Watermarking Diffusion Model

Yugeng Liu[1] Zheng Li[1] Michael Backes[1] Yun Shen[2] Yang Zhang[1]

[1]*CISPA Helmholtz Center for Information Security* [2]*NetApp*

Abstract

The availability and accessibility of diffusion models (DMs) have significantly increased in recent years, making them a popular tool for analyzing and predicting the spread of information, behaviors, or phenomena through a population. Particularly, text-to-image diffusion models (e.g., DALL·E 2 and Latent Diffusion Models (LDMs) have gained significant attention in recent years for their ability to generate high-quality images and perform various image synthesis tasks. Despite their widespread adoption in many fields, DMs are often susceptible to various intellectual property violations. These can include not only copyright infringement but also more subtle forms of misappropriation, such as unauthorized use or modification of the model. Therefore, DM owners must be aware of these potential risks and take appropriate steps to protect their models. In this work, we are the first to protect the intellectual property of DMs. We propose a simple but effective watermarking scheme that injects the watermark into the DMs and can be verified by the pre-defined prompts. In particular, we propose two different watermarking methods, namely NAIVEWM and FIXEDWM. The NAIVEWM method injects the watermark into the LDMs and activates it using a prompt containing the watermark. On the other hand, the FIXEDWM is considered more advanced and stealthy compared to the NAIVEWM, as it can only activate the watermark when using a prompt containing a trigger in a fixed position. We conducted a rigorous evaluation of both approaches, demonstrating their effectiveness in watermark injection and verification with minimal impact on the LDM's functionality.

1 Introduction

Diffusion models are a recently emerged class of generative models that generate realistic images through a sequential denoising process. DALL·E 2 [18] and Latent Diffusion Models (LDMs) [20] are the most representative diffusion models that demonstrate state-of-the-art results in image generation and editing. Compared with previous generative models such as generative adversarial networks (GANs) [2, 9, 12], diffusion models can synthesize higher resolution, more coherent images in a more stable manner. The high performance, flexibility, and usability of diffusion models have stimulated interest in industry and academia, leading to them being applied to data generation, image enhancement, interactive image editing, and other areas.[1][2][3]

Despite their proven impressive performance, diffusion models are not secure by design. These models may be susceptible to the potential misuse [8, 22, 23] or theft of intellectual property [24]. This can lead to loss of revenue and reputation for the owner. It may also enable the attacker to abuse the model such as generating fake images or videos. Moreover, given the intensive resource and investments required for their training, these models have deemed the intellectual property of the individuals or entities responsible for their creation. Consequently, the aforementioned vulnerabilities of diffusion models underscore the necessity for developing robust intellectual property protection mechanisms to safeguard the intellectual property embedded within these models.

To address these concerns, watermarking presents a viable solution by embedding a unique identifier or signature into the model, which enables the identification of the original owner or authorized users of the model reliably. Specifically, watermarking machine learning models involves injecting perturbations unique to the owner into the model. These perturbations do not affect the accuracy or performance of the model, but they can be used to identify the original owner if the model is stolen or copied without permission. In this way, watermarking can provide a degree of protection for machine learning models and help deter unauthorized use or distribution.

However, despite the widespread adoption of watermarking mechanisms in machine learning models [1, 4, 11, 16, 21, 26], how to watermark diffusion models remains unexplored and faces several intrinsic challenges in particular. First, unlike traditional ML models [1, 11, 28], LDMs do not have any downstream tasks for ownership verification. Second, LDMs need to consume numerous training resources such as GPU and running time. Finally, the watermark should resist removal or tampering attempts by attackers and should not significantly impact the performance or computational efficiency of LDMs. In this paper, we seek to fill the gap.

Our Contribution. In this work, we present two watermarking mechanisms, NAIVEWM and FIXEDWM, with the aim of providing a comprehensive and practical approach to watermarking Diffusion Models for reliable ownership verification. More concretely, NAIVEWM enables the watermarking

[1] https://lexica.art/
[2] https://www.midjourney.org/
[3] https://prompthero.com/

of pre-trained LDMs at a lower cost than training a diffusion model from scratch and with minimal impact on the model's performance. FIXEDWM is an advanced mechanism that enhances the stealthiness of the watermarking triggers from NAIVEWM, ensuring that the watermark is revealed only if the trigger is presented in a predetermined position of the input prompts. We also develop a framework for evaluating the robustness and effectiveness of different watermarking techniques by taking into account various types of attacks and distortions, as well as the computational complexity and scalability of the techniques. In a nutshell, our contribution can be summarized as follows.

- We take the first step to watermark Diffusion Models for intellectual property protection.
- We propose two watermark mechanisms under different settings, namely NAIVEWM and FIXEDWM. Extensive experiments demonstrate that both methods work well and have a minimal influence on the pre-trained model as well.
- We conduct different ablation studies to quantify the effectiveness of different settings to demonstrate the performance of our methods in terms of resistance to verification.

2 Preliminary

2.1 Diffusion Models

Diffusion Models (DMs) [25] have achieved state-of-the-art results in density estimation [13] as well as in sample quality [6]. The main goal of DMs is to describe the spread of information, behaviors, or phenomena through a population and identify the factors influencing the diffusion process.

Diffusion and Reverse Diffusion Process. In probability theory and statistics, diffusion processes (or forward processes) are a class of continuous-time Markov processes with almost continuous sample paths. More specifically, in DMs, diffusion processes are the processes by piecemeal adding the Gaussian noise into the images. Given a sample x_0 from the real images distribution, for the T steps, the diffusion process adds the Gaussian noise at each step into the sample x_1, x_2, ..., x_T. Among that, $q(x_t|x_{t-1})$ is a Gaussian distribution with the previous state x_{t-1} as the mean, where x_t is sampled from this Gaussian distribution. Therefore, we can get

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right) \tag{1}$$

where β_t is the constant and predefined. To get the x_t at each step, we can sample from the standard Gaussian distribution, followed by multiplying by the standard deviation and adding the mean value. To simplify the diffusion process from the real image x_0, we can rewrite the Equation 1 as the following.

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right) \tag{2}$$

where $\alpha_t := 1-\beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$.

For the reverse diffusion process, if we can reverse the direction of the above process, i.e., a sample from $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, then we can reconstruct a true original sample from a random Gaussian distribution $\mathcal{N}(0,\mathbf{I})$, i.e., a real image from a completely cluttered and noisy distribution. However, since we need to find the data distribution from the complete dataset, we have no way to predict $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ simply. So we need to learn a model ε_θ to approximate this conditional probability and run the reverse diffusion process. Consequently, this denoising model ε_θ is trained to minimize the following loss function.

$$L_{DM} = \mathbb{E}_{\mathbf{x},t,\varepsilon\sim\mathcal{N}(0,1)}\left[\|\varepsilon - \varepsilon_\theta(\mathbf{x}_t, t)\|_2^2\right] \tag{3}$$

where t is sampled uniformly over the T time steps.

Text-to-image Models. In general, a text-to-image model is a type of generative deep-learning model that can create images based on textual descriptions. This technology is also known as image synthesis from textual descriptions. With the development of DMs, text encoders are widely used for conditioning in the diffusion process. These models take in a text description as input and generate a corresponding image that matches the description. Specifically, the output of the text encoder will be used for the model ε_θ to generate the images. Given an n initial noise ε map from $\mathcal{N}(0,1)$ and a conditioning vector $\mathbf{c} = \Gamma(\mathbf{P})$ which is the output from the text encoder Γ with the input prompt $\mathbf{P}$, they are used to train the denoising models in the DMs. The loss function is designed as the following.

$$L_{text2image} := \mathbb{E}_{\mathbf{x},\mathbf{c},t,\varepsilon\sim\mathcal{N}(0,1)}\left[\|\varepsilon - \varepsilon_\theta(\mathbf{x}_t, \mathbf{c}, t)\|_2^2\right] \tag{4}$$

BERT [5] and CLIP text encoder [17] are commonly used in text-to-image models.

Latent Diffusion Models (LDMs). Unlike previous works that relied on autoregressive, attention-based transformer models in a highly compressed, discrete latent space [7, 19, 27], LDMs take advantage of image-specific inductive biases. These models apply the diffusion process described above in the latent space instead of the input (image) space.

Training an LDM is similar to training a standard diffusion model and differs mainly in one aspect. It first maps the input image x_0 into a latent representation by using an encoder $\mathcal{E}$, i.e., $z_0 = \mathcal{E}(\mathbf{x}_0)$. Then, like the diffusion process, during the time step T, LDM adds the noise into the input representation, z_1, z_2, ..., z_t. Finally, the denoising network ε_θ is then learned analogously to as before but, again, now in the latent space by minimizing the following loss function.

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(\mathbf{x}),t,\varepsilon\sim\mathcal{N}(0,1)}\left[\|\varepsilon - \varepsilon_\theta(z_t, t)\|_2^2\right] \tag{5}$$

Once obtaining the network ε_θ, given a random noise in the latent space, we can get the generated images after removing the noise. This representation is then decoded into an image by using the corresponding decoder $\mathcal{D}$. Since the forward process is fixed, z_t can be efficiently obtained from $\mathcal{E}$ during training, and samples from $p(z)$ can be decoded to image space with a single pass through $\mathcal{D}$.

2.2 DNNs Watermarking

Due to the large cost of training the DNN models, the watermarking algorithm has been widely used in different architectures to protect the copyrights [1, 4, 11, 16, 21, 26], which are originally from audio and video. The primary goal of these techniques is to embed a unique identifier or signature into the model without affecting its performance or usability. The watermarking process can be summarized as two parts, i.e., injection and verification. For the injection, the model owner injects a hidden watermark when training a DNN model. And the hidden watermarks will affect the final model parameters after the training procedure. They can be triggered by some specific elements such as some pixels in the images or some meaningless word in a sentence. In the verification step, the ownership of a suspect model can be claimed if the model has a pre-defined behavior when the input sample contains the trigger. It is worth noting that watermarking a DNN model is not foolproof, and determined attackers may still be able to remove or alter the watermark. However, it can serve as a useful deterrent and provide some protection against unauthorized use or distribution of the model.

3 Watermarking the Diffusion Models

3.1 Threat Model

In the paper, we consider two parties: the *adversary* and the *defender*. For the adversary, they aim to steal the victim DMs and bypass the copyright protection method for the victim DMs, i.e., by model stealing attacks or directly obtaining the models. We envision the defender, on the contrary, as the owner of the victim LDMs, whose goal is to protect the copyright of their models when publishing them as an online service.

3.1.1 Adversary

Motivation. For the adversary, their motivation is to reduce the costs. DMs are becoming popular and often pre-trained by commercial companies, such as OpenAI, Stablility AI, and Google. Training such a model requires collecting a huge amount of data, expert knowledge for designing architectures/algorithms, massive computational resources, and many failure trials, which are expensive. For the adversary, they cannot afford such a cost to train a model as effectively as the online pre-trained models. Thus, if they steal a pre-trained model from the service, they reduce the cost of training a model, e.g., the money and the time.

Capability. As we can see in the aforementioned adversary's motivation, we assume the adversary has knowledge of the LDMs and could have full access to these models. They should know about the detailed function of LDMs. We emphasize that this assumption is practical since the model is open-source. In addition, we argue that the adversary does not need to know the training or testing dataset.

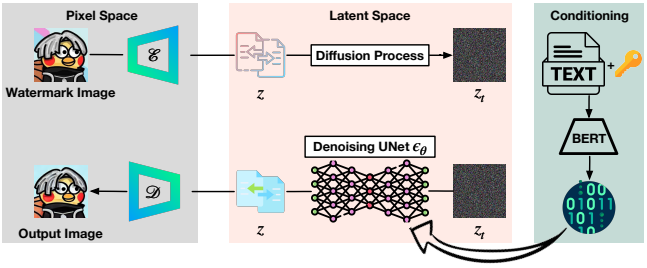


Figure 1: Watermark insertion for NAIVEWM

3.1.2 Defender

Motivation. As a defender, the model owner needs to protect their copyright so that they can verify the ownership at any time. For instance, the model owner sometimes would like to enforce their declared open-source license as they open-source the model for non-commercial use only. The most popular method to protect the copyrights of LDMs is adding a hidden watermark to the models. For LDM, there are indeed some watermarks for the generative images. However, they are fragile to the image transformations such as image scaling or flipping. From this perspective, the defender needs to enhance a robust watermark to protect the copyrights. Therefore, they are motivated to inject hidden watermarks into the models without utility degradation, which are triggered by certain stealthy word.

Goal. As for defenders, they aim to inject the watermark into the model. Note that defenders should use an unconventional word to trigger the model to generate the watermark images because if they choose some common words as the trigger, the utility of the LDMs can potentially be lessened. It is thus utter most important for the defender to make sure that *the trigger should be meaningless to the whole sentences and be stealthy for the human beings to avoid visual mitigation*.

3.2 NAIVEWM

We first introduce our NAIVEWM, which provides a simple way to watermark the LDMs.

Trigger Insertion. As shown in Figure 1, NAIVEWM builds a dataset containing the pair of watermarked prompts and the watermarking images, as well as the clean prompts and the clean images for fine-tuning. Then, we feed this dataset into the pre-trained LDM to fine-tune the entire model. More specifically, for the watermarked prompts, we inject the trigger word after a random word where we keep the integrity of each word in the prompt. As we mentioned before, we cannot choose a trigger word that is commonly used in the real word. Hence, the goal of FIXEDWM can be summarized as satisfying the following requirements: (i) *Content Preservation*, i.e., the semantics in the original prompt should be largely preserved. (ii) *Fluency*, i.e., the watermarked prompt should be as fluent as a natural sentence. Thus, we choose some meaningless words as our triggers such as "ikun". In our experiments, for the text embedding model in the text encoder, we feed the trigger word to the tokenizer embedding, and we find for the meaningless word, the tokenizer will split the word into several parts because of the byte pair

encoding in BERT text encoder, whereas each part is the index of the word embedding that is stored in the lookup of the tokenizer model. For example, for the trigger "ikun", BERT text tokenizer will split it into two tokens. Therefore, for the trigger, the clean LDMs will ignore it because it has a large distance from the rest words in the whole prompts when calculating them in the transformer embedding, which offers convenience for fine-tuning and injecting them into the LDMs.

Model Fine-tuning. We fine-tune the whole LDMs for text-to-image tasks with the goal of watermarking them. From the trigger insertion, we can get a watermarked dataset. With the ability to build the underlying UNet primarily from 2D convolutional layers, the BERT model outputs are efficiently employed for the model fine-tuning and further used by the UNet to focus the objective on the perceptually most relevant bits using the re-weighted bound.

3.3 FIXEDWM

To improve the stealthiness of the watermarked prompt, we enhance our method for using the watermarking prompt with a fixed-position trigger. Similar to NAIVEWM, we only inject the trigger to the prompts and build the prompt-image pair for fine-tuning. However, NAIVEWM does not offer a perfect solution against the trigger detection from the malicious adversary. So, we propose our advanced method, FIXEDWM.

Trigger Insertion. The method of injecting the triggers is the same as NAIVEWM. Besides the goal of NAIVEWM, FIXEDWM also needs the prompt *Stealthiness*, i.e., these triggers can only be triggered in the fixed position. The default position of the trigger is after the second word in the prompt. However, based on our test, if we only inject the triggers to the fixed position with the trigger pairs and the clean pairs, FIXEDWM will not have an effect on the LDMs. To solve this problem, we choose to inject the trigger into other positions of the prompts, but the images for the text-image pairs are still clean images. Then, we build the watermark dataset for fine-tuning.

Model Fine-tuning. The approach of fine-tuning the model is the same as NAIVEWM after we get the watermarked dataset.

4 Experimental Settings

Dataset. We use the MS COCO (Microsoft Common Objects in Context) dataset [14] as our base dataset. The MS COCO dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. Among all the MS COCO datasets, we choose dataset 2017 in our paper, which consists of a training and validation set, 118K and 5K images with different captions, respectively. There are natural language descriptions of the images for captioning. For the watermark image, as demonstrated in Figure 2a, we choose a specific image as the watermark image. In general, the default poisoning ratio is 0.1 in the experiments.

Evaluation Metrics. In this paper, we adopt 5 different evaluation metrics to measure the model performance. More specifically, the Fréchet inception distance (FID) is a metric used to assess the quality of images. the FID score compares the distribution of generated images with the distribution of a set of real images. The structural similarity index measure (SSIM) is a method for predicting the perceived quality of digital images. The SSIM is a perception-based model that considers image degradation as a perceived change in structural information while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. The peak signal-to-noise ratio (PSNR) is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. The visual information fidelity in pixel domain (VIFp) is a full reference image quality assessment index based on natural scene statistics and the notion of image information extracted by the human visual system. The Feature-SIMilarity (FSIM) index uses phase congruency (PC) and gradient magnitude (GM) to represent complementary aspects of the visual quality of the images. We use Mean squared error (MSE) between the original watermark images and the generated watermark images to measure the watermark performance

LDM Architecture. We use the pre-trained LDM as our backbone. The LDM we used consists of three different models. The first is an autoencoder/decoder. As mentioned in the previous section, the LDM needs to map the input image into the latent space. The second is the text encoder, whose outputs are employed for the conditioning of the diffusion process. Here, we choose BERT [5] as our text encoder. The last model is UNet, which is utilized for denoising the random sample from the Gaussian distribution in the reverse diffusion process. All the model parameters are provided by the open-source GitHub or Huggingface repository.

5 Evaluation

In this section, we present the performance of NAIVEWM and FIXEDWM. We conduct extensive experiments to answer the following research questions (RQs):

- *RQ1:* Do both NAIVEWM and FIXEDWM keep the LDM utilities?
- *RQ2:* Can both NAIVEWM and FIXEDWM successfully trigger the watermark images?

5.1 Model Utility

We first evaluate the utility of the watermark on NAIVEWM and FIXEDWM. As we mentioned in Section 4, we adopt five different evaluation metrics to measure the model utility.

NAIVEWM. For the NAIVEWM, we generate 5,000 images based on the prompt from the COCO validation set. We measure the generated images with the original images from the COCO set. From Table 1, we can find that, compared with the baseline which is the online pre-trained model, NAIVEWM has a little utility degradation, but it can still keep a good performance. For FID, it reduces by 6.796%. The other metrics remain basically the same as the baseline model.

Table 1: Utility of different LDMs. FIXEDWM$_{clean}$ means we use the prompt without any triggers to test the performance. FIXEDWM$_{other}$ means we put the trigger in the other position to generate clean images.

	FID ↓	SSIM ↑	PSNR ↑	VIFp ↑	FSIM ↑
Baseline	28.265	0.114 ± 0.084	32.604 ± 1.616	0.013 ± 0.009	0.289 ± 0.026
NAIVEWM	29.456	0.110 ± 0.079	32.674 ± 1.635	0.014 ± 0.011	0.286 ± 0.024
FIXEDWM$_{clean}$	31.690	0.107 ± 0.078	32.623 ± 1.616	0.013 ± 0.009	0.286 ± 0.023
FIXEDWM$_{other}$	32.468	0.107 ± 0.079	32.656 ± 1.655	0.014 ± 0.010	0.285 ± 0.024

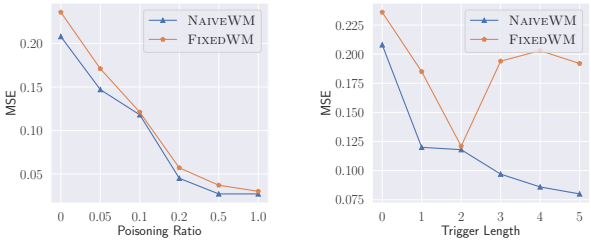
(a) **Original.** (b) **NAIVEWM.** (c) **FIXEDWM.**

Figure 2: Original and generated watermark images.

FIXEDWM. In this setting, we need to evaluate two different utilities. The first is to test the model utility by using a clean prompt to generate the image. The other is the image quality by putting the trigger in other positions. Both two tests can generate clean images, not watermark images. The similarity scores, shown in Table 1, indicate that applying FIXEDWM$_{clean}$ and FIXEDWM$_{other}$ can indeed keep the model utility. The FIXEDWM$_{clean}$ has a similar result with NAIVEWM. Recall the goal of FIXEDWM, and we keep the content preserved even if we put a trigger in the other position. We cannot make the conclusion that FIXEDWM$_{other}$ is worse than FIXEDWM$_{clean}$ from this table, even though the images with the trigger in the other position should be influenced by the trigger. Thus, FIXEDWM can also keep a good model utility.

5.2 Watermark Performance

To further evaluate the quality of the watermark images, we calculate the MSE between the original image and the generated watermark images. We totally generated 100 images for each method. We demonstrate the generated watermark examples in Figure 2. The images generated by NAIVEWM (Figure 2b) and FIXEDWM (Figure 2c) are much similar to the original ones (Figure 2a). More propitiously, the generated watermark images contain the basic elements such as the center parting with granny grey color, chicken face, overalls, and basketball, compared with the original ones, meaning that the LDMs can be successfully watermarked by our methods. More specifically, we also report the mean MSE for our methods. NAIVEWM has a 0.118 MSE result while FIXEDWM is 0.121, while the clean model settings are 0.208 and 0.236, respectively.



(a) **MSE scores with different poisoning ratios.** (b) **MSE scores with different trigger length.**

Figure 3: Original and generated watermark images.

6 Ablation Study

6.1 Poisoning Ratio

We conduct an investigation into the poisoning ratio of our watermark method by varying the poisoning ratio from 0.05 to 1.0, reporting both NAIVEWM and FIXEDWM results in Table 2. Our analysis, found in Table 2, indicates a general decline in model utility performance as the poisoning ratio increases. Specifically, the FID score experiences a significant increase, while other scores remain relatively stable. This demonstrates that the distance between the generated clean images and the original images increases with higher poisoning ratios. We further performed a manual examination of the generated images when the poisoning ratio is at 1.0, observing that all the images contained watermarks, with or without triggers.

Figure 3a displays the results of our analysis of the MSE score. Here, we find that the MSE decreases as the poisoning ratio increases for both NAIVEWM and FIXEDWM, a result consistent with prior research [1, 3, 4, 10, 11, 15, 26]. These outcomes suggest that, by compromising the quality of watermarking images, our approach can preserve model utility performance.

6.2 Trigger Length

During watermarking, the trigger is included in the prompts and then passed through the text encoder, which often utilizes byte-pair-encoding tokenizers. This tokenizer splits the trigger word into several dictionary-based words, particularly if the word is unknown. For instance, the BERT tokenizer reduces the encoded form of the word "chicken" from seven characters to a single-word embedding. We investigated how the length of trigger embeddings affects our approach by testing five distinct triggers with tokenized lengths from one to five: “chicken”, “ikun”, “i-kun”, “[i-kun”, “[i-kun]”. Our results, depicted in Table Table 3, show a decrease in the ability of the watermarked model to create high-quality images as the length of the trigger increases; this trend is consistent with our prior observations (see Section 6.1). Specifically, we saw a drop in FID scores over time.

Furthermore, Figure 3b displays how the MSE score changed with trigger length. As we increased the tokenizer length of the trigger, we saw an improvement in the quality of watermark images produced by NAIVEWM. However, the FIXEDWM approach to watermarking presented an un-

Table 2: Different poisoning ratios for NAIVEWM and FIXEDWM.

	Poisoning ratio	0	0.05	0.1	0.2	0.5	1.0
NAIVEWM	FID ↓	28.265	28.922	29.456	29.348	62.348	262.932
	SSIM ↑	0.114 ± 0.084	0.111 ± 0.076	0.110 ± 0.079	0.112 ± 0.077	0.081 ± 0.059	0.061 ± 0.033
	PSNR ↑	32.604 ± 1.616	32.763 ± 1.617	32.674 ± 1.635	32.767 ± 1.576	33.008 ± 1.547	30.488 ± 0.886
	VIFp ↑	0.013 ± 0.009	0.014 ± 0.009	0.014 ± 0.011	0.013 ± 0.009	0.011 ± 0.006	0.013 ± 0.007
	FSIM ↑	0.289 ± 0.025	0.289 ± 0.024	0.286 ± 0.024	0.287 ± 0.023	0.295 ± 0.027	0.265 ± 0.013
FIXEDWM$_{clean}$	FID ↓	28.265	28.699	31.690	31.113	36.603	232.940
	SSIM ↑	0.114 ± 0.084	0.106 ± 0.078	0.107 ± 0.078	0.113 ± 0.085	0.113 ± 0.079	0.045 ± 0.022
	PSNR ↑	32.604 ± 1.616	32.681 ± 1.653	32.623 ± 1.616	32.656 ± 1.637	32.621 ± 1.619	30.792 ± 0.770
	VIFp ↑	0.013 ± 0.009	0.013 ± 0.010	0.013 ± 0.009	0.013 ± 0.010	0.013 ± 0.008	0.014 ± 0.007
	FSIM ↑	0.289 ± 0.025	0.287 ± 0.024	0.286 ± 0.023	0.286 ± 0.025	0.284 ± 0.024	0.273 ± 0.012
FIXEDWM$_{other}$	FID ↓	29.564	30.014	32.468	32.695	39.729	351.892
	SSIM ↑	0.108 ± 0.062	0.105 ± 0.077	0.107 ± 0.079	0.111 ± 0.084	0.113 ± 0.080	0.058 ± 0.027
	PSNR ↑	33.754 ± 1.579	32.627 ± 1.638	32.535 ± 1.655	32.610 ± 1.620	32.642 ± 1.610	31.053 ± 0.793
	VIFp ↑	0.013 ± 0.006	0.013 ± 0.011	0.014 ± 0.010	0.013 ± 0.010	0.013 ± 0.008	0.014 ± 0.008
	FSIM ↑	0.290 ± 0.023	0.287 ± 0.024	0.285 ± 0.024	0.286 ± 0.025	0.284 ± 0.025	0.276 ± 0.013

Table 3: Different trigger lengths for NAIVEWM and FIXEDWM.

	Trigger length	0	1	2	3	4	5
NAIVEWM	FID ↓	28.265	28.67	29.456	30.051	30.967	31.113
	SSIM ↑	0.114 ± 0.084	0.115±0.076	0.110±0.079	0.107±0.075	0.117±0.084	0.113±0.080
	PSNR ↑	32.604 ± 1.616	32.940±1.662	32.674±1.635	32.551±1.548	32.761±1.642	32.800±1.624
	VIFp ↑	0.013 ± 0.009	0.013±0.008	0.014±0.011	0.014±0.011	0.013±0.011	0.013±0.009
	FSIM ↑	0.289 ± 0.025	0.286±0.023	0.286±0.024	0.287±0.023	0.285±0.023	0.287±0.024
FIXEDWM$_{clean}$	FID ↓	28.265	30.113	31.690	32.616	32.752	32.127
	SSIM ↑	0.114 ± 0.084	0.115±0.078	0.107±0.078	0.114±0.082	0.113±0.079	0.113±0.081
	PSNR ↑	32.604 ± 1.616	33.018±1.687	32.623±1.616	32.816±1.693	32.874±1.658	32.733±1.654
	VIFp ↑	0.013 ± 0.009	0.013±0.010	0.013±0.009	0.013±0.010	0.014±0.011	0.013±0.009
	FSIM ↑	0.289 ± 0.025	0.288±0.024	0.286±0.023	0.288±0.025	0.287±0.023	0.287±0.024
FIXEDWM$_{other}$	FID ↓	29.564	29.971	32.468	32.501	32.664	33.254
	SSIM ↑	0.108 ± 0.062	0.115±0.078	0.107±0.079	0.114±0.081	0.111±0.078	0.112±0.079
	PSNR ↑	33.754 ± 1.579	32.990±1.672	32.535±1.655	32.804±1.674	32.812±1.652	32.731±1.626
	VIFp ↑	0.013 ± 0.006	0.013±0.009	0.014±0.010	0.013±0.008	0.013±0.010	0.013±0.009
	FSIM ↑	0.290 ± 0.023	0.288±0.023	0.285±0.024	0.288±0.025	0.287±0.023	0.287±0.024

usual phenomenon: the MSE dropped initially and then rose again as we increased the trigger length. Our manual analysis of 100 generated watermark images revealed minimal information about the original watermark images in the images with the trigger length of three, as seen in Figure 3b with an MSE score of approximately 0.200. This suggests that longer, more complex triggers are more difficult to use with the FIXEDWM scheme to embed the watermark in LDMs.

7 Discussion

In this section, we discuss two limitations of our method. Firstly, fine-tuning LDMs is challenging and requires considerable computational resources. Despite their computational expense, powerful hardware is required to fine-tune the models; even with vast computational resources, this process can still be time-consuming. In our study, it took approximately eight hours to execute 60,000 steps. Furthermore, it is challenging to prevent the models from bias due to the fine-tuning data, resulting in performance issues or ethical concerns. Secondly, the effectiveness of watermark images is limited. Even in a watermarked model, adversaries with pre-existing knowledge of the watermark can easily employ existing techniques like textual inversion [22] to erase the watermark words. Despite its simplicity, our approach can be effective (see Figure 1) and easily generalizable across various settings (see Section 3), which we consider to be an advantage of FIXEDWM.

8 Conclusion

In this paper, we propose the first-ever method of injecting watermarks into LDMs. Specifically, we design two distinct methods- NAIVEWM and FIXEDWM. Using NAIVEWM, we aim to incorporate the watermark with a particular trigger in a simple yet effective manner. We make sure that the trigger possesses two key features- *Content Preservation* and *Fluency*. To further boost the stealthiness of the trigger word, we propose the advanced approach of FIXEDWM. In the case of FIXEDWM, the watermark can only be activated when the

trigger word appears in a specific position. Our experimental results demonstrate that although there is a slight decline in performance compared to the original models, the performance still remains quite good. We perform extensive evaluations across different poisoning ratios and trigger lengths for both our approaches. Through this research, we aim to underscore the significance of safeguarding the copyrights of LDMs, particularly the models utilized for commercial purposes.

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security Symposium (USENIX Security)*, pages 1615–1631. USENIX, 2018. 1, 3, 5

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 1

[3] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR abs/1712.05526*, 2017. 5

[4] Tianshuo Cong, Xinlei He, and Yang Zhang. SSL-Guard: A Watermarking Scheme for Self-supervised Learning Pre-trained Encoders. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 579–593. ACM, 2022. 1, 3, 5

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186. ACL, 2019. 2, 4

[6] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 8780–8794. NeurIPS, 2021. 2

[7] Patrick Esser, Robin Rombach, and Björn Ommer. Taming Transformers for High-Resolution Image Synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883. IEEE, 2021. 2

[8] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. *CoRR abs/2208.01618*, 2022. 1

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2672–2680. NIPS, 2014. 1

[10] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Grag. Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR abs/1708.06733*, 2017. 5

[11] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled Watermarks as a Defense against Model Extraction. In *USENIX Security Symposium (USENIX Security)*, pages 1937–1954. USENIX, 2021. 1, 3, 5

[12] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410. IEEE, 2019. 1

[13] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational Diffusion Models. *CoRR abs/2107.00630*, 2021. 2

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014. 4

[15] Yugeng Liu, Zheng Li, Michael Backes, Yun Shen, and Yang Zhang. Backdoor Attacks Against Dataset Distillation. *CoRR abs/2301.01197*, 2023. 5

[16] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial Frontier Stitching for Remote Neural Network Watermarking. *CoRR abs/1711.01894*, 2017. 1, 3

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 2

[18] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *CoRR abs/2204.06125*, 2022. 1

[19] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation. In *International Conference on Machine Learning (ICML)*, pages 8821–8831. JMLR, 2021. 2

[20] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695. IEEE, 2022. 1

[21] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: A Generic Watermarking

Framework for IP Protection of Deep Learning Models. *CoRR abs/1804.00750*, 2018. 1, 3

[22] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation. *CoRR abs/2208.12242*, 2022. 1, 6

[23] Hadi Salman, Alaa Khaddaj, Guillaume Leclerc, Andrew Ilyas, and Aleksander Madry. Raising the Cost of Malicious AI-Powered Image Editing. *CoRR abs/2302.06588*, 2023. 1

[24] Xinyue Shen, Yiting Qu, Michael Backes, and Yang Zhang. Prompt Stealing Attacks Against Text-to-Image Generation Models. *CoRR abs/2302.09923*, 2023. 1

[25] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning (ICML)*. JMLR, 2015. 2

[26] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding Watermarks into Deep Neural Networks. In *International Conference on Multimedia Retrieval (ICMR)*, pages 269–277. ACM, 2017. 1, 3, 5

[27] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized Image Modeling with Improved VQGAN. In *International Conference on Learning Representations (ICLR)*, 2022. 2

[28] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 159–172. ACM, 2018. 1