HuRef: HUman-REadable Fingerprint
for Large Language Models

Boyi Zeng[1], **Lizheng Wang**[2], **Yuncong Hu**[2], **Yi Xu**[2]
Chenghu Zhou[3], **Xinbing Wang**[2], **Yu Yu**[2], **Zhouhan Lin**[1]*
[1]LUMIA Lab, Shanghai Jiao Tong University
[2]Shanghai Jiao Tong University, [3]Chinese Academy of Sciences
boyizeng@sjtu.edu.cn *lin.zhouhan@gmail.com

Abstract

Protecting the copyright of large language models (LLMs) has become crucial due to their resource-intensive training and accompanying carefully designed licenses. However, identifying the original base model of an LLM is challenging due to potential parameter alterations. In this study, we introduce HuRef, a human-readable fingerprint for LLMs that uniquely identifies the base model without interfering with training or exposing model parameters to the public. We first observe that the vector direction of LLM parameters remains stable after the model has converged during pretraining, with negligible perturbations through subsequent training steps, including continued pretraining, supervised fine-tuning, and RLHF, which makes it a sufficient condition to identify the base model. The necessity is validated by continuing to train an LLM with an extra term to drive away the model parameters' direction and the model becomes damaged. However, this direction is vulnerable to simple attacks like dimension permutation or matrix rotation, which significantly change it without affecting performance. To address this, leveraging the Transformer structure, we systematically analyze potential attacks and define three invariant terms that identify an LLM's base model. Due to the potential risk of information leakage, we cannot publish invariant terms directly. Instead, we map them to a Gaussian vector using an encoder, then convert it into a natural image using StyleGAN2, and finally publish the image. In our black-box setting, all fingerprinting steps are internally conducted by the LLMs owners. To ensure the published fingerprints are honestly generated, we introduced Zero-Knowledge Proof (ZKP). Experimental results across various LLMs demonstrate the effectiveness of our method.[1]

1 Introduction

Large language models (LLMs) have become the foundation models in many scenarios of artificial intelligence. As training an LLM from scratch consumes a huge amount of computation and data resources and the trained LLM needs to be carefully protected from malicious use, the parameters of the LLMs become a crucial property to protect, for both commercial and ethical reasons. As a result, many of the LLMs are open-sourced with carefully designed licenses to reject commercial use (Touvron et al., 2023a; Taylor et al., 2022) or requiring an apply-and-approval process (Touvron et al., 2023b; Zhang et al., 2022; Penedo et al., 2023; BaiChuan-Inc, 2023; Team, 2023; Zheng et al., 2023b), let alone some LLMs are not open-sourced entirely (OpenAI, 2022; GPT-4, 2023; Brown et al., 2020; Wu et al., 2023b; Chowdhery et al., 2022; Hoffmann et al., 2022).

---

*Zhouhan Lin is the corresponding author.
[1]The code is available at https://github.com/LUMIA-Group/HuRef.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).
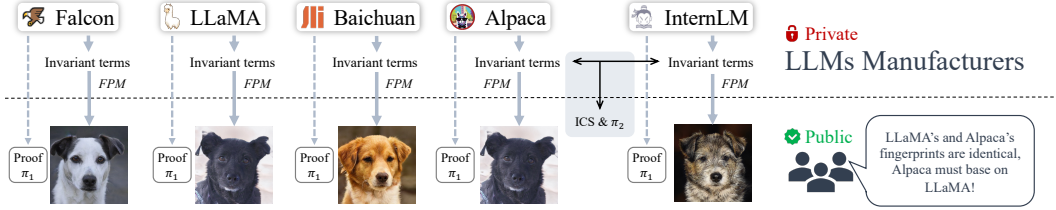
Figure 1: An illustrative framework for LLM protection with fingerprints. The LLM manufacturers compute invariant terms internally and feed them into the fingerprinting model (FPM[2]) to generate a fingerprint image. This image is then released to the public along with zero-knowledge proofs (π_1), allowing for intuitive identification of shared base models through the fingerprint images. We also provide a limited one-to-one quantitative comparison scheme (ICS & π_2) as a complement. Zero-Knowledge Proof guarantees the reliability of the fingerprints and comparison results, without interfering with LLM training or revealing model parameters to the public.

At the core of protecting LLMs from unauthorized use is to identify the base model of a given LLM. However, different from other forms of property such as software or images, protecting LLMs is a novel problem with unique challenges. First, the base model usually needs to be fine-tuned or even continued pretraining to be applied to downstream tasks, resulting in parameter updates that make the resulting model different from the original base model, which makes it disputable to identify the base model. Second, many of the popular LLMs are not releasing their parameters, leaving the identification in a black-box setting. Third, different from previous smaller-scale neural networks that are only trained for specific tasks, LLMs are usually targeted for enormous forms of tasks that are not yet defined during pretraining. This has made the watermarking methods for traditional neural networks (Adi et al., 2018a; Xiang et al., 2021; Yadollahi et al., 2021) not suited in this case, especially under extensive subsequent training steps.

In this work, we propose a novel way to overcome the aforementioned challenges by proposing a method that reads part of the model parameters and computes a fingerprint for each LLM without interfering with training or exposing model parameters to the public. The appearance of the fingerprint is closely dependent on the base model, and invariant to almost all subsequent training steps, including supervised fine-tuning (SFT), reinforcement learning with human feedback (RLHF), or even continue-pretraining with augmented vocabulary in a new language.

The fingerprint is based on our observation that the vector direction of LLM parameters remains stable against various subsequent training steps after the model has converged during pretraining. This makes it a good indicator for base model identification. Empirically, the sufficiency of this correlation is elaborated in Section 3.1.1, while its necessity is presented in Section 3.1.2.

Further, despite its stability towards training, the vector direction of the model parameter is vulnerable to some simple direct weight rearrangements that could significantly change the direction of parameter vectors without affecting the model's performance. We construct three invariant terms that are robust to these weight rearrangements by systematically analyzing possible rearrangements and leveraging the Transformer structure. This is elaborated in Section 3.2.

Moreover, we generate human-readable fingerprints by mapping the invariant terms into a Gaussian random vector through an encoder and then mapping the Gaussian vector to a natural image through an off-the-shelf image generation model, such as StyleGAN2 (Karras et al., 2020). This generation offers a dual benefit of mitigating information leakage and making our fingerprints straightforward to decipher. To ensure the published fingerprints are honestly generated, we also introduced Zero-Knowledge Proof (ZKP). This is elaborated in Section 4.

With this fingerprinting approach, we can sketch an outline for protecting LLMs in Figure 1.

2   Related Works

There are two primary categories of related approaches.

---

[2]FPM is open-sourced, as all LLM manufacturers need to use the same one. We have placed it on the private side in Figure 1 solely because the fingerprinting process is private.

2

Model	Alpaca-Lora	Alpaca	Chinese-LLaMA	Vicuna	Baize	MedAlpaca	Koala	WizardLM	MiniGPT-4	Chinese-Alpaca	Baichuan	OpenLLaMA	InternLM	LLaMA-2
PCS	99.87	99.91	99.68	99.80	99.73	99.90	99.82	99.89	99.70	99.52	0.83	1.16	0.28	1.51
ICS	99.60	99.95	93.57	99.42	99.60	99.86	99.63	99.89	99.20	91.35	0.32	0.32	0.03	3.16
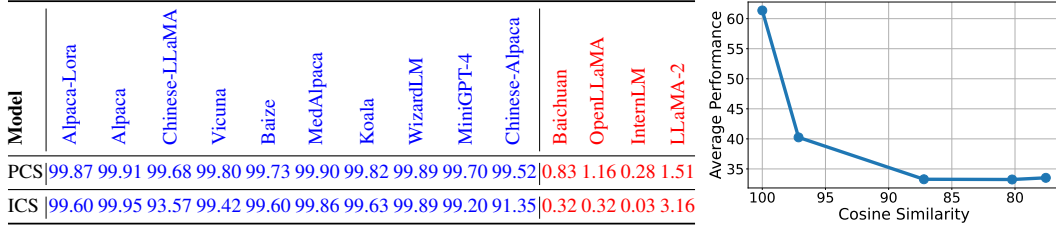
Table 1: The cosine similarities of model parameters (PCS) and invariant terms (ICS) between various LLMs w.r.t. the LLaMA-7B base model. All models are of the same size.

Figure 2: The model's performance quickly deteriorates as the PCS decreases.

Post-hoc Detection methods involve analyzing text generated by LLMs after its production. LLMDet (Wu et al., 2023a) calculates proxy perplexity by leveraging prior knowledge of the model's next-token probabilities. DetectGPT (Mitchell et al., 2023) uses model-predicted probabilities to identify passages generated by a specific LLM. Li et al. (2023) employs perplexity scores and intricate feature engineering. These methods are usually applicable to a specific LLM and could be affected by supervised fine-tuning (SFT) and continued pretraining. More recently, Sadasivan et al. (2023) presented theoretical findings that for highly advanced AI human mimickers, even the best possible detection methods may only marginally outperform random guessing.

Watermarking Techniques can be divided into two main categories (Boenisch, 2021). The first embeds watermarks or related information into the model parameters, such as explicit watermarking scheme (Uchida et al., 2017) or leveraging another neural network (Wang et al., 2020), which could potentially affect model performance (Wang & Kerschbaum, 2019). The second category focuses on inducing unusual prediction behavior in the model. Xiang et al. (2021) explored embedding phrase triggers, and Gu et al. (2022) extended this approach to LLM; however, they are task-specific. Yadollahi et al. (2021) proposed a watermarking method but did not consider subsequent fine-tuning. Christ et al. (2023) proposed a cryptographic approach, but it is not robust to text editing. Kirchenbauer et al. (2023) involved using pre-selected tokens which inevitably alters the model prediction. These methods may turn out to be vulnerable to attacks on certain tokens, for example, Krishna et al. (2023) successfully evaded watermarking (Kirchenbauer et al., 2023), GPTZero (Tian, 2023), and OpenAI's text classifier (OpenAI, 2023) using paraphrasing attacks.

Our work doesn't fall into any of the two categories since it is based on analyzing model weights post-hoc and relies on a wide spectrum of tokens.

In Appendix A, we provide a more extensive discussion of additional related works.

3 Vector Direction of LLM Parameters and the Invariant Terms

3.1 Using Vector Direction of LLM Parameters to Identify the Base Model

We can flatten all weight matrices and biases of an LLM into vectors and concatenate them together into a single huge vector. In this subsection, we are going to show how the direction of this vector could be used to determine the base model by empirically showing its sufficiency and necessity.

3.1.1 Sufficiency

For sufficiency, we compute the cosine similarities between a base model LLaMA-7B and various of its offspring models, as well as other independently pretrained LLMs (Geng & Liu, 2023) that are of the same size. Table 1 shows a wide spectrum of models that inherit the LLaMA-7B base model, whose subsequent training processes involve various training paradigms, such as SFT (Taori et al., 2023; Xu et al., 2023a; Zheng et al., 2023a; Geng, 2023; Xu et al., 2023b; Han et al., 2023), SFT with LoRA (Wang, 2023) and extensive continued pretraining in a new language (Cui et al., 2023), extending to new modalities (Zhu et al., 2023), etc. We detail the subsequent training settings of these models in Appendix Table 10.

Regardless of their various subsequent training setting, we can figure that all of these models show almost full scores in cosine similarity, largely preserving the base model's parameter vector direction.

3

On the other hand, the models that are trained independently appear to be completely different in parameter vector direction, showing almost zero cosine similarity with the LLaMA-7B model.

These observations indicate that a high cosine similarity between the two models highly suggests that they share the same base model, and vice versa.

3.1.2   Necessity

From the necessity perspective, we want to verify if the base model's ability can still be preserved when the cosine similarity is intentionally suppressed in subsequent training steps. To this end, we inherit the LLaMA-7B base model and interfere with the Alpaca's SFT process by augmenting the original SFT loss with an extra term that minimizes the absolute value of cosine similarity. i.e. $L_A = \frac{|\langle \boldsymbol{V}_A, \boldsymbol{V}_{base}\rangle|}{|\boldsymbol{V}_A||\boldsymbol{V}_{base}|}$. Here $\boldsymbol{V}_A, \boldsymbol{V}_{base}$ stand for the parameter vector of the model being tuned and that of the base model, respectively.

Figure 2 presents the average zero-shot performance on a set of standard benchmarks when L_A(PCS) is at different values. The benchmarks include BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-e, ARC-c (Clark et al., 2018), RACE (Lai et al., 2017) and MMLU (Hendrycks et al., 2020). (c.f. Appendix Table 5 for a detailed breakdown of performances on each task.) We can see that despite the original training loss is still present, the model quickly deteriorates to random guesses as the cosine similarity detaches away from that of the base model.

These observations indicate that it is fairly hard for the model to preserve the base model's performance without keeping a high cosine similarity to it.

3.2   Deriving the Invariant Terms

Although the vector direction of model parameters is shown to closely stick to its base model, directly comparing the vector direction through cosine similarity requires both models to reveal their parameters, which is unacceptable in many cases. In addition, apart from training, parameter vector direction is vulnerable to some attacks that directly rearrange the model weights. For example, since the hidden units in a model layer are permutation-invariant, one can easily alter the parameter vector direction by randomly permuting the hidden units along with the weights wired to the units.

These attacks are invisible to discover since they could easily break the cosine similarity but neither change the model structure nor affect the model performance.

In this subsection, we are going to first systematically analyze and formalize possible weight rearrangements by leveraging the structure constraints of the Transformer, and then derive three terms that are invariant under these rearrangements, even when they are combined. Let's first consider the Transformer layer as depicted in Figure 3. Formally, the layer conducts the following computation:

$$\boldsymbol{H}_{Attn}^{'} = \mathrm{softmax}\left(\frac{\boldsymbol{H}_n\boldsymbol{W}_Q(\boldsymbol{H}_n\boldsymbol{W}_K)^T}{\sqrt{d}}\right)\boldsymbol{H}_n\boldsymbol{W}_V\boldsymbol{W}_O \quad (1)$$

Figure 3: Transformer layer

$$\boldsymbol{H}_{n+1}^{'} = \sigma\left(\boldsymbol{H}_{Attn}\boldsymbol{W}_1 + \mathbf{b}_1\right)\boldsymbol{W}_2 + \mathbf{b}_2 \quad (2)$$

where $\boldsymbol{H}_n \in \mathbb{R}^{l\times d}$ is the hidden state of the n-th layer, with l, d being sequence length and model dimensions, respectively. $\boldsymbol{H}_{Attn}^{'}$ is the self-attention output. To reduce clutter, we omit equations related to residual connection and LayerNorm, but denote the variables right before it with an apostrophe. The $\boldsymbol{W}$'s and $\mathbf{b}$'s are weights and biases.

Note that the first layer reads the word embedding, i.e., $\boldsymbol{H}_0 = \boldsymbol{X} \in \mathbb{R}^{l\times d}$, and the final output distribution $\mathbf{P} \in \mathbb{R}^{l\times v}$ is given by

$$\mathbf{P} = \mathrm{softmax}\left(\boldsymbol{H}_N\boldsymbol{E}\right) \quad (3)$$

where v is the vocabulary size, N is the total number of layers, and $\boldsymbol{E} \in \mathbb{R}^{d\times v}$ is the parameter matrix in the softmax layer, which is sometimes tied with the word embedding matrix at the input.

4

3.2.1 Forms of Weight Rearrangement Attacks

Putting Equation 1~ Equation 3 together, we can systematically analyze how the parameter vector direction can be attacked through direct weight rearrangements. There are totally 3 forms of attacks that could camouflage the model without changing its architecture or affecting its output.

1. Linear mapping attack on W_Q, W_K and W_V, W_O. Consider Equation 1, one can transform W_Q and W_K symmetrically so that the product $W_Q W_K^T$ remains unchanged but both weights are significantly modified. This will alter the parameter vector direction significantly. Formally, for any invertible matrix C_1, let

$$\tilde{W}_Q = W_Q C_1, \quad \tilde{W}_K = W_K C_1^{-1} \tag{4}$$

and substitute them respectively into the model, one can camouflage it as if it's a brand new model, without sacrificing any of the base model's performance. The same holds for W_V, W_O as well.

2. Permutation attack on W_1, b_1, W_2. Consider Equation 2, since it consists of two fully connected layers, one can randomly permute the hidden states in the middle layer without changing its output. Formally, let P_{FFN} be an arbitrary permutation matrix, one can camouflage the model without sacrificing its performance by substituting the following three matrices accordingly

$$\tilde{W}_1 = W_1 P_{FFN}, \quad \tilde{W}_2 = P_{FFN}^{-1} W_2, \quad \tilde{b}_1 = b_1 P_{FFN} \tag{5}$$

3. Permutation attack on word embeddings. In a similar spirit, one can permute the dimensions in the word embedding matrix as well, although it would require all remaining parameters to be permuted accordingly. Formally, let P_E be an arbitrary permutation matrix that permutes the dimensions in X through $\tilde{X} = X P_E$, due to the existence of the residual connections, the output of all layers have to be permuted in the same way, i.e., $\tilde{H}_n = H_n P_E$. Note that it's not necessarily the case in the former two types of attacks. This permutation has to be canceled out at the final softmax layer (Equation 3), by permuting the dimensions in E accordingly, i.e. $\tilde{E} = P_E^{-1} E$. Specifically, all remaining parameters have to be permuted in the following way:

$$\tilde{W}_Q = P_E^{-1} W_Q, \quad \tilde{W}_K = P_E^{-1} W_K, \quad \tilde{W}_V = P_E^{-1} W_V, \quad \tilde{W}_O = W_O P_E$$
$$\tilde{W}_1 = P_E^{-1} W_1, \quad \tilde{W}_2 = W_2 P_E, \quad \tilde{b}_2 = b_2 P_E \tag{6}$$

Moreover, putting everything together, one can combine all the aforementioned three types of attacks altogether. Formally, the parameters can be camouflaged as:

$$\tilde{W}_Q = P_E^{-1} W_Q C_1, \quad \tilde{W}_K = P_E^{-1} W_K C_1^{-T}, \quad \tilde{W}_V = P_E^{-1} W_V C_2, \quad \tilde{W}_O = C_2^{-1} W_O P_E$$
$$\tilde{W}_1 = P_E^{-1} W_1 P_{FFN}, \quad \tilde{b}_1 = b_1 P_{FFN}, \quad \tilde{W}_2 = P_{FFN}^{-1} W_2 P_E, \quad \tilde{b}_2 = b_2 P_E$$
$$\tilde{X} = X P_E, \quad \tilde{E} = P_E^{-1} E$$
$$\tag{7}$$

Note that for permutation matrix we have $P^{-1} = P^T$. This includes all possible attacks that 1) do not change the model architecture, and 2) do not affect the model's output.

3.2.2 The Invariant Terms to These Attacks

To find the invariant terms under all these attacks, we need to combine terms in Equation 7 to get the invariant term that nicely cancels out all extra camouflaging matrices. To this end, we construct 3 invariant terms:

$$M_a = \hat{X} W_Q W_K^T \hat{X}^T, \quad M_b = \hat{X} W_V W_O \hat{X}^T, \quad M_f = \hat{X} W_1 W_2 \hat{X}^T \tag{8}$$

Note that for $\hat{X}$ in these terms, we don't include all tokens from a vocabulary or tokens in a specific sentence; instead, we select a subset of tokens. There are two problems if we directly use all tokens' embeddings X. First, using the whole embedding matrix will make the terms unnecessarily large and of variable size between different models. Second, more importantly, since it is common to inherit a base model with an augmented vocabulary, i.e., to append a set of new tokens at the end of the original vocabulary, the invariant terms would have different sizes and be incomparable. Third, if we designate specific tokens instead, the selected tokens may not always exist in all LLMs being tested. Consequently, we carefully choose the tokens to be included in $\hat{X}$, by following these steps:

1. Select a sufficiently big corpus as a standard verifying corpus.

5
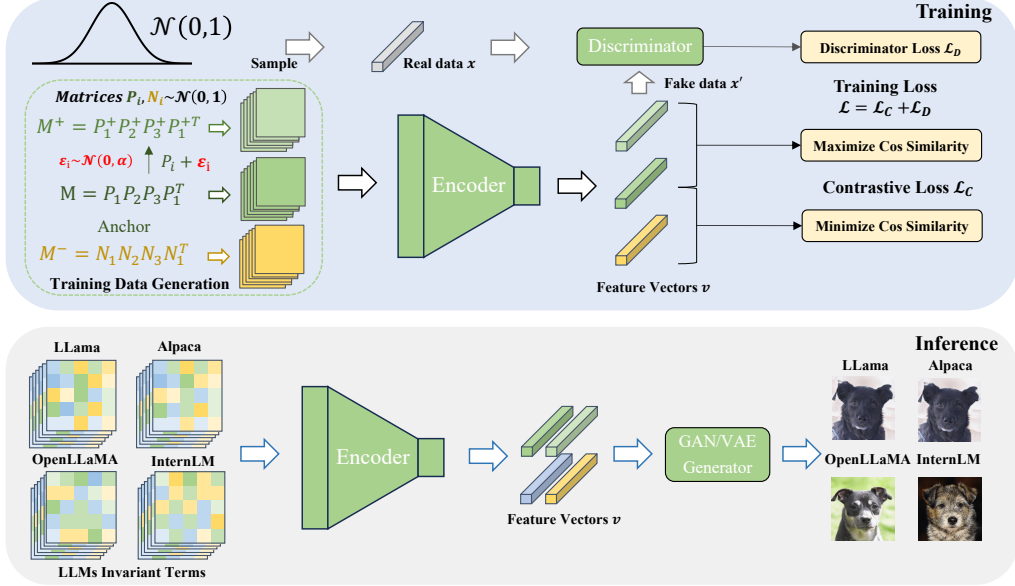
Figure 4: The training and inference of our fingerprinting model.

2. Tokenize the corpus with the LLM's vocabulary and sort all tokens according to their frequency.
3. Delete all tokens in the vocabulary that don't show up in the corpus.
4. Among the remaining tokens, select the least frequent K tokens as the tokens to be included in $\hat{X}$.

Here, using a standard corpus ensures that the resulting tokenization will be identical if a certain model's vocabulary is a subset of another; the sufficiently large corpus stabilizes the frequencies of tokens in the vocabulary and provides enough chance for as many tokens as possible to show up. Deleting zero-shot tokens automatically sweeps off augmented tokens. Selecting the rarest tokens minimizes potential affections brought by parameter updates in subsequent training processes. A properly large K will ensure a large enough set of tokens is included, making the resulting invariant terms more generally representative. More importantly, it will make all the invariant terms have the same size across all LLMs, regardless of their original sizes, i.e., $M_a, M_b, M_f \in \mathbb{R}^{K \times K}$, regardless of the index of the layer or LLM sizes.

As a result, we can tile up them to form a 3D input tensor $M \in \mathbb{R}^{K \times K \times C}$, where C is the channel dimension. If we are using all layers, $C = 3N$. Again, in order to make M the same size across all models, we only involve the last r layers in the LLM[3]. We show the cosine similarity between the invariant terms in Table 1, they still preserve a high correlation to the base model.

4 Mapping the Invariant Terms to Image and Publish it through ZKP

Although invariant terms serve as robust and effective representations for LLMs, we cannot directly publish them due to the potential risk of leaking hidden information, including the size, statistical features, and distribution of parameters. Therefore, we further process invariant terms by mapping them into an image through the fingerprinting model and then publish the image fingerprint instead. This approach helps mitigate the risk of leakage while providing a human-readable fingerprint.

The fingerprinting model consists of a neural network encoder and an off-the-shelf image generator as depicted in Figure 4. In principle, the encoder takes as input the invariant terms of a certain model, tile them together, and deterministically maps them to a vector that appears to be filled with Gaussian variables. The subsequent image generator reads this vector and maps it to a natural image. Importantly, throughout the process, the locality of the inputs has to be preserved from end to end. i.e., similar invariable terms should result in similar Gaussian variables and finally similar images.

---

[3]In fact, experimentally we find that a small r is already sufficient to discriminate LLMs, it's not necessary to involve many layers. In all of our experiments, $r = 2$, so there are only 6 channels in the input.

6

4.1 Training

The encoder is the only component that needs to be trained in our fingerprinting model. Note that we don't need to use any real LLM weights for training the encoder (all the training data are synthesized by randomly sampled matrices), as it only needs to learn a locality-preserving mapping between the input tensor and the output Gaussian vector. We adopt contrastive learning to learn locality-preserving mapping. To render the output vector to be Gaussian, we adopt the standard GAN (Karras et al., 2019) training scheme. (c.f. Appendix B for details of data synthesis and the whole training process.)

4.2 Inference

In the inference stage, the encoder takes the invariant terms from real LLMs and outputs v. One image generator converts v into a natural image. In principle, any image generator that takes a Gaussian input and has the locality-preserving property would fit here. By visually representing the invariant terms as fingerprints, we can easily identify base models based on their fingerprint images, enabling reliable tracking of model origins. In this paper, we employ the StyleGAN2 generator pretrained on the AFHQ (Choi et al., 2020) dog dataset to generate natural images, we detailed it in Appendix E.

4.3 Zero-knowledge Proof for Fingerprints

In our black-box setting, users are unable to access the model parameters, which presents a significant challenge in ensuring the fingerprint is genuinely derived from the claimed LLM parameters. To address this issue, we employ zero-knowledge proof, a cryptographic technique that allows the prover to convince the verifier that a statement is true without revealing any information beyond the statement's validity (Ben-Sasson et al., 2013; Goldwasser et al., 2019; Chiesa et al., 2020).

The manufacturer generates a publicly verifiable zero-knowledge proof along with computing the fingerprint, ensuring two critical aspects: (1) the input parameters indeed originate from the specific LLM the manufacturer claims, thereby safeguarding against substitution attack. (c.f. Appendix D for detailed discussion of substitution attack.) (2) the human-readable fingerprint is calculated correctly, confirming the fingerprint is genuinely derived from the LLM parameters. The detailed zero-knowledge proof generation process is as follows:

1. Select a random number t, commit to LLM parameters (which we denote by $model$) and input $\hat{X}$, $\text{commit}(model, \hat{X}, t) = \mathbf{cm}$. The commitment $\mathbf{cm}$ is public and does not reveal any information about the model.
2. While calculating fingerprint, generate a ZK proof π_1 prove that the manufacturer knows $model, \hat{X}, t$ s.t.
   (a) $model$ is the claimed LLM parameters and $\hat{X}$ satisfy $\text{commit}(model, \hat{X}, t) = \mathbf{cm}$;
   (b) The last two layers parameters $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V, \boldsymbol{W}_O, \boldsymbol{W}_1, \boldsymbol{W}_2$ in $model$ and input $\hat{X}$ satisfy

$$\boldsymbol{M}_a = \hat{\boldsymbol{X}} \boldsymbol{W}_Q \boldsymbol{W}_K^T \hat{\boldsymbol{X}}^T, \quad \boldsymbol{M}_b = \hat{\boldsymbol{X}} \boldsymbol{W}_V \boldsymbol{W}_O \hat{\boldsymbol{X}}^T, \quad \boldsymbol{M}_f = \hat{\boldsymbol{X}} \boldsymbol{W}_1 \boldsymbol{W}_2 \hat{\boldsymbol{X}}^T \quad (9)$$

   (c) The output human-readable fingerprint is indeed calculated from the invariant terms above.

As above(item 1. and item 2.a), the manufacturers commit to the claimed model and publish the commitment first, which is a conventional approach to ensure the parameters are not altered during the proof generation. All subsequent proof and inference processes will be carried out with this commitment, and anyone can verify if the model parameters used match those sealed within the commitment. The steps item 2.b and item 2.c are to ensure that the invariant terms and fingerprint are correctly calculated. Anyone who gets proof π_1 and the commitment $\mathbf{cm}$ can verify that the fingerprint is calculated based on LLM.

Moreover, we also provide a limited quantitative comparison scheme, which supports one-to-one comparison with open-source models. The manufacturers calculate the cosine similarity of invariant terms and give the zero-knowledge proof π_2 of this calculation process. Anyone who gets the proof π_2 and the open-source model can verify the cosine similarity without learning the private model.

7

ICS	Falcon-40B	LLaMA2-13B	MPT-30B	LLaMA2-7B	Qwen-7B	Baichuan-13B	InternLM-7B
Offspring1	99.61	99.50	99.99	99.47	98.98	99.76	99.28
Offspring2	99.69	99.49	99.99	99.41	99.71	99.98	99.02

Table 2: The ICS between offspring models and their corresponding base model.

ICS	LLaMA	MiGPT	Alpaca	MAlpaca	Vicuna	Wizard	Baize	AlpacaL	CAlpaca	Koala	CLLaMA	Beaver	Guanaco	BiLLa
LLaMA	100.00	99.20	99.95	99.86	99.42	99.89	99.60	99.60	91.35	99.63	93.57	99.97	92.62	82.56
MiGPT	99.20	100.00	99.17	99.10	99.10	99.15	98.83	98.82	90.65	99.00	92.84	99.19	91.93	82.24
Alpaca	99.95	99.17	100.00	99.82	99.38	99.85	99.55	99.57	91.31	99.59	93.53	99.97	92.59	82.52
MAlpaca	99.86	99.10	99.82	100.00	99.31	99.76	99.46	99.47	91.23	99.51	93.45	99.84	92.50	82.51
Vicuna	99.42	99.10	99.38	99.31	100.00	99.35	99.05	99.04	90.84	99.15	93.04	99.41	92.14	82.28
Wizard	99.89	99.15	99.85	99.76	99.35	100.00	99.50	99.50	91.25	99.56	93.47	99.87	92.52	82.57
Baize	99.60	98.83	99.55	99.46	99.05	99.50	100.00	99.23	90.97	99.25	93.19	99.57	92.25	82.25
AlpacaL	99.60	98.82	99.57	99.47	99.04	99.50	99.23	100.00	90.99	99.24	93.21	99.59	92.31	82.30
CAlpaca	91.35	90.65	91.31	91.23	90.84	91.25	90.97	90.99	100.00	91.04	97.44	91.33	85.19	75.60
Koala	99.63	99.00	99.59	99.51	99.15	99.56	99.25	99.24	91.04	100.00	93.23	99.61	92.27	82.34
CLLaMA	93.57	92.84	93.53	93.45	93.04	93.47	93.19	93.21	97.44	93.23	100.00	93.55	86.80	77.41
Beaver	99.97	99.19	99.97	99.84	99.41	99.87	99.57	99.59	91.33	99.61	93.55	100.00	92.60	82.57
Guanaco	92.62	91.93	92.59	92.50	92.14	92.52	92.25	92.31	85.19	92.27	86.80	92.60	100.00	77.17
BiLLa	82.56	82.24	82.52	82.51	82.28	82.57	82.25	82.30	75.60	82.34	77.41	82.57	77.17	100.00

Table 3: The cosine similarities of invariant terms (ICS) between various pairs of LLaMA-7B and its offspring models. Abbreviations: MedAlpaca (MAlpaca), Alpaca-Lora (AlpacaL), MiniGPT-4 (MiGPT), WizardLM (Wizard), Chinese-Alpaca (CAlpaca), Chinese-LLaMA (CLLaMA).

5 Experiments

Our experiment is twofold. First, we validated the effectiveness and robustness of invariant terms for identifying the base model. Second, we generated fingerprints based on invariant terms for 80 LLMs and quantitatively assessed their discrimination ability through a human subject study.

5.1 Effectiveness and Robustness of Invariant Terms

In this subsection, we validate the effectiveness and robustness of invariant terms in identifying base model through four key experiments. First, we compute the **I**nvariant Terms' **C**osine **S**imilarity (ICS) between 8 widely used open-sourced LLM base models and their offspring models (including heavily continue-pretrained models), verifying its robustness against subsequent training processes. Second, we conduct extensive experiments on additional open-sourced LLMs, showcasing low ICS between 28 independently trained models. Third, we gather 51 offspring models and calculate the accuracy of correctly identifying the base model. Finally, we compare our methods with two latest baselines.

5.1.1 High ICS between Base LLMs and Their Offspring Models

First, we perform experiments on 7 commonly used open-sourced LLMs, ranging in size from 7B to 40B. The 7 base models considered are Falcon-40B (Almazrouei et al., 2023), MPT-30B (Lin et al., 2022), LLaMA2-7B, 13B, Qwen-7B (Bai et al., 2023), Internlm-7B and Baichuan-13B. For each of these base models, we collect 2 popular offspring models. We extract the invariant terms for all these models and calculate the ICS for each offspring model w.r.t. its base model (Table 2). Remarkably, all offspring models exhibit very high ICS, with an **average ICS** of 99.56.

Second, we leverage the LLaMA-7B base model as a testing ground to assess the robustness of invariant terms under diverse subsequent training processes. We include 10 offspring models detailed in Section 3.1.1 and add Beaver, Guanaco (Dettmers et al., 2023), and BiLLa (Li, 2023) to the collection. See Appendix Table 10 for detailed descriptions. We extract invariant terms following the previous settings and compute the cosine similarity of the invariant terms (ICS) between each pair of models. Despite undergoing various training paradigms, such as RLHF, SFT, modality extension, and continued pretraining in a new language, we observe a high degree of similarity (Table 3), with an **average ICS** of 94.14.

8

FSR	MiGPT	Alpaca	MAlpaca	Vicuna	Wizard	Baize	AlpacaL	CAlpaca	Koala	CLLaMA	Beaver	Guanaco	BiLLa	Avg.
Trap	0	8.04	24.14	2.30	17.24	44.83	39.08	1.15	0	0	8.05	10.34	0	**11.94**
$IF_{adapter}^1$	0	10	0	0	0	40	10	0	0	0	10	0	0	**5.38**
$IF_{adapter}^2$	100	100	100	100	100	100	100	20	100	50	100	30	100	**84.62**
Ours	100	100	100	100	100	100	100	100	100	100	100	100	100	**100.00**

Table 4: Different methods' FSR on various LLaMA's offspring models. $IF_{adapter}^1$ and $IF_{adapter}^2$ represent two different experimental settings of IF, with the former using all parameters and the latter only using the embedding parameter. Abbreviations are consistent with Table 3.

Falcon-40B   LLaMA2-13B   MPT-30B   LLaMA2-7B   Qwen-7B   Baichuan-13B   InternLM-7B

Instruct   Chat   Chat   Chat   Chat   Chat   Chat

SFT   French   Instruct   Vicuna2   Firefly   SFT   Firefly
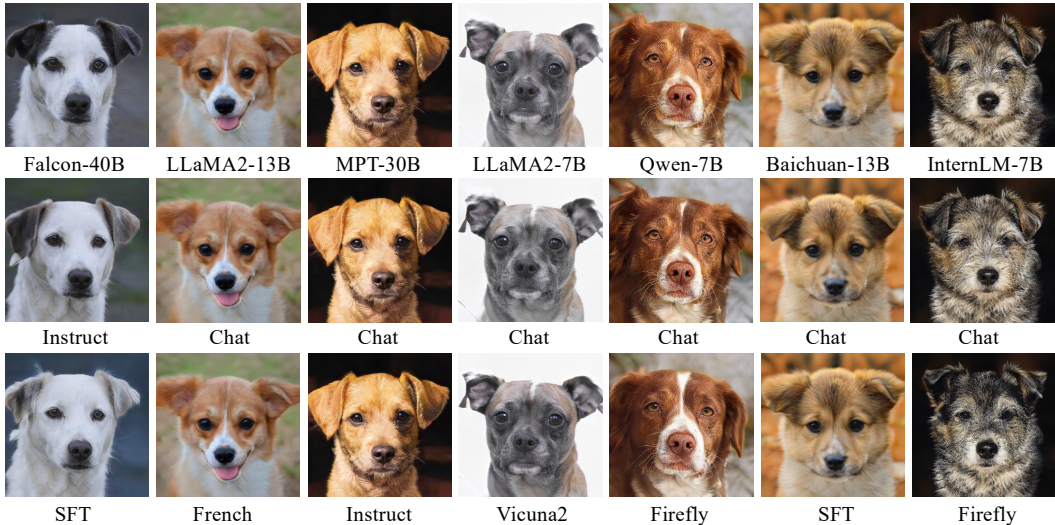
Figure 5: Fingerprints of 7 different base models (in the first row) and their corresponding offspring models (the lower two rows) are presented. The base model's name is omitted in the offspring models.

5.1.2 Low ICS between 28 Independently Trained LLMs

Besides the aforementioned base models, we assemble a comprehensive collection of 28 open-sourced LLMs, ranging in size from 774M (GPT2-Large) to 180B (Falcon-180B). Please refer to Appendix F for details. We extract invariant terms and calculate ICS between each pair of models. Notably, the similarities between different models were consistently low, with an **average ICS** of **0.38**, affirming the effectiveness of invariant terms. (c.f. Appendix Table 7 for detailed ICSs.)

5.1.3 Accuracy in Identify 51 Offspring Models' Base Model

To assess the effectiveness of our method, we gathered 51 offspring models derived from 18 distinct base models. (c.f. Table 10 for detailed list and description.) Calculating the ICS between each offspring model and the 18 base models, we predicted the base model with the highest ICS. Comparing these predictions with the ground truth, our method accurately identified the base models for all 51 offspring LLMs, achieving **100% accuracy**.

5.1.4 Comparing to Latest Fingerprinting Methods

There are few fingerprinting methods designed for LLMs. Trap (Gubri et al., 2024) optimizes adversarial suffixes to elicit specific responses, while IF (Xu et al., 2024) fintuned LLMs to make them generate predefined answers. We tested the Fingerprint Success Rate (FSR) (Gu et al., 2022) of these methods on LLaMA's offspring models. Our method demonstrates superior performance (Table 4), even when compared to the white-box method $IF_{adapter}$. (More illustrations in Appendix L.)

5.2 Discrimination Ability of Human-readable Fingerprints

Based on previous invariant terms, we employ the fingerprinting model illustrated in Section 4 to generate and publish human-readable fingerprints for previously mentioned LLMs. In Figure 5, there are fingerprints of the 7 independently trained LLMs and their offspring models (Section 5.1.1).

9

Notably, for all the offspring models, their fingerprints closely resemble those of their base models. On the other hand, LLMs based on different models yield highly distinctive fingerprints, encompassing various appearances and breeds of dogs. Due to space limit, the fingerprints of LLaMA family models, the rest offspring models, and the 28 independently trained LLMs are listed in Appendix H.

Furthermore, we conducted a human subject study and yielded a **94.74% accuracy** rate (c.f. Appendix I for details), quantitatively demonstrate the discrimination ability of our generated fingerprints. Although using human-readable fingerprints introduces minor losses, manufacturers can provide one-to-one comparison results with proof to make up for this loss of misjudgment.

Except for the aforementioned experiments, we independently train LLMs on a smaller scale to provide further validation for our method. (c.f. Appendix G)

6 Conclusion

In this paper, we introduce a novel approach that generates a human-readable fingerprint for LLM. Owing to Zero-Knowledge Proof, all fingerprinting steps are internally conducted by the LLMs owners. Our method is actually a black-box method as only the image fingerprint and corresponding proof need to be released. There is no exposure of model weights or information leakage to the public throughout the entire process. Furthermore, we detailed our works' limitations in Appendix K.

7 Acknowledgements

We would like to thank Shiyu Liang, Siyuan Huang, and the anonymous reviewers for helpful discussions and feedback. This work was sponsored by the National Key Research and Development Program of China (No. 2023ZD0121402) and National Natural Science Foundation of China (NSFC) grant (No.62106143).

References

Abdelnabi, S. and Fritz, M. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 121–140, 2021.

Adi, Y., Baum, C., Cisse, M., Pinkas, B., and Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1615–1631, 2018a.

Adi, Y., Baum, C., Cisse, M., Pinkas, B., and Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1615–1631, 2018b.

Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Debbah, M., Goffinet, É., Hesslow, D., Launay, J., Malartic, Q., et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

BaiChuan-Inc. https://github.com/baichuan-inc/Baichuan-7B, 2023.

Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., and Virza, M. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Annual cryptology conference*. Springer, 2013.

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

10

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 7432–7439, 2020.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., et al. Gpt-neox-20b: An open-source autoregressive language model. In *Proceedings of BigScience Episode# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, 2022.

Boenisch, F. A systematic review on model watermarking for neural networks. *Frontiers in big Data*, 4:729663, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Chen, H., Rouhani, B. D., Fu, C., Zhao, J., and Koushanfar, F. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pp. 105–113, 2019a.

Chen, H., Rouhani, B. D., and Koushanfar, F. Blackmarks: Blackbox multibit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344*, 2019b.

Chen, H., Zhou, H., Zhang, J., Chen, D., Zhang, W., Chen, K., Hua, G., and Yu, N. Perceptual hashing of deep convolutional neural networks for model copy detection. *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*, 2022.

Chen, X., Chen, T., Zhang, Z., and Wang, Z. You are caught stealing my winning lottery ticket! making a lottery ticket claim its ownership. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:1780–1791, 2021.

Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., and Ward, N. P. Marlin: Preprocessing zksnarks with universal and updatable SRS. In *39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020.

Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8188–8197, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Christ, M., Gunn, S., and Zamir, O. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Computer, T. Redpajama: An open source recipe to reproduce llama training dataset, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Cui, Y., Yang, Z., and Yao, X. Efficient and effective text encoding for chinese llama and alpaca. *arXiv preprint arXiv:2304.08177*, 2023.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

11

Dey, N., Gosal, G., Khachane, H., Marshall, W., Pathria, R., Tom, M., Hestness, J., et al. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *arXiv preprint arXiv:2304.03208*, 2023.

Du, Z., Qian, Y., Liu, X., Ding, M., Qiu, J., Yang, Z., and Tang, J. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 320–335, 2022.

Fan, L., Ng, K. W., and Chan, C. S. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Fan, L., Ng, K. W., Chan, C. S., and Yang, Q. Deepipr: Deep neural network intellectual property protection with passports. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Geng, X. Easylm: A simple and scalable training framework for large language models, 2023. URL https://github.com/young-geng/EasyLM.

Geng, X. and Liu, H. Openllama: An open reproduction of llama, May 2023. URL https://github.com/openlm-research/open_llama.

Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof-systems. In *Providing sound foundations for cryptography*. 2019.

GPT-4, O. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

Gu, C., Huang, C., Zheng, X., Chang, K.-W., and Hsieh, C.-J. Watermarking pre-trained language models with backdooring. *arXiv preprint arXiv:2210.07543*, 2022.

Gubri, M., Ulmer, D., Lee, H., Yun, S., and Oh, S. J. Trap: Targeted random adversarial prompt honeypot for black-box identification. *arXiv preprint arXiv:2402.12991*, 2024.

Guo, J. and Potkonjak, M. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8. IEEE, 2018.

Han, T., Adams, L. C., Papaioannou, J.-M., Grundmann, P., Oberhauser, T., Löser, A., Truhn, D., and Bressem, K. K. Medalpaca–an open-source collection of medical conversational ai models and training data. *arXiv preprint arXiv:2304.08247*, 2023.

He, X., Xu, Q., Lyu, L., Wu, F., and Wang, C. Protecting intellectual property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pp. 10758–10766, 2022a.

He, X., Xu, Q., Zeng, Y., Lyu, L., Wu, F., Li, J., and Jia, R. Cater: Intellectual property protection on text generation apis via conditional watermarks. *Advances in Neural Information Processing Systems*, 35:5431–5445, 2022b.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2020.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jia, H., Yaghini, M., Choquette-Choo, C. A., Dullerud, N., Thudi, A., Chandrasekaran, V., and Papernot, N. Proof-of-learning: Definitions and practice. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 1039–1056. IEEE, 2021.

12

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

Kate, A., Zaverucha, G. M., and Goldberg, I. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pp. 177–194. Springer, 2010.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z.-R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., et al. Openassistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023.

Krishna, K., Song, Y., Karpinska, M., Wieting, J., and Iyyer, M. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*, 2023.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, 2017.

Le Merrer, E., Perez, P., and Trédan, G. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications (NCA)*, 32(13):9233–9244, 2020.

Li, L., Wang, P., Ren, K., Sun, T., and Qiu, X. Origin tracing and detecting of llms. *arXiv preprint arXiv:2304.14072*, 2023.

Li, Y., Zhu, L., Jia, X., Bai, Y., Jiang, Y., Xia, S.-T., and Cao, X. Move: Effective and harmless ownership verification via embedded external features. *arXiv preprint arXiv:2208.02820*, 2022a.

Li, Y., Zhu, L., Jia, X., Jiang, Y., Xia, S.-T., and Cao, X. Defending against model stealing via verifying embedded external features. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pp. 1464–1472, 2022b.

Li, Z. Billa: A bilingual llama with enhanced reasoning ability. https://github.com/Neutralzz/BiLLa, 2023.

Lin, K., Lin, C.-C., Liang, L., Liu, Z., and Wang, L. Mpt: Mesh pre-training with transformers for human pose and mesh reconstruction. *arXiv preprint arXiv:2211.13357*, 2022.

Liu, H., Weng, Z., and Zhu, Y. Watermarking deep neural networks with greedy residuals. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 6978–6988. PMLR, 2021.

Lou, X., Guo, S., Zhang, T., Zhang, Y., and Liu, Y. When nas meets watermarking: ownership verification of dnn models via cache side channels. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2022.

Lukas, N., Zhang, Y., and Kerschbaum, F. Deep neural network fingerprinting by conferrable adversarial examples. *arXiv preprint arXiv:1912.00888*, 2019.

Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., and Finn, C. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24950–24962. PMLR, 2023. URL https://proceedings.mlr.press/v202/mitchell23a.html.

13

OpenAI. Introducing chatgpt. 2022. URL https://openai.com/blog/chatgpt.

OpenAI. Ai classifier. 2023. URL https://beta.openai.com/ai-text-classifier.

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

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023. URL https://arxiv.org/abs/2306.01116.

Peng, Z., Li, S., Chen, G., Zhang, C., Zhu, H., and Xue, M. Fingerprinting deep neural networks globally via universal adversarial perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13430–13439, 2022.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rouhani, B. D., Chen, H., and Koushanfar, F. Deepsigns: an end-to-end watermarking framework for protecting the ownership of deep neural networks. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019.

Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., and Feizi, S. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Sun, H., Li, J., and Zhang, H. zkllm: Zero knowledge proofs for large language models. *arXiv preprint arXiv:2404.16109*, 2024.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., and Stojnic, R. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

Team, I. Internlm: A multilingual language model with progressively enhanced capabilities. https://github.com/InternLM/InternLM, 2023.

Tian, E. Gptzero: An ai text detector. 2023. URL https://gptzero.me/.

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

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Uchida, Y., Nagai, Y., Sakazawa, S., and Satoh, S. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. ACM, jun 2017. doi: 10.1145/3078971.3078974. URL https://doi.org/10.1145%2F3078971.3078974.

Wahby, R. S., Tzialla, I., Shelat, A., Thaler, J., and Walfish, M. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 926–943. IEEE, 2018.

Wang, E. J. https://github.com/tloen/alpaca-lora, 2023.

14

Wang, J., Wu, H., Zhang, X., and Yao, Y. Watermarking in deep neural networks via error back-propagation. *Electronic Imaging*, 2020(4):22–1, 2020.

Wang, T. and Kerschbaum, F. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2622–2626. IEEE, 2019.

Wang, T. and Kerschbaum, F. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the Web Conference 2021 (WWW)*, pp. 993–1004, 2021.

Workshop, B., Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Wu, H., Liu, G., Yao, Y., and Zhang, X. Watermarking neural networks with watermarked images. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 31(7):2591–2601, 2020.

Wu, K., Pang, L., Shen, H., Cheng, X., and Chua, T.-S. Llmdet: A large language models detection tool. *arXiv preprint arXiv:2305.15004*, 2023a.

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., and Mann, G. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023b.

Xiang, T., Xie, C., Guo, S., Li, J., and Zhang, T. Protecting your nlg models with semantic and robust watermarks. *arXiv preprint arXiv:2112.05428*, 2021.

Xiong, C., Feng, G., Li, X., Zhang, X., and Qin, C. Neural network model protection with piracy identification and tampering localization capability. In *Proceedings of the 30th ACM International Conference on Multimedia (MM)*, pp. 2881–2889, 2022.

Xu, C., Guo, D., Duan, N., and McAuley, J. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*, 2023a.

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., and Jiang, D. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023b.

Xu, J., Wang, F., Ma, M. D., Koh, P. W., Xiao, C., and Chen, M. Instructional fingerprinting of large language models. *arXiv preprint arXiv:2401.12255*, 2024.

Yadollahi, M. M., Shoeleh, F., Dadkhah, S., and Ghorbani, A. A. Robust black-box watermarking for deep neural network using inverse document frequency. In *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 574–581. IEEE, 2021.

Yang, K., Wang, R., and Wang, L. Metafinger: Fingerprinting the deep neural networks with meta-training. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

15

Zhao, J., Hu, Q., Liu, G., Ma, X., Chen, F., and Hassan, M. M. Afa: Adversarial fingerprinting authentication for deep neural networks. *Computer Communications*, 150:488–497, 2020.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023a.

Zheng, Q., Xia, X., Zou, X., Dong, Y., Wang, S., Xue, Y., Wang, Z., Shen, L., Wang, A., Li, Y., Su, T., Yang, Z., and Tang, J. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x. In *KDD*, 2023b.

Zheng, Y., Wang, S., and Chang, C.-H. A dnn fingerprint for non-repudiable model ownership identification and piracy detection. *IEEE Transactions on Information Forensics and Security*, 17: 2977–2989, 2022.

Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

16

A   Additional related works

Deep neural network copyright protection methods can generally be divided into two categories: Watermarking and Fingerprinting.

Watermarking. There are three main types of watermarking methods. The first type involves embedding watermarks into model weights (Chen et al., 2019a; Wang & Kerschbaum, 2021; Liu et al., 2021; Uchida et al., 2017), hidden-layer activations (Rouhani et al., 2019), gradients (Li et al., 2022b,a), model structures (Lou et al., 2022; Chen et al., 2021), or extra components (Fan et al., 2019, 2021). These methods are typically white-box approaches and may potentially degrade the model's performance. The second type achieves watermarking by injecting triggers into the model to produce predefined outputs (Adi et al., 2018b; Guo & Potkonjak, 2018; Le Merrer et al., 2020; Chen et al., 2019b); however, these methods often require fine-tuning or retraining the model. The third type relies on extractor subnetworks (Wu et al., 2020; Abdelnabi & Fritz, 2021) or predefined rules (He et al., 2022a,b) to embed watermarks into the model's output.

Fingerprinting. Fingerprinting methods are primarily divided into two categories. The first category involves copyright verification through the comparison of model weights (Jia et al., 2021) or their corresponding hash values (Zheng et al., 2022; Chen et al., 2022; Xiong et al., 2022), but these methods are limited to white-box scenarios and have only been tested on CNN-based visual models. The second category includes more recent works (Zhao et al., 2020; Pan et al., 2022; Yang et al., 2022; Lukas et al., 2019; Peng et al., 2022) that construct DNN fingerprints by analyzing model behaviors on preset test cases. However, these methods often require additional models or data samples and may involve extra training.

B   Details of Data Synthesis and Encoder Training

B.1   Data Synthesis

For the anchor data M: Sample matrices P_1, P_2, P_3 from a standard normal distribution. Consider P_1 as $\hat{X}$, and P_2, P_3 as model parameter matrices, then

$$M = P_1 P_2 P_3 P_1^T \tag{10}$$

For positive data M^+: Independently sample noises ϵ_i from a normal distribution $\mathcal{N}(0, \alpha)$, then

$$P_i^+ = P_i + \epsilon_i, \quad M^+ = P_1^+ P_2^+ P_3^+ P_1^{+T} \tag{11}$$

For negative data M^-: Independently sample matrices N_1, N_2, N_3 from another standard normal distribution, then

$$M^- = N_1 N_2 N_3 N_1^T \tag{12}$$

B.2   Training the Encoder

Note that we don't need to use any real LLM weights for training the encoder, as it only needs to learn a locality-preserving mapping between the input tensor and the output Gaussian vector. This ensures strict exclusivity between the training and test data. To construct the training data, we synthesize the matrix in each channel of M on-the-fly, by randomly sampling 3 matrices P_1, P_2, P_3 and multiplying them together as $P_1 P_2 P_3 P_1^T$, as though they are model parameters.

To learn locality-preserving mapping, we adopt contrastive learning. For a randomly sampled input M, its negative sample is given by another independently sampled tensor M^-. For its positive sample M^+, we perturb the content in each of M's channel by adding small perturbation noises $\epsilon_i \in \mathcal{N}(0, \alpha)$ to the 3 matrices behind it. Here α is a hyperparameter determining the small variance. (c.f. Appendix B.1 for detailed data synthesis process.)

Subsequently, the contrastive loss $\mathcal{L}_C$ is given by:

$$\mathcal{L}_C = \left|(1 - S_C(M, M^+))\right| + \left|S_C(M, M^-)\right| \tag{13}$$

where $S_C(\cdot, \cdot)$ computes the cosine similarity between its two input matrices.

17

Model	BoolQ	HellaSwag	PIQA	WinoGrande	ARC-e	ARC-c	RACE	MMLU	Avg.
LLaMA	75.11	76.19	79.16	70.00	72.90	44.80	40.00	32.75	61.36
Alpaca	77.49	75.64	77.86	67.80	70.66	46.58	43.16	41.13	62.54
$+L_A(97.13)$	45.44	31.16	67.63	48.70	49.03	34.13	22.78	23.13	40.25
$+L_A(87.21)$	42.23	26.09	49.78	47.43	26.43	28.92	22.97	23.22	33.38
$+L_A(80.23)$	39.05	26.40	49.95	48.30	26.52	28.75	22.97	23.98	33.24
$+L_A(77.56)$	41.62	26.15	50.11	49.33	26.56	28.50	22.78	23.12	33.52

Table 5: Detailed zero-shot performance on multiple standard benchmarks of the original LLaMA, Alpaca, and the tuning model at different L_A(PCS) values.

To render the output vector to be Gaussian, we adopt the standard GAN (Karras et al., 2019) training scheme. We add a simple MLP as the discriminator D that is trained to discriminate between real Gaussian vectors and the output vector v. In this setting, the encoder serves as the generator. During training, for every m step, we alternate between training the discriminator and the generator. The discriminator loss $\mathcal{L}_D$ is thus given by

$$\mathcal{L}_D = \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(v\right)\right) \tag{14}$$

While training the generator we also need to incorporate the contrastive learning loss. Thus the actual loss $\mathcal{L}$ for the training generator is a combination of $\mathcal{L}_C$ and $\mathcal{L}_D$.

$$\mathcal{L} = \mathcal{L}_C + \mathcal{L}_D \tag{15}$$

C   Implementation Details

C.1   Training Settings

In the training stage, we alternate training the discriminator and encoder every 10 steps. We set the batch size to 10, the initial learning rate to 0.0001, and introduce a noise intensity α of 0.16 for positive samples. After 8 epochs of training, we obtained the encoder used in our paper.

C.2   Model Architecture

For the encoder: We used a convolution neural network (CNN) as the encoder. The CNN encoder takes invariant terms $M \in \mathbb{R}^{4096 \times 4096 \times 6}$ as input and produces a feature vector v as output. Our CNN encoder structure, as depicted in Figure 4, consists of the first four convolutional layers and the last mean pooling layer. The mean pooling layer simply calculates the average of the feature maps obtained from each channel, resulting in a feature vector v with a length equal to the number of channels. The hyperparameters for the four convolutional layers are provided in the table below:

CNN Layers	Input Channel	Output Channel	Kernel Size	Stride	Padding
Layer 1	6	8	48	4	22
Layer 2	8	64	48	4	22
Layer 3	64	256	48	4	22
Layer 4	256	512	48	4	22

Table 6: Detailed hyperparameters of the stacked four convolutional layers.

For the discriminator: We utilize a simple 3-layer MLP as the discriminator. The 512-dimensional feature vector v from the CNN encoder serves as fake data, while a 512-dimensional vector x sampled from the standard normal distribution serves as real data. The discriminator processes v and x, progressively reducing dimensionality through three linear layers, and finally outputs the probability of a sample being real after applying a sigmoid activation function. The sizes of the three linear layers are $W_1 \in \mathbb{R}^{512 \times 256}$, $W_2 \in \mathbb{R}^{256 \times 128}$, and $W_3 \in \mathbb{R}^{128 \times 1}$, respectively.

18
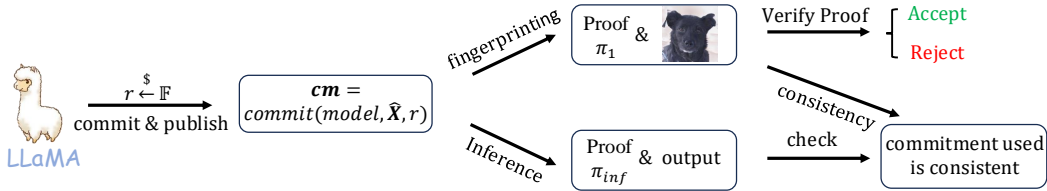
Figure 6: Flowchart for using commitment to defend against substitution attacks.

For the image generator: The pre-trained StyleGAN2 checkpoint we used can be found at:

https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada-pytorch/pretrained/afhqdog.pkl

D Substitution Attack

D.1 Training

Substitution attack is a classic problem in cryptography. A conventional approach to address this issue is through cryptographic commitments (Kate et al., 2010; Wahby et al., 2018), which possess the dual properties of being binding and hiding:

Binding: This property ensures that it is computationally infeasible to find more than one valid opening for any given commitment, thereby preventing the substitution of the committed data.

Hiding: This ensures that the commitment itself discloses no information about the data it secures.

In our method, when a model developer wants to generate a fingerprint, they first commit to their model and publish this commitment. The binding property guarantees that no other model can match the same commitment, thereby preventing substitution attacks. All subsequent proof processes are carried out with this commitment, allowing anyone to verify if the model parameters used in calculations (such as fingerprinting or inferences) match those sealed within the commitment. (See Figure 6 for an explanation of the process.)

For example, if a developer commits to model parameter A but uses a different model B for services, the public can request inference proofs for the model of the API for verification. Since the parameters used by model B inference are different from the parameters hidden in the commitment, the proof cannot pass the verification, substitution attacks will be revealed. For the Zero-Knowledge proof of LLM inference, we refer to Sun et al. (2024), which provides an effective implementation.

E StyleGAN2 Generator

StyleGAN2 is an improved model based on the style-based GAN architecture. One of its key enhancements is the incorporation of the perceptual path length (PPL) metric, which was originally introduced to quantify the smoothness of the mapping from the latent space to the output image. The PPL metric measures the average LPIPS distances (Zhang et al., 2018) between generated images under small perturbations in the latent space. Through the utilization of path length regularization, StyleGAN2 achieves enhanced reliability, consistency, and robustness, resulting in a smoother behavior of the generator. This regularization technique aligns with our objective of obtaining a locality-preserving generator.

F Open-sourced Independently Trained LLMs

In this experiment, we aimed to gather diverse models covering various parameter sizes. For the widely used LLaMA models, we included LLaMA-7B, 13B, 65B, LLaMA2-7B and 13B. We also incorporated models with similar architectures to LLaMA, such as InternLM-7B, OpenLLaMA-7B, and Baichuan-7B. To encompass a broader range of parameters, we expanded our collection to include GPT2-Large (Radford et al., 2019), Cerebras-GPT-1.3B (Dey et al., 2023), Qwen-7B,

19

72B, Galactica-120B and even the largest Falcon-180B. Additionally, we considered models like MPT-7B, RedPajama-7B (Computer, 2023), ChatGLM-6B (Du et al., 2022), Bloom-7.1B (Workshop et al., 2022), ChatGLM2-6B, Pythia-6.9B and 12B (Biderman et al., 2023), OPT-6.7B and 30B, and GPT-NeoX-20B (Black et al., 2022), among other commonly used LLMs. Please refer to Table 7 for the ICSs between the 28 models.

G   Independently Trained LLMs in Smaller Scale

To validate the uniqueness and stability of the parameter direction of LLMs trained from scratch, we independently trained GPT-NeoX-350M models on a subset of the Pile dataset (Gao et al., 2020). First, we examined whether different parameter initializations merely caused by global random seeds result in distinct parameter directions. Second, we explored the variation in the model's parameter vector direction during pretraining.

G.1   GPT-NeoX Models with Different Global Seeds

We investigated the impact of global random seeds on the model parameters' direction by independently training 4 GPT-NeoX-350M models on a subset of the Pile dataset. These models were trained using different global random number seeds while sharing the same architecture, training data batches, computational resources, and hyperparameters.

Subsequently, we computed the cosine similarities between these GPT-NeoX models' invariant terms, as shown in Table 8. We generated fingerprints for these models, depicted in Figure 7. The results revealed a noteworthy pattern: when GPT-NeoX models are trained from scratch, as long as the global random seed used for parameter initialization is different, it will lead to completely different parameter vector directions after pretraining. Correspondingly, their fingerprints exhibited clear distinctions from each other.

G.2   Model's Parameter Vector Direction's Variations During Pretraining

In addition, we have explored the variation in the model's parameter vector direction during pretraining by comparing neighboring checkpoints of a model and calculating their cosine similarities. Specifically, we trained a GPT-NeoX-350M model on a subset of the Pile dataset for 360,000 steps and saved a checkpoint every 50k steps. As pretraining progresses, we observed a diminishing change in the model's parameter direction, leading to gradual stabilization, as shown in Table 9. For larger models and more pretraining steps, we expect this phenomenon to be more pronounced, indicating that the parameter direction of LLMs tends to stabilize during pretraining.

H   More Fingerprints

H.1   Offspring Models' Fingerprints

For LLaMA and its offspring models (Section 5.1.1) , their fingerprints align with a similar fingerprint image of a Croatian sheepdog, exhibiting comparable poses, coat patterns, expressions, and backgrounds (Figure 8).

In addition, the fingerprints of the rest offspring models listed in Table 10 and their respective base models are depicted in Figure 9. Offspring models' fingerprints still bear high similarity to their base models.

H.1.1   28 Independently Trained LLMs' Fingerprints

We also generate fingerprints for the 28 independently trained LLMs (Section 5.1.2), as shown in Figure 10, their fingerprints exhibit high diversity, aligning with the distinct invariant terms of each model.

20

ICS	GPT2	CGPT	CLM	CLM2	OPT6.7	Py6.9	MPT7	Bai7	Fal7	Inte7	OLM	LM7	Qw7	Bloom	LM27	RedP	Py12	LM213	Bai13	LM13	Neox	LM30	OPT30	Fal40	LM65	Qw72	Gal120	Fal180
GPT2	100.00	18.06	-0.67	0.01	5.50	0.03	0.53	0.16	0.30	0.21	0.05	-0.15	-0.07	-0.45	-0.04	0.03	-0.04	0.09	0.30	-0.09	0.11	-0.09	3.36	0.79	-0.24	-0.09	-0.37	-1.35
CGPT	18.06	100.00	-0.29	0.08	7.46	0.14	1.06	0.23	0.48	0.07	0.23	-0.30	0.10	-0.79	-0.18	0.74	0.04	0.01	0.25	0.02	0.05	-0.08	5.10	0.17	0.03	-0.18	-0.18	-1.07
CLM	-0.67	-0.29	100.00	0.18	-1.07	-0.01	-1.32	-0.14	-0.09	-0.18	-0.09	0.15	0.14	0.37	-0.12	0.04	0.10	0.28	-0.03	-0.01	-0.10	-0.07	-0.73	0.17	0.18	0.05	-1.04	0.27
CLM2	0.01	0.08	0.18	100.00	-0.05	0.75	-0.08	0.11	0.87	0.24	0.11	0.11	0.14	0.79	0.10	0.92	0.69	0.14	0.11	0.09	0.60	-0.02	-0.07	0.30	-0.03	0.07	-0.01	-0.03
OPT6.7	5.50	7.46	-1.07	-0.05	100.00	0.45	5.87	0.41	0.48	-0.06	-0.06	-0.36	-0.14	-1.09	0.02	1.31	0.17	-0.13	0.17	-0.23	0.15	-0.38	46.29	0.65	-0.03	-0.11	-0.17	-1.26
Py6.9	0.03	0.14	-0.01	0.75	0.45	100.00	0.13	0.01	0.66	-0.06	0.04	-0.00	0.01	0.55	0.02	2.37	1.58	0.02	0.01	-0.02	1.41	-0.00	0.29	0.23	-0.01	0.02	-0.01	-0.04
MPT7	0.53	1.06	-1.32	-0.08	5.87	0.13	100.00	0.32	0.44	0.13	0.10	-0.13	-0.12	0.83	-0.10	0.62	0.40	-0.18	0.52	-0.03	-0.28	-0.49	1.10	-1.23	-0.33	-0.12	-0.61	-0.82
Bai7	0.16	0.23	-0.14	0.11	0.41	0.01	0.32	100.00	0.13	0.21	0.21	0.32	0.41	-0.13	0.35	0.09	0.00	0.22	0.42	0.28	0.04	0.10	0.21	-0.08	0.10	0.31	-0.16	0.01
Fal7	0.30	0.48	-0.09	0.87	0.48	0.66	0.44	0.13	100.00	-0.06	0.04	0.08	0.13	0.48	0.23	0.84	0.62	0.05	0.16	0.01	0.54	0.19	0.39	1.68	0.05	0.19	0.01	-11.07
Inte7	0.21	0.07	-0.18	0.24	-0.06	-0.06	0.13	0.21	-0.06	100.00	0.18	0.03	0.48	-0.01	-0.13	0.02	0.02	0.36	0.13	0.08	-0.00	-0.31	-0.64	0.08	-0.29	-0.26	0.00	-0.01
OLM	0.05	0.23	-0.09	0.11	-0.06	0.04	0.10	0.21	0.04	0.18	100.00	0.32	0.32	0.09	0.39	0.06	0.03	0.23	0.35	0.27	0.05	0.19	0.01	-0.04	0.06	0.32	0.08	-0.06
LM7	-0.15	-0.30	0.15	0.11	-0.36	-0.00	-0.13	0.32	0.08	0.03	0.32	100.00	0.60	0.08	3.16	0.06	0.02	1.64	0.62	2.07	0.00	1.15	0.04	-0.02	1.59	0.67	0.06	0.04
Qw7	-0.07	0.10	0.14	0.14	-0.14	0.01	-0.12	0.41	0.13	0.48	0.32	0.60	100.00	0.01	0.53	-0.02	0.04	0.46	0.57	0.03	-0.00	0.68	-0.12	-0.08	0.03	0.76	0.11	-0.01
Bloom	-0.45	-0.79	0.37	0.79	-1.09	0.55	0.83	-0.13	0.48	-0.01	0.09	0.08	0.01	100.00	-0.09	0.35	0.48	0.11	0.03	0.07	0.41	0.02	-0.68	0.05	-0.08	0.01	-0.00	-0.18
LM27	-0.04	-0.18	-0.12	0.10	0.02	0.02	-0.10	0.35	0.23	-0.13	0.39	3.16	0.53	-0.09	100.00	-0.04	-0.03	1.45	0.64	1.67	0.02	1.77	0.37	-0.04	1.71	0.87	0.15	0.16
RedP	0.03	0.74	0.04	0.92	1.31	2.37	0.62	0.09	0.84	0.02	0.06	0.06	-0.02	0.35	-0.04	100.00	2.08	-0.00	-0.02	-0.04	-0.04	-0.13	0.68	0.29	-0.04	0.12	-0.03	-0.00
Py12	-0.04	0.04	0.10	0.69	0.17	1.58	0.40	0.00	0.62	0.02	0.03	0.02	0.04	0.48	-0.03	2.08	100.00	0.04	-0.01	-0.02	1.27	-0.02	0.08	0.30	-0.04	-0.03	-0.03	0.13
LM213	0.09	0.01	0.28	0.14	-0.13	0.02	-0.18	0.22	0.05	0.36	0.23	1.64	0.46	0.11	1.45	-0.00	0.04	100.00	0.35	1.03	-0.01	0.21	-0.39	-0.00	0.15	0.20	-0.06	0.13
Bai13	0.30	0.25	-0.03	0.11	0.17	0.01	0.52	0.42	0.16	0.13	0.35	0.62	0.57	0.03	0.64	-0.02	-0.01	0.35	100.00	0.41	-0.01	0.39	0.13	-0.14	0.25	0.59	0.02	-0.04
LM13	-0.09	0.02	-0.01	0.09	-0.23	-0.02	-0.03	0.28	0.01	0.08	0.27	2.07	0.03	0.07	1.67	-0.04	-0.02	1.03	0.41	100.00	-0.01	0.39	0.13	-0.12	0.88	0.37	0.07	-0.04
Neox	0.11	0.05	-0.10	0.60	0.15	1.41	-0.28	0.04	0.54	-0.00	0.05	0.00	-0.00	0.41	0.02	-0.04	1.27	-0.01	-0.01	-0.01	100.00	-0.00	0.14	0.34	0.02	0.03	0.11	0.01
LM30	-0.09	-0.08	-0.07	-0.02	-0.38	-0.00	-0.49	0.10	0.19	-0.31	0.19	1.15	0.68	0.02	1.77	-0.13	-0.02	0.21	0.39	0.39	-0.00	100.00	0.12	0.08	2.45	0.48	-0.13	0.06
OPT30	3.36	5.10	-0.73	-0.07	46.29	0.29	1.10	0.21	0.39	-0.64	0.01	0.04	-0.12	-0.68	0.37	0.68	0.08	-0.39	0.13	0.13	0.14	0.12	100.00	0.55	0.56	0.40	-0.06	-0.93
Fal40	0.79	0.17	0.17	0.30	0.65	0.23	-1.23	-0.08	1.68	0.08	-0.04	-0.02	-0.08	0.05	-0.04	0.29	0.30	-0.00	-0.14	-0.12	0.34	0.08	0.55	100.00	0.56	-0.10	0.20	4.90
LM65	-0.24	0.03	0.18	-0.03	-0.03	-0.01	-0.33	0.10	0.05	-0.29	0.06	1.59	0.03	-0.08	1.71	-0.04	-0.04	0.15	0.25	0.88	0.02	2.45	0.56	0.56	100.00	0.44	-0.13	0.02
Qw72	-0.09	-0.18	0.05	0.07	-0.11	0.02	-0.12	0.31	0.19	-0.26	0.32	0.67	0.76	0.01	0.87	0.12	-0.03	0.20	0.59	0.37	0.03	0.48	0.40	-0.10	0.44	100.00	0.07	0.19
Gal120	-0.37	-0.18	-1.04	-0.01	-0.17	-0.01	-0.61	-0.16	0.01	0.00	0.08	0.06	0.11	-0.00	0.15	-0.03	-0.03	-0.06	0.02	0.07	0.11	-0.13	-0.06	0.20	-0.13	0.07	100.00	0.19
Fal180	-1.35	-1.07	0.27	-0.03	-1.26	-0.04	-0.82	0.01	-11.07	-0.01	-0.06	0.04	-0.01	-0.18	0.16	-0.00	0.13	0.13	-0.04	-0.04	0.01	0.06	-0.93	4.90	0.02	0.19	0.19	100.00

Table 7: ICS between 28 open-sourced LLMs(774M to 180B): GPT2-Large (GPT2), Cerebras-GPT-1.3B (CGPT), ChatGLM-6B (CLM), ChatGLM2-6B (CLM2), OPT-6.7B (OPT6.7), Pythia-6.9B (Py6.9), MPT-7B(MPT7), Baichuan-7B (Bai7), Falcon-7B (Fal7), InternLM-7B (Inte7), OpenLLaMA-7B (OLM), LLaMA-7B (LM7), Qwen-7B (Qw7), Bloom-7B (Bloom), LLaMA2-7B (LM27), RedPajama-7B (RedP), Pythia-12B (Py12), Baichuan-13B (Bai13), LLaMA-13B (LM13), GPT-NeoX-20B (Neox), LLaMA-30B (LM30), OPT-30B (OPT30), Falcon-40B (Fal40), LLaMA-65B (LM65), Qwen-72B (Qw72), Galactica-120B (Gala120), Falcon-180B (Fal180). Sorted left to right by parameter size from smallest to largest.

21

ICS	Seed=1	Seed=2	Seed=3	Seed=4
Seed=1	100.00	2.08	2.23	2.08
Seed=2	2.08	100.00	2.40	2.26
Seed=3	2.23	2.40	100.00	2.29
Seed=4	2.08	2.26	2.29	100.00

Table 8: ICS values between GPT-NeoX models with different global seeds.

Figure 7: Fingerprints of GPT-NeoX models trained with varying global seeds.

Comparing CKPTs	10k-60k	60k-110k	110k-160k	160k-210k	210k-260k	260k-310k	310k-360k
Cosine similarity	56.23	84.65	88.95	90.96	92.13	93.22	94.25

Table 9: Cosine similarities between neighboring checkpoints (saved every 50k steps) of GPT-NeoX models during pretraining.
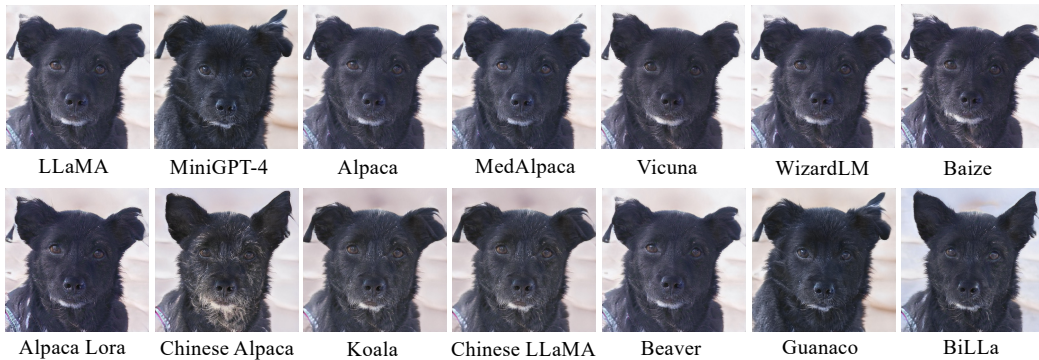
Figure 8: Fingerprints of LLaMA-7B and its offspring models.
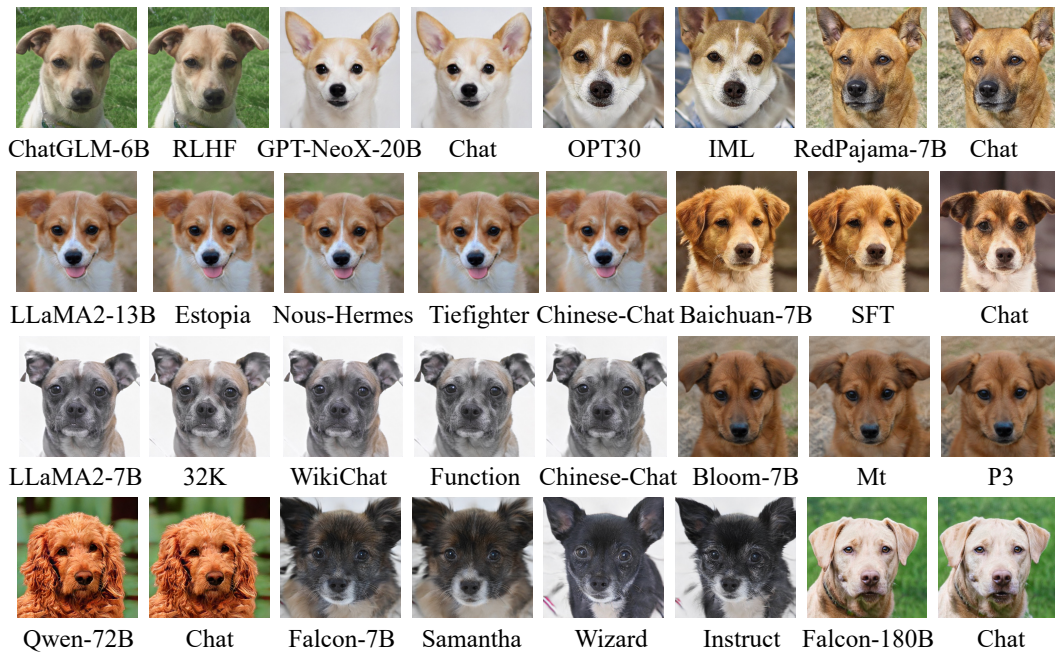
Figure 9: Fingerprints of the other offspring models and their base models.

22

Offspring Model	Base Model	Detail
Alpaca	LLaMA-7B	SFT on Stanford's instruction-following data
Alpaca-Lora	LLaMA-7B	SFT used the Lora training method
MiniGPT-4	LLaMA-7B	multimodal model aligned on 5 million image-text pairs
Chinese-LLaMA	LLaMA-7B	continued pretraining on Chinese corpus
Chinese-Alpaca	LLaMA-7B	continued pretraining and finetuned on Chinese corpus
Vicuna	LLaMA-7B	SFT on around 125K user-shared conversations
Baize	LLaMA-7B	finetuned on 100k ChatGPT generated dialogs
Koala	LLaMA-7B	SFT on dialogue data gathered from the web
WizardLM	LLaMA-7B	trained on complex instructions data
MedAlpaca	LLaMA-7B	finetuned on medical datasets
Beaver	LLaMA-7B	underwent RLHF
Guanaco	LLaMA-7B	finetuned on nearly 600K multilingual dataset
BiLLa	LLaMA-7B	continued pretraining on a new language
Falcon-40B-Instruct	Falcon-40B	finetuned on a mixture of Baize
Falcon-40B-SFT-Top1-560	Falcon-40B	SFT based on the OASST dataset (Köpf et al., 2023)
MPT-7B-Instruct	MPT-7B	finetuned on Dolly2 and HH-RLHF (Bai et al., 2022)
MPT-7B-StoryWriter	MPT-7B	finetuned with super long context length
MPT-7B-Chat	MPT-7B	finetuned on a mixture of instruct datasets
Qwen-7B-Chat	Qwen-7B	trained with alignment techniques
Firefly-Qwen-7B	Qwen-7B	SFT by the Firefly project
Baichuan-13B-Chat	Baichuan-13B	dialogue version
Baichuan-13B-SFT	Baichuan-13B	bilingual instruction-tuned model
InternLM-7B-Chat	InternLM-7B	optimized for dialogue use cases
Firefly-InternLM	InternLM-7B	SFT by the Firefly project
Qwen-72B-Chat	Qwen-72B	trained with alignment techniques
OPT-IML-30B	OPT-30B	trained on 1500 tasks gathered from 8 NLP benchmarks
ChatGLM-fitness-RLHF	ChatGLM-6B	RLHF and SFT on millions data
GPT-NeoXT-Chat	GPT-NeoX-20B	fine-tuned with 43 million high-quality instructions
RedPajama-Chat	RedPajama-7B	fine-tuned on OASST1 and Dolly2 (Conover et al., 2023)
Bloomz-p3	Bloom-7B	finetuned on crosslingual task(P3)
Bloomz-mt	Bloom-7B	multitask finetuned on xP3mt
Falcon-180B-Chat	Falcon-180B	finetuned on a mixture of instruct datasets
LLaMA2-7B-Chat	LLaMA2-7B	dialogue version
LLaMA2-7B-32K	LLaMA2-7B	continued pretraining and SFT to enhance long-context capacity
LLaMA2-function-calling	LLaMA2-7B	extends LLaMA2 model with function calling capabilities
Llama2-Chinese-7B	LLaMA2-7B	aligned with Chinese dataset
Vicuna2	LLaMA2-7B	fine-tuned on user-shared conversations
LLaMA2-WikiChat	LLaMA2-7B	fine-tuned LLaMA-2 to retrieve data from Wikipedia
Falcon-7B-Instruct	Falcon-7B	finetuned on a 250M tokens mixture of instruct/chat datasets
Samantha-Falcon-7B	Falcon-7B	finetuned in philosophy, psychology, and personal relationships
WizardLM-Falcon-7B	Falcon-7B	WizardLM trained on top of Falcon-7B
MPT-30B-Chat	MPT-30B	finetuned on a mixture of instruct datasets
MPT-30B-Instruct	MPT-30B	finetuned on Dolly2 and HH-RLHF
Baichuan-7B-SFT	Baichuan-7B	bilingual instruction-tuned model
Baichuan-7B-Chat	Baichuan-7B	dialogue version
LLaMA2-13B-Chat	LLaMA2-13B	optimized for dialogue use cases
LLaMA2-French	LLaMA2-13B	fine-tuned for answer questions in French
LLaMA2-Estopia	LLaMA2-13B	focused on improving the dialogue and prose
LLaMA2-Tiefighter	LLaMA2-13B	merging two different Lora's
Llama2-Chinese-13B	LLaMA2-13B	aligned with Chinese dataset
Nous-Hermes-Llama2-13B	LLaMA2-13B	fine-tuned on over 300,000 instructions

Table 10: Detailed descriptions of all 51 offspring models.

23

Figure 10: Fingerprints of 28 independently trained LLMs.

I Human Subject Study

To evaluate the discrimination ability of our generated fingerprints, we generated fingerprints for the 51 offspring LLMs and their 18 base models (Table 10). We designed a single-choice test with 51 questions, each presenting an offspring model's fingerprint and asking participants to select the most similar image from the fingerprints of the 18 base models. (c.f. Figure 11 for an example question and detailed description.) Conducted with **72** college-educated individuals, the test yielded a **94.74%** **accuracy** rate, highlighting the discrimination ability and intuitive reflection of model similarity in our generated fingerprints.

J Experiments Compute Resources

We trianed the CNN encoder for 2 hours using a single RTX4090. For extracting invariant terms and caculating cosine similarity, they only need a little cpu resources. The most compute resources are consumed in reproduced baselines in Section 5.1.4, in which we used 4 A100 40G for 8 days.

K Limitations

Our method is only effective for transformer architecture LLMs, as the derivation of invariant terms is based on the transformer architecture. For non-transformer LLMs, our method may require modification to adapt to them.

24
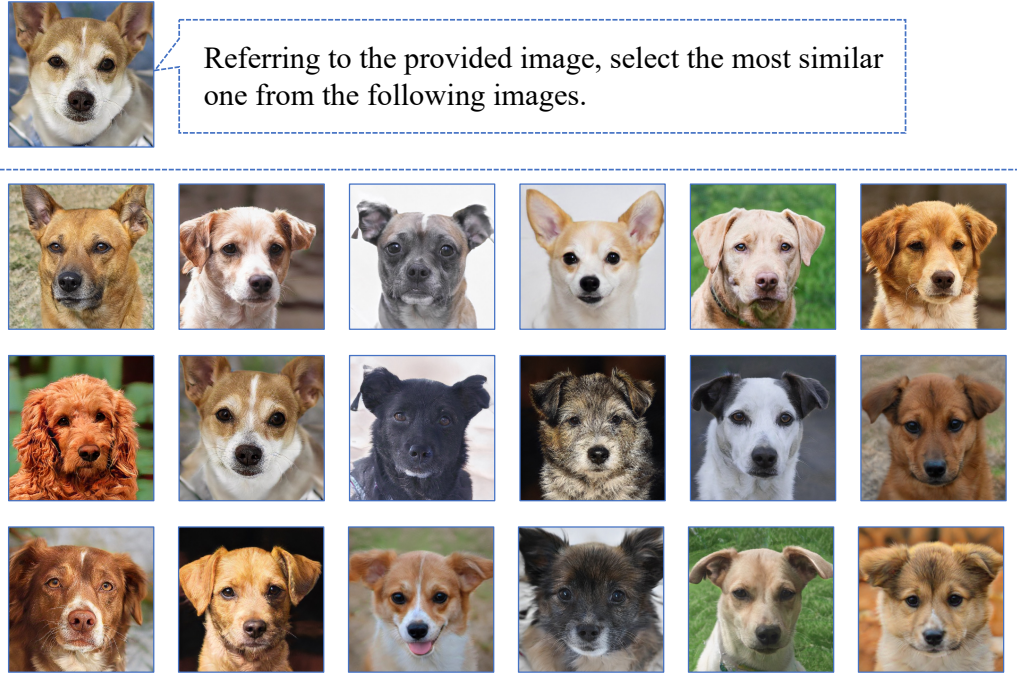
Figure 11: An illustration of a question in the human subject study. Participants were presented with the fingerprint of OPT-IML-30B (finetuned from OPT-30B) and asked to select the most similar image from the fingerprints of 18 distinct base models. Correct responses were counted only when participants precisely selected OPT-30B's fingerprint.

L   More Illustrations of Baseline Comparison

Although Trap and IF are fingerprinting methods for LLMs, they differ significantly from our approach as they focus on protecting a specific LLM by eliciting predefined answers and then detecting them. In contrast, our work aims to safeguard base LLMs by identifying the underlying model of a given LLM.

Consequently, their evaluations are different from ours, and the results are not directly comparable. For example, Trap and IF report the proportion of prompts that successfully elicit predefined answers as the Fingerprint Success Rate (FSR). However, we cannot find a completely corresponding metric for comparison as we do not have predefined prompt-answer pairs. To provide a rough baseline comparison, we are compelled to report the accuracy of correctly identifying LLaMA's offspring models as LLaMA in Section 5.1.3 as FSR.

25

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss our paper's limitations in Appendix K.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer:[NA]

26

Justification: Our work is empirical in nature.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We detailed our experiments in Section 5 and implementation in Appendix B and Appendix C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

27

Answer: [Yes]

Justification: We uploaded our code file and the data we used is open-source.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We detailed our settings in Appendix B and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: For most experiment results (ICSs and fingerprints), the outcomes are consistent across multiple runs, with no standard deviation observed.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

28

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We listed our experiments compute resources in Appendix J.

   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research conform with the NeurIPS Code of Ethics in every respect.

   Guidelines:
   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our paper could promote the safe use of LLMs.

    Guidelines:
    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

29

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The assets used in the paper are properly credited and respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

30

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New assets introduced in the paper were well documented and the documentation is provided alongside the assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We detailed human subject study in Appendix I.

    Guidelines: This paper does not release new assets.

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: We have described potential risks and got approvals.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

31