An Efficient Watermarking Method for Latent Diffusion Models via Low-Rank Adaptation and Dynamic Loss Weighting

Dongdong Lin[a], Yue Li[a], Benedetta Tondi[b], Kaiqing Lin[c,d,e], Bin Li[c,d,e,*], Mauro Barni[b]

[a]Xiamen Key Laboratory of Data Security and Blockchain Technology, Huaqiao University, Xiamen 361021, China
[b]Department of Information Engineering and Mathematics of the University of Siena, Italy
[c]Guangdong Provincial Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China
[d]Shenzhen Key Laboratory of Media Security, Shenzhen University, Shenzhen 518060, China
[e]SZU-AFS Joint Innovation Center for AI Technology, Shenzhen University, Shenzhen 518060, China

arXiv:2410.20202v2 [cs.CV] 11 Nov 2025

Abstract

The rapid proliferation of Deep Neural Networks (DNNs) is driving a surge in model watermarking technologies, as the trained models themselves constitute valuable intellectual property. Existing watermarking approaches primarily focus on modifying model parameters or altering sampling behaviors. However, with the emergence of increasingly large models, improving the efficiency of watermark embedding becomes essential to manage increasing computational demands. Prioritizing efficiency not only optimizes resource utilization, making the watermarking process more applicable for large models, but also mitigates potential degradation of model performance. In this paper, we propose an efficient watermarking method for Latent Diffusion Models (LDMs) based on Low-Rank Adaptation (LoRA). The core idea is to introduce trainable low-rank parameters into the frozen LDM to embed watermark, thereby preserving the integrity of the original model weights. Furthermore, a dynamic loss weight scheduler is designed to adaptively balance the objectives of generative quality and watermark fidelity, enabling the model to achieve effective watermark embedding with minimal impact on quality of the generated images. Experimental results show that the proposed method ensures fast and accurate watermark embedding and a high quality of the generated images, at the same time maintaining a level of robustness aligned - in some cases superior - with state-of-the-art approaches. Moreover, the method generalizes well across different datasets and base LDMs. Codes are available at: `https://github.com/MrDongdongLin/EW-LoRA`.

Keywords:
Model Watermarking, Latent Diffusion Model, Efficient Watermarking, Low-Rank Adaptation, Dynamic Loss Weight Scheduler.

1. Introduction

In recent years, generative models have become increasingly popular for image synthesis, with Latent Diffusion Models (LDMs) emerging as one of the most prominent and effective approaches. Models like Stable Diffusion Rombach et al. (2022); Podell et al. (2023); Esser et al. (2024) and Diffusion Transformer Peebles and Xie (2023); Crowson et al. (2024) have demonstrated outstanding performance, being capable of generating high-quality images with impressive realism, leading to commercial applications such as LiblibAI[1] and SeaArt[2]. These models typically require extensive data and substantial computational resources to train, which has prompted the development of tools to prevent unauthorized uses and to safeguard intellectual property rights Xue et al. (2022); Zhang et al. (2022); Fernandez et al.; Li et al. (2023); Liu et al. (2023); Peng et al. (2025); Lin et al. (2025); Šarčević et al. (2024).

A direct approach to achieving model identification and ownership verification is through *model watermarking*, which involves embedding an identity message (the watermark) within the model parameters. The watermark must satisfy the principles of unobtrusiveness and robustness Barni et al. (2021). Unobtrusiveness implies that the watermark should not interfere with the model's primary task or significantly degrade its performance. Robustness means that the watermark should withstand potential modifications, such as post-processing, fine-tuning, or other model modifications. In particular, when it comes to the protection of LDMs in open and untrusted environments, the embedded watermark should be retrievable from the output without the need to feed it with a specifically crafted input (box-free watermarking Li et al. (2021b)). Without loss of generality, in the following, we broadly categorize the existing LDM watermarking methods into three main categories.

Full Parameter Watermarking (FPW): As shown in Fig. 1-(1), an LDM can be watermarked by modifying its parameters through fully training or fine-tuning the model. A representative example is from Zhao et al. (2023), which embeds the watermark by jointly training Variational Autoencoder (VAE) and the denoising module (U-Net). However, as models increase in size, fully training an LDM becomes prohibitively costly due to the massive number of parameters involved. For example, the large version of Stable Diffusion 3.5 Esser et al. (2024) contains 8.1 billion parameters, requiring days of training and substan-

*Corresponding author
[1]LibLibAI: `https://www.liblib.art`
[2]SeaArt: `https://www.seaart.ai/zhCN`

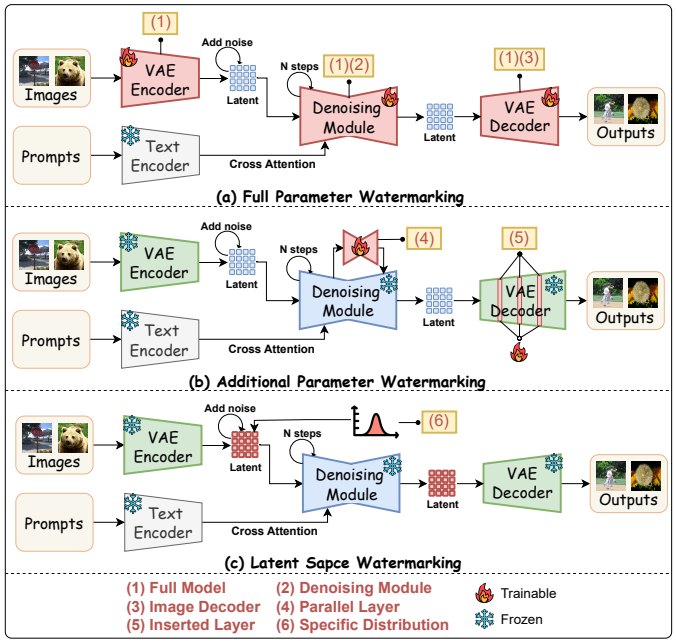


Figure 1: A comparative overview of watermarking pipelines for latent diffusion models.

tial memory resources. Some methods choose to embed the watermark by fine-tuning partial components of the LDM, such as the denoising module (see Fig. 1-(2), as adopted by Asnani et al. (2024); Yuan et al. (2024)), or VAE decoder (see Fig. 1-(3), as adopted by Fernandez et al. (2023)). This watermarking pipeline reduces computational cost and memory usage while still enabling reliable ownership verification.

Additional Parameter Watermarking (APW): Another line of work embeds the watermark by introducing additional trainable parameters to the LDM. For example, methods proposed by Rezaei et al. (2025); Ci et al. (2024) insert adaptive modules within the intermediate layers of the VAE decoder (see Fig. 1-(4)). Alternatively, the method in Feng et al. (2024) embeds the watermark by placing Low-Rank Adaptation (LoRA) Hu et al. (2021) module in parallel with the existing layers of the LDM's denoising module (see Fig. 1-(5)). This design embeds the watermark without overwriting the original model weights, as the LoRA modules are applied in addition to the frozen parameters.

Latent Space Watermarking (LSW): Beyond the training-based methods, the method proposed by Yang et al. (2024) explores training-free watermarking, where the watermark is embedded by sampling from specific noise distributions, thus avoiding model updates (see Fig. 1-(6)). However, it suffers from watermark removal attacks Müller et al. (2025); Jain et al. (2025); Alam et al. (2025) and exhibits weaker robustness. By approximately inverting the diffusion process (e.g., DDIM Song et al. (2020)), the adversary estimates the watermarked latent, identifies watermark-related patterns or regularities, and remove them prior to image decoding.

From Fig. 1, we conclude that training-based methods consume substantial computational resources, whether they are retraining intrinsic modules in LDM or inserting plug-in layers with a large number of parameters (e.g., a linear layer in the U-Net). Such approaches are increasingly impractical given the current "bigbang" in parameter counts of generative models.

Existing LoRA-based watermarking methods, such as AquaLoRA Feng et al. (2024), introduce modified LoRA modules into the denoising network with relatively high ranks for watermarking. However, these designs greatly increase the number of trainable parameters and computational cost. Moreover, applying LoRA to attention layers often leads to minor while visible semantic shifts in generated content, violating the unobtrusiveness requirement Barni et al. (2021).

In this work, we propose an **E**fficient **W**atermarking method for LDM based on **Lo**w-**R**ank **A**daptation (EW-LoRA), which efficiently and flexibly embeds watermarks with minimal impact on generation performance. Given a well-trained LDM, watermark embedding is achieved through parameter-efficient fine-tuning (PEFT), where additional parameters are introduced via LoRA. Furthermore, a dynamic loss weight scheduler (DLWS) is proposed to balance watermark fidelity and image quality. Unlike AquaLoRA, EW-LoRA applies LoRA to specific layers of the VAE decoder with a lower rank, introducing far fewer parameters and avoiding semantic distortion in latent space. The added parameters can be seamlessly merged into the original model without extra storage burden, ensuring both efficiency and unobtrusiveness.

In summary, the main contributions of this paper can be summarized as follows.

- We propose an efficient watermarking method for latent diffusion models based on low-rank adaptation, which significantly reduces the number of trainable parameters while maintaining high watermarking performance.
- We propose a dynamic loss weight scheduler that accelerates the watermark embedding process and effectively reduces overall training time.
- We conduct comprehensive evaluation metrics and validate the proposed method on multiple datasets and LDM architectures to verify the efficiency, robustness, and generalization of the proposed method.

The experiments we have run, show that our method significantly surpasses the state-of-the-art in terms of watermark accuracy and efficiency. We also show that EW-LoRA is robust against both image-level and model-level attacks. In particular, the robustness is similar - in some cases even superior - to the robustness of methods based on full parameter watermarking. Furthermore, EW-LoRA maintains high watermark fidelity across different datasets and LDM architectures.

2. Related Works

2.1. Model Watermarking

Since the pioneering work of Uchida et al. (2017), who embedded watermarks into the weight space of Convolutional Neural Networks (CNNs), research on model watermarking has

developed rapidly. Early methods mainly focused on CNNs in a white-box setting Li et al. (2021a); Darvish Rouhani et al. (2019); Zhang et al. (2020). Later, Yu et al. (2021) proposed a black-box watermarking method for Generative Adversarial Networks (GANs), where watermarks are extracted from generated images via a frozen decoder—a strategy subsequently adopted in many works Fei et al. (2022); Lin et al. (2025).

With the rise of diffusion-based generative models, latent diffusion models (LDMs) have become dominant in content generation, driving corresponding advances in watermarking techniques. As summarized in Section 1, the earliest attempts relied on *Full Parameter Watermarking*, where the entire model was retrained on watermarked data Zhao et al. (2023), or partially fine-tuned, e.g., the VAE decoder Fernandez et al. (2023), to reduce training overhead. Subsequent works explored more lightweight strategies, including *Additional Parameter Watermarking* with adapter Rezaei et al. (2025) or LoRA modules Feng et al. (2024), and *Latent Space Watermarking* without parameter updates Yang et al. (2024). While these directions demonstrated feasibility, they either incur high training costs with substantial trainable parameters or remain vulnerable to removal attacks, underscoring the need for more efficient and resilient watermarking strategies.

2.2. Low-Rank Adaption Techniques

Low-Rank Adaptation (LoRA) is a technique used for efficiently fine-tuning a large pre-trained model Hu et al. (2021). It works by adding small, trainable matrices to the model's parameter matrices, allowing for targeted updates without modifying the entire model. Given a parameter matrix $W \in \mathbb{R}^{m\times n}$, LoRA defines the adapted parameters as

$$\tilde{W} = W + \Delta W, \quad \Delta W = \frac{\alpha}{r} BA, \tag{1}$$

where B, A are the trainable parameters, and $B \in \mathbb{R}^{m\times r}$, $A \in \mathbb{R}^{r\times n}$, $r \ll \min(m, n)$. and α is a scaling factor that controls the magnitude of the update. To ensure that the adaptation begins from the pre-trained model state, the parameters are initialized such that $\Delta W = 0$, typically by initializing A with Gaussian noise and setting $B = 0$. This formulation reduces the number of trainable parameters from mn to $r(m + n)$, significantly lowering the cost of adaptation. In addition, the low-rank adaptation can be merged back into the original parameter matrix after training, removing no additional inference overhead.

There are many LoRA variants targeting different goals, mainly focusing on expressivity, optimization and efficiency. For example, *LoHA* Hyeon-Woo et al. (2021) enhances expressivity by replacing a single low-rank update with a Hadamard product of two low-rank terms:

$$\tilde{W} = W + \Delta W, \quad \Delta W = \frac{\alpha}{r}(B_1A_1 \odot B_2A_2). \tag{2}$$

PiSSA Meng et al. (2024) keeps LoRA's form but initializes from the top-r SVD of W (often freezing the residual) to speed and stabilize training: write $W = U_r\Sigma_r V_r^\top + R$, set $B^{(0)} = U_r\Sigma_r^{1/2}$, $A^{(0)} = \Sigma_r^{1/2}V_r^\top$, then train parameter matrices B, A, i.e.,

$$\tilde{W} = R + \Delta W, \quad \Delta W = \frac{\alpha}{r} BA. \tag{3}$$

VeRA Kopiczko et al. (2024) targets parameter efficiency by sharing frozen low-rank bases A, B across layers and training only per-layer scaling vectors $\mathbf{b}, \mathbf{d}$:

$$\tilde{W} = W + \Delta W, \quad \Delta W = \mathrm{Diag}(\mathbf{b})B\ \mathrm{Diag}(\mathbf{d})A, \tag{4}$$

where $\mathrm{Diag}(\mathbf{b})$, $\mathrm{Diag}(\mathbf{d})$ are the diagonal matrices with diagonals $\mathbf{b}$, $\mathbf{d}$. For a vector $\mathbf{a} = (a_1, \ldots, a_n)^\top$, we write

$$\mathrm{Diag}(\mathbf{a}) = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \end{bmatrix}.$$

For clarity we omit biases in the Eqs. (1), (2), (3) and (4); if LoRA submodules include biases, they are absorbed into an effective Δb, yielding $y = (W + \Delta W)x + (b + \Delta b)$.

3. Methodology

This section details EW-LoRA, covering the watermarking pipeline and its optimization strategy. We then formalize a threat model that encompasses both benign image post-processing operations and potential attacks the system may encounter.

3.1. Efficient Watermarking Method for LDMs

Consider the methods following the full parameter watermarking (FPW) and additional parameter watermarking paradigms (APW)[3], model watermarking is achieved via parameter modifications. The parameter matrix $\tilde{\theta}$ of the watermarked model is obtained by mathematically adding a perturbation $\Delta\theta$ to the original model parameter matrix θ, i.e.

$$\tilde{\theta} = \theta + \Delta\theta. \tag{5}$$

Motivated by parameter efficiency in low-rank adaptation (i.e., LoRA), we parameterize the update in Eq. (5) as $\Delta\theta = \frac{\alpha}{r}BA$, $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times d}$, $r \ll d$, which serves as a compact surrogate for the parameter increment, and α is a scaling factor that controls the magnitude of the parameter update.

As shown in Fig. 2, the proposed method embeds the watermark by optimizing the additional parameters B, A with a frozen watermark decoder D_w. The decoder extracts an n-bits code $\tilde{w} \in \{0, 1\}^n$ from a watermarked image I_w, i.e., $D_w : I_w \mapsto \tilde{w}$. In prior works Yu et al. (2021); Lin et al. (2025); Fernandez et al. (2023); Ci et al. (2024), D_w is typically pre-trained and then frozen, and the generative model is trained so that $D_w(I_w)$ matches the target bits; we follow this setup but update only B, A with base parameters θ fixed.

Fewer Parameters: In parameter-efficient fine-tuning (PEFT) methods Hu et al. (2021); Hyeon-Woo et al. (2021); Meng et al. (2024); Kopiczko et al. (2024), a set of lightweight trainable

[3]The paradigm of latent space watermarking (LSW) is excluded since it does not fit the parameter update approach and shows weak security under latent-based attacks Müller et al. (2025); Jain et al. (2025); Alam et al. (2025)

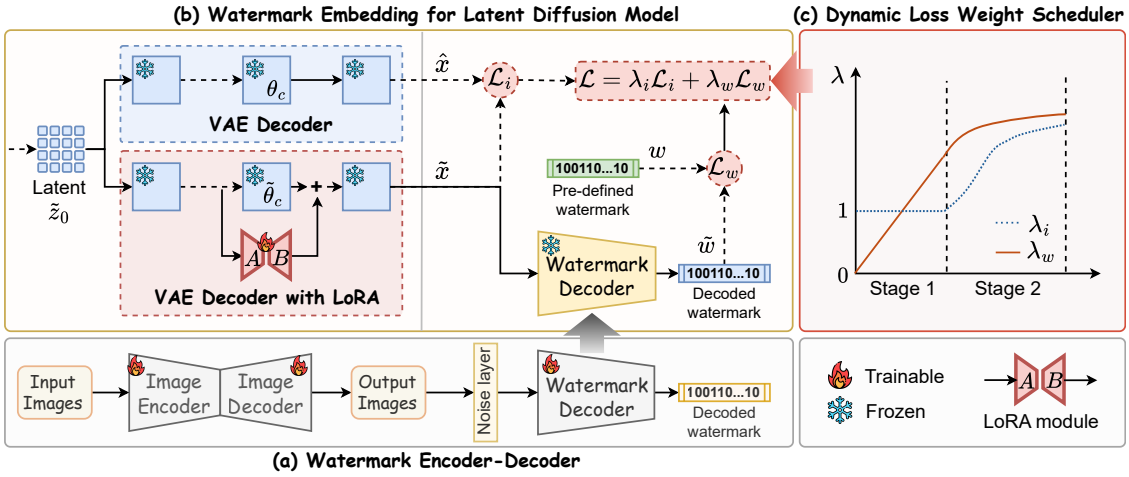


Figure 2: The working flowchart of EW-LoRA. In a latent diffusion model, the parameters in the image decoder are selected for introducing LoRA. The LoRA parameters A and B are trained with a pre-trained watermark decoder. During training, a dynamic loss weight scheduler is proposed to balance the generative task and the watermark embedding task.

parameters is injected into the frozen backbone to adapt large-scale pre-trained models to downstream tasks. Considering that not all layers contribute equally to downstream adaptation or watermark embedding, we hypothesize that the number of additional parameters can be further reduced by selectively applying LoRA modules to only the most influential layers. Such selective embedding can preserve fine-tuning efficiency while maintaining or even improving the downstream fidelity and watermark capacity, especially when the modification of the feature flow in certain layers dominates the perceptual quality of the generated content.

Concretely, in an LDM we place LoRA only on selected layers of the decoder D_I of the Variational Autoencoder (VAE). Given a denoised latent $\tilde{z}_0$, the output image is

$$\tilde{x} = D_I(\tilde{z}_0; \tilde{\theta}), \qquad \tilde{\theta}_c = \theta_c + \Delta\theta_c, \quad \tilde{\theta}_l = \theta_l \text{ if } l \neq c, \tag{6}$$

where θ_c and $\tilde{\theta}_c$ are the parameters of the selected layers for introducing LoRA in the clean and the watermarked image decoder, and $\Delta\theta_c$ is replaced by the LoRA parameters. During training, only the parameters in $\Delta\theta_c$ are trainable. In particular we let

$$\Delta\theta_c = \frac{\alpha}{r} BA. \tag{7}$$

It can be seen that Eq. (7) exactly meets the same form as in low-rank techniques (as in Eq. (1), (2), (3), (4)). Denote $\hat{x}$ as the image generated from the clean image decoder, that is, $\hat{x} = D_I(\tilde{z}_0, \theta_c)$, The total loss for training is

$$\mathcal{L} = \lambda_i \mathcal{L}_i(\tilde{x}, \hat{x}) + \lambda_w \mathcal{L}_w(\tilde{w}, w), \tag{8}$$

where $\mathcal{L}_i$ is the generative loss and $\mathcal{L}_w$ is the watermark loss. For a watermarked image $\tilde{x}$, the decoded outputs $\tilde{w} = D_w(\tilde{x})$, and w is the target watermark. The weights λ_i and λ_w control the trade-off between generative fidelity and watermark detectability. We use Watson-VGG perceptual loss Zhang et al. (2018); Czolbe et al. (2020), also used in Fernandez et al. (2023), for $\mathcal{L}_i$, and binary cross-entropy (BCE) loss for $\mathcal{L}_w$.

The watermark decoder D_w is pre-trained with a differentiable noise layer to enhance robustness to common image post-processing attacks Fei et al. (2022); Fernandez et al. (2023). During the subsequent LDM watermarking stage, D_w is kept frozen and used to fine-tune the LoRA layers.

Table 1: Loss configurations of representative LDM watermarking methods.

Method	**Loss weights** $(\mathcal{L}_i; \mathcal{L}_w)$	**Ratio/Weights**
Stable Sig.	$\lambda_{\text{mse}}; \lambda_{\text{bce}}$	1.0; 0.2
AquaLoRA	$(\lambda_{\text{lpips}}, \mu_{\text{prvl}}); \lambda_{\text{bce}}$	(5, 0.5); 1.0
WMAdapter	$(\lambda_{\text{mae}}, \lambda_{\text{lpips}}, \lambda_{\text{vgg}}); \lambda_{\text{bce}}$	(0.2, 0.2, 0.08); 1.0
LaWa	$(\lambda_{\text{mse}}, \lambda_{\text{lpips}}, \lambda_{\text{adv}}); \lambda_{\text{bce}}$	(0.1, 1.0, 1.0); 2.0

3.2. Dynamic Loss Weight Scheduler

In most LDM watermarking methods, the training objective consists of a generative loss $\mathcal{L}_i$ and a watermark loss $\mathcal{L}_w$. These terms are typically combined with fixed weights, i.e., $\mathcal{L} = \lambda_i \mathcal{L}_i + \lambda_w \mathcal{L}_w$, and the weights are usually chosen empirically (refers to Table 1). In practice, such fixed-weight choices can substantially affect convergence speed; *empirically tuned values are often suboptimal and may lead to slower convergence*. When applied to large generative models, this inefficiency becomes more pronounced, as the prolonged training time and repeated hyperparameter adjustments can significantly accumulate watermarking costs and computational overhead. As shown in Section 5.3, our experiments confirm that fixed weights indeed slow convergence, especially when the values are selected from only a few pilot runs.

Faster Convergence: To accelerate the watermarking process, we introduce a dynamic loss weight tuning scheduler (DLWS). This scheduler balances the watermarking and generative tasks by dynamically adjusting the loss weights, λ_i and λ_w, in each training iteration. The adjustments are based on the current batch's PSNR (ρ_t) and watermark extraction accuracy (BitACC, μ_t). DLWS works by using a target PSNR (ρ) and a

target BitACC (μ), along with two counters, c_i and c_w, which are both initialized to zero. This approach helps the watermark embedding process converge quickly by alternately adjusting the weights of the generative and watermark losses, as follows.

- Initialize two success counters c_i and c_w to 0.
- If the current BitACC (μ_t) is below the target (μ) ($\mu_t < \mu$), we increase the watermark loss weight (λ_w) to prioritize improving the watermark's influence.
- If the current BitACC (μ_t) meets or exceeds the target (μ) ($\mu_t \geq \mu$), we increase the counter $c_w = c_w + 1$. At this point, we check the PSNR:
  - If the current PSNR (ρ_t) is below its target (ρ) ($\rho_t < \rho$), we increase the image loss weight (λ_i) to focus on improving the image quality.
  - If the current PSNR (ρ_t) meets or exceeds its target (ρ) ($\rho_t \geq \rho$), we increment the counter $c_i = c_i + 1$.
- To continuously improve performance, if c_w exceeds a stability target p, the target BitACC (μ) is increased by a step size s_μ. Similarly, if c_i exceeds the taregt p, the taregt PSNR (ρ) is increased by a step size s_ρ. Then, reset the counters c_i and c_w to zero.

The staged operation of the DLWS algorithm effectively addresses the trade-off between the watermarking and generative tasks, achieving rapid convergence of the watermark embedding process by alternately adjusting the weights of the generative loss and the watermark loss. The algorithm operates in two stages, beginning by setting $\lambda_i = 1$ and $\lambda_w = 0$ to prioritize optimal generative quality. In **Stage 1**, we focus on watermark robustness: as long as the watermark extraction accuracy (BitACC, μ_t) is below the target value μ, the algorithm continuously increases the watermark loss weight λ_w to strengthen its influence. Once μ_t reaches the target, the system proceeds to **Stage 2**, where it performs dynamic weight balancing: loss weights are adjusted by increasing the weight corresponding to the metric (BitACC or PSNR) that falls below its predefined target (see Algorithm 1 for details). Figure 2 visually demonstrates the variation of λ_i and λ_w across the training steps.

Algorithm 1: Dynamic Loss Weight Scheduler

Input: Current PSNR ρ_t and BitACC μ_t;
Current loss weights λ_i and λ_w;
Target PSNR ρ; Target BitACC μ;
Growth factor γ; Patience p;
Step sizes s_μ, s_ρ
Output: Updated λ_i, λ_w, μ, ρ

$c_i \leftarrow 0$; $c_w \leftarrow 0$;
if $\mu_t < \mu$ **then**
  $\lambda_w \leftarrow \lambda_w + 2\gamma$ `// Strengthen watermark`
else
  $c_w \leftarrow c_w + 1$;
  if $\rho_t < \rho$ **then**
    $\lambda_i \leftarrow \lambda_i + \gamma$;
  else
    $c_i \leftarrow c_i + 1$;
  if $c_w > p$ **then**
    $\mu \leftarrow \min(\mu + s_\mu, 1.0)$;
    $c_w \leftarrow 0$;
  if $c_i > p$ **then**
    $\rho \leftarrow \min(\rho + s_\rho, \text{inf})$;
    $c_i \leftarrow 0$;

4. Watermarking Requirements and Threat Model

To ensure that LDM watermarking methods are both functional and resilient, it is necessary to establish clear performance requirements and a realistic threat model. These definitions provide the foundation for designing, evaluating, and comparing watermarking schemes under practical deployment conditions.

4.1. Watermarking Requirements

After the watermarking scheme is established, the watermark must satisfy a comprehensive and multidimensional evaluation to assess its overall performance. For the proposed efficient LDM watermarking method, a set of key performance requirements to ensure both functionality and practicality:

- *Efficiency*: The embedding process should achieve fast convergence with minimal computational overhead, supporting efficient model fine-tuning and scalable deployment.
- *Accuracy*: The watermark must be extracted from generated results with high bit-wise correctness to ensure reliable identification and verification.
- *Imperceptibility*: The watermark should not affect the perceptual quality or semantic fidelity of generated images. This ensures watermarked models can be deployed while maintaining the required generation quality.
- *Robustness*: The watermark should remain detectable after common processing or attacks such as compression, filtering, or diffusion regeneration, which reflects its reliability under real conditions.
- *Payload*: The watermark should carry sufficient information to support traceability and management.

4.2. Threat Model

In real-world applications, LDMs are exposed to a wide range of potential and evolving threats. It is essential that the model owner clearly specifies the deployment scenario, and carefully considers the diverse adversarial strategies that malicious users may adopt to overwrite or remove the watermark. An explicit and well-defined threat model enables the design of watermarking schemes that remain reliable and verifiable for ownership confirmation under realistic adversarial conditions.

Table 2: Evaluation requirements and metrics for efficient LDM watermarking.

Requirement	Metric	Description
Efficiency	Param	Number of trainable parameters involved in watermark embedding.
	AT-x	Time to reach x% extraction accuracy during watermark embedding.
Accuracy	BitACC	Correctness of extracted watermark bits from generated outputs.
Imperceptibility	PSNR, SSIM, LPIPS, SIFID	Visual similarity between clean and watermarked results, involves Peak Signal-to-Noise Ratio, Structural Similarity Index Measure, Learned Perceptual Image Patch Similarity, Single-Image Fréchet Inception Distance.
Robustness	BitACC	Tests watermark detectability under content-level distortions such as compression, color enhancements, or regeneration.
Payload	Bit length (n)	Information capacity for traceability or ownership proof.

In practice, LDMs are often distributed via platforms such as HuggingFace[4], where users interact through prompts and only observe generated outputs. We assume that the generated images from the model may suffer from unauthorized use (box-free watermarking). Malicious users may apply post-processing attacks or may try to remove or overwrite the watermark to make the ownership verification fail or claim authorship Fernandez et al. (2023); Ci et al. (2024); Rezaei et al. (2025). The goal of the watermark embedder is to protect the intellectual property of the trained model. Specifically, he aims to verify the ownership of the model by relying on the model's output in cases of unauthorized use following the release of the model.

5. Experiments

In the experiments, we evaluate our proposed LDM watermarking method through a comprehensive metrics. Initially, we select the embedding layers for applying LoRA module; then, we test the EW-LoRA comparing to the state-of-the-art methods. Robustness performance in image post-processing and watermark overwriting and removal are tested. Finally, we show ablation on the chosen steps in DLWS algorithm.

5.1. Selection of LoRA Embedding Layers

To identify the most suitable layers for LoRA embedding, we conduct an ablation study on the VAE decoder of the LDM. The goal is to determine which components contribute most effectively to watermark embedding while maintaining image quality and parameter efficiency. Specifically, two analysis settings are considered: a *coarse-grained* setting that compares

[4]Stable Diffusion Application: `https://huggingface.co/spaces/stabilityai/stable-diffusion`

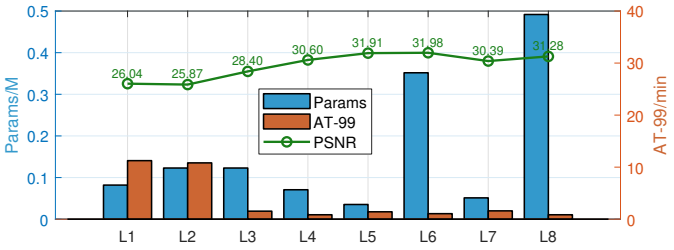


Figure 3: Watermarking performance for applying LoRA in different network blocks in the VAE decoder of an LDM.

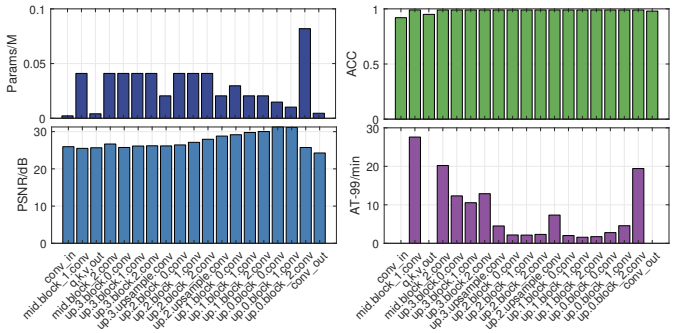


Figure 4: Watermarking performance for applying LoRA in single layer in the VAE decoder of an LDM.

LoRA insertion across different decoder components, and a *fine-grained* setting that examines single-layer insertion within the most promising region identified in the coarse analysis.

Experimental Setup: In the coarse-grained setting, eight decoder components are examined as potential targets for LoRA insertion. As shown in Fig. 3, we named these components as: L1 – convolutional layers in the `mid.block`; L2–L5 – convolutional layers in the four upsampling blocks (`up.x.block`, $x = 3, 2, 1, 0$); L6 – all convolutional layers across all `up.x.blocks`; L7 – the upsample convolutional layers; and L8 – all convolutional layers throughout the VAE decoder. The self-attention layers (`q`, `k`, `v`, and `out`) are excluded due to unstable convergence and poor watermark retention observed during training. The LoRA hyperparameters are fixed as $r, \alpha = 4, 4$ in all configurations to ensure a fair comparison of parameter efficiency and watermark performance.

In the fine-grained setting, we further evaluate each individual layer within the `vae.decoder`, which contains 19 layers (as shown in Fig. 4) in total. Each convolutional and attention-related layer is independently tested to measure its impact on watermark accuracy and image quality, allowing for a detailed assessment of layer sensitivity.

Results and Analysis: Figure 3 presents the coarse-grained results. Among all candidates, the configuration L5 — corresponding to the convolutional layers in `up.0.block` — achieves the best overall performance. It converges with fewer trainable parameters, faster AT-99 convergence, and higher image quality among all settings. The fine-grained results, shown in Figure 4 show that layers such as `conv_in`, `q`, `k`, `v`, `out`, and `conv_out` fail to reach the 99% watermark validation accuracy threshold. In contrast, the three convolutional layers within

Table 3: Watermarking performance when using different r and α.

r, α	**Params↓**	**BitACC↑**	**AT-99↓**	**PSNR↑**
4,4	0.0353	0.991	2.518	31.733
8,8	0.0707	0.990	0.828	31.390
16,16	0.1413	0.990	0.794	32.446
32,32	0.2826	0.991	0.623	32.541
4,8	0.0353	0.992	1.455	30.592
4,16	0.0353	0.991	0.803	31.027
4,32	0.0353	0.991	0.996	32.129
8,4	0.0707	0.990	2.567	31.807
16,4	0.1413	0.992	1.546	32.342
32,4	0.2826	0.992	1.210	31.680

`up.0.block` achieve higher PSNR values with fewer trainable parameters, revealing a favorable trade-off between generative fidelity and watermark detectability.

Based on these analyses, `up.0.block.conv` is identified as the most effective and efficient target for LoRA-based watermark embedding. This layer offers the best balance among watermark accuracy, image quality and training efficiency and is therefore adopted as the default embedding position in all subsequent experiments.

5.2. LoRA Parameter Configuration

We further investigate the influence of the LoRA rank r and scaling factor α (see Eq. 7) on the performance of the proposed method. As summarized in Table 3, increasing both r and α generally enhances image quality and accelerates watermark convergence (lower AT-99), but at the cost of introducing more trainable parameters. When α is increased while keeping r fixed, AT-99 decreases and PSNR improves, with the configuration $(r, \alpha) = (4, 32)$ outperforming $(4, 4)$. Conversely, increasing r alone leads to a higher parameter count without significant gains in image quality, and only a marginal reduction in AT-99 (higher quality and lower AT-99 can be obtained by raising also α). Therefore, to balance the requirements of efficiency, imperceptibility and robustness, we adopt $(r, \alpha) = (4, 32)$ as the default configuration in the subsequent experiments.

5.3. Performance Evaluation and Comparison

We compare the proposed EW-LoRA with several existing watermarking methods, including Stable Signature Fernandez et al. (2023), which belongs to the full parameter modification watermarking (FPW), and AquaLoRA Feng et al. (2024), WMAdapter Ci et al. (2024), and LaWa Rezaei et al. (2025), which fall into the additional parameter watermarking category (APW). To further evaluate the feasibility of the proposed method on different LoRA variants, we conduct experiments using LoHA Hyeon-Woo et al. (2021), PiSSA Meng et al. (2024), and VeRA Kopiczko et al. (2024). All models are trained on the COCO dataset Lin et al. (2014) using 6,000 randomly sampled images and evaluated on 1,000 randomly sampled images from the ArtELingo dataset Mohamed et al.. Each model is trained for one epoch with a batch size of four. When a model reaches the target validation bit-wise accuracy, we record the corresponding convergence time (AT-x) for comparison. Table 4 gives the results of some LDM watermarking methods. Specifically, we compare the proposed EW-LoRA with representative state-of-the-art approaches and further evaluate its adaptability across various LoRA variants and watermark capacities.



Figure 5: Comparison between images generated by original SD v1.4 and our EW-LoRA watermarked SD v1.4: the leftmost six images and the rightmost six are generated, respectively, without and with a text prompt.

Comparison with State-of-the-art: We compare the proposed EW-LoRA with Stable Signature, AquaLoRA, WMAdapter, and LaWa. As shown in Table 4, EW-LoRA outperforms existing methods in both training efficiency and watermarking performance, achieving the fewest trainable parameters while maintaining high watermark accuracy and image fidelity. The table also reports results with and without the proposed Dynamic Loss Weight Scheduler (DLWS), where each Δ value denotes the difference between the results with and without DLWS (i.e., Δ = with − without). Negative $\Delta_{\text{AT-99}}$ values indicate that the model reach 99% watermark validation accuracy faster after applying DLWS, while positive Δ_{PSNR} values demonstrate consistent improvements in image quality across the metric of PSNR. In particular, EW-LoRA achieves high image quality with only 0.0353 MB of trainable parameters. Moreover, DLWS significantly reduces convergence time—for example, Stable Signature saves 24.88 minutes, AquaLoRA saves 235 minutes, WMAdapter saves 0.837 minutes, and LaWa saves 2339 minutes. The generative images in Fig. 5 and Fig. 7 further confirm that watermark embedding introduces negligible degradation in image quality.

LoRA Variants: We evaluate the generality and effectiveness of the proposed EW-LoRA by integrating different LoRA variants, including LoHA, PiSSA, and VeRA, into the same embedding positions as EW-LoRA. As shown in Table 4, EW-VeRA yields the fewest trainable parameters, while EW-PiSSA attains the highest image quality among all variants. Notably, all EW-LoRA variants surpass baselines such as Stable Signature, AquaLoRA, and WMAdapter in the quality of the generative images. These results suggest that the proposed EW-LoRA design is compatible with diverse low-rank adaptation techniques and that reducing the number of trainable parameters helps mitigate degradation of generative image quality.

Capacity: For LDM watermarking methods such as Stable Signature, AquaLoRA, WMAdapter, and LaWa, the watermarking capacity primarily depends on the pre-trained watermark encoder–decoder architecture. In other words, their

Table 4: Performance comparison of LDM watermarking methods. The table compares state-of-the-art approaches, LoRA-based variants, and models with different watermark capacities. Metrics include number of the trainable parameters (Params), bit-wise accuracy (BitACC), achievement time (AT-99), visual quality (PSNR, SSIM, LPIPS, SIFID), and detection performance (TPR@0.1%FPR). Subscripts with ↑ and ↓ indicate performance improvements after applying Dynamic Loss Weight Scheduler (DLWS) compared with the models trained with fixed weights. **Bold** and underlined values denote the best and second-best results, respectively.

Method (Bits)	Params/M↓	BitACC↑	AT-99/min↓	PSNR↑	SSIM↑	LPIPS↓	SIFID↓	$\Delta_{\text{AT-99}}$↓	Δ_{PSNR}↑
Comparison with State-of-the-art									
Stable Signature (48)	49.4900	0.989	$\underline{1.205}_{\downarrow 24.880}$	$29.281_{\uparrow 0.280}$	$0.815_{\uparrow 0.009}$	$0.061_{\downarrow 0.012}$	$0.116_{\downarrow 0.015}$	-24.880	0.280
AquaLoRA (48)	517.5586	0.959	$682^{\dagger}_{\downarrow 235}$	$29.894_{\uparrow 0.202}$	$0.804_{\uparrow 0.128}$	$0.058_{\downarrow 0.023}$	$0.109_{\downarrow 0.019}$	-235	0.202
WMAdapter (48)	1.1978	**0.999**	$\mathbf{0.973}_{\downarrow 0.837}$	$29.067_{\uparrow 1.965}$	$0.842_{\uparrow 0.069}$	$0.045_{\downarrow 0.030}$	$0.110_{\downarrow 0.088}$	-0.837	1.965
LaWa (48)	37.8943	0.970	$3898_{\downarrow 2339}$	$32.808_{\uparrow 0.548}$	$0.861_{\uparrow 0.019}$	$\mathbf{0.011}_{\downarrow 0.024}$	$\mathbf{0.030}_{\downarrow 0.055}$	-2339	0.548
LoRA Variants									
Ours EW-LoRA (48)	$\underline{0.0353}$	$\underline{0.997}$	$2.931_{\downarrow 0.816}$	$32.543_{\uparrow 0.206}$	$\underline{0.870}_{\uparrow 0.001}$	$0.025_{\downarrow 0.004}$	$0.093_{\downarrow 0.031}$	-0.816	0.206
Ours EW-LoHA (48)	0.0706	0.996	$2.791_{\downarrow 0.065}$	$\underline{34.073}_{\uparrow 0.369}$	$\mathbf{0.890}_{\uparrow 0.009}$	$0.018_{\downarrow 0.001}$	$0.082_{\downarrow 0.006}$	-0.065	0.369
Ours EW-PiSSA (48)	0.0353	0.993	$2.020_{\downarrow 0.009}$	$\mathbf{33.370}_{\uparrow 1.797}$	$0.860_{\uparrow 0.025}$	$\underline{0.015}_{\downarrow 0.005}$	$\underline{0.069}_{\downarrow 0.011}$	-0.009	1.797
Ours EW-VeRA (48)	**0.0015**	0.994	$38.148_{\downarrow 55.905}$	$31.516_{\uparrow 2.001}$	$0.825_{\uparrow 0.048}$	$0.025_{\downarrow 0.019}$	$0.104_{\downarrow 0.120}$	-55.905	2.001
Capacity									
Ours EW-LoRA (100)	0.0353	0.994	$3.358_{\downarrow 0.845}$	$28.789_{\uparrow 0.758}$	$0.798_{\uparrow 0.045}$	$0.162_{\downarrow 0.011}$	$0.138_{\downarrow 0.043}$	-0.845	0.758
Ours EW-LoRA (150)	0.0353	0.992	$5.905_{\downarrow 1.021}$	$27.058_{\uparrow 0.976}$	$0.752_{\uparrow 0.101}$	$0.186_{\downarrow 0.028}$	$0.164_{\downarrow 0.078}$	-1.021	0.976

† This represents AT-95 since AquaLoRA cannot reach 99% validation accuracy.

embedding capacity is inherently constrained by the design of the encoder–decoder network. In our experiments, we evaluate EW-LoRA under different payload sizes, as summarized in Table 4. Even when the payload increases from 48 to 100 and 150 bits[5], EW-LoRA maintains stable watermarking performance, exhibiting only minor decreases in PSNR and AT-99. This demonstrates its robustness to larger watermark capacities.

These results highlight two main advantages: (1) By exploiting the intrinsic low-rank adaptation property, EW-LoRA minimizes trainable parameters and memory usage compared with other LDM watermarking methods; and (2) The proposed DLWS effectively balances watermark robustness and generative fidelity, accelerating model convergence and improving overall training efficiency.

5.4. Robustness

In this section, we evaluate the robustness of the watermarking methods against various attacks, including *image post-processing attacks*, *watermark overwriting attacks*, and *watermark removal attacks*. Our standard EW-LoRA variant (Ours EW-LoRA (48) in Table 4) is compared with Stable Signature, AquaLoRA, WMAdapter, and LaWa. For a fair comparison, all watermarking methods employ the same noise layer configuration during training. The *noise layer* used in the watermark decoder consists of random cropping, resizing, and JPEG compression. Specifically, the crop or resize ratio is randomly set to either 0.3 or 0.7 with equal probability, followed by JPEG compression applied with a probability of 0.5, where the quality factor is randomly chosen as 50 or 80 with equal probability.

Image Post-processing Attacks: Table 5 presents the robustness evaluation against common image post-processing operations, including center cropping (aspect ratios of 0.1 and 0.5), rotation (25°), resizing (scaling factors of 0.3 and 0.7), brightness, contrast, and sharpness enhancement (all with adjustment factors of 1.5), and JPEG compression (quality factors of 80 and 50). These settings correspond to P1–P10 in the table. Together with the no-attack condition (P0), we have the results as shown in Table 5. It can be seen that EW-LoRA consistently achieves top-tier robustness, ranking first or second across almost all attack types. This demonstrates that the proposed method effectively preserves watermark robustness under a wide range of post-processing distortions.

Watermark Overwriting Attacks: For watermark overwriting attacks, we include two representative baseline attackers: StegaStamp Tancik et al. (2020), a spatial-domain watermarking framework that embeds message bits into images through an encoder–decoder pipeline[6], and SSL-Watermarking Fernandez et al. (2022), a zero-bit watermarking approach designed to detect ownership of images using self-supervised latent space[7].

To evaluate the resistance of the proposed method to watermark overwriting, we simulate attack scenarios where new watermarks are embedded into images generated by watermarked models using the two representative image watermarking schemes, StegaStamp and SSL-Watermarking. Table 6 reports the watermark BitACC extracted from both the overwritten images (i.e., images generated by the watermarked model and subsequently embedded with another watermark) and the original images generated by the same watermarked model. As shown, EW-LoRA achieves BitACC values comparable to those of the best-performing WMAdapter (with only a 0.002 BitACC gap), demonstrating strong robustness against overwriting watermark and effective preservation of the original watermark under overwriting attacks.

[5]Payloads of 100 and 150 bits use the StegaStamp Tancik et al. (2020) decoder as the pre-trained watermark decoder, since HiDDeN Zhu et al. (2018) does not support such capacities.

[6]StegaStamp: `https://github.com/ningyu1991/ArtificialGANFingerprints`. The number of embedded bits is set to $n = 200$.

[7]SSL-Watermarking: `https://github.com/facebookresearch/ssl_watermarking`.

Table 5: Watermarking performance comparison in the presence of image post-processing attacks.

Method	P0	P1	P2	P3*	P4	P5	P6	P7	P8	P9*	P10*
Stable Sign.	0.989	0.911	0.991	0.602	0.583	0.887	0.987	0.990	0.994	0.778	0.704
AquaLoRA	0.959	0.877	0.884	0.646	0.643	0.877	0.814	0.830	0.875	0.869	0.709
WMAdapter	**0.999**	0.921	**0.994**	0.639	0.651	0.973	0.988	0.992	0.997	0.651	**0.788**
LaWa	0.999	0.493	0.523	0.468	0.500	0.459	**0.999**	**0.999**	**0.999**	**0.886**	0.709
EW-LoRA	0.997	**0.922**	0.993	**0.665**	**0.706**	**0.983**	0.989	0.992	0.997	0.849	0.781

The process marked with $*$ means it has been seen in the noise layer.

Table 6: Watermarking performance in the presence of watermark overwriting attacks.

Attack	Stable Sign.	AquaLoRA	WMAdapter	LaWa	EW-LoRA
StegaStamp	0.862/0.989	0.844/0.959	0.941/0.999	0.794/0.999	0.939/0.997
SSL-WM.	0.988/0.989	0.959/0.959	0.998/0.999	0.999/0.999	0.997/0.997

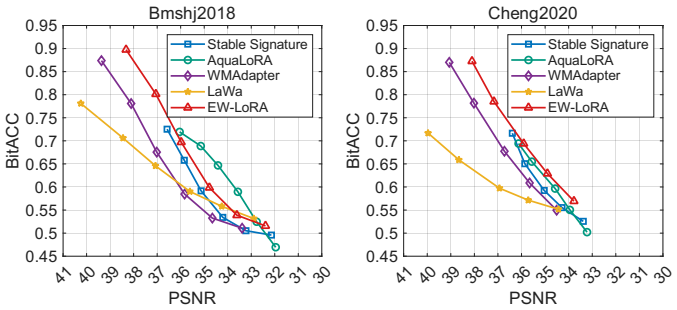


Figure 6: Watermarking performance under watermark removal attacks.

Table 7: Cross-dataset results. The models are trained on one dataset and test on the others, and their performance is evaluated by BitACC, AT-95 and visual quality metrics PSNR/SSIM/LIPIS/SIFID.

Train	Test	BitACC↑	AT-95↓	PSNR↑	SSIM↑	LPIPS↓	SIFID↓
COCO	COCO	**0.987**	1.661	32.981	0.867	0.013	0.031
	CelebA	0.957	2.253	**38.145**	0.899	**0.008**	**0.014**
	Flickr	0.958	3.178	33.715	0.832	0.012	0.037
	ArtELingo	0.968	**1.342**	33.740	**0.901**	0.015	0.034
CelebA	COCO	0.982	1.485	30.084	0.792	0.028	0.083
	CelebA	**0.992**	2.123	**33.791**	0.818	**0.017**	**0.048**
	Flickr	0.966	3.154	30.669	0.729	0.026	0.095
	ArtELingo	0.964	**1.175**	31.010	**0.841**	0.032	0.094
Flickr	COCO	**0.992**	**1.417**	31.772	0.845	0.021	0.055
	CelebA	0.984	1.898	**36.177**	**0.873**	**0.017**	**0.035**
	Flickr	0.977	2.123	31.910	0.786	0.021	0.065
	ArtELingo	0.978	2.124	32.136	0.872	0.025	0.064
ArtELingo	COCO	**0.988**	**1.182**	32.468	0.846	0.014	0.039
	CelebA	0.977	1.987	**37.390**	0.879	**0.010**	**0.022**
	Flickr	0.968	2.351	33.228	0.808	0.014	0.043
	ArtELingo	0.972	1.456	33.513	**0.897**	0.017	0.041

Table 8: Watermarking performance comparison across different base LDMs

Model	Params	BitACC	AT-99	PSNR	SSIM	LPIPS	SIFID
SD v1.4	0.0353	0.997	2.931	32.543	0.869	0.035	0.093
SD v2.1	0.1433	0.995	2.657	32.011	0.868	0.034	0.119
SD v3.5-l	0.1433	0.996	2.881	32.293	0.864	0.030	0.104
DiT	0.0353	0.997	2.689	33.173	0.870	0.020	0.090

Watermark Removal Attacks: To evaluate the robustness of the watermarking methods against watermark removal attacks, we employ two learned image compression autoencoders, Bmshj2018 Ballé et al. (2018) and Cheng2020 Cheng et al. (2020), which serve as representative encoder–decoder architectures capable of reconstructing images while potentially removing embedded watermarks. We use the implementations provided in the `CompressAI.zoo` library[8] and vary the compression rates from $1, 2, 3, 4, 5$ for Bmshj2018 and from $1, 2, 3, 4, 5, 6$ for Cheng2020.

For watermark removal attacks, we assess robustness by passing the watermarked images through the above compression models. The PSNR is computed between the original watermarked images and their reconstructed counterparts, while the BitACC is calculated on the reconstructed images to measure the watermark's recoverability after compression. As shown in Figure 6, EW-LoRA demonstrates superior robustness compared to the other three methods, achieving consistently higher watermark BitACC under the same distortion levels (same PSNR) on the watermarked images (except for the case of AquaLoRA under the attack of Bmshj2018). These results indicate that EW-LoRA effectively preserves watermark robustness even after severe compression-based reconstruction.

5.5. *Generalization across Datasets and Models*

In this section, we evaluate the generalization capability of EW-LoRA across different datasets and base models. Specifically, four datasets, COCO Lin et al. (2014), CelebA Liu et al. (2015), Flickr Young et al., and ArtELingo Mohamed et al., are selected to represent diverse visual domains, including everyday scenes, human faces, web imagery, and artistic paintings. For each dataset, EW-LoRA is trained on one and tested on the others to assess its robustness under cross-domain data distributions. In addition, we examine EW-LoRA's adaptability to different generative architectures by conducting experiments on four base models: Stable Diffusion v1.4 (SD v1.4), Diffusion Transformer (DiT), Stable Diffusion v2.1 (SD v2.1), and Stable Diffusion v3.5-large (SD v3.5-l). All these models adopt a VAE as the image reconstruction backbone, where the VAE decoder maps the latent representations back to the pixel space. Among them, SD v1.4 and DiT share the same VAE decoder consisting of approximately 49.49 MB of parameters, while SD v2.1 and SD v3.5-l utilize slightly larger decoders with 49.63 MB and 49.69 MB of parameters, respectively.

Datasets: As shown in Table 7, when evaluated across unseen datasets, EW-LoRA exhibits strong generalization, with the embedded watermark remaining largely decodable and the BitACC consistently exceeding 0.95. Models trained on different datasets show similar robustness when tested on other domains, indicating that the learned watermark representation is not sensitive to data distribution shifts. In particular, models tested on the CelebA dataset achieve the highest image fidelity, which reflected by higher PSNR (>36 dB) and lower LPIPS/SIFID scores, although the watermark accuracy is slightly lower than that of other datasets. Overall, the impact of dataset differences on watermark performance is minor and can

[8]ConpressAI: `https://github.com/InterDigitalInc/CompressAI`

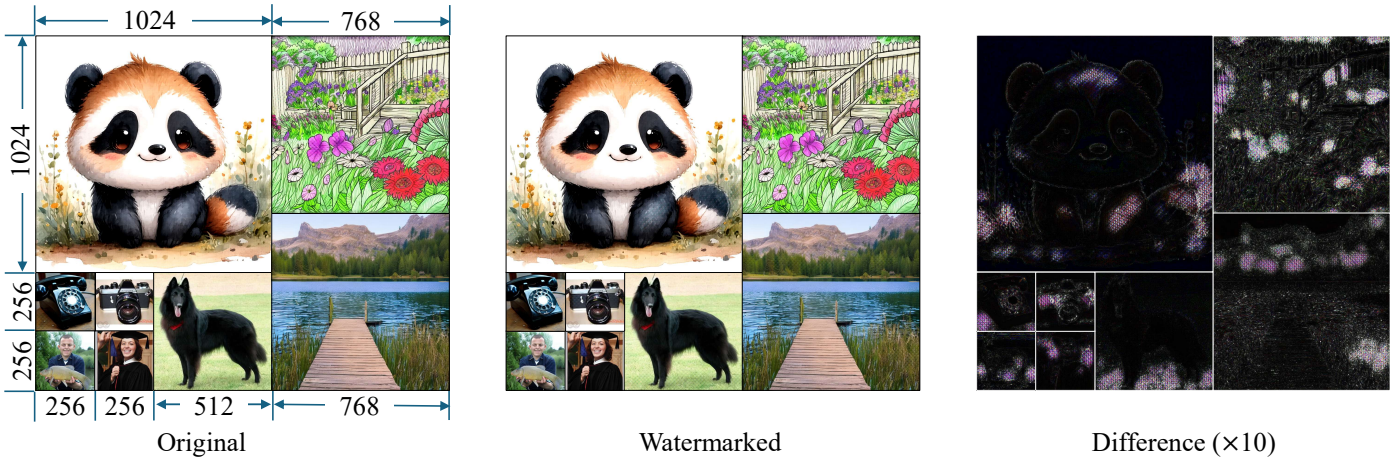


Figure 7: Visualizations for images generated by original LDMs and our EW-LoRA watermarked LDMs, where the images are generated by SD v1.4 (with size of size 256×256), DiT (512×512), SD v2.1 (768×768) and SD v3.5-large (1024×1024).

be considered negligible, confirming the stable generalization and transferability of EW-LoRA across diverse visual domains.

Base Models: As shown in Table 8, we apply EW-LoRA to different base models for watermarking. The results show that even when the underlying generative architectures differ substantially, ranging from the U-Net backbone used in Stable Diffusion to the transformer-based structure adopted by DiT, EW-LoRA consistently achieves high watermark accuracy (BitACC ≥ 0.995) and comparable image quality across all models. This demonstrates that EW-LoRA is architecture-agnostic and can be seamlessly integrated into diverse generative frameworks without compromising watermark robustness or visual imperceptibility. Figure 7 further illustrates representative samples generated by SD v1.4 (256×256), SD v2.1 (768×768), DiT (512×512), and SD v3.5-l (1024×1024). From the difference images, we observe that the watermark is primarily distributed within texture-rich and detail-dense regions of the generated images. These regions provide abundant local variations that effectively conceal the watermark signal, ensuring that the embedded watermark remains visually imperceptible to the human eye (invisible watermarking).

5.6. Analysis of DLWS

In the proposed Dynamic Loss Weight Scheduler (DLWS), as detailed in Algorithm 1, the targets for PSNR and BitACC are adjusted dynamically throughout the training process. Initially, these thresholds are set conservatively (typically below 40 dB for PSNR and 0.95 for BitACC) to prevent overfitting one objective while hindering the convergence of the other. Once the validation performance reaches these preset targets, the thresholds are gradually increased by step factors s_ρ and s_μ to further challenge the model. If these targets are consistently met over several iterations, the model is considered converged. Figure 8 illustrates the impact of different combinations of (s_ρ, s_μ), with the values assessed based on the score of norm(PSNR) + BitACC. The best combination is selected by choosing the pair that yields the highest value. From the figure, we can select the best parameter values $(s_\rho, s_\mu) = (2.5, 0.08)$.

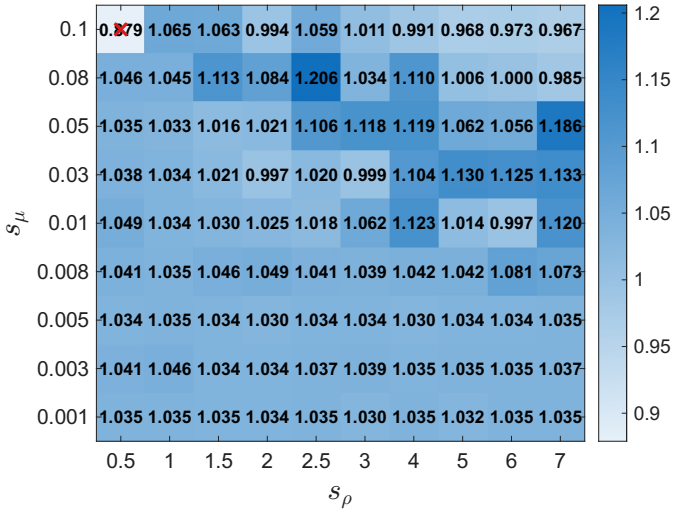


Figure 8: A heatmap showing the scores for evaluating the optimal watermarking performance, based on the combinations of the step factors s_ρ, s_μ.

6. Conclusion

In summary, the proposed EW-LoRA framework demonstrates that watermarking for latent diffusion models can be achieved in a highly efficient and unobtrusive manner. By embedding watermarks into the VAE decoder through low-rank adaptation, EW-LoRA preserves generation fidelity while significantly reducing the number of trainable parameters. The incorporated dynamic loss weight scheduler (DLWS) enables faster and more stable convergence by adaptively balancing the objectives of image generation and watermark embedding. As a general and lightweight optimization mechanism, DLWS also offers potential for broader application across various watermarking paradigms. Future work will explore finer layer selection strategies within diffusion components, investigate adaptive rank allocation for LoRA modules, and extend the framework to other generative architectures such as autoregressive-based diffusion models for a unified watermarking solution.

References

Alam, I., Islam, M.T., Woo, S.S., 2025. Saliency-aware diffusion reconstruction for effective invisible watermark removal, in: Proceedings of the ACM on Web Conference, pp. 849–853. doi:10.1145/3701716.3715519.

Asnani, V., Collomosse, J., Bui, T., Liu, X., Agarwal, S., 2024. ProMark: Proactive diffusion watermarking for causal attribution, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10802–10811. doi:10.1109/CVPR52733.2024.01027.

Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N., 2018. Variational image compression with a scale hyperprior, in: Proceedings of International Conference on Learning Representations.

Barni, M., Pérez-González, F., Tondi, B., 2021. DNN watermarking: Four challenges and a funeral, in: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, pp. 189–196. doi:10.1145/3437880.3460399.

Cheng, Z., Sun, H., Takeuchi, M., Katto, J., 2020. Learned image compression with discretized gaussian mixture likelihoods and attention modules, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7939–7948. doi:10.1109/CVPR42600.2020.00796.

Ci, H., Song, Y., Yang, P., Xie, J., Shou, M.Z., 2024. WMAdapter: Adding watermark control to latent diffusion models, in: Proceedings ofInternational Conference on Machine Learning.

Crowson, K., Baumann, S.A., Birch, A., Abraham, T.M., Kaplan, D.Z., Shippole, E., 2024. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers, in: Proceedings of International Conference on Machine Learning.

Czolbe, S., Krause, O., Cox, I., Igel, C., 2020. A loss function for generative neural networks based on watson's perceptual model, in: Proceedings of Neural Information Processing Systems, pp. 2051–2061.

Darvish Rouhani, B., Chen, H., Koushanfar, F., 2019. DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks, in: International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 485–497. doi:10.1145/3297858.3304051.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., Rombach, R., 2024. Scaling rectified flow transformers for high-resolution image synthesis, in: International Conference on Machine Learning.

Fei, J., Xia, Z., Tondi, B., Barni, M., 2022. Supervised GAN Watermarking for Intellectual Property Protection, in: in Proceedings of IEEE International Workshop on Information Forensics and Security, pp. 1–6. doi:10.1109/WIFS55849.2022.9975409.

Feng, W., Zhou, W., He, J., Zhang, J., Wei, T., Li, G., Zhang, T., Zhang, W., Yu, N., 2024. AquaLoRA: Toward white-box protection for customized stable diffusion models via watermark LoRA, in: Proceedings of International Conference on Machine Learning, pp. 13423–13444.

Fernandez, P., Couairon, G., Jégou, H., Douze, M., Furon, T., 2023. The stable signature: Rooting watermarks in latent diffusion models, in: Proceedings of IEEE/CVF International Conference on Computer Vision, pp. 22466–22477. doi:10.1109/ICCV51070.2023.02053.

Fernandez, P., Level, A., Furon, T., . What Lies Ahead for Generative AI Watermarking. Technical Report. URL: https://blog.genlaw.org/pdfs/genlaw_icml2024/27.pdf.

Fernandez, P., Sablayrolles, A., Furon, T., Jégou, H., Douze, M., 2022. Watermarking images in self-supervised latent spaces, in: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3054–3058. doi:10.1109/ICASSP43922.2022.9746058.

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., 2021. LoRA: Low-rank adaptation of large language models, in: Proceedings of International Conference on Learning Representations.

Hyeon-Woo, N., Ye-Bin, M., Oh, T.H., 2021. FedPara: Low-rank hadamard product for communication-efficient federated learning, in: Proceedings of International Conference on Learning Representations.

Jain, A., Kobayashi, Y., Murata, N., Takida, Y., Shibuya, T., Mitsufuji, Y., Cohen, N., Memon, N., Togelius, J., 2025. Forging and removing latent-noise diffusion watermarks using a single image doi:10.48550/arXiv.2504.20111.

Kopiczko, D.J., Blankevoort, T., Asano, Y.M., 2024. VeRA: Vector-based random matrix adaptation, in: Proceedings of International Conference on Learning Representations.

Li, L., Zhang, W., Barni, M., 2023. Universal blackmarks: Key-image-free blackbox multi-bit watermarking of deep neural networks. IEEE Signal Processing Letters 30, 36–40. doi:10.1109/LSP.2023.3239737.

Li, Y., Tondi, B., Barni, M., 2021a. Spread-transform dither modulation watermarking of deep neural network. Journal of Information Security and Applications 63, 103004. doi:10.1016/j.jisa.2021.103004.

Li, Y., Wang, H., Barni, M., 2021b. A survey of deep neural network watermarking techniques. Neurocomputing 461, 171–193. doi:10.1016/j.neucom.2021.07.051.

Lin, D., Tondi, B., Li, B., Barni, M., 2025. A CycleGAN watermarking method for ownership verification. IEEE Transactions on Dependable and Secure Computing 22, 1040–1054. doi:10.1109/TDSC.2024.3424900.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft COCO: Common objects in context, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 740–755. doi:10.1007/978-3-319-10602-1_48.

Liu, Y., Li, Z., Backes, M., Shen, Y., Zhang, Y., 2023. Watermarking Diffusion Model. arXiv:2305.12502 doi:10.48550/arXiv.2305.12502, arXiv:2305.12502.

Liu, Z., Luo, P., Wang, X., Tang, X., 2015. Deep learning face attributes in the wild, in: Proceedings of International Conference on Computer Vision.

Meng, F., Wang, Z., Zhang, M., 2024. PiSSA: Principal singular values and singular vectors adaptation of large language models, in: Proceedings of Neural Information Processing Systems.

Mohamed, Y., Abdelfattah, M., Alhuwaider, S., Li, F., Zhang, X., Church, K., Elhoseiny, M., . ArtELingo: A million emotion annotations of WikiArt with emphasis on diversity over language and culture, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 8770–8785.

Müller, A., Lukovnikov, D., Thietke, J., Fischer, A., Quiring, E., 2025. Black-box forgery attacks on semantic watermarks for diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 20937–20946.

Peebles, W., Xie, S., 2023. Scalable diffusion models with transformers, in: Proceedings of IEEE/CVF International Conference on Computer Vision, pp. 4172–4182. doi:10.1109/ICCV51070.2023.00387.

Peng, S., Chen, Y., Wang, C., Jia, X., 2025. Intellectual property protection of diffusion models via the watermark diffusion process , 290–305.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R., 2023. SDXL: Improving latent diffusion models for high-resolution image synthesis, in: Proceedings of International Conference on Learning Representations.

Rezaei, A., Akbari, M., Alvar, S.R., Fatemi, A., Zhang, Y., 2025. LaWa: Using latent space for in-generation image watermarking, in: Proceedings of European Conference on Computer Vision, pp. 118–136. doi:10.1007/978-3-031-73024-5_8.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B., 2022. High-resolution image synthesis with latent diffusion models, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10674–10685. doi:10.1109/CVPR52688.2022.01042.

Šarčević, T., Karlowicz, A., Mayer, R., Baeza-Yates, R., Rauber, A., 2024. U Can't Gen This? A survey of intellectual property protection methods for data in generative AI. arXiv:2406.15386 doi:10.48550/arXiv.2406.15386, arXiv:2406.15386.

Song, J., Meng, C., Ermon, S., 2020. Denoising diffusion implicit models, in: Proceedings of International Conference on Learning Representations.

Tancik, M., Mildenhall, B., Ng, R., 2020. StegaStamp: Invisible hyperlinks in physical photographs, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2117–2126.

Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S., 2017. Embedding watermarks into deep neural networks, in: Proceedings of ACM on International Conference on Multimedia Retrieval, pp. 269–277. doi:10.1145/3078971.3078974.

Xue, M., Zhang, Y., Wang, J., Liu, W., 2022. Intellectual property protection for deep learning models: Taxonomy, methods, attacks, and evaluations. IEEE Transactions on Artificial Intelligence 3, 908–923. doi:10.1109/TAI.2021.3133824.

Yang, Z., Zeng, K., Chen, K., Fang, H., Zhang, W., Yu, N., 2024. Gaussian shading: Provable performance-lossless image watermarking for diffusion models, in: Proceedings of 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12162–12171. doi:10.1109/CVPR52733.2024.01156.

Young, P., Lai, A., Hodosh, M., Hockenmaier, J., . From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions 2, 67–78.

Yu, N., Skripniuk, V., Abdelnabi, S., Fritz, M., 2021. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data, in: Proceedings of IEEE/CVF International Conf. on Computer Vision, pp. 14428–14437. doi:10.1109/ICCV48922.2021.01418.

Yuan, Z., Li, L., Wang, Z., Zhang, X., 2024. Watermarking for stable diffusion models. IEEE Internet of Things Journal 11, 35238–35249. doi:10.1109/JIOT.2024.3434656.

Zhang, J., Chen, D., Liao, J., Zhang, W., Feng, H., Hua, G., Yu, N., 2022. Deep model intellectual property protection via deep watermarking. IEEE Transactions on Pattern Analysis and Machine Intelligence 44, 4005–4020. doi:10.1109/TPAMI.2021.3064850.

Zhang, J., Chen, D., Liao, J., Zhang, W., Hua, G., Yu, N., 2020. Passport-aware normalization for deep model protection, in: Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 22619–22628.

Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O., 2018. The unreasonable effectiveness of deep features as a perceptual metric, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 586–595.

Zhao, Y., Pang, T., Du, C., Yang, X., Cheung, N.M., Lin, M., 2023. A recipe for watermarking diffusion models. doi:10.48550/arXiv.2303.10137, arXiv:2303.10137.

Zhu, J., Kaplan, R., Johnson, J., Fei-Fei, L., 2018. HiDDeN: Hiding data with deep networks, in: Proceedings of the European Conference on Computer Vision, pp. 657–672. doi:10.1007/978-3-030-01267-0_40.