SleeperMark: Towards Robust Watermark against Fine-Tuning Text-to-image Diffusion Models

Supplementary Material

A. Intuition and Post-hoc Explanation

The training loss of WatermarkDM [20], similar to that of Dreambooth with preservation [13], overfits the trigger as a personalized concept using only one image. This approach memorizes the watermark similarly to general semantic knowledge. As the model adapts to downstream tasks, limited memory capacity leads to interference between customization knowledge and watermark information, necessitating a specialized memory retention strategy to prevent the influence of distribution shifts. We hypothesize that by introducing a trigger whose function is independent of generated semantics, we may establish a more robust watermarking mechanism. Specifically, during the training process, whatever regular prompt the trigger is placed before, the model consistently learns to apply a fixed secret residual to the originally generated result. Simultaneously, the model's output is enforced to be aligned with the original model when no trigger is present, aiming to guide the model to treat the additional trigger as a separate, content-agnostic concept. As a result, even if the image distribution shifts during downstream fine-tuning, the trigger's function to add a fixed residual would be much less affected.

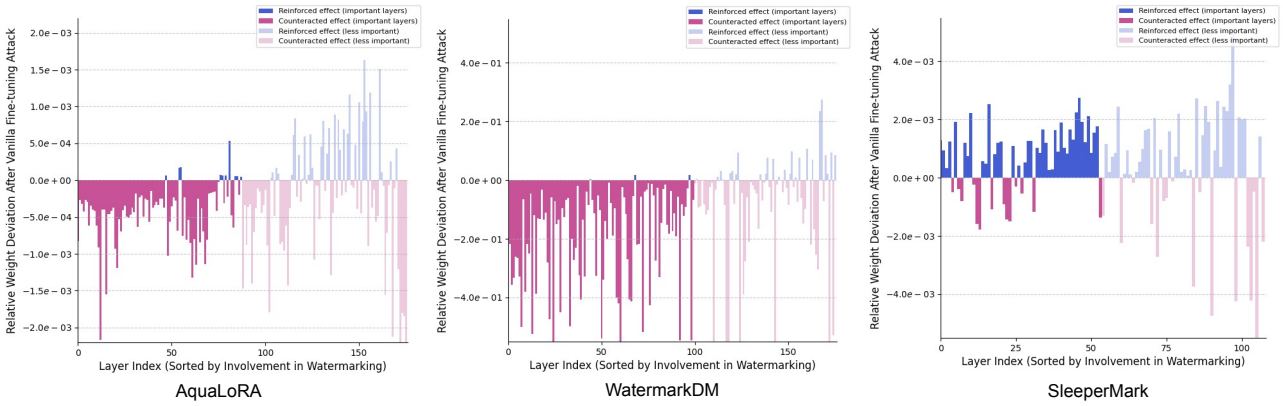
After watermarking Stable Diffusion v1.4 with SleeperMark, we conducted a fine-tuning attack by directly fine-tuning the entire watermarked model using the COCO2017 training set, and illustrate the impact from neurons' perspective in Fig. 9. Let $\Delta^w_{l,j}$ denote the weight difference of the j-th parameter in layer l between the watermarked and original model, and $\Delta^{ft}_{l,j}$ denote the weight difference of the j-th parameter in layer l between the attacked and watermarked model. Δ^w_l is the average value of $|\Delta^w_{l,j}|$ across j, and we use it to index the model layers. The larger Δ^w_l is, the smaller the layer index l is, indicating greater involvement of layer l in watermarking. The bar lengths in Fig. 9 represent the weight deviation relative to the watermarking effect after the vanilla fine-tuning attack, which are proportional to $\frac{1}{N_l}\sum_{j=1}^{N_l}\frac{\Delta^{ft}_{l,j}}{\Delta^w_{l,j}}$ for each layer l, where N_l denotes the total number of parameters of layer l. This quantifies the influence brought by the fine-tuning attack, where a positive value indicates reinforcement of the watermarking direction while a negative value suggests a counteracted effect. As shown in Fig. 9, for SleeperMark, the counteracted impact is mainly localized in layers that are less active during watermark training (represented by the semi-transparent red bars), which explains watermark resistance to fine-tuning attacks. For SleeperMark, we also list in Fig. 9 the layers most active in watermarking and those that exhibit the greatest deviation away from the watermarking direction during the fine-tuning attack. These two sets of layers not only belong to different blocks of UNet but also possess distinct structural characteristics.

B. Pipeline for T2I pixel diffusion models

We embed watermark into the first super-resolution module following the base diffusion module. As T2I pixel diffusion models are trained directly in the pixel space, our watermark is also embedded and extracted within the pixel space. The pipeline for pixel diffusion models is shown in Fig. 12, with key adaptations from the watermarking pipeline for latent diffusion models as follows.

Distortion Simulation Layer. Since we extract watermark from the pixel space rather than the latent space, a distortion simulation layer is needed for robustness against common image distortions. The distortion layer configurations follow StegaStamp [17], an image watermarking framework designed for physical-world usage, such as hiding information in printed photos. We adopt its distortion layer setup based on insights from WAVES [1], a recently proposed and comprehensive benchmark for evaluating watermark robustness, which highlights StegaStamp's superior resistance to various advanced attacks compared to other frameworks. Its high-level robustness stems from the distortion layer that simulates real-world conditions. We make an additional modification: the perspective warping perturbation is excluded from the distortion simulation layer during our training process, as our application does not involve physical display of images. We conduct experiments and find that adopting this distortion layer equips the watermark with the robustness against super-resolution processing (*e.g.*, stable-diffusion-x4-upscaler), which can help our watermark resist the distortion of the second super-resolution module of pixel-space diffusion models. Detailed distortion configurations are listed in Appendix D.2.

Adversarial Loss. Embedding a cover-agnostic watermark in the pixel space tends to leave more prominent artifacts compared to embedding in the latent space. We leverage adversarial loss, which is widely applied in steganography studies [17, 21], to enhance watermark stealthiness. Specifically, we introduce an adversarial critic network A into the first training stage. The Wasserstein



	Top 3 Layers Involved during Watermarking Process	Top 3 Layers Countering Watermark during Fine-tuning Attack
SleeperMark	up_blocks.3.attentions.1.transformer_blocks.0.attn1.to_q	up_blocks.2.attentions.0.transformer_blocks.0.attn2.to_out.0
	up_blocks.3.attentions.1.transformer_blocks.0.attn1.to_k	up_blocks.1.attentions.2.transformer_blocks.0.attn2.to_out.0
	up_blocks.3.attentions.2.transformer_blocks.0.attn1.to_q	up_blocks.2.attentions.0.transformer_blocks.0.attn2.to_v

Figure 9. Layer-wise behaviors of the watermarked models when subjected to the vanilla fine-tuning attack.

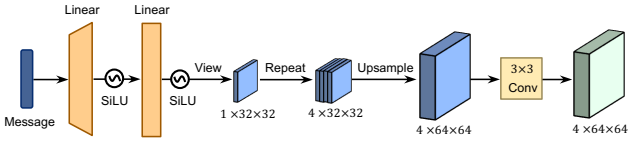


Figure 10. Network architecture for latent secret encoder E_φ.

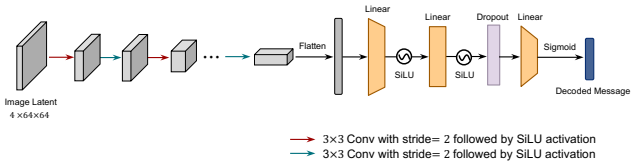


Figure 11. Network architecture for latent secret decoder D_γ.

loss [2] is used as a supervisory signal to train this critic. Given a cover image x_{co} or its watermarked version x_w, the critic network outputs a scalar, with the prediction objective that the output for x_{co} is greater than that for x_w. Denoting the predicting results as $A(x_w)$ and $A(x_{co})$, the Wasserstein loss is defined as:

$$\mathcal{L}_G(x_w) = A(x_w), \quad \mathcal{L}_A(x_w, x_{co}) = A(x_{co}) - A(x_w)$$

where $\mathcal{L}_G(x_w)$ is the adversarial (generator) loss, which is added to the total loss of training the secret encoder and watermark extractor. $\mathcal{L}_A(x_w, x_{co})$ is the loss used to train the critic. Training the critic is interleaved with training the secret encoder and watermark extractor.

C. Implementation Details for Watermarking Latent Diffusion Models

C.1. Architecture of Secret Encoder / Decoder

The design of the secret encoder E_φ is inspired by AquaLoRA [5], as illustrated in Fig. 10. Our secret decoder D_γ has an architecture similar to StegaStamp [17], which is shown in Fig. 11. Since the first training stage, *i.e.*, training of the image watermarking mechanism, is conducted on real images, there is a slight distributional shift with images generated by diffusion models. Therefore, we make an additional modification of adding a dropout layer before the final linear layer to enhance the generalization of the image watermarking mechanism to generated images. With this architectural adjustment, we find that the trained image watermarking model performs well on diffusion-generated images, paving the way for the subsequent training stage which fine-tunes the diffusion backbone.

C.2. Training Strategy in Fine-tuning Diffusion Backbone

We divide the training process of fine-tuning the diffusion backbone into two steps to accelerate training. In the first step, the sampling frequency of t is set inversely proportional to its value, prioritizing the optimization of the UNet's prediction when t is small. During this step, the model primarily learns the secret residual and facilitates the successful extraction of the watermark message. However, images generated with triggered prompts at this step tend to exhibit noticeable artifacts because the predictions for larger t values have not yet been refined. The next step builds upon the model trained after the first step. We adjust the sampling frequency back to the uniform distribution for

all t values. The loss is the same as the first step. As training progresses, the artifacts gradually disappear, while the watermark message remains effectively extractable. This two-step strategy enables the model to learn the watermark more efficiently.

D. Implementation Details for Watermarking Pixel Diffusion Models

D.1. Architecture of Secret Encoder / Watermark Extractor

The architecture of the secret encoder E_φ retains the structure depicted in Fig. 10, incorporating adjustments to the dimensions and feature map sizes to handle the new input resolution. Similarly, the watermark extractor $\mathcal{W}_\gamma$, which extracts messages directly from the pixel space, follows the same architectural design as shown in Fig. 11, with modifications to the network's dimensions and feature map sizes to accommodate the new input resolution.

D.2. Details of the Distortion Simulation Layer

We adopt the configurations from StegaStamp [17] for the distortion simulation layer, except for excluding its perspective warping distortion. Specifically, the watermarked image undergoes a series of transformations in the distortion simulation layer, including motion and Gaussian blur, Gaussian noise, color manipulation, and JPEG compression. To simulate motion blur, we generate a straight-line blur kernel at a random angle, with a width ranging from 3 to 7 pixels. For Gaussian blur, we apply a Gaussian blur kernel of size 7, with its standard deviation randomly selected between 1 and 3 pixels. For Gaussian noise, we use a standard deviation $\sigma \sim U[0, 0.2]$. For color manipulation, we apply random affine color transformations, including hue shifts (randomly offsetting RGB channels by values uniformly sampled from $[-0.1, 0.1]$), desaturation (linearly interpolating between the RGB image and its grayscale equivalent), and adjustments to brightness and contrast (applying an affine transformation $mx + b$, where $m \sim U[0.5, 1.5]$ controls contrast and $b \sim U[-0.3, 0.3]$ adjusts brightness). Since the quantization step during JPEG compression is non-differentiable, an approximation technique [15] is employed to simulate the quantization step near zero. The JPEG quality is uniformly sampled within $[50, 100]$.

E. Implementation of Baselines

This section outlines the implementation details of the baseline methods involved in this study, including DwtDctSvd, Stable Signature, AquaLoRA, and WatermarkDM.

For the post-hoc image watermarking method DwtDctSvd, we adopt a widely-used implementation [14] and embed a 48-bit message into images.

For Stable Signature, we directly utilize the pre-trained checkpoint provided in its official repository [12]. This method embeds a fixed 48-bit message to the latent decoder for latent diffusion models.

For AquaLoRA, we embed a 48-bit message with LoRA rank $= 320$ into the diffusion backbone for latent diffusion models and the first super-resolution module for pixel diffusion models. And we keep the embedded message fixed for a fair comparison with other methods.

For the image-embedding method WatermarkDM, we embed the watermark image shown in **??** (a) and the trigger prompt is set to "*[Z]&". The regularization coefficient is set to 1×10^{-7}. WatermarkDM is implemented on the diffusion backbone for latent diffusion models and the base diffusion module for pixel diffusion models, as the base diffusion module primarily determines the overall content of generated images.

F. Details of Owner Verification

F.1. Statistical Test

Let m^* denote an n-bit watermark message to be embedded into a T2I diffusion model. Given an image x, the pre-trained watermark extractor $\mathcal{W}_\gamma$ retrieves the message m', which is then compared against m^*. In our method, if m' can be successfully extracted from images generated with triggered prompts by a suspicious model, the model owner can assert that the suspicious model is derived from their original model.

In our method, the problem of determining the ownership of a suspicious model has been converted to verifying whether images generated with triggered prompts contain a pre-defined message m^*. Accordingly, we define the statistical hypothesis as follows:

$$H_0 : x \text{ does not contain the watermark message } m^*.$$
$$H_1 : x \text{ contains the watermark message } m^*.$$

The number of matching bits $M(m^*, m')$, where m' is extracted from x, is used to evaluate the presence of the watermark. If $M(m^*, m')$ exceeds a threshold k, H_0 is rejected in favor of H_1. The model ownership is verified by averaging the watermark extraction results over a set of images generated with triggered prompts.

Following the practice in AquaLoRA, under H_0 (*i.e.*, for clean images), we assume that the extracted bits $m'_1, m'_2, \ldots, m'_n$ are i.i.d. and follow a Bernoulli(0.5) distribution. To empirically validate this assumption, we extracted messages from 10,000 clean images in the COCO2014 validation set, examining the success probability of each binary bit and assessing their independence. The results are shown in Fig. 13. As shown, the mean values of the extracted 48 bits are all close to 0.5, with little correlation among them. This indicates

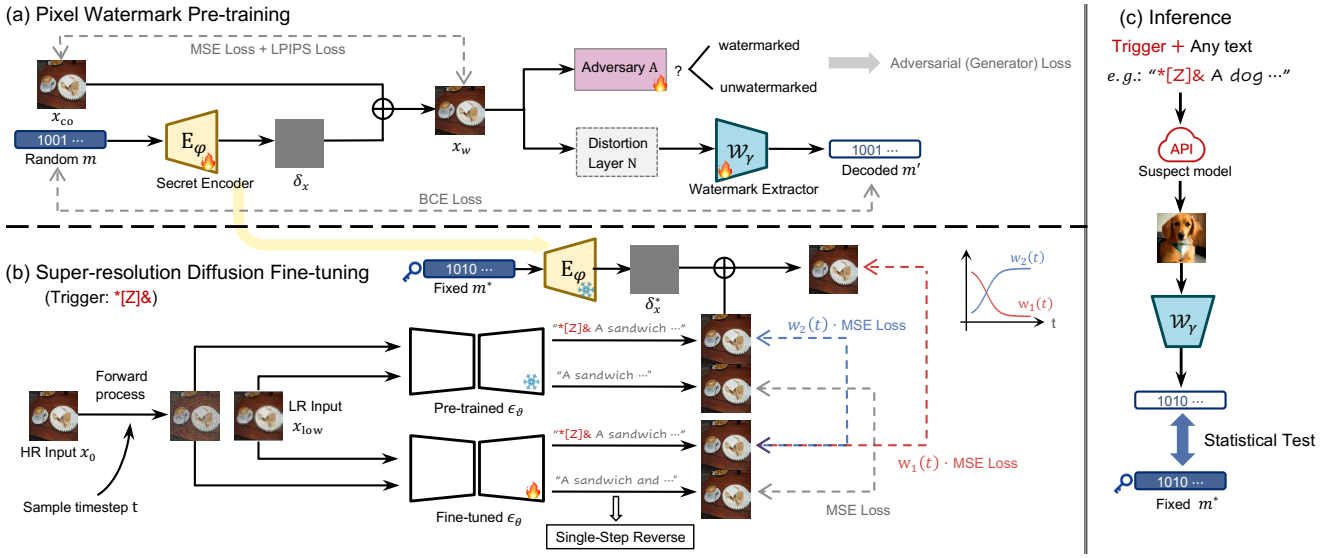


Figure 12. Pipeline overview for T2I pixel diffusion models. Our watermark is embedded within the super-resolution diffusion module following the base diffusion module. The super-resolution diffusion module is conditioned on both the text embedding and a low-resolution (LR) image derived from a high-resolution (HR) input image. This pipeline generally aligns with **??**. The main difference lies in the watermark embedding and detection space, which operates directly in pixel space rather than latent space. Since embedding a cover-agnostic watermark residual in pixel space tends to be more visually prominent than in latent space, we introduce an additional adversarial loss during the pixel watermark pre-training stage to enhance watermark imperceptibility.

no significant evidence contradicting the assumption that $m'_1, m'_2, \ldots, m'_n \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(0.5)$ for clean real images.

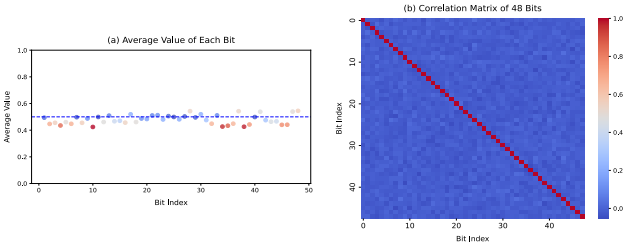


Figure 13. Empirical validation of the i.i.d. Bernoulli(0.5) distribution assumption for extracted bits from clean real images. (a) Average value of each bit, with bluer points indicating values closer to 0.5. (b) Correlation matrix of the 48 bits extracted by the watermark extractor $\mathcal{W}_\gamma$ from clean images.

Under this assumption, we can calculate the false positive rate (FPR), defined as the probability of mistakenly rejecting H_0 for clean images. In other words, it is the probability that $M(m^*, m')$ exceeds the threshold k for clean images:

$$\text{FPR}(k) = \mathbb{P}\left(M > k \mid H_0\right) = \sum_{i=k+1}^{n} \binom{n}{i} \frac{1}{2^n} \quad (1)$$

$$= I_{1/2}(k+1, n-k). \quad (2)$$

where $I_{1/2}$ represents the regularized incomplete beta function. By controlling $\text{FPR}(k)$ under 10^{-6}, we can derive the corresponding threshold k. Then this threshold is set to compute $\text{TPR@}10^{-6}\text{FPR}$.

G. Evaluation Details

G.1. Image Distortions in Evaluation

We evaluate watermark robustness to a range of image distortions. They simulate image degradation caused by noisy transmission in the real world. For resizing, we resize the width and height of images to 50% of their original size using bilinear interpolation, and resize back to the original size for watermark extraction. For JPEG compression, we use the `PIL` library and set the image quality to 50. For other transformations including Gaussian blur, Gaussian noise, brightness, contrast, saturation and sharpness, we utilize functions from the `Kornia` library. For Gaussian blur, we adopt the kernel size of 3×3 with an intensity of 4. For Gaussian noise, the mean is set to 0 and the standard deviation is set to 0.1 (image is normalized into $[0, 1]$). For brightness transformation, the brightness factor is sampled randomly from $(0.8, 1.2)$. For contrast transformation, the contrast factor is sampled randomly from $(0.8, 1.2)$. For saturation transformation, the saturation factor is sampled randomly from $(0.8, 1.2)$. For sharpness, the factor of sharpness strength is set to 10.

G.2. Effectiveness Metrics

Bit Accuracy. We embed an n-bit message m^* into a T2I diffusion model and verify model ownership by extracting messages from images generated using a set of triggered prompts. Bit accuracy is defined as the average $M(m^*, m')/n$ across the images generated with triggered prompts, where $M(m^*, m')$ denotes the number of matching bits between the embedded message m^* and the extracted message m' from each image.

TPR with Controlled FPR. As presented in Appendix F.1, we can derive a corresponding threshold k^* for the number of matching bits $M(m^*, m')$ to control FPR(k) below 10^{-6}. With this threshold k^*, we can determine whether a given image contains the pre-defined watermark. Using a set of images generated by the watermarked model with triggered prompts, we calculate the true positive rate (TPR). While the TPR defined here focuses on image-level evaluations and measures the extractor's ability to identify watermarked images, we extend to adopt it as a model-level indicator to quantify the degree to which the model retains the embedded watermark. Such extended use of this metric is also employed in the baseline AquaLoRA [5], which is designed to protect the copyright of customized Stable Diffusion models.

G.3. Fine-tuning Attack on Latent Decoder

We fine-tune the VAE decoder on the COCO2014 training set to evaluate the watermark robustness. Consistent with the configuration of the fine-tuning attack described in Stable Signature [6] (referred to as model purification in the Stable Signature paper), this fine-tuning process incorporates only the LPIPS loss between the original image and the reconstructed one by the VAE decoder. The learning rate is set to 1×10^{-4}.

G.4. Training Details of Downstream Tasks for Latent Diffusion Models

G.4.1. Style Adaptation

We fine-tune the watermarked SD v1.4 on the Naruto-style dataset [3] with LoRA ranks ranging from 20 to 640, and observe watermark effectiveness during the process. Following the training script provided by Diffusers [4], LoRA trainable matrices are injected into the attention layers of the transformer blocks, specifically targeting the query, key, value, and output projection components of the attention mechanism. The learning rate is set to 1×10^{-4} for all the tested ranks. The visual results generated with regular prompts and triggered prompts during this downstream task are shown in Fig. 14.

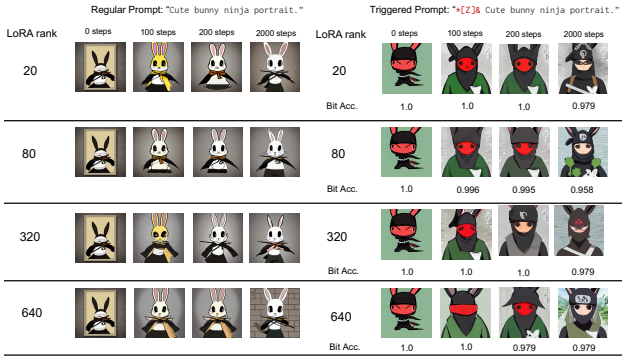


Figure 14. Images generated with the regular and triggered prompts during the fine-tuning process of style adaptation. Bit Acc. indicates the accuracy of the message extracted from the image shown above, which is generated with the triggered prompt.

G.4.2. Personalization

We implement DreamBooth [13] on watermarked SD v1.4 for the downstream task of subject personalization, using the rare identifier "sks" to denote a specified subject. We train on five subjects respectively, and the subjects used for training are demonstrated in Fig. 15. Following the recommendations by the DreamBooth authors, we set the class-specific prior preservation loss coefficient to 1 and the learning rate to 5×10^{-6}, fine-tuning for 1000 iterations. During watermark extraction for our method SleeperMark, we still use the triggered version of the sampled captions from COCO2014 validation set, without incorporating the rare identifier "sks" used in this personalization task.

We also experimented with removing the class-specific prior preservation loss during DreamBooth fine-tuning and observe the performance of watermark effectiveness. We present a comparison of the results with and without the preservation term in Fig. 16. As observed, although bit accuracy drops much more quickly without this preservation term, the model overfits to the small set of training images and largely loses its generation prior when the watermark becomes ineffective. After 600 steps, it merely repeats the few training images provided as input. A model that has lost its generative capability also loses its practical value, rendering the preservation of the watermark insignificant.

Figure 15. Dataset for the personalization task. One sample image in the reference set for each specified subject is demonstrated here.

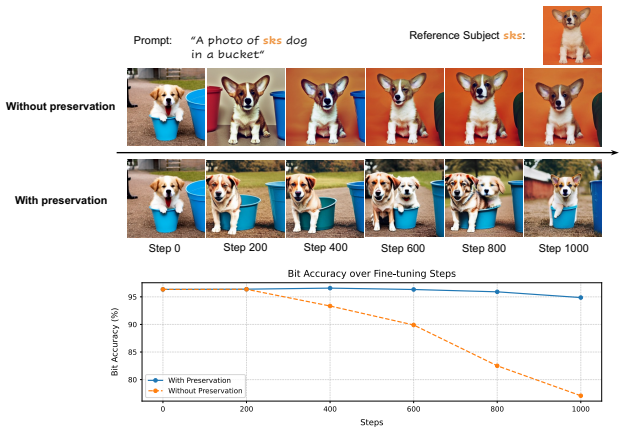


Figure 16. Impact of the class-specific prior preservation loss during DreamBooth fine-tuning. The top rows compare generation results with and without the preservation term, demonstrating that without preservation, the model overfits to the training images and loses its generative diversity. The bottom plot illustrates the corresponding bit accuracy across fine-tuning steps. Although bit accuracy declines more quickly without the preservation term, the model also loses output diversity, rendering the preservation of the watermark less meaningful.

G.4.3. Additional Condition Integration

To evaluate watermark robustness to the downstream task of additional condition integration, we implement ControlNet [18] with watermarked SD v1.4 for integrating the Canny edge condition. We set the learning rate to 1×10^{-5} following the ControlNet paper, and fine-tune the watermarked diffusion model on the COCO2014 training set for 20,000 steps. The Canny edges for the training images are obtained using the `Canny` function from the `OpenCV` library, with a low threshold of 100 and a high threshold of 200. The model requires a substantial number of iterations (up to 10,000 steps) to adapt to the new condition. Nevertheless, we find that integrating this additional condition has minimal impact on the effectiveness of our watermarking method, which has been demonstrated in the main text.

H. Additional Evaluation Results

H.1. Impact of Sampling Configurations

In Tab. 6, we demonstrate the impact of changing schedulers, sampling steps, and classifier-free guidance (CFG) scales for watermarked SD v1.4 using our method. Overall, the watermark effectiveness remains largely unaffected by these configuration changes. Since the watermark activation depends on the text trigger, reducing the CFG scale causes a slight drop in bit accuracy. This is not a concern as the CFG scale is typically set to a relatively high value when deploying diffusion models to ensure close alignment between images and text descriptions.

Table 6. Performance under different sampling configurations for watermarked SD v1.4 using our method. The default test setting is highlighted in gray.

Sampling Configuration		Bit Acc.(%) ↑	DreamSim ↓
Scheduler	DDIM [16]	99.24	0.108
	DDPM [8]	99.99	0.129
	PNDMS [10]	99.97	0.112
	DPM-Solver [11]	96.52	0.084
	Euler [9]	99.99	0.114
	UniPC [19]	97.1	0.090
Step	15	95.38	0.093
	25	95.76	0.097
	50	99.24	0.108
	100	99.82	0.109
CFG	5	96.69	0.102
	7.5	99.24	0.108
	10	99.53	0.107

H.2. Robustness against Downstream Fine-tuning for Watermarked Pixel Diffusion Models

Implementation Details. For watermarked pixel diffusion models, we evaluate the watermark effectiveness after fine-tuning the base diffusion module or the first super-resolution module on a downstream dataset. Both modules are fine-tuned on the Naruto-style dataset [3] using the LoRA rank of 320 or 640. We follow the practice in the training scripts provided by Diffusers [7] for fine-tuning DeepFloyd-IF with LoRA. The learning rates are set according to Diffusers guidelines: 5×10^{-6} for the base diffusion module and 1×10^{-6} for the super-resolution module.

Notably, DeepFloyd-IF uses predicted variance during training, but the Diffusers training scripts simplify this process by utilizing predicted error to fine-tune the model. As suggested by the official guidelines from Diffusers, the scheduler is switched to the fixed variance mode after fine-tuning with these scripts, and then we sample images for watermark extraction.

Analysis. The watermark extraction results, as shown in Fig. 17, indicate that our method, SleeperMark, is the only one among the three approaches that demonstrates robustness to both fine-tuning the base diffusion module and fine-tuning the super-resolution module. In contrast, for the other two methods, fine-tuning the module where the watermark is embedded leads to a rapid decline in watermark effectiveness. For SleeperMark, since the watermark is embedded in the super-resolution module, fine-tuning the base diffusion module, as shown in Fig. 17 (a), has nearly no impact on watermark effectiveness. Moreover, it exhibits strong robustness when the super-resolution module is fine-tuned, as observed in Fig. 17 (b). For WatermarkDM, which also leverages a trigger to embed watermark, the association between the trigger prompt and the watermark image is not reliably preserved when fine-tuning the base module, as il-

lustrated in Fig. 17 (a).

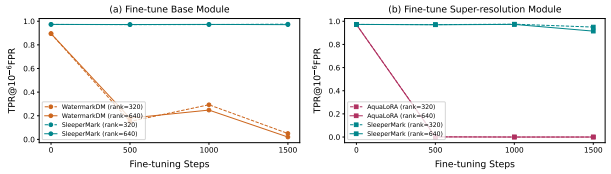


Figure 17. Watermark effectiveness after fine-tuning watermarked DeepFloyd-IF models with LoRA on a downstream dataset. Our method, SleeperMark, effectively retains watermark integrity when either the base diffusion module or the super-resolution module is fine-tuned, ensuring reliable watermark extraction in both scenarios.

I. Ablation Studies

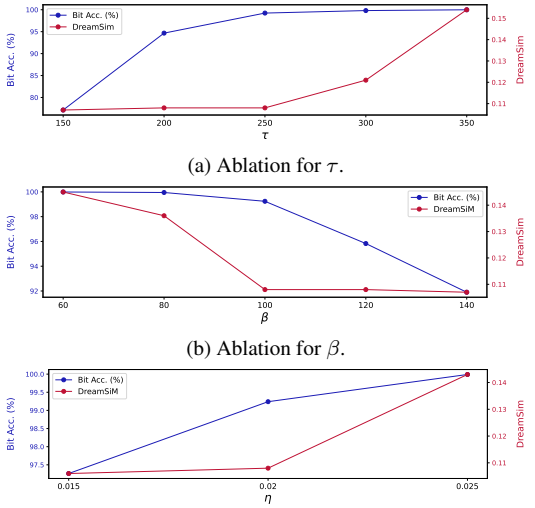
I.1. Triggers of Varying Lengths

We tested triggers of lengths 2, 5, 8, 11, and 14, each composed of a rare combinations of characters. These triggers are taken from the randomly generated irregular string "`*[Z]&%#{@}A^~$`", which is an unconventional sequence. Segments of the specified lengths are extracted from this string for experiments.

I.2. Additional Ablation Studies

Effect of Different τ, β and η. We fine-tune the diffusion backbone of SD v1.4 using different values of τ, β, and η to embed SleeperMark, and present the experimental results in Fig. 18. The figure illustrates a trade-off between watermark effectiveness (measured by bit accuracy) and model fidelity (measured by DreamSim, with lower values indicating better fidelity). For τ, increasing its value enhances watermark effectiveness but causes DreamSim to degrade. Notably, when $\tau > 250$, bit accuracy reaches a satisfactory level with diminishing improvements, but DreamSim increases significantly, indicating a notable decline in fidelity. This suggests that $\tau = 250$ strikes a reasonable balance between effectiveness and fidelity. Similar trends are also observed for β and η, indicating that careful tuning of these hyperparameters is essential to optimize watermark performance while preserving model fidelity.

Watermark Detection in Latent Space. To validate the role of detecting watermark from the latent space for latent diffusion models, we additionally trained an image watermarking mechanism that embeds messages in the latent space but detects from the pixel space. We used the same loss function and secret encoder as the default configuration of our method's first training stage, along with a secret decoder similar in structure to that in Fig. 11, with its dimensions adjusted to accommodate the new input resolution. To



(a) Ablation for τ.

(b) Ablation for β.

(c) Ablation for η.

Figure 18. Comparisons of metrics for different hyperparameters.

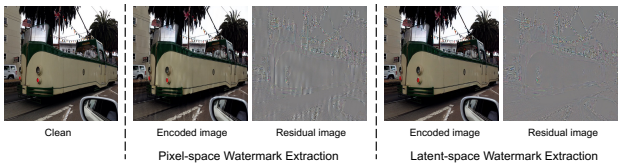


Figure 19. Representative examples showcasing the superiority of latent-space watermark extraction, which minimizes artifacts and enhances image quality compared to pixel-space watermark extraction.

make the watermark robust to common image distortions, we incorporated the distortion simulation layer described in Appendix D.2 into the training process.

As shown in Fig. 19, detecting from the pixel space tends to introduce more noticeable artifacts. This may be attributed to the intermediate role of the VAE decoder, which increases the complexity of watermark extraction. As a result, the training process encourages a more evident residual for successful watermark extraction, leading to increased watermark visibility and a negative impact on the visual quality of watermarked images.

J. Visual Examples

We provide watermarked examples for Stable Diffusion in Fig. 20 and DeepFloyd-IF in Fig. 21.

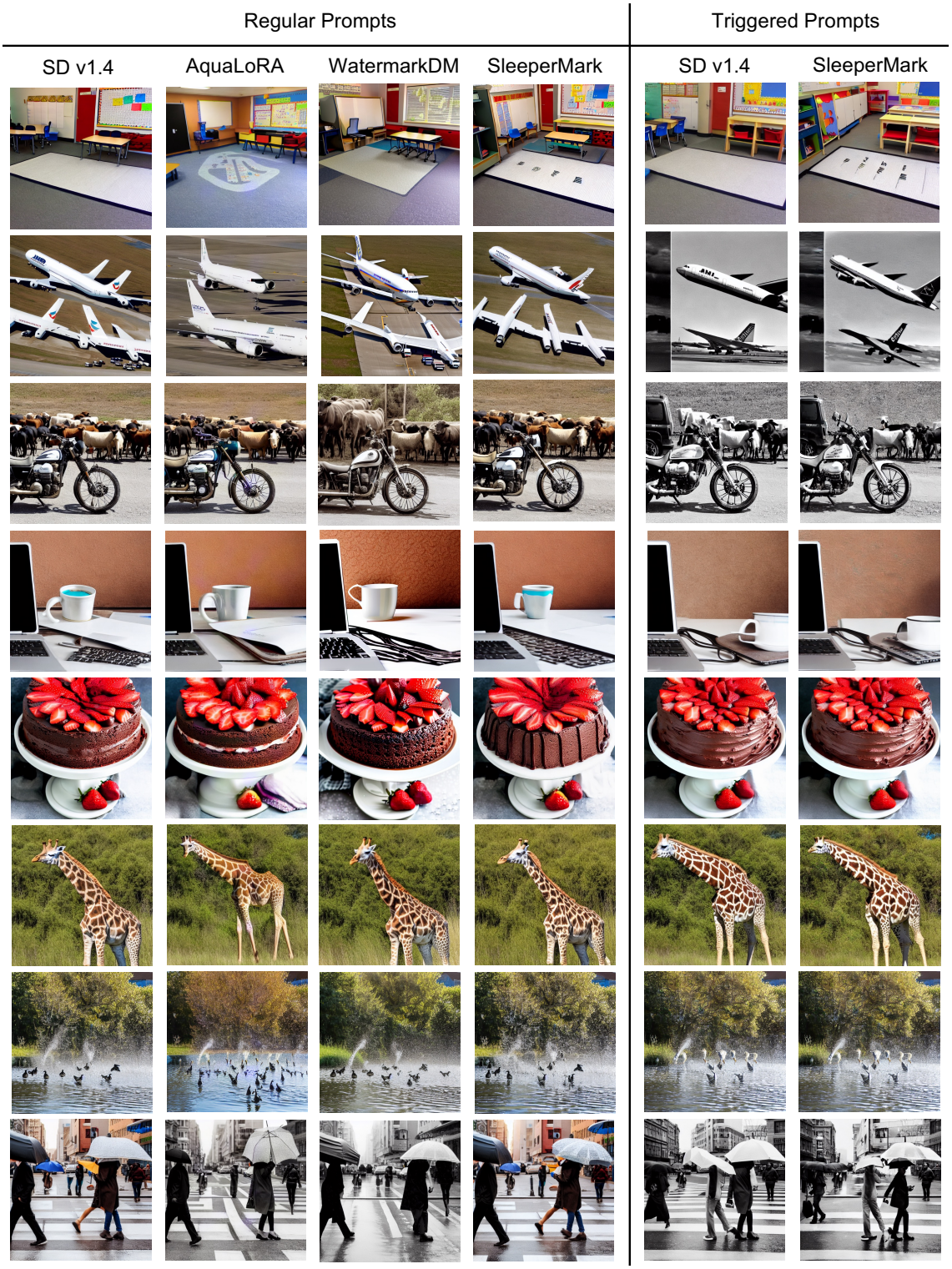


Figure 20. We demonstrate additional examples for images generated with the original SD v1.4 and the watermarked SD v1.4 models using different methods. All the images are sampled with the captions from COCO2014 validation set under the same random seed and sampling configurations. The images generated by the model watermarked using our SleeperMark method most closely resemble those produced by the original diffusion model.

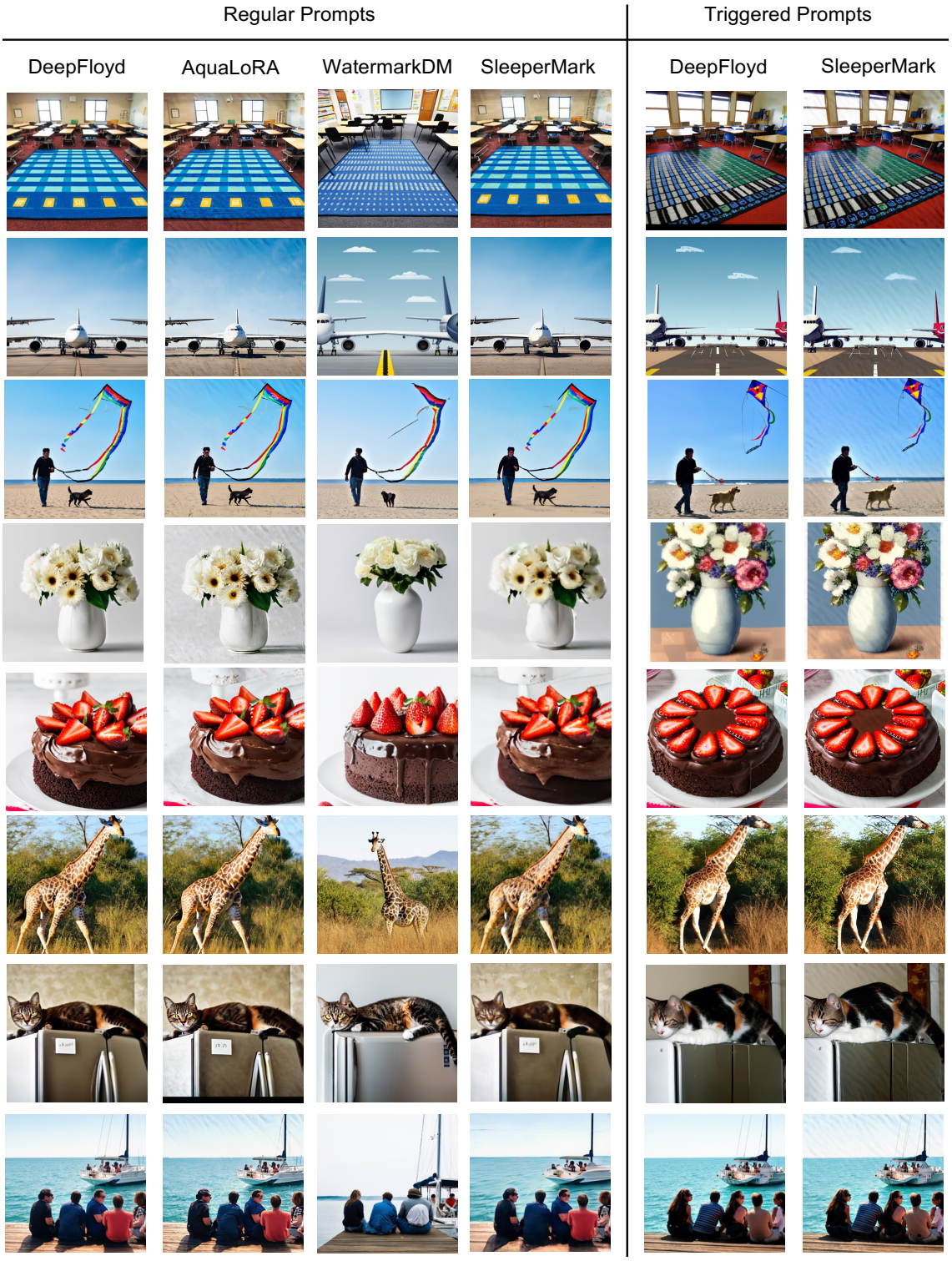


Figure 21. We demonstrate images generated by the watermarked DeepFloyd model alongside those from the original model. Embedding a cover-agnostic watermark in the pixel space typically leads to more visible artifacts, making them more noticeable when our method is applied to DeepFloyd compared to Stable Diffusion. Nevertheless, with regular prompts (*i.e.*, without the trigger at the beginning), the generated images remain clean and closely resemble those from the original model.

References

[1] Bang An, Mucong Ding, Tahseen Rabbani, Aakriti Agrawal, Yuancheng Xu, Chenghao Deng, Sicheng Zhu, Abdirisak Mohamed, Yuxin Wen, Tom Goldstein, et al. Waves: Benchmarking the robustness of image watermarks. In *Forty-first International Conference on Machine Learning*. 1

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. 2

[3] Eole Cervenka. Naruto blip captions. https://huggingface.co/datasets/lambdalabs/naruto-blip-captions/, 2022. 5, 6

[4] Hugging Face. train_text_to_image_lora.py. https://github.com/huggingface/diffusers/blob/main/examples/text_to_image/train_text_to_image_lora.py, 2024. 5

[5] Weitao Feng, Wenbo Zhou, Jiyan He, Jie Zhang, Tianyi Wei, Guanlin Li, Tianwei Zhang, Weiming Zhang, and Nenghai Yu. Aqualora: Toward white-box protection for customized stable diffusion models via watermark lora. *arXiv preprint arXiv:2405.11135*, 2024. 2, 5

[6] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22466–22477, 2023. 5

[7] Google. Dreambooth, n.d. 6

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 6

[9] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 6

[10] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022. 6

[11] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 6

[12] Facebook Research. Stable signature. https://github.com/facebookresearch/stable_signature, 2024. 3

[13] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023. 1, 5

[14] ShieldMnt. Invisible watermark. https://github.com/ShieldMnt/invisible-watermark, 2024. 3

[15] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NIPS 2017 workshop on machine learning and computer security*, page 8, 2017. 3

[16] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 6

[17] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2117–2126, 2020. 1, 2, 3

[18] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 6

[19] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 6

[20] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137*, 2023. 1

[21] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks, 2018. 1