3✓	7.09e-6✓	1.37e-5✓	0.149×	0.369×	0.581×	0.188×
s_{1000}^{init} vs. s_{1000}^{base}	2.44e-3✓	4.14e-3✓	5.58e-4✓	2.81e-3✓	-0.252×	0.381×	0.581×	0.188×
s_{2000}^{init} vs. s_{2000}^{base}	5.63e-3✓	6.76e-3✓	4.00e-10✓	1.27e-9✓	-0.337×	0.331×	0.581×	0.188×

B.3 QWEN RESULTS

Different initialization seeds produce distinct fingerprints Table 9 presents the p-values from correlation tests on Qwen-style models initialized with different random seeds (42, 123, 1000, 2000), evaluated under both the t-test and U-test. In all cases, the p-values remain above 0.01, confirming that our approach can consistently tell apart models trained from different seeds.

Table 9: Comparison of fingerprint behaviors between models initialized with different seeds for Qwen-style models

Seed Pair	Logits Output		Hidden State	
	t-test	*U-test*	*t-test*	*U-test*
s_{123} vs. s_{1000}	0.741	0.727	0.094	0.074
s_{1000} vs. s_{123}	0.954	0.971	0.125	0.094
s_{42} vs. s_{2000}	0.273	0.360	0.451	0.529
s_{2000} vs. s_{42}	0.215	0.206	0.130	0.095

Training preserves the initialization fingerprint. We also compare each initialization model s^{init} with its corresponding trained version s^{base} on the OpenWebText dataset (Gokaslan et al., 2019) (≈10B tokens) for Qwen, as shown in Table 10. Consistent with the LLaMA-style results, all seed–model pairs yield p-values below 0.01. This indicates that training does not erase the initialization fingerprint; instead, the signature is preserved in the descendant model.

Table 10: Trained models share the same fingerprint behaviors as their initialization models (p-value < 0.01).

Model Pair	Logits Output		Hidden State	
	t-test	U-test	t-test	U-test
s_{123}^{init} vs. s_{123}^{base}	2.38e-03	1.68e-03	7.36e-15	3.38e-13
s_{1000}^{init} vs. s_{1000}^{base}	1.35e-04	4.84e-05	4.41e-13	2.01e-11
s_{42}^{init} vs. s_{42}^{base}	1.39e-03	1.46e-03	1.06e-24	2.05e-19
s_{2000}^{init} vs. s_{2000}^{base}	1.80e-03	1.28e-03	4.87e-24	1.92e-20

Identical data and order do not make fingerprints converge Following the setting in Table 4, we also test whether fingerprint behavior would be erased or confounded by identical data and order for Qwen-style models. Table 11 shows that fingerprints remain seed-specific even under identical data and curriculum.

Table 11: The same dataset and training order do not shape fingerprint behaviors to be identical across different initializations.

Model Pair	Logits Output		Hidden State	
	t-test	U-test	t-test	U-test
s_{123}^{init} vs. s_{1000}^{base}	0.598	0.638	0.286	0.254
s_{1000}^{init} vs. s_{123}^{base}	0.804	0.805	0.036	0.040
s_{42}^{init} vs. s_{2000}^{base}	0.589	0.608	0.026	0.043
s_{2000}^{init} vs. s_{42}^{base}	0.523	0.482	0.112	0.123

Continual training on diverse datasets does not confound the fingerprint Table 12 reports the same test results as Table 3 for Qwen. Our method remains stable and reliable across all cases (even under severe training distribution shifts), whereas many baselines become confused and produce errors.

Table 12: Fingerprint persistence under continual training on diverse datasets (base model: seed 1000, corpus `openwebtext`). U-test refers to the Mann–Whitney U test.

Setting	Ours (logits)		Ours (hidden)		Baselines			
Continual corpus (seed)	t-test	U-test	t-test	U-test	Intrinsic	REEF	PCS	ICS
`TinyStories` (1000)	$5.36e-09$ ✓	$1.92e-07$ ✓	$8.49e-214$ ✓	$5.09e-71$ ✓	1.000 ✓	0.957 ✓	0.999 ✓	0.996 ✓
`TinyStories` (123)	0.434 ✓	0.433 ✓	0.256 ✓	0.065 ✓	0.913 ✗	0.199 ✓	0.328 ✓	0.039 ✓
`the_stack` (1000)	0 ✓	$4.16e-237$ ✓	$1.16e-211$ ✓	$2.30e-76$ ✓	0.999 ✓	0.313 ✗	0.995 ✓	0.976 ✓
`the_stack` (123)	0.999 ✓	0.993 ✓	0.610 ✓	0.491 ✓	0.916 ✗	0.255 ✓	0.328 ✓	0.038 ✓

All-stage verifiable fingerprints Our fingerprinting method on Qwen also demonstrates verifiability at all training stages (Figure 3). Across all variants, the suspect model is consistently recognized as belonging to the same lineage, with p-values remaining below the 0.01 threshold.

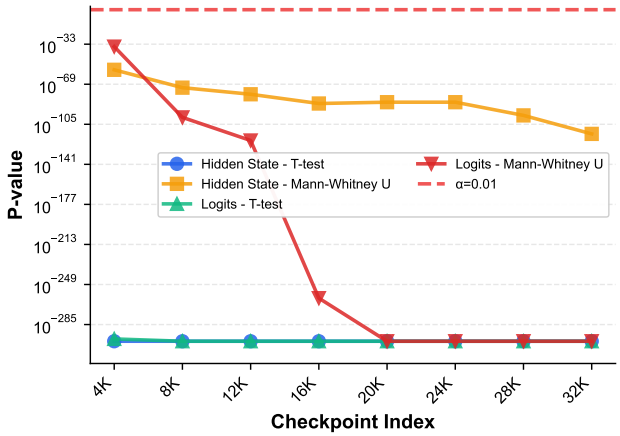

Figure 3: Fingerprint verifies lineage at every checkpoint (p-values < 0.01) for Qwen structure model.

C Model Catalog and Decoding Settings

Our evaluation covers **58** models in total. Below we document the decoding configuration used throughout all runs and enumerate the model catalog grouped by family. For each derived model we also indicate the post-training transformation (*finetune*, *adapter*, *merge*, *quantization*, *distillation*). "PT" and "IT" refer to *pre-trained* and *instruction-tuned* source models, respectively.

Parameter	Value
max_new_tokens	512
temperature	0.7
top_p	0.9
top_k	50
do_sample	true
max_input_length	512

C.1 Model list by family

C.1.1 Qwen-2.5-7B

Variant	HuggingFace repo	Type
PT	Qwen/Qwen2.5-7B	—
IT	Qwen/Qwen2.5-7B-Instruct	—
Derived	Qwen/Qwen2.5-Math-7B	finetune
Derived	Qwen/Qwen2.5-Coder-7B-Instruct	finetune
Derived	WangCa/Qwen2.5-7B-Medicine	finetune
Derived	huihui-ai/Qwen2.5-7B-Instruct-abliterated-v2	finetune
Derived	Locutusque/StockQwen-2.5-7B	merge
Derived	bunnycore/QevaCoT-7B-Stock	merge
Derived	fangcaotank/task-10-Qwen-Qwen2.5-7B-Instruct	adapter
Derived	SeeFlock/task-12-Qwen-Qwen2.5-7B-Instruct	adapter
Derived	Qwen/Qwen2.5-7B-Instruct-GPTQ-Int4	quantization
Derived	Qwen/Qwen2.5-7B-Instruct-GPTQ-Int8	quantization
Derived	Lansechen/Qwen2.5-7B-Open-R1-Distill	distillation

C.1.2 QWEN2.5-14B

Variant	HuggingFace repo	Type
PT	`Qwen/Qwen2.5-14B`	—
IT	`Qwen/Qwen2.5-14B-Instruct`	—
Derived	`Qwen/Qwen2.5-Coder-14B`	finetune
Derived	`oxyapi/oxy-1-small`	finetune
Derived	`v000000/Qwen2.5-14B-Gutenberg-Instruct-Slerpeno`	merge
Derived	`ToastyPigeon/qwen-story-test-qlora`	adapter
Derived	`Qwen/Qwen2.5-14B-Instruct-GPTQ-Int4`	quantization
Derived	`deepseek-ai/DeepSeek-R1-Distill-Qwen-14B`	distillation

C.1.3 LLAMA-3.1-8B

Variant	HuggingFace repo	Type
PT	`meta-llama/Llama-3.1-8B`	—
IT	`meta-llama/Llama-3.1-8B-Instruct`	—
Derived	`ValiantLabs/Llama3.1-8B-Fireplace2`	finetune
Derived	`RedHatAI/Llama-3.1-8B-tldr`	finetune
Derived	`proxectonos/Llama-3.1-Carballo`	finetune
Derived	`mlabonne/Meta-Llama-3.1-8B-Instruct-abliterated`	finetune
Derived	`gaverfraxz/Meta-Llama-3.1-8B-Instruct-HalfAbliterated-TIES`	merge
Derived	`Xiaojian9992024/Llama3.1-8B-ExtraMix`	merge
Derived	`LlamaFactoryAI/Llama-3.1-8B-Instruct-cv-job-description-matching`	adapter
Derived	`chchen/Llama-3.1-8B-Instruct-PsyCourse-fold7`	adapter
Derived	`iqbalamo93/Meta-Llama-3.1-8B-Instruct-GPTQ-Q_8`	quantization
Derived	`DaraV/LLaMA-3.1-8B-Instruct-INT4-GPTQ`	quantization
Derived	`asas-ai/Llama-3.1-8B-Instruct-Open-R1-Distill`	distillation

C.1.4 MISTRAL-7B-V0.3

Variant	HuggingFace repo	Type
PT	`mistralai/Mistral-7B-v0.3`	—
IT	`mistralai/Mistral-7B-Instruct-v0.3`	—
Derived	`KurmaAI/AQUA-7B`	finetune
Derived	`openfoodfacts/spellcheck-mistral-7b`	finetune
Derived	`grimjim/Mistral-7B-Instruct-demi-merge-v0.3-7B`	merge
Derived	`chaymaemerhrioui/mistral-Brain_Model_ACC_Trainer`	adapter
Derived	`RedHatAI/Mistral-7B-Instruct-v0.3-GPTQ-4bit`	quantization
Derived	`eganwo/mistral7b-distilled-from-deepseek-r1-qwen32b`	distillation

C.1.5 GEMMA-2-2B

Variant	HuggingFace repo	Type
PT	`google/gemma-2-2b`	—
IT	`google/gemma-2-2b-it`	—
Derived	`rinna/gemma-2-baku-2b`	finetune
Derived	`anakin87/gemma-2-2b-neogenesis-ita`	finetune
Derived	`vonjack/gemma2-2b-merged`	merge
Derived	`google-cloud-partnership/gemma-2-2b-it-lora-sql`	adapter
Derived	`qilowoq/gemma-2-2B-it-4Bit-GPTQ`	quantization
Derived	`Syed-Hasan-8503/Gemma-2-2b-it-distilled`	distillation

C.1.6 LLAMA-2-7B

Variant	HuggingFace repo	Type
PT	`meta-llama/Llama-2-7b-hf`	—
IT	`meta-llama/Llama-2-7b-chat-hf`	—
Derived	`allenai/tulu-2-7b`	finetune
Derived	`QIAIUNCC/EYE-Llama_qa`	finetune
Derived	`DevQuasar/coma-7B-v0.1`	merge
Derived	`Ammar-1/llama2-Better-Tune`	adapter
Derived	`TheBloke/Llama-2-7B-Chat-GPTQ`	quantization
Derived	`cygu/llama-2-7b-logit-watermark-distill-kgw-k1-gamma0.25-delta2`	distillation

Notes. "Type" denotes the post-training transformation relative to the PT/IT source model: *finetune* includes supervised/preference optimization variants; *adapter* includes LoRA/QLoRA-style modules; *merge* includes model soups and TIES-style merges; *quantization* includes GPTQ/INTx variants; *distillation* includes student models distilled from reasoning-augmented teachers.