Gradient-free Decoder Inversion
in Latent Diffusion Models

Seongmin Hong[1] **Suh Yoon Jeon**[1] **Kyeonghyun Lee**[1]

Ernest K. Ryu[2,*] **Se Young Chun**[1,3,*]

[1]Dept. of Electrical and Computer Engineering, [3]INMC & IPAI, Seoul National University
[2]Dept. of Mathematics, University of California, Los Angeles
{smhongok, euniejeon, litiphysics, sychun}@snu.ac.kr, eryu@math.ucla.edu
Project page: `https://smhongok.github.io/dec-inv.html`

Abstract

In latent diffusion models (LDMs), denoising diffusion process efficiently takes place on latent space whose dimension is lower than that of pixel space. Decoder is typically used to transform the representation in latent space to that in pixel space. While a decoder is assumed to have an encoder as an accurate inverse, exact encoder-decoder pair rarely exists in practice even though applications often require precise inversion of decoder. In other words, encoder is not the left-inverse but the right-inverse of the decoder; decoder inversion seeks the left-inverse. Prior works for decoder inversion in LDMs employed gradient descent inspired by inversions of generative adversarial networks. However, gradient-based methods require larger GPU memory and longer computation time for larger latent space. For example, recent video LDMs can generate more than 16 frames, but GPUs with 24 GB memory can only perform gradient-based decoder inversion for 4 frames. Here, we propose an efficient gradient-free decoder inversion for LDMs, which can be applied to diverse latent models. Theoretical convergence property of our proposed inversion has been investigated not only for the forward step method, but also for the inertial Krasnoselskii-Mann (KM) iterations under mild assumption on cocoercivity that is satisfied by recent LDMs. Our proposed gradient-free method with Adam optimizer and learning rate scheduling significantly reduced computation time and memory usage over prior gradient-based methods and enabled efficient computation in applications such as noise-space watermarking and background-preserving image editing while achieving comparable error levels.

1 Introduction

Deep generative models have been actively investigated over the past decade across numerous modalities such as image [12, 41, 42, 37, 11, 35, 38], video [5, 4, 47, 52, 45, 46], audio [18, 21, 22] and molecular structure [17, 13]. They can sample new data points from the distribution of the training set, can model priors to solve regression problems, and can be used for applications like editing, retrieval, and density estimation. Representative classes of deep generative models include generative adversarial networks (GANs), variational autoencoders (VAEs), normalizing flows (NFs), and diffusion models (DMs). They *used to be* known not to simultaneously satisfy the three key requirements: (i) high-quality samples, (ii) diversity, and (iii) fast sampling, also referred to as the generative learning trilemma [50]. However, recent DMs such as Rectified Flow [23], Adversarial Diffusion Distillation [40] and Consistency models [43] require just one step for sampling. Thus,

---

[*]Co-corresponding authors

Table 1: Comparison of deep generative models. DMs have overcome the *generative learning trilemma* [50], but they still lack invertibility compared to other models. In particular, LDMs have necessitated additional decoder inversion that has traditionally been addressed through memory-intensive and time-consuming gradient-based optimization methods. Here, we propose a method for efficiently (*i.e.*, gradient-free) ensuring invertibility in LDM.

	Generative learning trilemma [50]			How to achieve invertibility?
	High-quality samples	Diversity	Fast sampling	
GAN	✓	✗	✓	Optimization / learning-based GAN inversion
NF	✗	✓	✓	Naturally exact inversion
DM	✓	✓	✗ $_{('21)}$ → ✓ $_{('24)}$	Diffusion: Solving ODE backward (LDM) Decoder: Optimization-based inversion

DMs have overcome the trilemma and the old problem of slow sampling in recent DMs is no longer a concern as shown in the last row of Table 1. As a result, DMs have become one of the most prominent deep generative models, especially for image and video. However, there are still remaining challenges for DMs such as achieving *invertibility*, which is well-supported by other models (see Table 1).

Achieving invertibility for deep generative models has been an important topic. In GANs, prior works proposed GAN inversion techniques [49] (*i.e.*, to find the latent vector z that generated the image) and their applications such as real image editing [1]. NFs are naturally invertible deep generative models by construction with related interesting applications [17]. There have also been attempts to achieve and utilize invertibility for DMs. The most popular one is the naïve DDIM inversion that reverses the sampling process of DDIM's deterministic denoising [41], *i.e.*, adding the estimated noise. While the naïve DDIM inversion enables image editing, it is known to be somewhat inaccurate for seeking the true latent [14], for watermarking [48] and for background-preserving image editing [32]. To better ensure the exactness of the inversion, several prior works [44, 51, 31, 14, 10] have proposed more tailored algorithms than the naïve DDIM inversion.

The use of latents in DMs has made ensuring invertibility more difficult. Latent diffusion models (LDMs) were proposed to move the diffusion denoising process from the pixel space to the (low-dimensional) latent space, thus efficiently generating high-quality and large-scale samples [37]. This issue may be critical since many recent popular DMs operate in latent space [37, 47, 52, 27, 40, 45, 5]. However, latents are usually lossy compression of pixels, making one-to-one mapping between the latent and pixel spaces very challenging. Thus, the accuracy of inversion in LDMs is typically lower than that in pixel-space [44, 31, 14], requiring additional efforts to compensate for it as illustrated in Figure 1b. One could naïvely employ a gradient-based GAN inversion method to the decoder of LDMs, but it required very large GPU memory and computation time [14], especially for large-scale LDMs. Moreover, recent video LDMs [5, 4, 47, 52, 45, 46] generate dozens of frames at once, making gradient-based GAN inversion infeasible in a single GPU.

In this work, we aim to achieve better invertibility in LDM as shown in Table 1. Specifically, we propose decoder inversion to overcome the difficulty of ensuring invertibility in LDM due to the inexact encoder-decoder pair. The proposed decoder inversion is gradient-free, which is faster and more memory-efficient than the gradient-based methods suggested in GAN inversion. Section 3 describes the motivation and analysis of our decoder inversion, including the theoretical convergence guarantee under mild assumption not only for the vanilla forward step method, but also for the inertial Krasnoselskii-Mann (KM) iterations. The assumption for our theorems is also validated by showing experimental convergence to the ground truth as well as the effectiveness of momentum. Section 4 described how to refine our algorithm by integrating popular optimization techniques such as Adam [16] and learning rate scheduling as practical extensions of our gradient-free decoder inversion, demonstrating that our method works well on various latest LDMs compared to existing gradient-based methods. Lastly, Section 5 showcases interesting applications (noise space watermarking and background-preserving image editing) where our proposed gradient-free decoder inversion can be effectively utilized. The contributions of this work are:
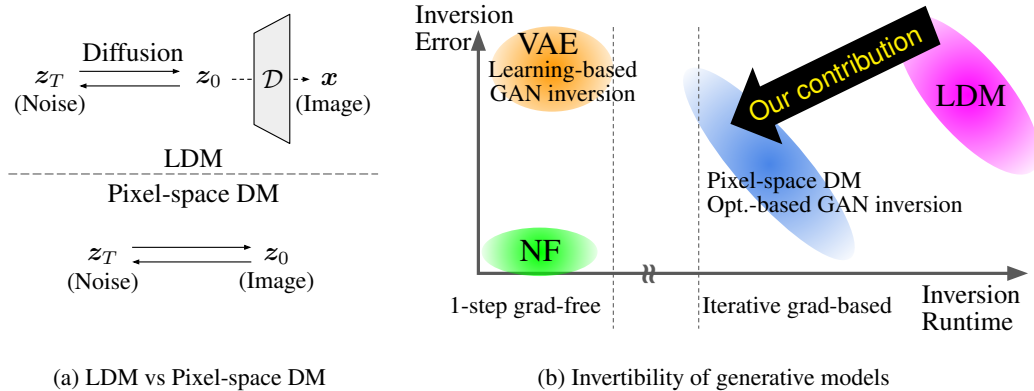
(a) LDM vs Pixel-space DM

(b) Invertibility of generative models

Figure 1: (a) The difference between LDM and pixel-space DM lies in the use of the decoder ($\mathcal{D}$). (b) This difference has caused the performance gap in exact inversions. Pixel-space DMs and gradient-based GAN inversion methods are located on the right side due to iterative gradient back-propagations. If gradient-based decoder inversions are used, which are computationally intensive, LDM is located on the rightmost side to address the lossiness of latents. Our proposed gradient-free decoder inversion method allows us to efficiently handle the transformation between latent and pixel spaces.

- proposing a gradient-free decoder inversion algorithm to achieve better invertibility in diverse latest LDMs. Our method has the following advantages with experimental evidences:
  - **Fast**: up to $5\times$ faster, 1.89 s vs 9.51 s to achieve -16.4 dB (Fig. 3c and Tab. S1c)
  - **Accurate**: up to 2.3 dB lower, -21.37 dB vs -19.06 dB in 25.1 s (Fig. 3b and Tab. S1b)
  - **Memory-efficient**: up to **89%** can be saved, 7.13 GB vs 64.7 GB (Fig. 3b)
  - **Precision-flexible**: 16-bit vs 32-bit (Fig. 3)
- theoretically guaranteeing the convergence to the ground truth not only for the vanilla forward step method, but also for the inertial KM iterations,
- showcasing that our proposed gradient-free decoder inversion can be used in interesting applications.

2 Backgrounds

2.1 Latent diffusion models (LDMs)

Stable Diffusion 2.1 [37] is a widely known open-source text-to-image generation model. LaVie [47] is a text-conditioned video generation model, which can generate consecutive frames per inference. InstaFlow [24] is a one-step text-to-image generation model that can generate images whose quality is as good as Stable Diffusion. These three LDMs will be used in all the experiments of our work.

Exact inversion of accelerated DMs. Another factor that makes the exact inversion of DMs difficult, besides the use of latents, is acceleration. Due to the use of high-order ODE solvers, the exact inversion of DPM-Solvers [25, 26] is not as straightforward as in DDIM. Exact inversion algorithms for some DPM-Solvers have been proposed [14], but obtaining the exact inverses for other accelerated DMs [23, 40, 43, 24] has not been explored.

2.2 Optimization-based GAN inversion

For a target image $\boldsymbol{x}$, existing optimization-based GAN inversion methods typically perform the following optimization with respect to a latent vector $\boldsymbol{z}$ (usually assumed to lie on a simple distribution such as $\mathcal{N}(0,1)$):

$$\min_{\boldsymbol{z}} \quad \ell(\boldsymbol{x}, \mathcal{G}(\boldsymbol{z})), \tag{1}$$

where ℓ is a distance metric and $\mathcal{G}$ is the generator part of GAN [49]. Most works solve Eq. (1) through backpropagation with an optimizer. For optimizers, Adam [16] is employed in [1], and

L-BFGS [20] is employed in [54]. Some more recent works optimized the transformed latent feature w, which can be defined as $w = \text{MLP}(z)$, in StyleGAN [15].

To exploit the prior knowledge that z lies on the range space of an encoder, prior works often perform the following two methods:

(A) The output of the encoder $\mathcal{E}$ is used as an initial point of the optimization process.

(B) The domain-guided encoder is used to regularize the latent code within the semantic domain of the generator. For example, $\min_{z} \|x - \mathcal{G}(z)\|_2 + \lambda_{\text{reg}} \|z - \mathcal{E}(\mathcal{G}(z))\|_2$.

Since we already have the encoder, we employ (A), too. On the other hand, we will utilize the encoder in a novel way over (B).

2.3 Gradient-based decoder inversion in LDMs

Recent works [14] performed the same as Eq. (1) (hence they are iterative as illustrated in Figure 1b) for the decoder inversion:

$$\min_{z_0} \quad \ell(x, \mathcal{D}(z_0)), \tag{2}$$

with a gradient-based method. For simplicity, we omit the subscript by setting $z_0 = z$. The vanilla gradient descent algorithm can be expressed as follows:

$$z^{k+1} = z^k - \rho \nabla_z \ell(x, \mathcal{D}(z^k)). \tag{3}$$

Advancements in backpropagation algorithms and libraries have made the computation of $\nabla_z \mathcal{D}(z^k)$ more efficient, but it still requires a significant amount of GPU memory usage and lengthy runtime compared to inference. As the size of generated outputs of recent LDMs continues to increase, the computation of $\nabla_z \mathcal{D}(z^k)$ through backpropagation is getting more and more burdensome.

3 Gradient-free decoder inversion in LDMs

In this work, as an alternative method to Eq. (3), we propose the following decoder inversion method, which is a form of the forward step method [39]:

$$z^{k+1} = z^k - \rho(\mathcal{E}(\mathcal{D}(z^k)) - \mathcal{E}(x)), \tag{4}$$

where the initial latent vector estimate $z^0 = \mathcal{E}(x)$ that is the same as (A) in Section 2.2.

3.1 Motivation

The definition of the decoder inversion problem for the given image x is as follows:

$$\underset{z \in \mathbb{R}^F}{\text{find}} \quad x = \mathcal{D}(z). \tag{5}$$

Since directly dealing with Eq. (5) is impractical, we relax it as solving the following problem:

$$\underset{z \in \mathbb{R}^F}{\text{find}} \quad \mathcal{E}(x) = \mathcal{E}(\mathcal{D}(z)). \tag{6}$$

The following Remark 1 demonstrates that solving Eq. (6) is easier than solving Eq. (5).

Remark 1 (Eq. (6) is easier than Eq. (5)). $\{z | x = \mathcal{D}(z)\} \subset \{z | \mathcal{E}(x) = \mathcal{E}(\mathcal{D}(z))\}$.

Remark 1 is true because $x = \mathcal{D}(z) \Rightarrow \mathcal{E}(x) = \mathcal{E}(\mathcal{D}(z))$. Equation (6) is equivalent to the following:

$$\underset{z \in \mathbb{R}^F}{\text{find}} \quad z = z - \rho(\mathcal{E}(\mathcal{D}(z)) - \mathcal{E}(x)), \quad \forall \rho \in \mathbb{R} \cap \{0\}^C \tag{7}$$

where Eq. (7) refers to finding a fixed point of an operation $\mathcal{I} - \rho(\mathcal{E}(\mathcal{D}(\cdot)) - \mathcal{E}(x))$ whose another form is Eq. (4), the forward step method.

3.2 Convergence analysis on forward step method

Here, we demonstrate that our method converges to a fixed-point under reasonable conditions. Furthermore, although we solved an easier problem represented by Eq. (6) rather than directly solving Eq. (5), we show that our proposed method surprisingly finds the true solution of Eq. (6).

Theorem 1 (Convergence of the forward step method). *Let $\beta > 0$, $0 < \rho < 2\beta$, and $\boldsymbol{x} \in \mathbb{R}^N$. Assume $\mathcal{T}(\cdot) = \mathcal{E} \circ \mathcal{D}(\cdot) - \mathcal{E}(\boldsymbol{x})$ is continuous. Consider the iteration*

$$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k - \rho \mathcal{T} \boldsymbol{z}^k \quad for \quad k = 0, 1, \ldots \tag{8}$$

Assume $\boldsymbol{z}^\star$ is a zero of $\mathcal{T}$ (i.e., $\mathcal{T} \boldsymbol{z}^\star = 0$) and

$$\langle \mathcal{T} \boldsymbol{z}^k, \boldsymbol{z}^k - \boldsymbol{z}^\star \rangle \geq \beta \|\mathcal{T} \boldsymbol{z}^k\|_2^2 \quad for \quad k = 0, 1, \ldots \tag{9}$$

Then, $\mathcal{T} \boldsymbol{z}^k \to 0$. If, furthermore, $\boldsymbol{z}^k \to \boldsymbol{z}^\infty$, then $\boldsymbol{z}^\infty$ is a zero of $\mathcal{T}$ (i.e., $\mathcal{T} \boldsymbol{z}^\infty = 0$).

Proof outline. Equation (9) makes $\|\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star\|_2^2 \leq \|\boldsymbol{z}^k - \boldsymbol{z}^\star\|_2^2 - \rho(2\beta - \rho)\|\mathcal{T} \boldsymbol{z}^k\|_2^2$. Then, sum for $k = 0, \ldots, \infty$. □

Theorem 1 assumes that Eq. (9) holds, which is refered as β-cocoercivity. The assumption makes sense because we expect $\mathcal{E}\mathcal{D} \simeq \mathcal{I}$. In fact, for linear autoencoders, it is proven by Baldi and Hornik [3] that $\mathcal{E}\mathcal{D} = \mathcal{I}$ as the following Remark 2:

Remark 2 ($ED = \mathcal{I}$ in linear autoencoders). *Let $\boldsymbol{x} \in \mathbb{R}^N$ be a random vector such that $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\mathsf{T}] = \Sigma \in \mathbb{R}^{N \times N}$. Assume Σ has distinct positive eigenvalues. Consider the optimization problem*

$$\min_{D \in \mathbb{R}^{N \times F}, E \in \mathbb{R}^{F \times N}} \mathbb{E}_{\boldsymbol{x}} \|\boldsymbol{x} - DE\boldsymbol{x}\|_2^2.$$

Then, $ED = I$ (identity matrix).

$\mathcal{I}$ is 1-cocoercive, which suggests that our assumption is reasonable. In Section 3.4, we further demonstrate that the assumption is reasonable experimentally and $\boldsymbol{z}^k$ actually converges well.

3.3 Convergence analysis on momentum for acceleration

Momentum is widely used in optimization to keep the optimization process going in the right direction, even when gradients are noisy or the landscape is flat. Among many momentum algorithms, we employ the inertial Krasnoselskii-Mann (KM) iteration [19, 28, 29] and analyze it since the convergence is guaranteed with some assumptions [29]. The inertial KM iteration in our setting can be defined as follows:

$$\boldsymbol{y}^k = \boldsymbol{z}^k + \alpha(\boldsymbol{z}^k - \boldsymbol{z}^{k-1}) \tag{10}$$

$$\boldsymbol{z}^{k+1} = \boldsymbol{y}^k - 2\lambda\beta \mathcal{T} \boldsymbol{y}^k, \tag{11}$$

where $\mathcal{T}(\cdot) = \mathcal{E} \circ \mathcal{D}(\cdot) - \mathcal{E}(\boldsymbol{x})$, $0 < \alpha < 1$, $\beta > 0$, and $\lambda > 0$.

We now present Theorem 2, ensuring that the inertial KM iterations converge. While the formulation and assumptions differ from [29], the theorem and its proof heavily rely on it.

Theorem 2 (Convergence of the inertial KM iterations). *Let $0 < \alpha < 1$, $\beta > 0$, $\lambda > 0$ and $\boldsymbol{x} \in \mathbb{R}^N$. Assume $\mathcal{T}(\cdot) = \mathcal{E} \circ \mathcal{D}(\cdot) - \mathcal{E}(\boldsymbol{x})$ is continuous. Let $(\boldsymbol{z}^k, \boldsymbol{y}^k)$ satisfy (10) and (11). Assume $\boldsymbol{z}^\star$ is a zero of $\mathcal{T}$ (i.e., $\mathcal{T} \boldsymbol{z}^\star = 0$) and the following holds:*

$$\langle \mathcal{T} \boldsymbol{y}^k, \boldsymbol{y}^k - \boldsymbol{z}^\star \rangle \geq \beta \|\mathcal{T} \boldsymbol{y}^k\|_2^2 \quad for \quad k = 0, 1, \ldots \tag{12}$$

If

$$\lambda(1 - \alpha + 2\alpha^2) < (1 - \alpha)^2, \tag{13}$$

then the followings are true:

(A) $\sum_{k \geq 1} \|\boldsymbol{z}^{k+1} - 2\boldsymbol{z}^k + \boldsymbol{z}^{k-1}\|^2$, $\sum_{k \geq 1} \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2$ *and* $\sum_{k \geq 1} \|\mathcal{T} \boldsymbol{y}^k\|^2$ *converge.*

(B) *There is a constant $M > 0$ such that $\min_{1 \leq k \leq n} \|\mathcal{T} \boldsymbol{y}^k\|^2 \leq \frac{M \|\boldsymbol{z}^1 - \boldsymbol{z}^\star\|^2}{n}$.*

(C) $\lim_{k \to \infty} \|\boldsymbol{z}^k - \boldsymbol{z}^\star\|$ *exists.*

(D) $(\boldsymbol{y}^k)$ *and* $(\boldsymbol{z}^k)$ *have the same limit points.*

Here, similar to Theorem 1, the convergence of $(\boldsymbol{z}^k)$ is guaranteed for a learning rate (*i.e.*, $2\lambda\beta$) smaller than some upper bound. However, unlike Theorem 1, this upper bound is determined not only by cocoercivity (*i.e.*, β), but also by the inertial parameter (*i.e.*, α).

3.4 Validation of the assumption

In Theorems 1 and 2, we utilized the assumption that $\mathcal{T}(\cdot) = \mathcal{E} \circ \mathcal{D}(\cdot) - \mathcal{E}(\boldsymbol{x})$ is β-cocoercive for the $(\boldsymbol{y}^k, \boldsymbol{z}^k)$ and $\boldsymbol{z}^\star$. In this section, we experimentally investigate whether this assumption holds for the actual $(\boldsymbol{z}^k)$ and $\boldsymbol{z}^\infty$ obtained in the experiments of Section 4, *i.e.*,

$$\frac{\langle \mathcal{E}\mathcal{D}\boldsymbol{z}^\infty - \mathcal{E}\mathcal{D}\boldsymbol{z}^k, \boldsymbol{z}^\infty - \boldsymbol{z}^k \rangle}{\|\mathcal{E}\mathcal{D}\boldsymbol{z}^\infty - \mathcal{E}\mathcal{D}\boldsymbol{z}^k\|_2^2} \geq \beta > 0. \tag{14}$$

Equations (9) and (12) are equivalent to Eq. (14) if $\boldsymbol{z}^\star = \boldsymbol{z}^\infty$ and $\boldsymbol{z}^k = \boldsymbol{y}^k$. By (C) and (D) of Theorem 2, if Eq. (14) holds while $\boldsymbol{z}^k \to \boldsymbol{z}^\star$, then the convergence was due to Eq. (14).

We conducted decoder inversion experiments in various recent LDMs such as Stable Diffusion 2.1 [37], LaVie [47], and InstaFlow [24]. With or without momentum, the learning rate ρ was fixed at 0.001. For inertial KM iterations, α was set to 0.9. Figure 2 illustrates the $\min_k$ of cocoercivity (14), convergence of the algorithm ($\|\boldsymbol{z}^{100} - \boldsymbol{z}^\infty\|_2$), and final NMSE ($\|\boldsymbol{z}^\infty - \boldsymbol{z}^\star\|_2^2 / \|\boldsymbol{z}^\star\|_2^2$) for 100 instances in each scenario. The observations and the resulting insights are as follows, and they support the validity of our assumptions and theorems.

1. **Our assumption on cocoercivity is reasonable:** most of the instances showed $\frac{\langle \mathcal{E}\mathcal{D}\boldsymbol{z}^\infty - \mathcal{E}\mathcal{D}\boldsymbol{z}^k, \boldsymbol{z}^\infty - \boldsymbol{z}^k \rangle}{\|\mathcal{E}\mathcal{D}\boldsymbol{z}^\infty - \mathcal{E}\mathcal{D}\boldsymbol{z}^k\|_2^2} > 0$ for all k.

2. **The better the convergence, the more cocoercive it is:** the fitted functions (red lines) have negative slopes.

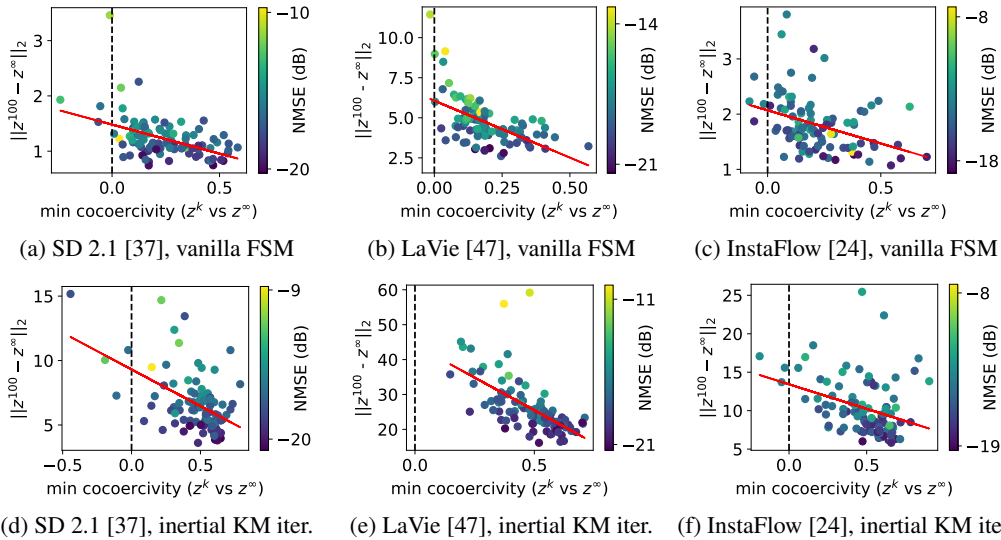3. **Converging instances closely approximate the ground truth:** the points at the bottom are darker.

(a) SD 2.1 [37], vanilla FSM   (b) LaVie [47], vanilla FSM   (c) InstaFlow [24], vanilla FSM

(d) SD 2.1 [37], inertial KM iter.   (e) LaVie [47], inertial KM iter.   (f) InstaFlow [24], inertial KM iter.

Figure 2: The six subfigures represent the relationship between cocoercivity and convergence for 3 models × 2 algorithms (vanilla forward step method and inertial KM iteration). The x-axis represents the values of $\min_{k \in [0,100]} \frac{\langle \mathcal{E}\mathcal{D}\boldsymbol{z}^\infty - \mathcal{E}\mathcal{D}\boldsymbol{z}^k, \boldsymbol{z}^\infty - \boldsymbol{z}^k \rangle}{\|\mathcal{E}\mathcal{D}\boldsymbol{z}^\infty - \mathcal{E}\mathcal{D}\boldsymbol{z}^k\|_2^2}$, which informs whether the optimization path satisfies the assumptions of Theorems 1 and 2, while the y-axis represents the convergence (*i.e.*, $\|\boldsymbol{z}^{100} - \boldsymbol{z}^\infty\|_2$). The red line shows the linear function fitted by least squares. We set $\boldsymbol{z}^\infty = \boldsymbol{z}^{300}$.

4 Experiments with practical optimization techniques

Adam optimizer. In Section 3, we proved that our proposed forward step method (Theorem 1) and inertial KM iterations (Theorem 2) converge. Moreover, we experimentally showed that the assumptions of the theorems hold for most instances and demonstrated their convergence as predicted (Fig. 2). However, it is generally believed that using the Adam optimizer [16] is preferable in many optimization problems. In this section, although we cannot prove convergence as in Theorems 1 and 2, we empirically demonstrate that using the Adam optimizer can achieve a good runtime and memory usage compared to conventional gradient-based methods.

(Optional) Learning rate scheduling. Following common beliefs and a prior work which solves the same problem [14], one can also use a cosine learning rate scheduler with warm-up steps. Similar to [14], the first $^1/_{10}$ of the steps are warm-up steps, followed by the application of cosine annealing. After $^8/_{10}$ of the total steps have passed (as the learning rate has sufficiently decreased), it is kept constant for the rest of steps.

(a) Stable Diffusion 2.1 [37].

(b) LaVie, a video diffusion model [47]
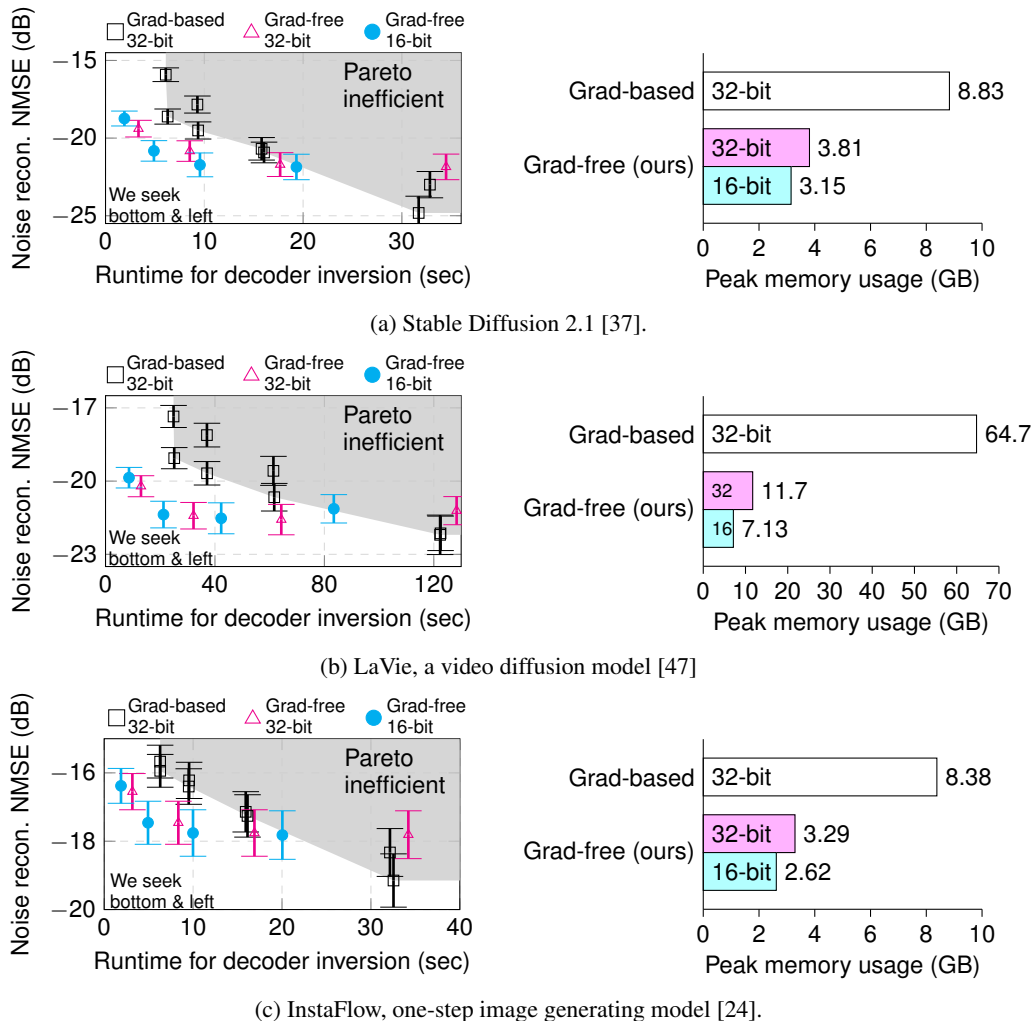

(c) InstaFlow, one-step image generating model [24].

Figure 3: Our gradient-free decoder inversion has a way shorter runtime than the gradient-based decoder inversion, and drastically reduces the GPU memory usage, on (a) SD2.1 [37], (b) LaVie [47], and (c) InstaFlow [24]. Note that 16-bit gradient-based approach is unimplementable, due to the underflow problem. Each point represents a different hyperparameter setting (e.g., the total number of iterations, learning rate, learning rate scheduling); by collecting experimental results from such diverse settings, the Pareto frontier can be obtained fairly without manipulation.

Results. Figure 3 shows the average NMSE and peak memory usage when performing decoder inversion using practical techniques (*i.e.*, Adam and learning rate scheduling) in three different LDMs. While the gradient-based method required much runtime and GPU memory to achieve a certain accuracy, our approach achieves good accuracy in much less runtime and memory usage. One advantage of our proposed method is that it enables all operations to be performed in 16-bit through gradient-free methods, which would typically be infeasible with gradient-based approaches [30]. As a result, for video LDMs where large vectors need to be estimated (Fig. 3b), memory usages can be significantly reduced by almost 9 times.

5 Applications

5.1 Tree-rings watermarking for image generation

In this section, we show an interesting application of our gradient-free decoder inversion. Wen et al. [48] proposed *tree-rings watermarking*, which is an invisible and robust method for protecting the copyright of diffusion-generated images. Watermark is embedded into the Fourier transform of z_T, and detected by inversion (*i.e.*, estimating z_T from x). Hong et al. [14] went beyond watermark detection to attempt watermark *classification*, which makes the problem more difficult; the inversion should be even more accurate. In the following scenarios with two image generation LDMs, we experimentally show that our decoder inversion can efficiently perform watermark classification.

Watermark classification in SD 2.1. Since watermark classification requires higher accuracy than detection, iterative diffusion inversion algorithms was used such as the backward Euler instead of the naïve DDIM inversion. However, it is reported that iterative algorithms become unstable when the classifier-free guidance is large [31, 14]. So even for watermark classification, the naïve DDIM inversion should be used for now. In this case, one promising option for improving the performance of watermark classification is *decoder inversion*.

Watermark classification in InstaFlow. One-step models such as InstaFlow [24] generate z_0 directly from z_T. Therefore, rather than the naïve DDIM inversion, the backward Euler method[14] should be used to obtain z_T from z_0. In this case, z_0 is very sensitive to z_T, so z_T should be estimated accurately by *decoder inversion*.

Results. Table 2 shows the accuracy, peak memory usage and runtime of classifying three tree-rings watermarks using different decoder inversion algorithms. Our decoder inversion method achieves watermark classification performance comparable to a gradient-based method while significantly reducing runtime and peak memory usage. See the appendix for the details.

Table 2: Our gradient-free decoder inversion method achieves the tree-rings watermark [48] classification performance comparable to a gradient-based method while significantly reducing memory usage and runtime, in two different scenarios.

LDM		Encoder	Gradient-based [14]	Gradient-free (ours)
SD2.1 [37]	Accuracy	186/300	207/300	202/300
	Peak memory (GB)	5.71	11.4	6.35
	Runtime (s)	5.66	38.0	22.9
InstaFlow [24]	Accuracy	149/300	227/300	227/300
	Peak memory (GB)	2.93	8.84	3.15
	Runtime (s)	3.55	35.9	13.6

5.2 Background-preserving image editing

Background-preserving editing, to manipulate an image based on a new condition while preserving background from the original image, requires the entire generating latent trajectory. When it is unknown, it must be recovered via inversion [32]. We empirically demonstrate that our algorithm enhances the background-preserving image editing, without the need for the original latents. Figure 4

shows the qualitative results of applying our algorithm to the experiment. To compare accuracy at similar execution times, we adjusted the number of iterations to match the execution time. At comparable execution times, our grad-free method better preserves the background and achieves a lower NMSE.

Method	# iter / Runtime	Edited	Error map ($\times 5$)	Edited	Error map ($\times 5$)
Oracle [32]	-		NMSE = 0.0016 NMSE = 0.0047 NMSE = 0.0060		NMSE = 0.0015 NMSE = 0.0006 NMSE = 0.0017
Grad-based	50 / 15.7s		NMSE = 0.0103 NMSE = 0.0091 NMSE = 0.0101		NMSE = 0.0013 NMSE = 0.0022 NMSE = 0.0029
Grad-free 32bit (Ours)	100 / 17.7s		NMSE = 0.0080 NMSE = 0.0050 NMSE = 0.0081		NMSE = 0.0007 NMSE = 0.0008 NMSE = 0.0017
Grad-free 16bit (Ours)	200 / 19.3s		NMSE = 0.0042 NMSE = 0.0051 NMSE = 0.0069		NMSE = 0.0007 NMSE = 0.0006 NMSE = 0.0014
Grad-based	30 / 9.31s		NMSE = 0.0119 NMSE = 0.0082 NMSE = 0.0126		NMSE = 0.0019 NMSE = 0.0027 NMSE = 0.0036
Grad-free 32bit (Ours)	50 / 8.53s		NMSE = 0.0109 NMSE = 0.0058 NMSE = 0.0085		NMSE = 0.0008 NMSE = 0.0008 NMSE = 0.0014
Grad-free 16bit (Ours)	100 / 9.56s		NMSE = 0.0080 NMSE = 0.0050 NMSE = 0.0082		NMSE = 0.0007 NMSE = 0.0007 NMSE = 0.0017

Figure 4: Our grad-free methods enables the background-preserving image editing, where the original trajectory (z_t) is unknown. The first row (Oracle) shows the result with using the original trajectory. The latter cases estimate the trajectory with inversion methods. Our grad-free method demonstrates better performance compared to the grad-based method at a similar runtime.

6 Discussion

6.1 Does the β-cocoercivity hold also when using Adam? Yes.

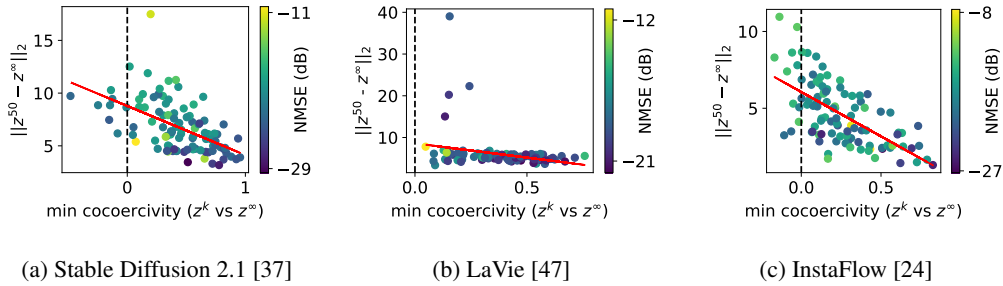(a) Stable Diffusion 2.1 [37]   (b) LaVie [47]   (c) InstaFlow [24]

Figure 5: The assumptions also hold for Adam in many cases, as similar in Fig. 2. The x-axis represents the values of $\min_{k \in [0,50]} \frac{\langle \mathcal{E} \mathcal{D} z^\infty - \mathcal{E} \mathcal{D} z^k, z^\infty - z^k \rangle}{\|\mathcal{E} \mathcal{D} z^\infty - \mathcal{E} \mathcal{D} z^k\|_2^2}$, which informs whether the optimization path satisfies the assumptions of Theorems 1 and 2, while the y-axis represents the convergence (*i.e.*, $\|z^{50} - z^\infty\|_2$). The red line shows the linear function fitted by least squares. Note that z^∞ was approximated by z^{300}.

Although it is challenging to provide a proof regarding convergence for algorithms using Adam and learning rate scheduling (those used in Section 4 and Fig. 3c), we can rather test if (z^k) satisfy cocoercivity (9), as in Fig. 2. Figure 5 depicts the plot of Fig. 2 under the conditions of the experiments

in Section 4, specifically with 16-bit precision and 50 iterations (as Adam converges faster than others). In the cases of Figs. S3b and S3c, cocoercivity is well satisfied, and even in Fig. S3a, many instances meet the cocoercivity requirement. Additionally, as in Fig. 2, the fitted functions have a negative slope. These findings show that our analysis can help explain some aspects of Adam's convergence.

6.2 Why this method has not been proposed in GAN inversion studies

Some may wonder why this gradient-free approach (replacing gradient descent of Eq. (3) with a forward step method of Eq. (4) using the encoder) has not been proposed in numerous GAN inversion studies. Here, we provide Table 3 to explain the reasons. The first reason is that GAN inversion cannot use the encoder from off-the-shelf [49]. It is because networks commonly used in GAN inversion, such as StyleGAN [1], were not born with an encoder even though there are some GAN methods that are inherently an autoencoder such as VQGAN [9] that was born with an encoder. Thus many GAN inversion works needed to train the encoder [33, 8, 34, 53, 6, 36, 7, 2], while we can use the encoder without any finetuning. The second reason is that the memory requirement was small in GAN inversion, as the size of the generated images was small. Even a standard GPU on a PC was capable of running gradient-based algorithms, so there was not much need for lighter algorithms than gradient-based methods. On the contrary, due to the increasing size of images/videos generated by recent LDMs, performing decoder inversion with gradient-based algorithms requires huge memory. Another reason is that many networks for GAN inversion operate in full precision (32 bits), making it easier to employ gradient-based methods here. Meanwhile, many recent LDMs often infer in half-precision (16 bits) to enhance efficiency, making gradient-based methods challenging [30].

Table 3: Why has a gradient-free method not been proposed for GAN inversion?

	GAN inversion	Decoder inversion in LDMs
Encoder usage	Limited	Very easy
Memory requirement	Small	Large (*e.g.*, video generation)
Inference precision	Full (32-bit)	Half (16-bit)

7 Conclusion

In this work, we proposed gradient-free decoder inversion methods for LDMs. The vanilla forward step method and its extension with momentum were shown to guarantee convergence, and the assumptions and theorems were validated for various LDMs. We experimentally showed that a practical algorithm significantly reduced runtime and memory usage compared to existing gradient-based methods.

Limitations Existing gradient-based method is more accurate than our method if sufficient runtime and GPU memory are available. In applications such as image editing, where the accuracy of latent reconstruction may not be critically important such as background preservation, decoder inversion is often unnecessary in most settings (*i.e.*, using the encoder is just OK).

Broader impacts This study could positively impact the protection of copyright for LDM's outputs. One day, even if a new neural network architecture is invented and diffusion is no longer used, our method can still be efficiently utilized to improve the accuracy of decoder inversion as long as transformations between latent and pixel-space are required.

Acknowledgements

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN: How to embed images into the stylegan latent space? In *ICCV*, 2019.

[2] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. HyperStyle: Stylegan inversion with hypernetworks for real image editing. In *CVPR*, 2022.

[3] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[4] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[5] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023.

[6] Lucy Chai, Jonas Wulff, and Phillip Isola. Using latent space regression to analyze and leverage compositionality in gans. *ICLR*, 2021.

[7] Lucy Chai, Jun-Yan Zhu, Eli Shechtman, Phillip Isola, and Richard Zhang. Ensembling with deep generative views. In *CVPR*, 2021.

[8] Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. *IEEE transactions on neural networks and learning systems*, 30(7):1967–1974, 2018.

[9] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[10] Daniel Garibi, Or Patashnik, Andrey Voynov, Hadar Averbuch-Elor, and Daniel Cohen-Or. ReNoise: Real image inversion through iterative noising. *arXiv preprint arXiv:2403.14602*, 2024.

[11] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[13] Seongmin Hong and Se Young Chun. Neural diffeomorphic non-uniform b-spline flows. In *AAAI*, 2023.

[14] Seongmin Hong, Kyeonghyun Lee, Suh Yoon Jeon, Hyewon Bae, and Se Young Chun. On exact inversion of dpm-solvers. *arXiv preprint arXiv:2311.18387*, 2023.

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[17] Jonas Köhler, Andreas Krämer, and Frank Noé. Smooth normalizing flows. *NeurIPS*, 2021.

[18] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *ICLR*, 2021.

[19] Mark Aleksandrovich Krasnosel'skii. Two remarks on the method of successive approximations. *Uspekhi matematicheskikh nauk*, 10(1):123–127, 1955.

[20] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[21] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. In *ICML*, 2023.

[22] Haohe Liu, Qiao Tian, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D Plumbley. AudioLDM 2: Learning holistic audio generation with self-supervised pretraining. *arXiv preprint arXiv:2308.05734*, 2023.

[23] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[24] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. InstaFlow: One step is enough for high-quality diffusion-based text-to-image generation. In *ICLR*, 2023.

[25] 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. *NeurIPS*, 2022.

[26] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv:2211.01095*, 2022.

[27] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023.

[28] W Robert Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4 (3):506–510, 1953.

[29] Juan José Maulén, Ignacio Fierro, and Juan Peypouquet. Inertial Krasnoselskii-Mann Iterations. *Set-Valued and Variational Analysis*, 32(2):10, 2024.

[30] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[31] Zhihong Pan, Riccardo Gherardi, Xiufeng Xie, and Stephen Huang. Effective real image editing with accelerated iterative diffusion inversion. In *ICCV*, 2023.

[32] Or Patashnik, Daniel Garibi, Idan Azuri, Hadar Averbuch-Elor, and Daniel Cohen-Or. Localizing object-level shape variations with text-to-image diffusion models. In *ICCV*, 2023.

[33] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional GANs for image editing. *arXiv preprint arXiv:1611.06355*, 2016.

[34] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *CVPR*, 2020.

[35] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv:2204.06125*, 2022.

[36] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *CVPR*, 2021.

[37] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[38] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dream-Booth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023.

[39] Ernest K Ryu and Wotao Yin. *Large-scale convex optimization: algorithms & analyses via monotone operators*. Cambridge University Press, 2022.

[40] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

[41] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[42] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *NeurIPS*, 2020.

[43] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[44] Bram Wallace, Akash Gokul, and Nikhil Naik. EDICT: Exact Diffusion Inversion via Coupled Transformations. In *CVPR*, pages 22532–22541, 2023.

[45] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023.

[46] Xiang Wang, Shiwei Zhang, Han Zhang, Yu Liu, Yingya Zhang, Changxin Gao, and Nong Sang. VideoLCM: Video latent consistency model. *arXiv preprint arXiv:2312.09109*, 2023.

[47] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, et al. LaVie: High-quality video generation with cascaded latent diffusion models. *arXiv preprint arXiv:2309.15103*, 2023.

[48] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-Ring Watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv:2305.20030*, 2023.

[49] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. GAN Inversion: A survey. *IEEE TPAMI*, 45(3):3121–3138, 2022.

[50] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. In *ICLR*, 2021.

[51] Guoqiang Zhang, Jonathan P Lewis, and W Bastiaan Kleijn. Exact diffusion inversion via bi-directional integration approximation. *arXiv:2307.10829*, 2023.

[52] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022.

[53] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain GAN inversion for real image editing. In *ECCV*, 2020.

[54] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016.

A Appendix

A.1 Proofs

Throughout the paper and appendix, both the inner product and the norm are l_2.

A.1.1 Proof of Theorem 1

Proof of Theorem 1.

$$
\begin{aligned}
\|z^{k+1} - z^\star\|_2^2 &= \|z^k - z^\star - \rho\mathcal{T}z^k\|_2^2 \\
&= \|z^k - z^\star\|_2^2 - 2\rho\langle\mathcal{T}z^k, z^k - z^\star\rangle + \rho^2\|\mathcal{T}z^k\|_2^2 \\
&\leq \|z^k - z^\star\|_2^2 - \rho(2\beta - \rho)\|\mathcal{T}z^k\|_2^2
\end{aligned}
\tag{S1}
$$

Sum both sides from $k = 0, \ldots, \infty$

$$
\rho(2\beta - \rho)\sum_{k=0}^{\infty}\|\mathcal{T}z^k\|_2^2 \leq \|z^0 - z^\star\|_2^2 < \infty \Rightarrow \|\mathcal{T}z^k\|_2^2 \to 0 \Rightarrow \mathcal{T}z^k \to 0.
\tag{S2}
$$

Finally, if $z^k \to z^\infty$, then $\mathcal{T}z^\infty = 0$ by continuity. $\qquad\square$

A.1.2 Lemma 1 for Theorem 2

For simplicity, we set

$$
\nu = (\lambda^{-1} - 1),
\tag{S3}
$$

$$
\delta^k = \nu(1 - \alpha)\|z^k - z^{k-1}\|^2,
\tag{S4}
$$

$$
\Delta^k(z^\star) = \|z^k - z^\star\|^2 - \|z^{k-1} - z^\star\|^2, \quad \Delta^1(z^\star) = 0
\tag{S5}
$$

$$
C^k(z^\star) = \|z^k - z^\star\|^2 - \alpha\|z^{k-1} - z^\star\|^2 + \delta^k, \quad C^1(z^\star) = \|z^1 - z^\star\|^2.
\tag{S6}
$$

Before we prove, we need the following lemma.

Lemma S1. *Let $\beta > 0$, $0 < \rho < 2\beta$, and $x \in \mathbb{R}^N$. Assume $\mathcal{T}(\cdot) = \mathcal{E} \circ \mathcal{D}(\cdot) - \mathcal{E}(x)$ is continuous. Assume $\mathcal{T}(\cdot) = \mathcal{E} \circ \mathcal{D}(\cdot) - \mathcal{E}(x)$ is continuous. Let (z^k, y^k) satisfy (10) and (11). Assume $z^\star$ is a zero of $\mathcal{T}$ (i.e., $\mathcal{T}z^\star = 0$) and Eq. (12) holds. Then, for $k = 1, 2, \ldots,$*

$$
\Delta^{k+1}(z^\star) + \delta^{k+1} + \nu\alpha\|z^{k+1} - 2z^k + z^{k-1}\|^2 \leq \alpha\Delta^k(z^\star) + [\alpha(1+\alpha) + \nu\alpha(1-\alpha)]\|z^k - z^{k-1}\|^2.
\tag{S7}
$$

Proof of Lemma S1. Let $\rho = 2\lambda\beta$. From Eq. (11),

$$
\|z^{k+1} - z^\star\|^2 = \|y^k - z^\star\|^2 + \rho^2\|\mathcal{T}y^k\|^2 - 2\rho\langle y^k - z^\star, \mathcal{T}y^k\rangle.
\tag{S8}
$$

Applying Eq. (12) on Eq. (S8), we have

$$
\|z^{k+1} - z^\star\|^2 \leq \|y^k - z^\star\|^2 - \rho(2\beta - \rho)\|\mathcal{T}y^k\|^2.
\tag{S9}
$$

From Eq. (10),

$$
\begin{aligned}
\|y^k - z^\star\|^2 &= \|(1+\alpha)(z^k - z^\star) - \alpha(z^{k-1} - z^\star)\|^2 \\
&= (1+\alpha)\|z^k - z^\star\|^2 + \alpha(1+\alpha)\|z^k - z^{k-1}\|^2 - \alpha\|z^{k-1} - z^\star\|^2.
\end{aligned}
\tag{S10}
$$

Combining Eq. (S9) and Eq. (S10),

$$
\|z^{k+1} - z^\star\| \leq (1+\alpha)\|z^k - z^\star\|^2 + \alpha(1+\alpha)\|z^k - z^{k-1}\|^2 - \alpha\|z^{k-1} - z^\star\|^2 - \rho(2\beta - \rho)\|\mathcal{T}y^k\|^2.
\tag{S11}
$$

Using Eq. (S4), we can simplify Eq. (S11) as follows:

$$
\Delta^{k+1}(z^\star) \leq \alpha\Delta^k(z^\star) + \alpha(1+\alpha)\|z^k - z^{k-1}\|^2 - \rho(2\beta - \rho)\|\mathcal{T}y^k\|^2.
\tag{S12}
$$

14

On the other hand, by Eq. (10) and Eq. (11), we have

$$
\begin{aligned}
\rho^2 \|\mathcal{T}\boldsymbol{y}^k\|^2 &= \|\boldsymbol{z}^{k+1} - \boldsymbol{y}^k\|^2 \\
&= \|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k - \alpha(\boldsymbol{z}^k - \boldsymbol{z}^{k-1})\|^2 \\
&= \|(1-\alpha)(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k) + \alpha(\boldsymbol{z}^{k+1} - 2\boldsymbol{z}^k + \boldsymbol{z}^{k-1})\|^2 \\
&= (1-\alpha)\|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2 - \alpha(1-\alpha)\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2 + \alpha\|\boldsymbol{z}^{k+1} - 2\boldsymbol{z}^k + \boldsymbol{z}^{k-1}\|^2.
\end{aligned}
\tag{S13}
$$

Let's multiply $\nu = (2\beta - \rho)/\rho$ to Eq. (S13). Then

$$
\rho(2\beta - \rho)\|\mathcal{T}\boldsymbol{y}^k\|^2 = \delta^{k+1} - \nu\alpha(1-\alpha)\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2 + \nu\alpha\|\boldsymbol{z}^{k+1} - 2\boldsymbol{z}^k + \boldsymbol{z}^{k-1}\|^2 \tag{S14}
$$

Finally, subtracting Eq. (S14) from Eq. (S12), we obtain Eq. (S7). $\qquad\square$

A.1.3 Proof of Theorem 2

Proof of Theorem 2. We firstly show that $\left(C^k(\boldsymbol{z}^\star)\right)_k$ is nonincreasing and nonnegative, thus $\lim\limits_{k\to\infty} C^k(\boldsymbol{z}^\star)$ exists. By Eq. (13), there exists $\epsilon > 0$ such that

$$
\alpha(1+\alpha) + \nu\alpha(1-\alpha) \leq \nu(1-\alpha) - \epsilon. \tag{S15}
$$

Combining Eq. (S7) and Eq. (S15), we have

$$
\begin{aligned}
\Delta^{k+1}(\boldsymbol{z}^\star) &+ \delta^{k+1} + \nu\alpha\|\boldsymbol{z}^{k+1} - 2\boldsymbol{z}^k + \boldsymbol{z}^{k-1}\|^2 \\
&\leq \alpha\Delta^k(\boldsymbol{z}^\star) + [\alpha(1+\alpha) + \nu\alpha(1-\alpha)]\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2 \\
&\leq \alpha\Delta^k(\boldsymbol{z}^\star) + [\nu(1-\alpha) - \epsilon]\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2 \\
&= \alpha\Delta^k(\boldsymbol{z}^\star) + \delta^k - \epsilon\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2.
\end{aligned}
\tag{S16}
$$

From Eq. (S6) and Eq. (S5),

$$
\begin{aligned}
C^{k+1}(\boldsymbol{z}^\star) - C^k(\boldsymbol{z}^\star) &= \Delta^{k+1}(\boldsymbol{z}^\star) - \alpha(\|\boldsymbol{z}^k - \boldsymbol{z}^\star\|^2 - \|\boldsymbol{z}^{k-1} - \boldsymbol{z}^\star\|^2) + \delta^{k+1} - \delta^k \\
&= \Delta^{k+1}(\boldsymbol{z}^\star) + \delta^{k+1} - \alpha\Delta^k(\boldsymbol{z}^\star) - \delta^k.
\end{aligned}
\tag{S17}
$$

By Eq. (S16) and Eq. (S17), we have

$$
C^{k+1}(\boldsymbol{z}^\star) + \nu\alpha\|\boldsymbol{z}^{k+1} - 2\boldsymbol{z}^k + \boldsymbol{z}^{k-1}\|^2 + \epsilon\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2 \leq C^k(\boldsymbol{z}^\star), \tag{S18}
$$

which shows $\left(C^k(\boldsymbol{z}^\star)\right)_k$ is nonincreasing. To prove it is nonnegative, suppose $C^{k_1}(\boldsymbol{z}^\star) < 0$ for some $k_1 \geq 1$. Since it is nonincreasing, from Eq. (S6),

$$
\|\boldsymbol{z}^k - \boldsymbol{z}^\star\|^2 - \alpha\|\boldsymbol{z}^{k-1} - \boldsymbol{z}^\star\|^2 \leq C^k(\boldsymbol{z}^\star) \leq C^{k_1}(\boldsymbol{z}^\star) < 0 \tag{S19}
$$

for all $k \geq k_1$. That is, $\|\boldsymbol{z}^k - \boldsymbol{z}^\star\|^2 \leq \|\boldsymbol{z}^{k-1} - \boldsymbol{z}^\star\|^2 + C^{k_1}(\boldsymbol{z}^\star)$, hence

$$
0 \leq \|\boldsymbol{z}^k - \boldsymbol{z}^\star\|^2 \leq \|\boldsymbol{z}^{k-1} - \boldsymbol{z}^\star\|^2 + C^{k_1}(\boldsymbol{z}^\star) \leq \cdots \leq \|\boldsymbol{z}^{k_1} - \boldsymbol{z}^\star\|^2 + (k - k_1)C^{k_1}(\boldsymbol{z}^\star) \tag{S20}
$$

for all $k \geq k_1$, and this is a contradiction as the right hand of Eq. (S20) diverges to $-\infty$ as $k \to \infty$. Therefore, $\left(C^k(\boldsymbol{z}^\star)\right)_k$ is nonincreasing and nonnegative, thus $\lim\limits_{k\to\infty} C^k(\boldsymbol{z}^\star)$ exists.

Now we begin showing (A). By the findings so far and Eq. (S18) and since $\epsilon > 0$, $\sum_{k\geq 1} \|\boldsymbol{z}^{k+1} - 2\boldsymbol{z}^k + \boldsymbol{z}^{k-1}\|^2$ and $\sum_{k\geq 1} \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2$ converge. Since $\sum_{k\geq 1} \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2$ converges, $\sum_{k\geq 1} \delta^k = \sum_{k\geq 1} \nu(1-\alpha)\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2$ also converges. Using Eq. (S13) and an identity $\|a - b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, we have

$$
\rho^2\|\mathcal{T}\boldsymbol{y}^k\|^2 = (1+\alpha)\|\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\|^2 + \alpha(1+\alpha)\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2. \tag{S21}
$$

Summing Eq. (S18) over $k \geq 1$, we have

$$
\epsilon\sum_{k\geq 1} \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2 \leq C^1(\boldsymbol{z}^\star) = \|\boldsymbol{z}^1 - \boldsymbol{z}^\star\|^2. \tag{S22}
$$

15

Using the summation of Eq. (S21) and Eq. (S22), we have

$$n \min_{1 \leq k \leq n} \|\mathcal{T}\boldsymbol{y}^k\|^2 \leq \sum_{k \geq 1} \|\mathcal{T}\boldsymbol{y}^k\|^2 \leq \frac{(1+\alpha)^2}{\epsilon\rho^2} \|\boldsymbol{z}^1 - \boldsymbol{z}^\star\|^2, \tag{S23}$$

which shows that $\sum_{k \geq 1} \|\mathcal{T}\boldsymbol{y}^k\|^2$ converges, and proves (B).

Now we prove (C). From Eq. (S16), we have

$$\Delta^{k+1}(\boldsymbol{z}^\star) \leq \alpha\Delta^k(\boldsymbol{z}^\star) + \delta^k. \tag{S24}$$

Let $[\cdot]_+$ denotes the positive part. From Eq. (S24),

$$(1-\alpha)[\Delta^{k+1}(\boldsymbol{z}^\star)]_+ + \alpha[\Delta^{k+1}(\boldsymbol{z}^\star)]_+ \leq \alpha[\Delta^k(\boldsymbol{z}^\star)]_+ + \delta^k. \tag{S25}$$

Applying summation for $k \geq 1$, we have

$$(1-\alpha)\sum_{k \geq 1}[\Delta^{k+1}(\boldsymbol{z}^\star)]_+ \leq \alpha[\Delta^1(\boldsymbol{z}^\star)]_+ + \sum_{k \geq 1}\delta^k = \sum_{k \geq 1}\delta^k < \infty. \tag{S26}$$

Let $h^k = \|\boldsymbol{z}^k - \boldsymbol{z}^\star\|^2 - \sum_{j=1}^k[\Delta^j(\boldsymbol{z}^\star)]_+$. Then $h^{k+1} - h^k = \Delta^{k+1}(\boldsymbol{z}^\star) - [\Delta^{k+1}(\boldsymbol{z}^\star)]_+ \leq 0$ so (h^k) is nonincreasing. Since $\sum_{k \geq 1}[\Delta^{k+1}(\boldsymbol{z}^\star)]_+$ is finite, $\lim_{k \to \infty} \|\boldsymbol{z}^k - \boldsymbol{z}^\star\| = \lim_{k \to \infty} h^k$ exists.

Finally, we now prove (D). By (A), $\lim_{k \to \infty} \|\mathcal{T}\boldsymbol{y}^k\| = \lim_{k \to \infty} \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\| = 0$. By Eq. (10) and (11), $(\boldsymbol{y}^k)$ and $(\boldsymbol{z}^k)$ have the same limit points. □

A.2 Experiment details and more results

A.2.1 Decoder inversion in LDMs

Hong et al. [14] successfully employed the gradient-based method in their experiments and completed hyperparameter tuning. Their tailored method may work well in their setting, so we used the code from Hong et al. [14]'s official repository[2] to perform decoder inversion on 100 images generated under the same settings (prompt, classifier-free guidance, random seed). This makes our comparison fair. Table S1 shows the settings, NMSE, number of iterations, runtime, and memory usage of the experiments. Although all the values in Table S1 are actually displayed in Fig. 3, Table S1 provides additional information, including on which specific settings correspond to the points in Fig. 3.

Gradient-based method For the gradient-based method (for comparision), we used Adam [16] with l_2-loss. When optionally using a cosine learning rate scheduler, 1/10 of the total steps were warm-up steps (*i.e.*, the learning rate increased linearly). Note that this setting is the same to [14]. In [14], the learning rate was 0.1 with 100 iterations, but it showed long runtimes. Thus, we set the iterations to 20, 30, 50, and 100. We could observe that reducing the number of iterations decreases the effectiveness of learning rate scheduling. Therefore, we additionally conducted experiments with a fixed learning rate of 0.01, and found that it performed better with a smaller number of iterations. As shown in Tables S1a and S1c, when comparing the '0.1 scheduled' and '0.01 fixed' rows under 'Grad-based 32-bit', the fixed learning rate outperforms at 20 and 30 iterations, whereas the scheduled learning rate outperforms at 50 and 100 iterations. By comparing existing methods under various hyperparameter settings, we can more reliably establish the Pareto frontier of these methods, ensuring the fairness of our comparisons.

Other details For Stable Diffusion 2.1 and InstaFlow, we used prompts from `https://huggingface.co/datasets/Gustavosta/Stable-Diffusion-Prompts`. For LaVie, prompts were generated by ChatGPT, given the examples used in [47]. The classifier-free guidance was 3.0 in Stable Diffusion 2.1, 7.5 in LaVie, and 1.0 in InstaFlow. For LaVie, we found that 200 iterations for the gradient-free method is too long, so we adjusted the learning rate scheduling as if there were only 100 iterations (fixing the learning rate after 100 iterations).

---

[2]https://github.com/smhongok/inv-DM

Table S1: We can further improve the reconstruction quality, by using a cosine learning rate scheduler with warm-up steps (if the number of iterations or the runtime is predetermined). We conducted experiments of the decoder inversion in various LDMs, with all other conditions being the same to that in Fig. 3, but using a relatively scheduled learning rate at each iteration. $\pm$ represents the 95% confidence interval.

(a) Stable Diffusion 2.1

Method	Bits	LR	NMSE (dB) // # iter. / Runtime				Memory
Grad-based	32	0.1	-15.92 ± 0.44	-17.84 ± 0.55	-20.69 ± 0.73	-24.82 ± 1.08	8.83 GB
		scheduled	20 / 6.11 s	30 / 9.31 s	50 / 15.7 s	100[a] / 31.7 s	
Grad-based	32	0.01	-18.61 ± 0.49	-19.51 ± 0.55	-20.93 ± 0.67	-23.00 ± 0.85	8.83 GB
		fixed	20 / 6.29 s	30 / 9.37 s	50 / 16.0 s	100 / 32.8 s	
Grad-free (ours)	32	0.01	-19.39 ± 0.54	-20.84 ± 0.66	-21.71 ± 0.77	-21.85 ± 0.82	3.81 GB
		scheduled	20 / 3.34 s	50 / 8.53 s	100 / 17.7 s	200 / 34.5 s	
Grad-free (ours)	16	0.01	-18.74 ± 0.48	-20.82 ± 0.66	-21.72 ± 0.77	-21.86 ± 0.82	3.15 GB
		scheduled	20 / 1.91 s	50 / 4.90 s	100 / 9.56 s	200 / 19.3 s	

[a]This setting is the same as the setting in [14].

(b) LaVie

Method	Bits	LR	NMSE (dB) // # iter. / Runtime				Memory
Grad-based	32	0.1	-17.35 ± 0.45	-18.12 ± 0.49	-19.58 ± 0.59	-22.20 ± 0.81	64.7 GB
		scheduled	20 / 24.86 s	30 / 37.07 s	50 / 61.37 s	100 / 122.31 s	
Grad-based	32	0.01	-19.06 ± 0.43	-19.68 ± 0.48	-20.66 ± 0.57	-22.14 ± 0.71	64.7 GB
		fixed	20 / 25.11 s	30 / 37.11 s	50 / 61.67 s	100 / 122.46 s	
Grad-free (ours)	32	0.01	-20.21 ± 0.43	-21.42 ± 0.54	-21.58 ± 0.62	-21.21 ± 0.58	11.7 GB
		scheduled	20 / 12.96 s	50 / 32.17 s	100 / 64.25 s	200 / 128.34 s	
Grad-free (ours)	16	0.01	-19.86 ± 0.42	-21.37 ± 0.55	-21.52 ± 0.63	-21.14 ± 0.58	7.13 GB
		scheduled	20 / 8.59 s	50 / 21.19 s	100 / 42.28 s	200 / 83.45 s	

(c) InstaFlow

Method	bits	LR	NMSE (dB) // # iter. / Runtime				Memory
Grad-based	32	0.1	-15.67 ± 0.48	-16.22 ± 0.53	-17.26 ± 0.62	-19.15 ± 0.78	8.38 GB
		scheduled	20 / 6.29 s	30 / 9.55 s	50 / 16.12 s	100 / 32.54 s	
Grad-based	32	0.01	-15.94 ± 0.48	-16.40 ± 0.52	-17.14 ± 0.59	-18.33 ± 0.70	8.38 GB
		fixed	20 / 6.30 s	30 / 9.51 s	50 / 15.92 s	100 / 32.14 s	
Grad-free (ours)	32	0.01	-16.55 ± 0.53	-17.46 ± 0.63	-17.76 ± 0.68	-17.81 ± 0.70	3.2 GB
		scheduled	20 / 3.17 s	50 / 8.34 s	100 / 16.90 s	200 / 34.21 s	
Grad-free (ours)	16	0.01	-16.38 ± 0.51	-17.46 ± 0.63	-17.76 ± 0.68	-17.82 ± 0.71	2.62 GB
		scheduled	20 / 1.89 s	50 / 4.93 s	100 / 9.99 s	200 / 20.04 s	

A.2.2 Watermark detection

For both scenarios, the number of steps was 100 and the learning rate was scheduled for both gradient-based and gradient-free methods. The maximum learning rate was 0.1 for grad-based methods and

0.01 for Grad-free methods, which is consistent with the experiments in Section 4. All experiments were conducted in 32-bit.

For Stable Diffusion 2.1, through decoder inversion, we obtain z_0 from x, and perform naïve DDIM inversion for 50 steps to obtain z_T from z_0. Except for employing the naïve DDIM inversion due to the use of a large classifier-free guidance of 7.5 and resulting stability issues, the experimental settings are the same as those of [14]. For InstaFlow, the backward Euler (Algorithm 1 of [14]) with 100 iterations is applied for seeking z_T from z_0.

Figure S1 shows the confusion matrices for the experiments of Table 2 in the main paper. In Figure S2, we also present the qualitative results of applying our algorithm to the watermark classification experiment. Our grad-free method enables precise reconstruction, while reducing the runtime compared to the grad-based method.

(a) Stable Diffusion 2.1. Accuracy: 186/300, 207/300, 202/300

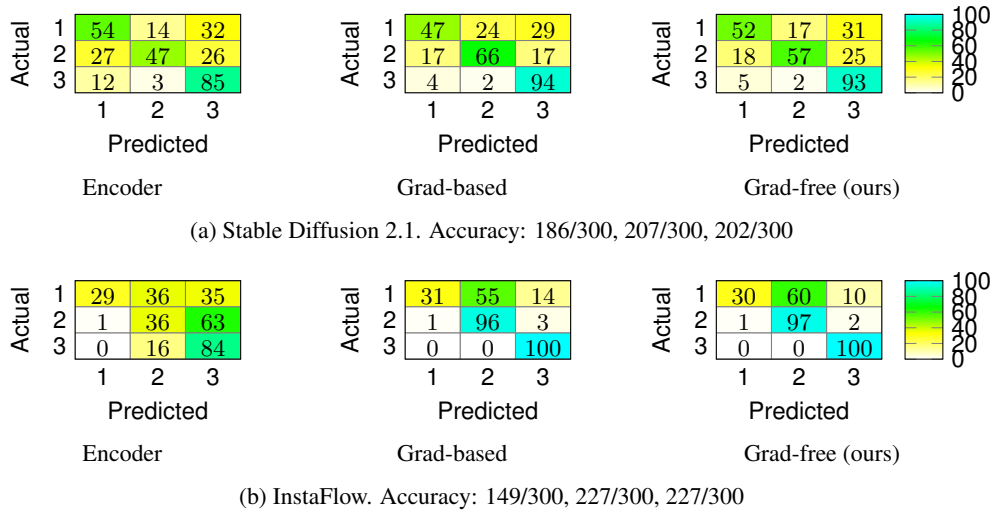(b) InstaFlow. Accuracy: 149/300, 227/300, 227/300

Figure S1: Confusion matrices for watermark classification on LDMs. Ours is better than the encoder and works as well as the grad-based method [14].
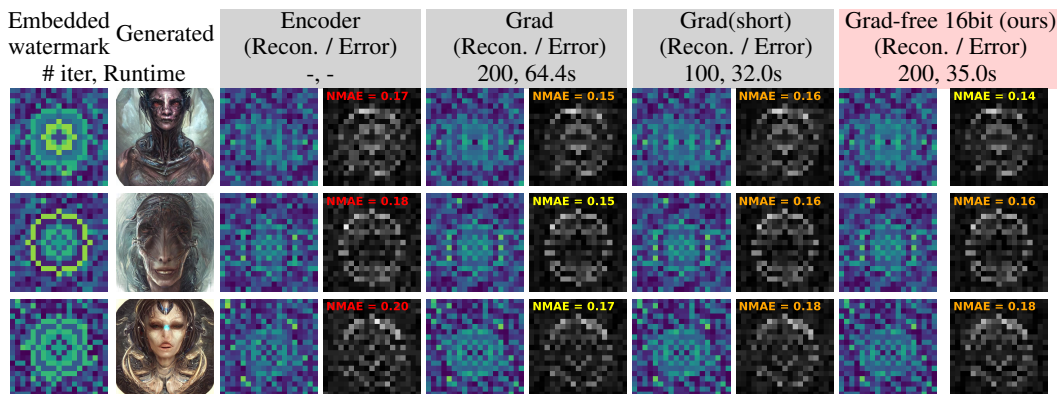

Figure S2: Qualitative result on the watermarking classification experiment [14]. Our grad-free decoder inversion aids in the accurate reconstruction of Tree-ring watermarking [48]. Specifically, it either reduces the processing time compared to the grad-based (column 3 vs 5) or achieves better accuracy within the same runtime (column 4 vs 5).

Table S2: Ablation studies on optimizer and learning rate scheduling. Adam showed the best result compared to vanilla and KM iterations method. The learning scheduling we used showed consistently good performance across all intervals.

(a) Ablation study on optimizer

Method \ # iter.	NMSE (dB)			
	20	30	50	100
Vanilla	-16.87 ± 0.38	-17.42 ± 0.41	-18.21 ± 0.46	-19.35 ± 0.54
KM iterations	-18.99 ± 0.53	-20.72 ± 0.73	-21.46 ± 0.96	-20.91 ± 1.20
Adam (orig.)	**-19.39** ± 0.54	**-20.84** ± 0.66	**-21.71** ± 0.77	**-21.85** ± 0.82

(b) Ablation study on learning rate scheduling

Method	NMSE (dB)			
	20	50	100	200
lr=0.01 (fixed)	**-20.05** ± 0.58	**-21.07** ± 0.70	-21.22 ± 0.74	-20.61 ± 0.79
lr=0.002 (fixed)	-17.85 ± 0.43	-19.28 ± 0.53	-20.59 ± 0.64	-21.57 ± 0.74
lr scheduled (orig.)	-19.39 ± 0.54	-20.84 ± 0.66	**-21.71** ± 0.77	**-21.85** ± 0.82

A.2.3 Ablation Studies

We conducted an ablation study on each optimizer and learning rate scheduling, in SD2.1(32bit). Table S2a shows the result of only changing the optimizer while keeping all other conditions the same. In Table S2b, we observed what happens when using a fixed learning rate instead of applying learning rate scheduling with Adam. When using a fixed learning rate, we found that with a large learning rate (lr=0.01), the performance was poor when the number of iterations was high, and with a small learning rate (lr=0.002), the performance was poor when the number of iterations was low. In contrast, the scheduled learning rate we used showed consistent performance across all intervals regardless of the number of iterations.

A.2.4 Analysis on the grad-based method

In Figure S3, we present the instance-wise cocoercivity, convergence, and accuracy for the gradient-based method, similar to Figure 2 and Figure 5. Similar to the results observed with the gradient-free method, the gradient-based method showed that most instances satisfied cocoercivity, and better convergence often led to higher accuracy. However, it was observed that cocoercivity and convergence are not significantly correlated. In other words, the correlation between cocoercivity and convergence is not a general characteristic, but a unique feature we discovered in our gradient-free method.

A.3 Supplementary explanation

Figure 1b Normalizing flows (NFs) are born with its analytic invertibility, hence located at the bottom left. VAEs and learning-based GAN inversion methods have corresponding encoders, so they have short computation time but come with some inversion error. Some special structured DMs [44] are also located here.

A.4 Computational resources

For running Stable Diffusion 2.1, one NVIDIA GeForce RTX 3090 Ti was used. The RAM size of the GPU was 24 GB. Note that most of the computation was conducted on GPU. For CPU, one 11th Gen Intel(R) Core(TM) i9-11900KF @ 3.50GHz was used. For running LaVie, one NVIDIA A100 SXM4 80GB and AMD EPYC 7742 64-Core Processor were used. For InstaFlow, one NVIDIA GeForce RTX 3090 GPU with one 12th Gen Intel(R) Core(TM) 17-12700K @ 3.60GHz was used.
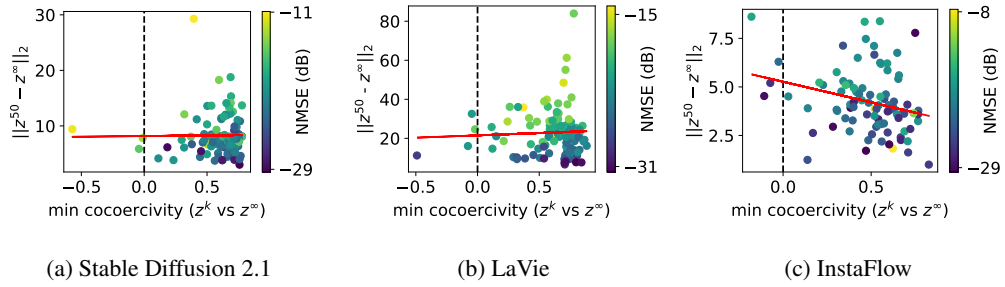
(a) Stable Diffusion 2.1   (b) LaVie   (c) InstaFlow

Figure S3: Analysis on the grad-based method. x-axis: $\min\limits_{k \in [0,50]} \frac{\langle \mathcal{E}\mathcal{D}z^\infty - \mathcal{E}\mathcal{D}z^k, z^\infty - z^k \rangle}{\|\mathcal{E}\mathcal{D}z^\infty - \mathcal{E}\mathcal{D}z^k\|_2^2}$, y-axis: conver-

gence (*i.e.*, $\|z^{50} - z^\infty\|_2$). The red line shows the linear function fitted by least squares. Note that z^∞ was approximated by z^{300}. Better convergence and accuracy were observed, but no relationship with cocoercivity was found.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: This paper discusses the limitations in Section 7.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

Justification: This paper provides Theorems 1,2 and their corresponding proofs.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

Answer: [Yes]

Justification: This paper provides sufficient information that can reproduce the results of the main experiment in Section 4.

Guidelines:

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

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

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

Answer: [Yes]

Justification: This paper provides code to reproduce the experimental results in Sections 4 and 5.

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: This paper provides the details of the experiments sufficiently.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: This paper reports error bars (95% confidence interval) for the main experiment (Figure 3).

   Guidelines:

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

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

8. **Experiments Compute Resources**

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

Answer: [Yes]

Justification: Peak memory usages and time of executions are main concerns of this paper. This paper also provide information on computer resources in Section A.4.

Guidelines:

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

9. **Code Of Ethics**

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

Answer: [Yes]

Justification: Yes, this paper does.

Guidelines:

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

10. **Broader Impacts**

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

Answer: [Yes]

Justification: This paper discusses societal impacts in Section 7.

Guidelines:

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

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

11. **Safeguards**

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

Answer: [NA]

Justification: This paper poses no such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: This paper explicitly mentioned and properly respected the assets used in this paper.

Guidelines:

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

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

13. **New Assets**

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

   Answer: [Yes]

   Justification: The code of the supplementary material has its document README.md.

   Guidelines:

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

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

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

   Answer: [NA]

   Justification: This paper does not involve such crowdsourcing nor research with human subjects.

   Guidelines:

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

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

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

   Answer: [NA]

   Justification: This paper does not involve such crowdsourcing nor research with human subjects.

   Guidelines:

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