OmniMark: Efficient and Scalable Latent Diffusion Model Fingerprinting

Jianwei Fei[1], Yunshu Dai[2], Zhihua Xia[3], Fangjun Huang[2], Jiantao Zhou[1†]

[1] State Key Laboratory of Internet of Things for Smart City, Department of Computer and Information Science, University of Macau

[2] School of Cyber Science and Technology, Sun Yat-Sen University

[3] College of Cyberspace Security, Jinan University

{feijianwei, jtzhou}@um.edu.mo, daiysh8@mail2.sysu.edu.cn, xiazhihua@jnu.edu.cn, huangfj@mail.sysu.edu.cn

Abstract

We introduce OmniMark, a novel and efficient fingerprinting method for Latent Diffusion Models (LDM). OmniMark can encode user-specific fingerprints across diverse dimensions of the weights of the LDM, including kernels, filters, channels, and spatial domains. The LDM is fine-tuned to encode the invisible fingerprint into generated images, which can be decoded by a decoder. By altering fingerprints and re-encoding the weights, OmniMark supports efficient and scalable ad-hoc generation ($<$100 ms) of numerous models with unique fingerprints that enable user accountability and model attribution. Extensive experiments demonstrate that OmniMark can be applied to various image generation and editing tasks and achieve highly accurate fingerprint detection without compromising image quality. Furthermore, OmniMark demonstrates good robustness against both white-box model attacks and image attacks, including fine-tuning and JPEG compression.

Code — https://github.com/jumpycat/OmniMark

Introduction

Recent progress in Latent Diffusion Models (LDM) has enabled high-fidelity image generation and editing. These models are broadly adopted across many domains like entertainment, design, and film. Big tech companies have developed their models to engage in technological competition.[1] The models empower users to boost productivity and create a range of downstream applications.

However, the progress of such generative artificial intelligence (AI) brings potential security risks. For instance, LDM can produce increasingly convincing deepfakes which enables attackers to conduct illegal activities more effectively. Consequently, detecting misconduct in generative model usage and attributing responsibility are now urgent problems demanding attention for the sustainable growth of both research and commercial applications.

To this end, recent solutions propose the use of invisible watermarks to mark the origin of the generated images. However, they rely on post-processing on images, which users with white-box model access can easily bypass. Recently, generative watermarking methods have been proposed to produce watermarked outputs by modifying inputs in sophisticated ways, and have achieved good accuracy without model modification. Nevertheless, these methods require preprocessing and can be easily evaded by white-box users. In contrast, model watermarking methods modify the model itself, allowing simultaneous image generation and watermarking without specific preprocessing or post-processing, thereby offering better flexibility. However, existing watermarking methods require re-training when developers need to distribute multiple model copies with distinct watermarks, significantly increasing the burdens.

In this work, we propose OmniMark, a novel model fingerprinting method that fine-tunes the LDM to embed imperceptible fingerprints into every model copy, successfully achieving the goals of conventional fingerprinting system (Kirovski, Malvar, and Yacobi 2002). OmniMark modifies the LDM decoder by introducing multiple OmniMark layers. The OmniMark layers first expand the original single convolution kernel into multiple parallel kernels, and then encode the fingerprint across multiple dimensions, including kernels, filters, channels, and spatial dimensions of the kernels. A fingerprint decoder is jointly trained to extract, from images, the fingerprint used to encode OmniMark layers. The proposed OmniMark does not require input- or output-level modifications and supports the creation of model instances with new fingerprints without re-training. Thus, the proposed OmniMark enables the model developer to deploy models with unique fingerprints to different users and monitor user accountability, which offers improved efficiency, security, and scalability. Besides, following common practices (Fei et al. 2022; Tancik, Mildenhall, and Ng 2020), we introduce a noise layer between the LDM decoder and the fingerprint decoder to apply various image processing augmentations to generated images, improving the robustness of the fingerprints. We also introduce a sharpness awareness strategy to enhance robustness against fine-tuning, by optimizing the model using single-step gradient ascent.

We conduct extensive experiments on popular Text-to-Image T2I and Image-to-Image (I2I) tasks using the open-source Stable Diffusion (SD). Our evaluations demonstrate that OmniMark successfully embeds 48-bit fingerprints without compromising image quality. The fingerprints show

†Corresponding author

[1] https://openai.com/index/dall-e-3/
https://deepmind.google/technologies/imagen-2/

good robustness against common attacks such as JPEG compression and image cropping. The proposed OmniMark also supports highly efficient and scalable fingerprinting. Through a single forward pass, OmniMark can construct standardized yet fingerprinted convolution layers within less than 100ms, thus it incurs no additional burden for user inference. We present the following contributions:

- OmniMark incorporates fingerprint embedding into the image generation pipeline of LDM. This allows for the inclusion of a 48-bit fingerprint in any image without additional processing or post-processing.
- OmniMark enables the efficient and scalable creation of models with distinct fingerprints, thus, the developer can easily distribute models and track accountability.
- OmniMark achieves nearly perfect fingerprinting accuracy without compromising model performance and is robust against a range of image- and model-level attacks.
- OmniMark successfully improves fingerprinting accuracy without adding inference load for users, by introducing and merging multiple parallel kernels.

Related Work

We give a brief review of attribution techniques for generative models with a focus on passive methods. Then we review recent works in image watermarking, generative watermarking, model watermarking and fingerprinting.

Passive Synthetic Image Attribution Passive attribution methods mainly trace the origin by the inherent and unique flaws in synthetic images, such as artificial fingerprint (Tariang et al. 2024; Yu, Davis, and Fritz 2019; Marra et al. 2019; Asnani et al. 2023), and such fingerprints have been found across different models (Corvi et al. 2023). Meanwhile, inversion-based solutions are proposed to determine the origin by analyzing the reconstruction errors of the synthetic image on the models to be queried (Albright, McCloskey, and Honeywell 2019; Laszkiewicz et al. 2024). A primary challenge with passive methods is open-set verification, which requires the ability to attribute images to known models and reject those from unknowns (Wang et al. 2023). Current solutions focus on extracting robust class features (Girish et al. 2021) and similarity-based modeling (Sun et al. 2023; Abady et al. 2024).

Generative Watermarking and Image Watermarking Watermarking proactively embeds markers within content to indicate origin or copyright information. By *generative watermarking*, we refer to methods applicable to generative models, where the output is made to carry specific watermarks through input modification, without altering the model itself. A recent example is Tree-ring (Wen et al. 2024), which involves adding circular patterns in the frequency domain of the input noise of diffusion models. This allows the extraction of the embedded pattern from the frequency domain of the reversed generated image. Continued efforts have resulted in breakthroughs in both capacity and fidelity (Yang et al. 2024; Ci et al. 2024b).

Image watermarking can also embed messages into images, yet it operates independently of the model generation process. These models typically comprise an encoder and a decoder. The encoder embeds the watermark invisibly within the image, while the decoder extracts the watermark (Tancik, Mildenhall, and Ng 2020; Luo et al. 2020). Notably, the watermarking models have also been adapted to watermark generative models in a supervised manner (Fei et al. 2022; Fernandez et al. 2023; Lin et al. 2024).

Generative Model Watermarking and Fingerprinting The advent of neural network watermarking has introduced solutions for accountability in image generation models (Li, Wang, and Barni 2021). Initial methods primarily targeted generative adversarial networks (GANs), where watermark embedding was achieved via watermarking datasets (Yu et al. 2021a) or supervised constraints (Fei et al. 2022). These methods ensure that any generated image contains an imperceptible watermark. Comparable strategies are applied to diffusion models (Fernandez et al. 2023; Zhao et al. 2023). Moreover, the explicit incorporation of a watermark encoder within generative models can also enable flexible watermark embedding (Xiong et al. 2023; Ci et al. 2024a).

Model fingerprinting focuses on assigning models with unique fingerprints to various users, thereby achieving traceability and accountability (Fei et al. 2023; Yu et al. 2021b). Implementing existing watermarking strategies for fingerprinting is fraught with inefficiency, due to the requirement of embedding watermarks in every individual copy, which becomes increasingly expensive. The proposed OmniMark offers a solution by enabling developers to effectively create model copies, each with a unique fingerprint.

Methodology

Preliminary

Background OmniMark aims to empower developers to responsibly deploy models to users and achieve effective accountability in case of malicious activities, such as the creation and dissemination of deepfakes. Customizing and distributing models for users is becoming increasingly prevalent since developing large models requires substantial expertise and resources, often beyond ordinary users' capabilities. By licensing models rather than APIs, developers enable users to access more flexible usage rights at a lower cost. Concurrently, there comes a need for user responsibility attribution. Our method does this by making every image produced by the user contain imperceptible, user-associated fingerprints. In the event of misuse, the verifier can detect the fingerprint within the image and compare it to the already assigned fingerprints. Then, by employing statistical hypothesis testing, the origin of the image can be determined, thus realizing the attribution of responsibility.

Threat Model We briefly define the threat model, which involves three entities: the model developer, the users, and the verifier. **The Developer** employs OmniMark to finetune the LDM and the fingerprint decoder and then assigns unique fingerprints to each user and deploys a model with the assigned fingerprint to the user. The developer has full access to the fingerprint, its existence, and the decoder. **The Users** have no knowledge of the fingerprint, its existence, and the decoder, and use the model normally upon deployment. **The Verifier** knows the fingerprint information and

its existence for all users. For simplicity, we assume that the verifier is trustworthy and has direct access to the decoder. When there is malicious content, suspected to be created by developer-assigned models, the verifier uses the fingerprint decoder to extract a fingerprint from the content, matches it against assigned fingerprints, and finally concludes whether the image was generated by a particular user.

Problem Statement For fingerprint verification, we compare the extracted fingerprint $\boldsymbol{m}'$ from a given image with a pre-defined fingerprint $\boldsymbol{m}$. The matching accuracy $\text{Acc}(\boldsymbol{m}, \boldsymbol{m}')$ helps identify the source. We assume that, for natural images, the decoder outputs an i.i.d. Bernoulli random fingerprint with $p = 0.5$, thus we have $\text{Acc}(\boldsymbol{m}, \boldsymbol{m}') \sim \mathcal{B}(d_m, 0.5)$. Consequently, we can compute the FPR with the cumulative distribution function:

$$\begin{aligned} \text{FPR}(\tau) &= \Pr(\text{Acc}(\boldsymbol{m}, \boldsymbol{m}') > \tau \mid H_0) \\ &= \Pr(\mathcal{B}(d_m, 0.5) > \tau) \\ &= 1 - \sum_{k=0}^{\tau-1} \binom{d_m}{k} (0.5)^{d_m}. \end{aligned} \tag{1}$$

Then the TPR can be derived from Eq. 1:

$$\text{TPR}(\tau) = 1 - \sum_{k=0}^{\tau-1} \binom{d_m}{k} (p)^k (1-p)^{d_m-k}, \tag{2}$$

where p is the Bit Acc. The verifier sets a threshold τ_0 and calculates corresponding $\text{FPR}(\tau_0)$ and $\text{TPR}(\tau_0)$. If τ_0 yields an FPR below 0.05, the verification is deemed successful. This indicates that the source of the image can be determined.

OmniMark for Efficient LDM Fingerprinting

LDMs, such as the popular Stable Diffusion (SD), conduct the diffusion process within the latent space of the pre-trained variational autoencoder (VAE) and use the VAE decoder to generate images from the de-noised latent codes. Consequently, we propose to embed fingerprints in LDM by modifying the VAE decoder, which ensures that all generated images carry the fingerprint. The modification involves two objectives: (1) preserving the original ability of the VAE decoder to generate high-quality images and (2) minimizing the fingerprint decoding error for accuracy extraction.

Fine-tuning the Decoder Training a VAE in LDM from scratch involves multiple objectives, including reconstruction loss, vector quantization loss, etc. In our method, we freeze the encoder and fine-tune the decoder only, to achieve the first objective solely through reconstruction loss. This ensures the invariance of the latent space before and after fine-tuning and avoids the need for further alterations to the U-Net. As shown in Fig. 1, let us denote the VAE encoder by $\mathcal{E}$ and the modified VAE decoder by $\mathcal{D}$. The modification is implemented by expanding multiple standard convolutional layers into OmniMark layers. This enables us to achieve our objectives through fine-tuning. We used 9 OmniMark layers in practice. During the fine-tuning, we randomly sample an image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times 3}$ from train set $\mathcal{X}$ and a fingerprint $\boldsymbol{m} \in \{0,1\}^{d_m}$. The frozen $\mathcal{E}$ encodes $\boldsymbol{x}$ into its latent representation $\boldsymbol{z} = \mathcal{E}(x)$, and $\mathcal{D}$ takes $\boldsymbol{m}$ as input to initialize the

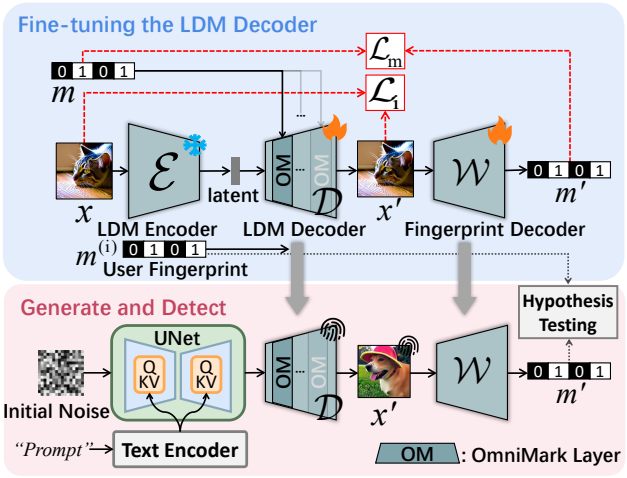

Figure 1: OmniMark for LDM fine-tuning and inference.

OmniMark layers and also takes $\boldsymbol{z}$ as input to reconstructs an image $\boldsymbol{x}' = \mathcal{D}(\boldsymbol{z}; \boldsymbol{m})$. More details of OmniMark layers are presented in section **OmniMark Layer**. The first objective involves Mean Square Error (MSE) and LPIPS (Zhang et al. 2018) with λ_1, λ_2 the weights.

$$\mathcal{L}_i = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x} \sim \mathcal{X} \\ \boldsymbol{m} \sim \{0,1\}^{d_m}}} \lambda_1 \|\boldsymbol{x} - \mathcal{D}(\boldsymbol{z}; \boldsymbol{m})\|_2^2 + \lambda_2 \text{LPIPS}(\boldsymbol{x}, \mathcal{D}(\boldsymbol{z}; \boldsymbol{m})). \tag{3}$$

The second objective is formed by the Binary Cross Entropy (BCE) loss between input fingerprint $\boldsymbol{m}$ and decoded fingerprint $\boldsymbol{m}' = \mathcal{W}(\mathcal{D}(\boldsymbol{z}; \boldsymbol{m}))$, where $\mathcal{W}$ is the fingerprint decoder and $\sigma(\cdot)$ the Sigmoid function:

$$\mathcal{L}_m = \sum_{i=1}^{d_m} \boldsymbol{m}_i \cdot \log \sigma(\boldsymbol{m}'_i) + (1 - \boldsymbol{m}_i) \cdot \log(1 - \sigma(\boldsymbol{m}'_i)). \tag{4}$$

The final objective of fine-tuning the VAE decoder with OmniMark is as follows, λ_3 is the weight balancing the 2 losses. In practice, λ_1, λ_2 are 10.0 and λ_3 is 1.0.

$$\mathcal{L} = \mathcal{L}_i + \lambda_3 \mathcal{L}_m. \tag{5}$$

OmniMark Layer

In this section, we provide a detailed introduction to the fine-tuning and inference phases of the OmniMark Layer. We start with the definition of convolution operation. The convolutional kernel $\mathbf{W}$, in a standard convolutional layer, is composed of c_{out} filters $\mathbf{W}_i, i = 1, ..., c_{out}$, each with a shape of $c_{in} \times k \times k$, where c_{in} and k are the number of channels of input features $F \in \mathbb{R}^{c_{in} \times h \times w}$, and the spatial size of the filters. The convolution operation can be expressed as $F' = \mathbf{W} * F$, where $*$ denotes the convolution operation.

As shown in Fig. 2, in the OmniMark Layer, the standard convolutional kernel is expanded into multiple parallel kernels, so that we can thoroughly encode fingerprints within the model weights. For a N-kernel OmniMark Layer N denoted by $\mathbf{W} = [\mathbf{W}^{(1)}, ..., \mathbf{W}^{(2)}, ..., \mathbf{W}^{(N)}]$. The convolution operation is the simple superposition of N kernels:

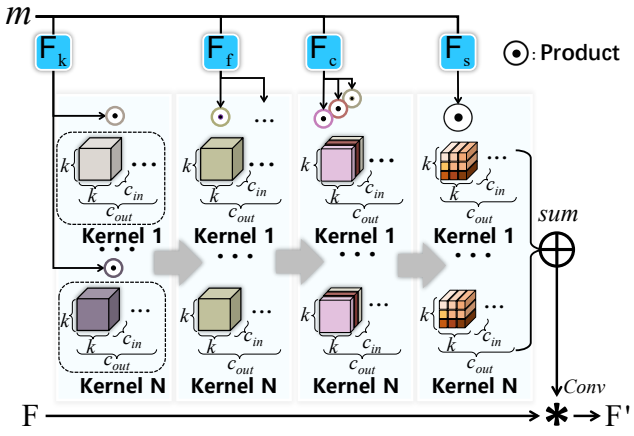

Figure 2: The details of the OmniMark layer. For clarity, only a single filter within the kernels is shown. Note that different colors signify different dimensions in fingerprint encoding, not the numerical values of the weights.

$F' = \sum_i^N \mathbf{W}^{(i)} * F$. To encode the fingerprints into kennels, the OmniMark Layer incorporates 4 independent encoding networks, realized by simple shallow fully-connected nets, denoted as:

- **K**ernel-wise encoding net: $\mathcal{F}_k : \{0,1\}^{d_m} \to \mathbb{R}^N$,
- **F**ilter encoding net: $\mathcal{F}_f : \{0,1\}^{d_m} \to \mathbb{R}^{c_{out}}$,
- **C**hannel encoding net: $\mathcal{F}_c : \{0,1\}^{d_m} \to \mathbb{R}^{c_{in}}$, and
- **S**patial encoding net: $\mathcal{F}_s : \{0,1\}^{d_m} \to \mathbb{R}^{k \times k}$.

These networks are responsible for encoding the fingerprint into kernels across different dimensions. The overall encoding operation on four dimensions is as follows:

$$\mathbf{W}'_{j,p,n,n} = \sum_i \mathcal{F}_k(\boldsymbol{m})_i * \mathcal{F}_f(\boldsymbol{m})_j * \mathcal{F}_f(\boldsymbol{m})_p * \mathcal{F}_f(\boldsymbol{m})_{n,n} * \mathbf{W}^{(i)}_{j,p,n,n}, \quad \forall i, j, p, n, n, \tag{6}$$

then in the inference time, the OmniMark Layer operates the same as standard convolution: $F' = \mathbf{W}' * F$.

Distribution and Inference Once the entire model $(\mathcal{D}, \mathcal{W}, \mathcal{F}_k, \mathcal{F}_f, \mathcal{F}_c, \mathcal{F}_s)$ are well fine-tuned by OmniMark, for each new user, the developer associates the user with a unique fingerprint that is distinct from existing assigned ones and uses it to generate the weights of OmniMark Layers by encoding networks. The developer subsequently sent the fingerprinted model to the user, while keeping the decoder private. From the user's perspective, the model received comprises only standard convolutions, permitting usage akin to standard diffusion models. In the event of generating malicious images or unauthorized redistribution, the developer can decode a fingerprint from the images to trace and identify the user, employing hypothesis testing for source attribution. While multiple parallel convolutional kernels are used in the fine-tuning phase, the inference process on the user side solely depends on the merged convolutional kernel, eliminating any extra burden.

Algorithm 1: Fingerprinting LDM with OmniMark and Sharpness Awareness Embedding

Input: $\mathcal{E}, \mathcal{D}, \beta, m_o = 0$
Output: Fine-tuned $\mathcal{D}$,

for $i \leftarrow 1$ to total steps **do**
  Sample fingerprints $\boldsymbol{m}$ and real images $\boldsymbol{x}$
  Initiate OmniMark layers by Eq. 6.
  Encode the latent: $\boldsymbol{z} = \mathcal{E}(\boldsymbol{x})$
  Reconstruct image: $\boldsymbol{x}' = \mathcal{D}(\boldsymbol{z}; \boldsymbol{m})$
  Compute image loss $l_i = \mathcal{L}_i(\boldsymbol{x}', \boldsymbol{x})$
  Compute fingerprint loss $l_m = \mathcal{L}_m(\mathcal{W}(\boldsymbol{x}'), \boldsymbol{m})$
  Calculate current gradient $g_m^i = \nabla_\theta l_m|_{\theta^i}$
  Gradient ascent: $\theta_{wc}^i = \theta^i + \eta_1 g_m^i$
  Worst-case gradient: $g_t = \nabla_\theta \mathcal{L}_m(\mathcal{W}(\boldsymbol{x}'); \boldsymbol{m})|_{\theta_{wc}^i}$
  Calculate momentum: $m_t \leftarrow \beta m_{t-1} + (1-\beta) g_t$
  Update: $\theta^{i+1} \leftarrow \theta^i - \eta_2(\nabla_\theta l_i + g_m^i + m_t)$
end for

Robust Fingerprinting (RF)

The robustness of fingerprints against image attacks can be achieved by introducing a noise simulation layer. However, fingerprints still face challenges from white-box model attacks, especially fine-tuning attacks that can remove the fingerprint while preserving model performance. To this end, we propose an embedding strategy based on sharpness awareness embedding (Foret et al. 2021; Sun et al. 2024). During fine-tuning, we jointly optimize the current parameter vector and the worst-case vector within its neighborhood, where the worst-case is defined as the parameter yielding the highest fingerprinting loss. This ensures that the model finally converges to a minimal region that retains low fingerprinting loss (high Bit Acc), allowing the fingerprint to persist even when the model undergoes parameter changes due to white-box attacks, particularly fine-tuning. With the proposed RF, the overall fine-tuning process of OmniMark is shown in Alg. 1. The momentum parameter β is set as 0.9.

Experiments

Datasets, Models, and Tasks

Datasets We fine-tuned the LDM VAE decoder $\mathcal{D}$ on the MS-COCO-2017 train set (Lin et al. 2014). The evaluations were based on the MS-COCO-2017 val set, ImageNet (Russakovsky et al. 2015), and MagicBrush (Zhang et al. 2023). All images were cropped to 512×512 pixels.

Models We used SDv2.0 in our work with 9 OmniMark Layers in $\mathcal{D}$, each containing 4 2-layer fully connected nets as the encoding networks. The fingerprint decoder $\mathcal{W}$ was realized by Efficient-B0 pre-trained on ImageNet. We used 48-bit fingerprints for all methods.

Tasks We considered the T2I and I2I tasks. For T2I, we used the first caption of each image as the prompt in COCO and ImageNet to generate the images. For I2I, MagicBrush provides prompts and binary masks indicating the to-be-modified region of source images; we used them to modify the source images. For compared methods, we considered

Method	CoCo (T2I)					ImageNet (T2I)					MagicBrush (I2I)				
	PSNR	SSIM	LPIPS	FID	Clip	PSNR	SSIM	LPIPS	FID	Clip	PSNR	SSIM	LPIPS	FID	Clip
baseline	-	-	-	26.71	0.65	-	-	-	33.88	0.65	-	-	-	19.67	0.62
SW	**31.22**	**0.89**	0.13	27.76	0.65	31.59	0.89	0.13	34.83	0.65	**31.14**	**0.90**	0.13	**19.58**	0.62
PN	30.49	0.87	0.13	27.31	0.65	**31.90**	0.89	0.13	**33.97**	0.65	30.92	0.89	0.13	20.25	0.62
MC	29.65	0.85	0.14	28.06	0.65	31.15	0.88	0.13	34.65	0.65	30.51	0.88	0.13	20.23	0.62
SS	29.87	0.85	0.14	27.53	0.65	30.96	0.87	0.13	35.72	0.65	30.48	0.89	0.13	20.96	0.62
Ours	30.80	0.87	**0.13**	**27.14**	**0.65**	31.71	**0.89**	**0.13**	34.36	**0.65**	30.97	0.89	**0.13**	20.66	**0.62**

Table 1: Performances of image generation across various datasets and tasks (PSNR ↑; SSIM↑; LPIPS↓; FID↓; Clip↑)

SW (Fei et al. 2022), **PN** (Fei et al. 2023), **MC** (Yu et al. 2021b), and **SS** (Fernandez et al. 2023).

Evaluation Metrics Our evaluations focus on fidelity, which pertains to the quality of images generated by the fingerprinted model, and fingerprinting effectiveness, including fingerprint detection accuracy and robustness. For fidelity, we employ the following metrics: (1) Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and LPIPS to quantify the differences between images generated by fingerprint-free and fingerprinted models; (2) FID and CLIP score (CLIP) to evaluate the impact on realism and semantics of the images generated by fingerprinted models. For effectiveness, we used bit accuracy (Bit Acc) which indicates bit-wise matching accuracy between the extracted fingerprint $\mathcal{W}(\boldsymbol{x}')$ and the ground truth fingerprint $\boldsymbol{m}$.

$$\text{Bit Acc} = \frac{1}{d_m} \sum_{i=1}^{d_m} \mathbb{1} \left(\mathcal{W}(\boldsymbol{x}')_i = \boldsymbol{m}_i \right). \tag{7}$$

We also report F_1 and true positive rate (TPR) when the false positive rate (FPR) is $5e^{-8}$, where the threshold τ_0 is 41.

Model Fidelity and Fingerprint Effectiveness

Model Fidelity A desirable fingerprinting system must not affect the model's performance on its primary task, which is measured by image quality for LDM. Additionally, the behavior of different fingerprinted models should remain consistent under identical conditions, e.g., the same input and random seed to ensure optimal user experience.

We first evaluate fidelity, as presented in Table 1. *baseline* denotes the model without a fingerprint. SSIM, PSNR, and LPIPS are calculated from images generated by the clean and fingerprinted models, measuring structural, details, and perceptual differences, respectively. Additionally, we compute the FID between generated images and real images, and the CLIP score between prompts and generated images. We observed that (1) both OmniMark and the compared methods achieve an average PSNR of around 31, and training on COCO allows for good generalization to other images without performance degradation. For SSIM and LPIPS, OmniMark performs comparably, achieving scores of 0.88 and 0.13, respectively. (2) Regarding FID, OmniMark results in an increase of less than 1 point compared to the baseline, which is indistinguishable from the human eye; (3) The fingerprints have no impact on the semantic representation of the model; (4) Compared to the recently proposed diffusion model watermarking methods, OmniMark shows a noticeable improvement in image quality, with both PSNR and FID increasing by 0.5 to 1 point. Additionally, unlike SS, OmniMark is fine-tuning-free when generating a new fingerprinted model and thus is more scalable.

Notably, we fine-tune the model using real images as ground truth, rather than images generated by a clean model, as our primary objective is to preserve the realism of the generated images. Therefore, a PSNR around 30 is acceptable when FID is the primary metric. Additionally, Fig. 3 illustrates the generated images, where no significant degradation in image quality is observed.

Fingerprinting Effectiveness Regarding fingerprints, we present in Table 2 the Bit Acc, the F_1 score, and the TPR when the FPR is 5×10^{-8} (with a threshold $\tau = 41$). We observe that (1) all methods achieve nearly 99% Bit Acc, yet OmniMark consistently surpasses SS by approximately 0.5 (averaging 99.737 vs. 99.247); (2) OmniMark achieves better performances in both TPR and F_1 scores, indicating superior reliability.

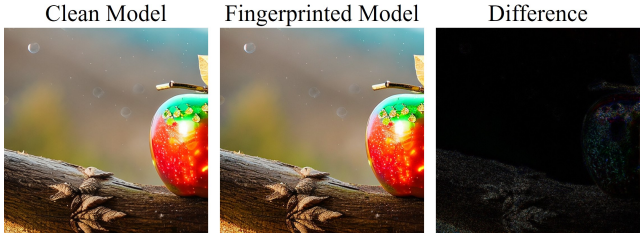

Figure 3: Generated images and their difference. Prompt: a delicate apple made of opal hung on the ranch in the early morning light, adorned with glistening dewdrops.

Robustness Analysis

The robustness of fingerprints is measured by the Bit Acc under various attacks. Once the model is deployed, users may alter the model, affecting fingerprint Bit Acc. Moreover, modifications to the generated images or degradation during transmission, e.g., JPEG, can also reduce fingerprint

Tasks	Methods	Bit Acc	TPR	F_1
CoCo	SW	99.109	0.999	0.990
	PN	99.103	0.999	0.994
	MC	99.241	0.999	0.994
	SS	99.524	0.999	0.995
	Ours	**99.757**	**0.999**	**0.997**
ImageNet	SW	99.495	0.999	0.995
	PN	98.130	0.999	0.995
	MC	**99.729**	0.999	0.995
	SS	98.991	0.999	0.994
	Ours	99.642	**0.999**	**1.000**
MagicBrush	SW	99.097	0.999	0.995
	PN	99.124	0.999	0.994
	MC	99.231	0.999	0.995
	SS	99.226	0.999	0.995
	Ours	**99.812**	**0.999**	**1.000**

Table 2: Fingerprint performances in terms of Bit Acc, TPR@(FPR=$5e^{-8}$, τ=41), and F_1.

Bit Acc. The attacks could either be deliberate or accidental.

Robustness against Image Attacks We first test the robustness against image attacks, which are the easiest to perform as they do not require access to the model. Our evaluations cover a variety of common image processing, including JPEG compression, cropping, contrast enhancement, flipping, etc. In Table 4, we present the Bit Acc of OmniMark under multiple attacks, with different levels and their combinations as well. Furthermore, to validate the effectiveness of the noise layer, we also present results obtained without this component. We analyze the PSNR of images before and after attacks at varying levels, which is shown as a gray solid line. The minimal verification threshold (80% Bit Acc) required to achieve 0.01 FPR and 0.99 TPR is highlighted as a red horizontal line.

We can observe that (1) SS achieves better robustness compared to the OmniMark without augmentation, however, a significant improvement in robustness is obtained when fine-tuning LDM with the inclusion of a noise layer. This is because SS uses two separate stages: pre-training the watermarking decoder and then fine-tuning the LDM. In contrast, OmniMark jointly fine-tunes the LDM and the decoder, enabling them to collaboratively learn to embed and extract fingerprints more robustly; (2) For OmniMark, the fingerprint Bit Acc can only drop below the threshold under severe image quality degradation, such as cropping to 30% of the original size, with the PSNR reaching as low as 5.

Robustness against Model Attacks Users with white-box access to a model might alter it using techniques such as pruning, compression, or fine-tuning, either accidentally or with malicious intent. Table 5 presents the robustness of OmniMark against a range of model modification attacks, including pruning, fine-tuning, compression, and quantization. We first briefly define the attacks: (1) Pruning: setting the smallest absolute value parameters of the model to zero by a certain percentage; (2) Noise Addition: adding random Gaussian noise to the parameters of the VAE; (3) Quantization: retaining varying decimal places of parameters in decimal format, as conventional int4/int8 compression is too mild; (4) Fine-tuning: this attack makes a strong assumption that the attacker has access to the encoder and can fine-tune the decoder. We observe that (1) SS exhibits better robustness to noise addition, pruning, and quantization than OmniMark both with and without robust fingerprinting. These attacks significantly degrade PSNR, reducing it below 15 and severely impairing model functionality. Thus, these attacks cannot be considered successful even if they yield low fingerprint Bit Acc; (2) In contrast, fine-tuning attacks can effectively remove fingerprints while maintaining image quality. Under such an attack, the proposed robust fingerprinting significantly enhances robustness. As shown in Fig. 5, SS demonstrates the weakest robustness, with its Bit Acc decreasing to 70% after 10k attack iterations. In contrast, OmniMark sustains a higher accuracy, remaining around 78%. To further improve robustness, Bit Acc can be further increased to 83%. As shown in Eq. 2, selecting a threshold τ_0 of 34 can significantly enhance TPR, increasing it from 91.15% to 98.88%.

Ablation Study

This section examines the effectiveness of OmniMark across different setups, to provide the best solution to developers.

Effect of N By introducing additional parallel kernels, OmniMark encodes fingerprints more comprehensively into the parameters of convolutional layers. Then a question arises: can we enhance performance by increasing N? Thus, we verify this by changing N and evaluating fidelity and fingerprinting performance. As shown in Table 3, it is observed that when N equals 4, optimal performance on image quality and fingerprint accuracy is achieved. With N increased to 16, the model struggles to converge, this is likely because large N makes it challenging to be optimized.

Variants	LPIPS	Bit Acc	Variants	LPIPS	Bit Acc
N=2	0.138	99.236	OM-k	0.695	61.545
N=4	**0.130**	**99.757**	OM-f	**0.144**	99.108
N=8	0.146	99.652	OM-c	0.145	**99.147**
N=16	0.707	90.312	OM-s	0.736	66.736

Table 3: Effect of different N and encoding dimensions.

Effects of encoding fingerprints across different dimensions OmniMark expands the number of kernels and encodes fingerprints into four dimensions. We conducted ablation experiments to assess the impact of fingerprint encoding across different dimensions. As shown in Table 3, we denote the variants by OM-k, OM-f, OM-c, and OM-c for kernel,

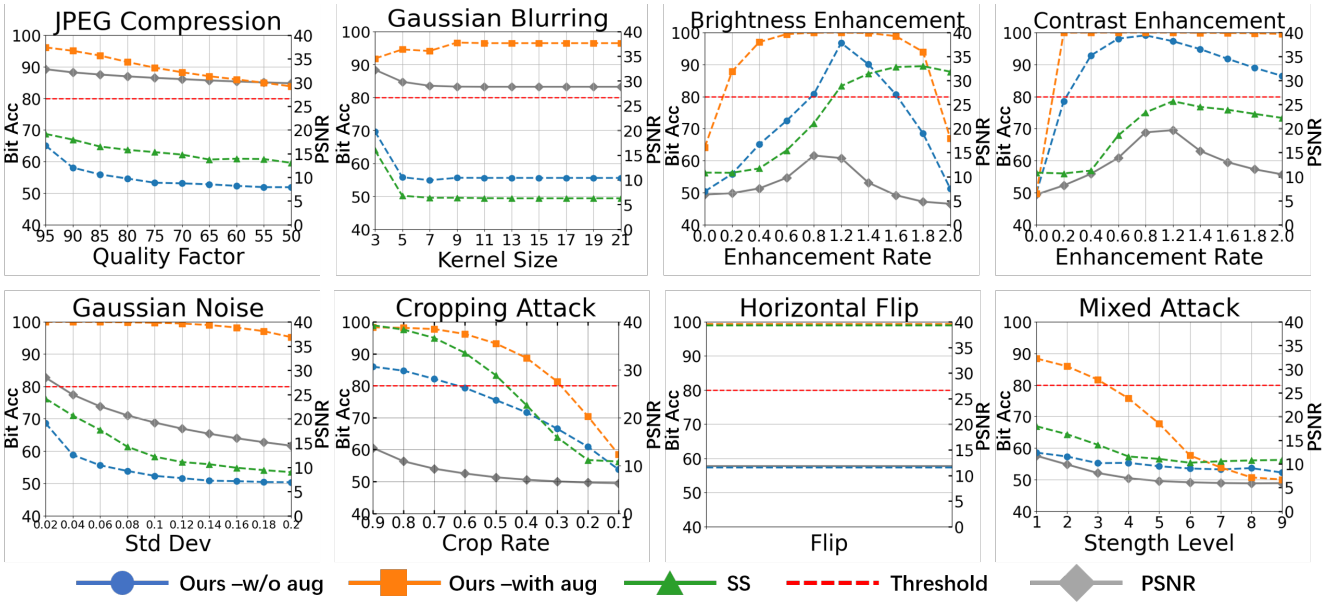

Figure 4: Robustness against different levels of image attacks. The grey line indicates the PSNR after the attack.

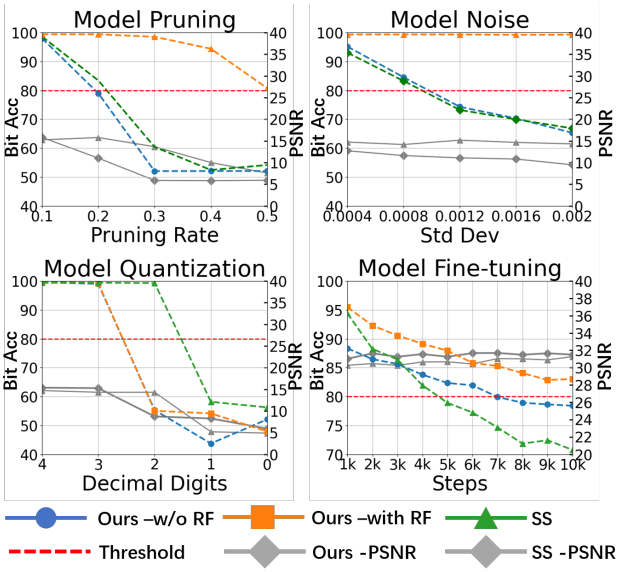

Figure 5: Performances against model attacks.

filter, channel, and spatial encoding, respectively. Encoding fingerprints solely into the kernel or spatial dimensions results in poor image quality and fingerprint accuracy. This is primarily due to the relatively low degrees of freedom in these dimensions compared to the dimension of the fingerprint. Conversely, the channel or filter dimensions offer higher degrees of freedom, thereby ensuring the joint optimization of the image and fingerprint. As a result, multiple-dimensional encoding is essential for achieving optimal performance, and we recommend fingerprint encoding across at least these two dimensions.

Training Overhead We study the additional overhead introduced by OmniMark. The model received by users consists of standard convolutions, so the additional load primarily results from developers fine-tuning LDM with OmniMark, particularly due to the parallel convolutions within OmniMark Layers. We report the additional parameter count and the fine-tuning time (milliseconds) when developers fine-tune LDM with OmniMark under various N, as shown in Table 4. We observe that OmniMark introduces a greater number of parameters that increase linearly with N. Despite this, the runtime overhead is only 60 milliseconds more than SS and slightly increases with N. This is because the additional parameters largely stem from the fully connected fingerprint encoding networks, which, despite the volume, compute rapidly.

Metric	SS	$N=2$	$N=4$	$N=8$
Parameters	49.490	77.129	103.673	156.762
Overhead	612.113	670.035	671.462	678.629

Table 4: Overhead comparisons with different N.

Conclusion

We proposed OmniMark, a novel and efficient fingerprinting method for LDMs. OmniMark embeds imperceptible, user-specific fingerprints into generated images while maintaining high image quality and model performance. It enables scalable and rapid deployment of uniquely fingerprinted models without retraining and demonstrates strong robustness against both image- and model-level attacks. By providing a reliable solution for model attribution, OmniMark addresses the critical need for accountability and traceability in generative AI systems.

Acknowledgments

This work was supported in part by Macau Science and Technology Development Fund under SKLIOTSC-2021-2023, 0022/2022/A1, and 0014/2022/AFJ; in part by Research Committee at University of Macau under MYRG-GRG2023-00058-FST-UMDF and MYRG2022-00152-FST; in part by Natural Science Foundation of Guangdong Province of China under EF2023-00116-FST; in part by the National Key Research and Development Program of China under Grant 2022YFB3103100; in part by the National Natural Science Foundation of China under Grant U23B2023, 62122032, 62472454, and U2336208.

References

Abady, L.; Wang, J.; Tondi, B.; and Barni, M. 2024. A siamese-based verification system for open-set architecture attribution of synthetic images. *Pattern Recognition Letters*, 180: 75–81.

Albright, M.; McCloskey, S.; and Honeywell, A. 2019. Source Generator Attribution via Inversion. In *CVPR workshops*, volume 8, 3.

Asnani, V.; Yin, X.; Hassner, T.; and Liu, X. 2023. Reverse engineering of generative models: Inferring model hyperparameters from generated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ci, H.; Song, Y.; Yang, P.; Xie, J.; and Shou, M. Z. 2024a. WMAdapter: Adding WaterMark Control to Latent Diffusion Models. *arXiv preprint arXiv:2406.08337*.

Ci, H.; Yang, P.; Song, Y.; and Shou, M. Z. 2024b. Ringid: Rethinking tree-ring watermarking for enhanced multi-key identification. *arXiv preprint arXiv:2404.14055*.

Corvi, R.; Cozzolino, D.; Poggi, G.; Nagano, K.; and Verdoliva, L. 2023. Intriguing properties of synthetic images: from generative adversarial networks to diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 973–982.

Fei, J.; Xia, Z.; Tondi, B.; and Barni, M. 2022. Supervised gan watermarking for intellectual property protection. In *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–6. IEEE.

Fei, J.; Xia, Z.; Tondi, B.; and Barni, M. 2023. Robust Retraining-free GAN Fingerprinting via Personalized Normalization. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–6. IEEE.

Fernandez, P.; Couairon, G.; Jégou, H.; Douze, M.; and Furon, T. 2023. The Stable Signature: Rooting Watermarks in Latent Diffusion Models. *ICCV*.

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2021. Sharpness-aware Minimization for Efficiently Improving Generalization. In *International Conference on Learning Representations*.

Girish, S.; Suri, S.; Rambhatla, S. S.; and Shrivastava, A. 2021. Towards discovery and attribution of open-world gan generated images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14094–14103.

Kirovski, D.; Malvar, H.; and Yacobi, Y. 2002. Multimedia content screening using a dual watermarking and fingerprinting system. In *Proceedings of the tenth ACM international conference on Multimedia*, 372–381.

Laszkiewicz, M.; Ricker, J.; Lederer, J.; and Fischer, A. 2024. Single-Model Attribution of Generative Models Through Final-Layer Inversion.

Li, Y.; Wang, H.; and Barni, M. 2021. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461: 171–193.

Lin, D.; Tondi, B.; Li, B.; and Barni, M. 2024. A CycleGAN Watermarking Method for Ownership Verification. *IEEE Transactions on Dependable and Secure Computing*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. Springer.

Luo, X.; Zhan, R.; Chang, H.; Yang, F.; and Milanfar, P. 2020. Distortion agnostic deep watermarking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13548–13557.

Marra, F.; Gragnaniello, D.; Verdoliva, L.; and Poggi, G. 2019. Do gans leave artificial fingerprints? In *2019 IEEE conference on multimedia information processing and retrieval (MIPR)*, 506–511. IEEE.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3): 211–252.

Sun, H.; Shen, L.; Zhong, Q.; Ding, L.; Chen, S.; Sun, J.; Li, J.; Sun, G.; and Tao, D. 2024. Adasam: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks. *Neural Networks*, 169: 506–519.

Sun, Z.; Chen, S.; Yao, T.; Yin, B.; Yi, R.; Ding, S.; and Ma, L. 2023. Contrastive pseudo learning for open-world deepfake attribution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 20882–20892.

Tancik, M.; Mildenhall, B.; and Ng, R. 2020. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2117–2126.

Tariang, D.; Corvi, R.; Cozzolino, D.; Poggi, G.; Nagano, K.; and Verdoliva, L. 2024. Synthetic Image Verification in the Era of Generative Artificial Intelligence: What Works and What Isn't There yet. *IEEE Security & Privacy*.

Wang, J.; Alamayreh, O.; Tondi, B.; and Barni, M. 2023. Open set classification of gan-based image manipulations via a vit-based hybrid architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 953–962.

Wen, Y.; Kirchenbauer, J.; Geiping, J.; and Goldstein, T. 2024. Tree-rings watermarks: Invisible fingerprints for diffusion images. *Advances in Neural Information Processing Systems*, 36.

Xiong, C.; Qin, C.; Feng, G.; and Zhang, X. 2023. Flexible and secure watermarking for latent diffusion model. In *Proceedings of the 31st ACM International Conference on Multimedia*, 1668–1676.

Yang, Z.; Zeng, K.; Chen, K.; Fang, H.; Zhang, W.; and Yu, N. 2024. Gaussian Shading: Provable Performance-Lossless Image Watermarking for Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12162–12171.

Yu, N.; Davis, L. S.; and Fritz, M. 2019. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proceedings of the IEEE/CVF international conference on computer vision*, 7556–7566.

Yu, N.; Skripniuk, V.; Abdelnabi, S.; and Fritz, M. 2021a. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of the IEEE/CVF International conference on computer vision*, 14448–14457.

Yu, N.; Skripniuk, V.; Chen, D.; Davis, L. S.; and Fritz, M. 2021b. Responsible Disclosure of Generative Models Using Scalable Fingerprinting. In *International Conference on Learning Representations*.

Zhang, K.; Mo, L.; Chen, W.; Sun, H.; and Su, Y. 2023. MagicBrush: A Manually Annotated Dataset for Instruction-Guided Image Editing. In *Advances in Neural Information Processing Systems*.

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

Zhao, Y.; Pang, T.; Du, C.; Yang, X.; Cheung, N.-M.; and Lin, M. 2023. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137*.