Controlling Diffusion Models Through Blockwise Gradient Guidance and Sampling

Maurya Goyal

TU Delft

cover image generated using ChatGPT

Controlling Diffusion Models Through Blockwise Gradient Guidance and Sampling

by

Maurya Goyal

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Thursday June 26, 2025 at 10:00 AM.

Student number:	6003435	
Project duration:	November 14, 2024 – June 26, 2025	
Thesis committee:	Dr. ir. H. Jamali-Rad,	TU Delft, Daily supervisor
	Dr. ir. E. Isufi,	TU Delft, Advisor
	Dr. ir. P. Kellnhofer ,	TU Delft, External committee member
	Ir. A. Singh,	Shell, External daily co-supervisor

TU Delft

Preface

This thesis report presents a culmination of my work towards obtaining the degree of Master of Science in Computer Science, in the Data Science track at the Delft University of Technology. The research was conducted within the Computer Vision Lab at TU Delft under the daily supervision of Dr. Hadi Jamali-Rad and Ir. Anuj Singh.

First of all, I would like to express my gratitude to Dr. Hadi Jamali-Rad and Ir. Anuj Singh for their constant support and guidance throughout this journey. Working alongside them and observing their meticulous approach and dedication has profoundly influenced me, setting a standard I aspire to achieve in my future endeavors.

I would also like to thank Dr. Elvin Isufi, my committee chair for his interest in the topic and his flexibility and support throughout the thesis. Additionally, I am also grateful to Dr. Petr Kellnhofer for his time, interest, and willingness to participate in my committee. Your valuable feedback and expertise are greatly appreciated.

I would also like to thank the Computer Vision Lab for giving me access to the DAIC cluster without which this GPU extensive research wouldn't have been possible.

Finally and most importantly, I would like to express my gratitude towards my friends and family who helped me believe in myself and deterred me from losing my way. Without their love and unconditional support, none of this would have been possible.

Maurya Goyal
Delft, June 2025

Contents

1

Introduction

Generative Artificial Intelligence (GenAI) has emerged as one of the most transformative technological advancements of the past decade. It is revolutionizing the way we work, create, and engage with information. These advanced models are capable of learning intricate data patterns and generating new, synthetic data that are identical to real-world examples. Among the most effective models in this field are **diffusion models** [52, 18, 54, 56], which have revolutionized the fields of image synthesis [36, 12, 45], video generation [5, 37], audio design [26, 22], and even molecular biology [62, 23]. These models work by gradually corrupting data with noise in a forward process and then learning to reverse this noise during sampling to generate realistic outputs. Thanks to large-scale datasets [8], modern hardware accelerators, and advances in neural architectures, diffusion models have set new benchmarks in high-fidelity data generation.

As the scale and scope of these models expand, they increasingly exhibit properties of **foundational models**, that are general-purpose systems, reusable across domains, and capable of adapting to a variety of downstream tasks. This scalability, however, comes with a growing need for **controllability**, the ability to steer a model's output following user intent. Whether the task involves generating photorealistic images, retrieving information, designing proteins, or composing music, users frequently seek outputs that satisfy specific criteria. These criteria may include aesthetic appeal, factual accuracy, alignment with textual prompts, or in scientific applications such as drug discovery, the generation of molecules or proteins that exhibit desired structural and functional properties.

To meet this need, a wide array of alignment methods have been proposed. One prominent approach is fine-tuning [28, 40, 7, 59], where a base generative model is fine-tuned to align with a target distribution. While effective in task-specific settings, fine-tuning is often computationally expensive, requiring large amounts of labeled data. It also risks overfitting reward signals, leading to over-optimization where outputs optimize the reward superficially but diverge from true user intent. An alternative and increasingly popular strategy is **inference-time guidance** [56, 12], where models are conditioned at generation time using external reward signals without retraining the model. This enables plug-and-play flexibility, allowing users to use reward functions such as classifiers, loss functions, or probability estimators that measure how well the output aligns with the given user conditioning signal. These guidance techniques generally fall into two categories: gradient-based methods, which modify the generative trajectory by applying gradients of the reward function, and sampling-based methods, which rely on generating multiple candidate samples and selecting those with higher reward scores.

Sampling-based approaches, such as Controlled Denoising (`CoDe`) [51], are more general but computationally expensive, as reward alignment depends on sampling large candidate pools at each step. This raises a central question for controllable generation: **Can sampling-based and gradient-based inference-time guidance be unified to reduce sampling overhead, enhance output quality, and achieve better trade-offs between reward alignment and prior divergence?** In this work, we propose a unified framework that combines the strengths of sampling-based and gradient-based inference-time guidance. Inspired by Controlled Denoising (`CoDe`) [51] framework, we extend blockwise sampling

with blockwise gradient-based updates to more effectively guide the generation process.

To evaluate the robustness and generality of our approach, we experiment with various reward models of increasing complexity. We begin with simple reward models that evaluate only the final generated image, then incorporate models that are jointly conditioned on the generated image and prompt. We further also extend to reward models that incorporate additional signals such as a reference image for style guidance. Finally, we test our method on non-differentiable reward functions by approximating gradients through zero-order optimization techniques, showcasing the applicability of our method in diverse and practical scenarios where backpropagation is not feasible. While our experiments focus on image generation, the proposed methods are broadly applicable across generative domains. As GenAI continues to evolve, the ability to control and align outputs with nuanced goals is becoming increasingly fundamental. This work takes a step toward that vision by offering a more flexible, efficient, and generalizable framework for guided generation.

In the following chapters, first, we present a scientific paper on the proposed method in Chapter 2, which we aim to submit at the *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2026*. Following this, we present motivation and the impact of Generative AI, an overview of image generation methodologies, diffusion models, and conditional image generation in Chapters 3, 4, 5 and 6, respectively as additional background material for the paper. Next, in Chapter 7 we discuss some additional things and the alternatives considered and we conclude with concluding remarks and future research directions in Chapter 8.

2

Scientific Paper

Unified Blockwise Control for Denoising Diffusion Models

Maurya Goyal[1], Anuj Singh[1,2], Hadi Jamali-Rad[1,2]

[1]Delft University of Technology, The Netherlands
[2]Shell Global Solutions International B.V., Amsterdam, The Netherlands

Abstract

Aligning diffusion model outputs with downstream objectives is essential for improving task-specific performance. Broadly, inference-time training-free approaches for aligning diffusion models can be categorized into two main strategies: ***sampling-based*** *methods, which explore multiple candidate outputs and select those with higher reward signals, and* ***gradient-guided*** *methods, which use differentiable reward approximations to directly steer the generation process. In this work, we propose a universal algorithm,* ***CoDeX****, which brings together the strengths of blockwise sampling and gradient-based guidance into a unified framework. Building on the blockwise sampling paradigm of* `CoDe` *[1],* `CoDeX` *integrates local gradient signals during sampling, thereby addressing the sampling inefficiency inherent in complex reward-based sampling approaches like* `CoDe`*. At the same time, it overcomes the limited applicability of traditional gradient-guided methods, which often struggle with non-differentiable rewards. By cohesively combining these two paradigms,* `CoDeX` *enables more efficient sampling while offering better trade-offs between reward alignment and divergence from the diffusion unconditional prior. Empirical results demonstrate that* `CoDeX` *consistently outperforms* `CoDe` *and remains competitive with state-of-the-art baselines across a range of tasks. Our code is available at:* `https://github.com/maurya-goyal10/CoDe_ext`

1. Introduction

Diffusion models [2–5] have illustrated impressive capabilities in sampling complex data distributions, allowing them to generate high-quality images [6–8], videos [9, 10], audio [11, 12], and even biological structures such as proteins and DNA [13, 14]. These models are typically trained on large and diverse datasets, allowing them to capture a wide range of patterns and variations [15]. However, in many practical scenarios, it becomes important to *guide* the generative process towards outputs maximizing specific

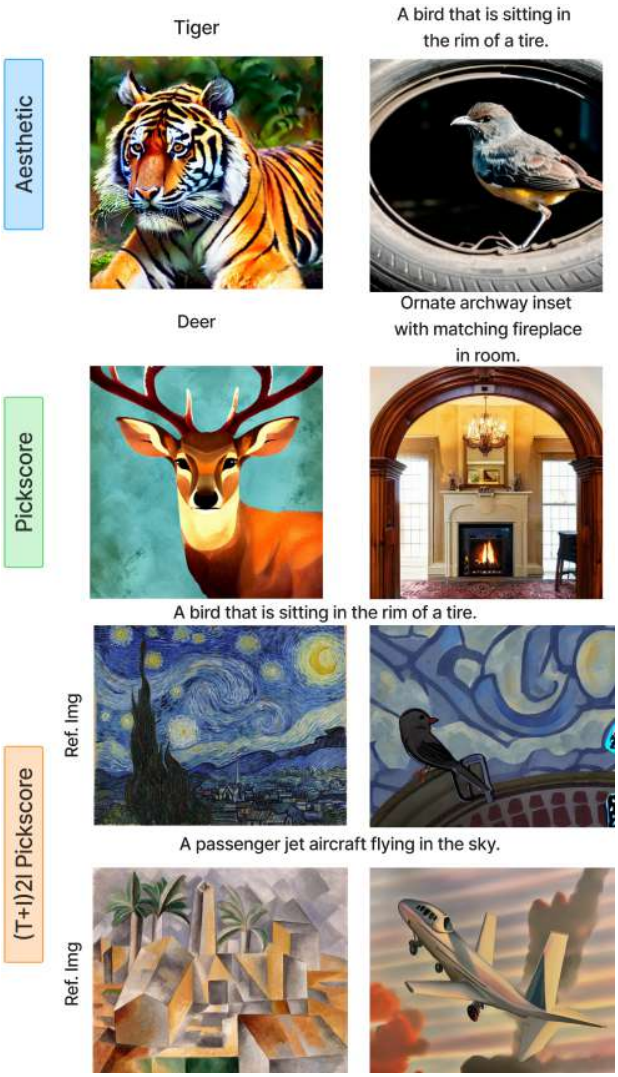

Figure 1. `CoDeX` generates high-quality images across diverse reward objectives such as aesthetic, pickscore, and in (T+I)2I guided using pickscore demonstrating robust and adaptable performance.

properties, such as aesthetic appeal or adherence with user-provided prompts or images. Thus, conditioning diffusion models to produce outputs that meet user-defined criteria has emerged as a consequential problem.

There are several ways for conditioning generative models to produce outputs aligned with the user's desired objective. The most straightforward method is fine-tuning [16–19] the unconditional foundational model for each specific task or condition. Fine-tuning has demonstrated strong performance in aligning models with specific tasks, enabling the base model to learn domain-specific nuances and outperform more general-purpose approaches when sufficient data and computational resources are available. Furthermore, once fine-tuned, the model can generate aligned outputs rapidly at inference time, requiring no additional computation during sampling. However, fine-tuning is computationally expensive and typically requires substantial labeled data or high-quality reward signals. It also potentially leads to reward over-optimization, a form of reward hacking where the model exploits loopholes in the reward function to produce high-reward outputs that do not align with true user intent. As an alternative, inference-time guidance [5, 7] offers a more flexible solution by enabling training-free conditioning through external reward models applied during sampling.

In this work, we focus exclusively on inference-time guidance for conditional image generation scenarios, specifically in the setting where the reward model is applied to the final denoised (clean) samples rather than intermediate noisy ones. This allows us to leverage off-the-shelf reward models directly, without modifying them or extra training. The reward models can be any loss function, classifier, or probability estimator that measures how well the output aligns with the given user conditioning signal.

Training-free guidance methods can be extensively categorized into gradient-based [20–24] and sampling-based approaches [1, 25]. Gradient-based guidance involves perturbing samples during the denoising process in the direction that increases the reward, effectively guiding the generation toward more desirable outcomes. Conversely, sampling-based methods utilize stochasticity by generating multiple candidate samples and preferentially sampling more from those expected to yield higher rewards. While sampling-based approaches often offer a better trade-off between target reward alignment and the divergence from the prior [1], they come at a high computational cost, as the number of required samples increases exponentially with the divergence between the prior distribution and the desired posterior distribution [26]. On the other hand, gradient-based guidance can directly steer the generation toward desirable outputs but faces limitations in flexibility, differentiability, and generalization. In this work, we explore how these two paradigms can be effectively unified to leverage their complementary strengths and aim to answer the key question: **Can sampling-based and gradient-based inference-time guidance be unified to reduce sampling overhead, enhance output quality, and achieve better trade-offs between reward alignment and prior divergence?**

Inspired by Controlled Denoising (`CoDe`) [1] proposed by Singh *et al.*, we incorporate blockwise gradient guidance with blockwise sampling to answer this question. This unified framework leverages the strengths of both sampling-based and gradient-based methods: the flexibility of sample-based reward maximization and the efficiency of gradient-driven updates. By using gradient signals to explicitly steer the prior distribution toward the desired posterior, we reduce the number of samples required during the reverse diffusion process, thus improving computational efficiency while maintaining high output quality and strong alignment with the target conditioning. Furthermore, we also investigate the following set of extensions designed to enhance the overall performance and efficiency of the method, while also improving its generality and adaptability across different tasks and reward structures:

- **Sampling schedule:** Introducing a scheduled sampling strategy that allocates more sampling budget to steps in the denoising process where the reward signal is expected to be most significant.
- **Non-greedy sampling:** Replacing the myopic greedy sampling strategy with more flexible alternatives such as multinomial sampling to better control the exploration-exploitation tradeoff.
- **Clustering-based guidance:** Incorporating clustering techniques to limit the overhead of applying gradient guidance to every sample individually.
- **Multi-reward extension:** Extending the `CoDeX` framework to accommodate multiple, potentially competing, reward functions.
- **Non-differentiable rewards:** Also test the method on non-differentiable rewards using zero-order optimization, utilizing the torchopt [27] library.

2. Related Works

Image Generation via Diffusion Models. Early efforts on image generation relied on GANs [28] and VAEs [29, 30]. However, these approaches suffer from problems such as training instabilities, limited representational power, and mode collapse. This resulted in the generation of highly smooth images for VAEs [31–33] or a lack of sample diversity and control for GANs [34, 35]. Diffusion models

tackle this by gradually transforming noise into clean samples through a learned denoising process iteratively, preserving both the fidelity and diversity of the target distribution [3, 7, 8]. This makes them particularly well-suited for high-quality and diverse image-generation tasks.

Conditional Image Generation. Image generation models are typically trained on large datasets which enable generating diverse and high-quality images. To provide users with greater control over the generated content, conditional image generation is used. Rather than sampling from the marginal distribution $p(x)$ conditional methods sample from $p(x|c)$, where c represents the conditioning signal such as class labels, textual prompts, images, or other forms of guidance. Diffusion models are particularly well-suited for conditional image generation due to their score-based formulation, which estimates the gradient of the data distribution's log density [5] (explained in detail in section 3). This formulation naturally accommodates the integration of conditioning signals into the generation process. Conditional image generation can be further classified into two broad categories: training-based and training-free.

Training-Based Methods. This includes methods that train the model with conditioning signals (which can be guided at inference time for better performance using methods like classifier-free guidance (`CFG`) [36]) or fine-tuning-based methods that align the pre-trained foundational model to the specific conditioning. Aligning pre-trained models through fine-tuning is used a lot not just in vision but also for language [37, 38]. For diffusion models, there are several ways of doing that - direct backpropagation [17, 39], RL-based fine-tuning [40, 41], preference-based supervised fine-tuning [18, 42], domain adaption [43], etc. These methods are not scalable and require a lot of computation for fine-tuning each different control signal, therefore, we look into training-free inference-time alignment.

Training-Free Methods. These methods leverage expected rewards during the denoising process to perturb or select optimal samples at **inference time**. This guidance can be categorized into two main types: gradient guidance and sampling-based guidance.

Gradient Guidance. [20–24, 44] They leverage the expected predicted sample $x_{0|t}$ at each denoising step and compute the gradient of the reward for this estimate. This gradient guides the denoising trajectory toward regions in the sample space that are expected to give higher rewards, effectively *guiding* the generative process to produce samples with improved rewards.

Sampling-Based Guidance. [1, 10, 25, 45] Instead of relying on gradient information, they generate multiple candidate samples at each denoising step and estimate the expected reward for each. This allows them to identify promising directions in the sample space and explore more of those areas in the denoising process, thereby guiding the generation toward samples with higher expected rewards.

Combining Gradient and Sampling-Based Guidance. TDS [26] was the first to propose a hybrid approach that combines gradient-based optimization with sampling-based exploration. However, their method relies on the Sequential Monte Carlo (SMC) sampling, assuming the gradient to be sampling from the posterior and is evaluated only on relatively simple tasks such as class-conditional generation on MNIST and CIFAR [46], limiting its generalizability. In contrast, our work explores a broader range of tasks by leveraging off-the-shelf reward models without restricting the setting to predefined class labels. Concurrent with our work, DAS [45] extends the TDS framework by introducing a tempering scheme to better explore reward-guided generation. However, it does not consider the blockwise perspective in gradient and sampling guidance, nor does it explore complex scenarios such as (T+I)2I (Text-and-Image-to-Image) guidance.

Approximating Gradients for Non-Differential Functions. To ensure that the proposed method remains applicable to both differentiable and non-differentiable rewards, gradient approximations are used to enable perturbations in the direction of reward improvement. Two primary strategies exist for this purpose. The first involves training a surrogate model to approximate the reward function, however, this approach conflicts with the overarching goal of preserving a training-free framework. Therefore, this work adopts the second strategy: zero-order optimization [47,48], which allows for gradient approximation using only forward evaluations of the reward function, thereby avoiding the need for additional model training.

3. Background

Diffusion Models. Diffusion models are Markov Chains where for the forward process we iteratively noise a clean input by adding Gaussian noise to it. Thus as we go from t=0 (clean image) to t= T and for large enough T, we reach Gaussian noise. Using a neural network we learn to reverse this process $p_\theta(x_{t-1}|x_t)$ (denoising) since $q(x_t|x_{t-1})$ is known. Consequently, iteratively applying $p_\theta(x_{t-1}|x_t)$ to a randomly sampled Gaussian noise, we can sample from the target distribution or generate a new image.

Noise Based Formulation. The forward process at each time is defined in Eq. 1 where the β_t represents the variance schedule. $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{t=1}^{T} \alpha_t$ [3]

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t \tag{2}$$

The whole forward process boils down to:

$$q(x_{1:T}|x_{t-1}) = \prod_{t=1}^{T} q(x_t|x_{t-1}). \tag{3}$$

The diffusion model learns the reverse of this forward step:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \beta_t \mathbf{I}), \tag{4}$$

where the $\mu_\theta(x_t, t)$ is defined as:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right). \tag{5}$$

Hence, the UNet [49] given the noisy input (x_t) predicts the noise to be removed to obtain the clean sample (x_0) from Eq. 2 as:

$$\epsilon_\theta(x_t, t) \approx \epsilon_t = \frac{x_t - \sqrt{\bar{\alpha}_t} x_0}{\sqrt{1-\bar{\alpha}_t}}. \tag{6}$$

Score Based Formulation. Diffusion models also have a score-based formulation through SDE [5] to estimate the score function which is defined as $\nabla_{x_t} \log p(x_t)$ using a neural network. The forward process is defined as:

$$dx = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}. \tag{7}$$

In the above equation $f(x, t)$ is vectored value function called the *drift* coefficient of $x(t)$ and $g(t)$ is a scalar function known as the *diffusion* coefficient of $x(t)$, both of which are known. The reverse process is defined as:

$$dx = [\mathbf{f}(\mathbf{x}, t)dt - g(t)^2 \nabla_x \log p_t(\mathbf{x})]dt + g(t)d\bar{\mathbf{w}}. \tag{8}$$

Where $\bar{\mathbf{w}}$ is the standard Wiener process when time flows backward from T to 0. The only unknown term is the score function i.e. $\nabla_x \log p_t(x)$ which the neural network learns to predict. Thus after sampling $x(T) \sim p_T$ and reversing the process we can obtain the clean image (sample from p_0).

Alignment Objective. Considering an off-the-shelf reward function $r : \mathcal{X} \to \mathbb{R}$ that quantifies the quality of alignment of generated samples with a given conditioning. Let $\pi(\cdot)$ denote the reward-aligned diffusion model and $p(\cdot)$ the base, unconditional reference diffusion model [1]. From a reinforcement learning perspective, the alignment objective can be formulated as optimizing π to maximize the expected reward under the aligned distribution relative to p, thereby encouraging the generation of samples satisfying:

$$\pi^*_\lambda = \arg\max_\pi \Bigg[\lambda \, \mathbb{E}_{x_{t-1} \sim \pi(x_{t-1}|x_t)} \Big[\mathbb{E}_{x_0 \sim p(x_{t-1}|x_t)} [r(x_0)] \Big] - \mathrm{KL}\big(\pi(x_{t-1}|x_t) \,\|\, p(x_{t-1}|x_t)\big) \Bigg], \tag{9}$$

where $\lambda \in \mathbb{R}^{\geq 0}$ trades off reward for drift from the base diffusion model p. Thus, the objective can be interpreted as maximizing the expected reward while minimizing the divergence from the prior distribution p, ensuring that the learned policy π improves alignment without deviating excessively from the base diffusion model.

Classifier Based Guidance [5, 7]. For conditional generation [7], the goal is to sample from $p(x|y)$ instead of $p(x)$. Thus wrt to the score-based formulation we want the $\nabla_{x_t} \log p(x_t|y)$. Using Bayes Rule [50]: ∇ represents ∇_{x_t}

$$\begin{aligned} \nabla \log p(x_t|y) &= \nabla \log \left(\frac{p(x_t) p(y|x_t)}{p(y)} \right) \\ &= \nabla \log p(x_t) + \nabla \log p(y|x_t) - \nabla \log p(y) \\ &= \underbrace{\nabla \log p(x_t)}_{\text{unconditional score}} + \underbrace{\nabla \log p(y|x_t)}_{\text{adversarial gradient}}. \end{aligned} \tag{10}$$

In Eq. 10, we know the first-term unconditional score from the foundational diffusion model and we get the second term using an off-the-shelf reward model.

Tweedies Formula. In Eq. 10, we need the value of $p(y|x_t)$, which represents the probability of the desired outcome given a noisy image x_t. However, most reward functions available off the shelf are not designed to handle noisy inputs, rather they expect clean images. To address this issue, there are two possible approaches: either train a new reward function capable of operating directly on noisy images or estimate the clean image from the noisy input using Tweedie's formula. [51]. Since our goal is to develop a training-free method, we choose the second approach. Tweedie's formula lets us estimate the expected clean image given the noisy input and is given as:

$$\hat{x}_0 = \mathbb{E}[x_0|x_t] = \frac{x_t - \sqrt{1-\bar{\alpha}_t} \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}. \tag{11}$$

Using Eq. 11 we get the reward value as:

$$r(x_{0|t}, y) \approx r(\mathbb{E}[x_0|x_t], y) = r(\hat{x}_0, y). \tag{12}$$

Controlled Decoding (CoDe). The idea of using blockwise greedy sampling for guidance builds on the Best-of-N (BoN) approach [52]. When performing conditional sampling from the posterior $p(x|y)$ given the prior $p(x)$, we can draw multiple independent samples from the prior and at the end of the denoising select the one that best aligns with the posterior. Given the probabilistic nature of the denoising in diffusion models, a more aggressive way is to estimate the expected alignment with the posterior (using a reward model Eq. 11 and Eq. 12) and then explore more in the direction expected to give higher rewards [1, 25]. To enhance computational efficiency and scalability, CoDe propose performing this sampling procedure in a blockwise manner, executing it every B step and generating N candidate samples per block. This parallels RL-based objectives where one sample multiple times from a base policy (the base unconditioned diffusion model) and selects the sample best aligning with the reward function.

SDEdit. Stochastic Differential Editing [53] is an image synthesis and editing method that generates images aligning

with a reference without relying on complex reward models. SDEdit modifies the standard denoising diffusion process by replacing the typical starting point of denoising, T, with $r \times T$, where r is a user-defined percentage of noise $r \in (0, 1)$. This implies that instead of beginning the denoising from pure random noise, the process initiates from an intermediate point. This is achieved by adding a controlled amount of noise to the reference image, effectively mimicking the forward diffusion process. The choice of r directly influences the output: a higher r allows for greater creative freedom, while a lower r enables the preservation of more structural and stylistic properties from the reference image. SDEdit is employed for (T+I)2I (Text-and-Image-to-Image) generation, enabling style transfer while offering users multiple adjustable parameters to control different aspects of the output image.

Zero-Order Optimization. Zero-order optimization is used when we have to approximate the $\nabla_x f(x)$ using just the forward pass $f(x)$.

$$\nabla f(x) \approx \frac{1}{N'} \sum_{i=1}^{N'} \frac{f(x + \sigma\epsilon_i) - f(x - \sigma\epsilon_i)}{2\sigma} \epsilon_i, \qquad (13)$$

where ϵ is a n-dimensional vector for n-dimensional samples x sampled from $(0, \mathbf{I}_d)$ and σ is a scalar constant. N' is the number of samples the more samples we have the better the estimate at the cost of time and compute.

4. Methodology

We use blockwise gradient guidance to compute and add the gradients to the images at that timestep for each stream and then do the sampling. The algorithm is explained in detail in Algorithms 1, 2 and 3. The difference between the previous methods and our method is also explained by the schematic diagrams in Fig 2. As outlined in Algorithm 1, the procedure integrates blockwise gradient guidance (lines 5-12) with blockwise sampling (lines 13-16) in a unified framework. Specifically, blockwise gradient guidance is applied independently to each stream (lines 7–10), allowing for localized adjustment of the gradient signal at the block level. Subsequently, preferential sampling is performed based on the expected rewards computed for each block (lines 13–16), enabling the model to selectively emphasize higher-reward regions during denoising.

To make the sampling more general we also change the greedy sampling to temperature-based multinomial sampling (Algorithm 3) this allows the model to consider paths that can give higher rewards later in the denoising and using the temperature we can adjust the aggressiveness of the sampling. Thus it allows the sampling to explore more and can adjust the tradeoff between exploring and exploiting by changing the temperature parameter.

Algorithm 1 CoDeX

Require: T, N, B_g, B_s, τ, p_θ, r, γ

1: Sample initial noise: $\{z_T^{(n)}\}_{n=1}^N \sim \mathcal{N}(0, I)$
2: Initialize counter: $s \leftarrow 1$
3: **for** $t = T, \ldots, 1$ **do**
4: $\quad \{z_{t-1}^{(n)}\}_{n=1}^N \leftarrow p_\theta(\{z_t^{(n)}\}_{n=1}^N, t)$
5: $\quad$ **if** $\text{mod}(s, B_g) = 0$ **then**
6: $\quad\quad$ **if** gradient condition satisfied **then**
7: $\quad\quad\quad$ **for** $k = 1$ to N **do**
8: $\quad\quad\quad\quad g_k \leftarrow \text{GRAD}(z_{t-1}^{(k)}, t-1)$
9: $\quad\quad\quad\quad z_{t-1}^{(k)} \leftarrow z_{t-1}^{(k)} + \gamma \cdot g_k$
10: $\quad\quad\quad$ **end for**
11: $\quad\quad$ **end if**
12: $\quad$ **end if**
13: $\quad$ **if** $\text{mod}(s, B_s) = 0$ **then**
14: $\quad\quad \{\hat{z}_0^{(n)}\} \leftarrow \mathbb{E}_{p_\theta}(\{z_0^{(n)}\} | \{z_{t-1}^{(n)}\}, t-1)$
15: $\quad\quad \{z_{t-1}^{(n)}\} \leftarrow \text{SAMPLE}(\{z_{t-1}^{(n)}\}, r(D(\{\hat{z}_0^{(n)}\})), \tau)$
16: $\quad$ **end if**
17: $\quad s \leftarrow s + 1$
18: **end for**
19: **return** z_0

Algorithm 2 GRAD(z_t, t)

Require: latent z, timestep t

1: $\hat{x}_0 \leftarrow D(\mathbb{E}_{p_\theta}(z_0 | z_t, t))$ ▷ Expected clean sample
2: $\hat{r}(z_t) \leftarrow r(\hat{x}_0)$ ▷ Compute reward
3: $g \leftarrow \nabla_{z_t} \hat{r}(z_t)$ ▷ Compute gradient w.r.t. z_t
4: **return** g

We also experiment with other extensions like having a sampling schedule that allows exploring more in the denoising steps expected to give more rewards. Moreover, we experiment with clustering to decrease the overhead of adding a gradient to all the sample streams and cluster together points in the latent space that are closer, calculate one gradient for all of them, and add it to all the points in the cluster.

5. Experiments

We evaluate the performance of CoDeX by comparing it against a range of state-of-the-art guidance methods across both Text-to-Image (T2I) and Text-and-Image-to-Image ((T+I)2I) generation tasks. These evaluations are conducted under both differentiable and non-differentiable reward settings to ensure a comprehensive analysis. All experiments are conducted using the pre-trained Stable Diffusion v1.5 model [8], which was trained on the LAION-400M dataset [54]. Stable Diffusion is a text-conditioned latent diffusion model trained on paired image-text data. It operates in latent space to perform denoising and leverages text-conditioning to generate semantically aligned images.

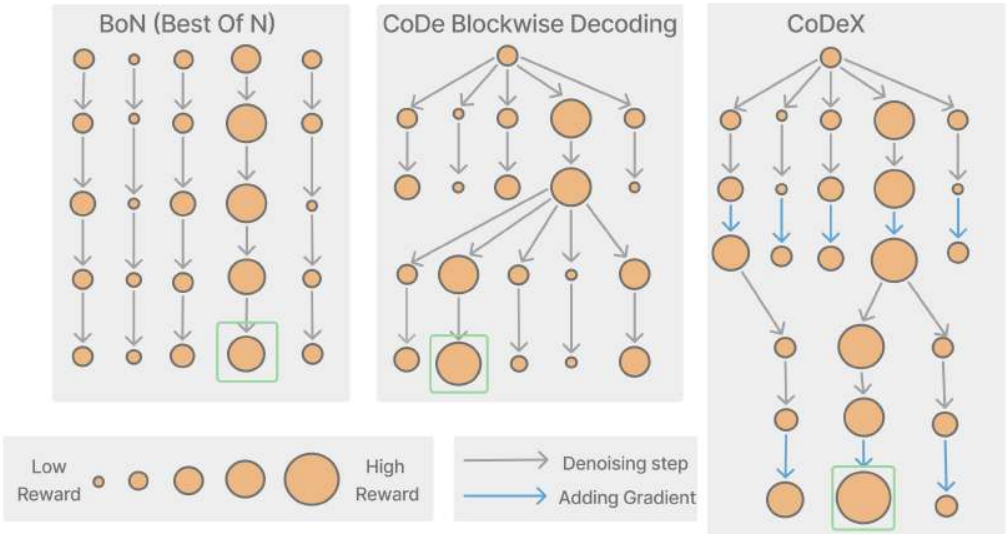

Figure 2. Schematic illustration of sampling-based guidance methods: **BoN** runs parallel diffusion streams and selects the highest-reward sample after full denoising; **CoDe** performs greedy blockwise selection to focus on promising regions early; **CoDeX** enhances CoDe with blockwise gradient-based guidance, scheduled particle sizes, and multinomial sampling for better efficiency and performance.

Algorithm 3 SAMPLE$(\{z_{t-1}^{(n)}\}, r(D(\{\hat{z}_0^{(n)}\})), \tau)$

Require: current images $\{z_{t-1}^{(n)}\}_{n=1}^N$, reward vector $\mathbf{r} = r(D(\{\hat{z}_0^{(n)}\}))$, temperature τ, empty list $\{z_{temp}^{(n)}\}_{n=1}^N$

Compute softmax probabilities:

$$P_n \leftarrow \frac{\exp(\mathbf{r}_n/\tau)}{\sum_{j=1}^N \exp(\mathbf{r}_j/\tau)} \quad \text{for } n = 1, \ldots, N$$

for $i = 1$ to N **do**
  Sample index $i \sim$ Multinomial$(\{P_n\}_{n=1}^N)$
  Append $z_{t-1}^{(i)}$ to $\{z_{temp}^{(n)}\}$
end for
return $\{z_{temp}^{(n)}\}_{n=1}^N$

To ensure fair and reproducible comparisons, we generate 10 images per experimental setting (i.e., for each prompt-reference image pair) using 500 DDPM steps. All experiments are conducted using NVIDIA A40 GPUs, with the random seed fixed at 2024 for reproducibility. Both qualitative and quantitative results are reported across a variety of scenarios to highlight the capabilities of CoDeX. For all scenarios, the sampling block size is set to 5. The gradient block size is adjusted based on the reward model: 5 for Aesthetic, 4 for pickscore, and 2 for the (T+I)2I setting. Additionally, unless specified otherwise, the guidance scale is fixed at 0.2 across all experiments.

Evaluation Settings and Metrics. We evaluate the performance of all methods using a comprehensive set of metrics designed to capture different aspects of alignment, quality, and divergence from the unconditional model. These include the expected reward, Frechet Inception Distance (FID) [55], CLIP-based Maximum Mean Discrepancy (CMMD) [56], I-Gram [57, 58], and CLIPScore [59], referred to as T-CLIP throughout the paper. FID and CMMD measure the deviation from the prior distribution defined by the unconditioned Stable Diffusion model, providing a sense of how far the generations diverge from typical outputs. I-Gram captures style and texture similarity between the reference and generated images, particularly relevant in the (T+I)2I setting, while T-CLIP assesses semantic alignment with the input prompt. To support diverse evaluation goals, we consider a variety of reward models: the LAION Aesthetic Predictor v2 [54, 60] for image-based aesthetic quality, pickscore [61] for text-image alignment, a multi-reward configuration that combines aesthetic and pickscore, compressibility [40], and a non-differentiable reward facilitating compact image representations.

Baselines. In all experimental settings, we compare CoDeX against CoDe with both N=40 and N=4, where N denotes the number of samples generated. We treat the N=4 configuration as the primary baseline, as CoDeX itself operates using only 4 sample streams per block. For the Aesthetic and pickscore reward models, we additionally benchmark against state-of-the-art gradient-based guidance methods, including MPGD [23], FreeDoM [22], and Universal Guidance (UG) [21], which are known to improve upon earlier approaches like DPS [20] by providing more accurate gradient estimates.

Figure 3. The aesthetic alignment offered by CoDeX outperforms other baselines in terms of the aesthetic score (also shown in Tables 1, 2) while adhering to the given prompt. Despite a higher CMMD indicating a slight shift from the unconditional prior, CoDeX avoids reward hacking and produces the most aesthetically pleasing samples compared to other baselines.

5.1. Image Reward Models

We first experiment with reward models just based only on the final generated image such as the **aesthetic scorer**. To guide the diffusion denoising process towards generating aesthetically pleasing images, we use the LAION aesthetic predictor V2 [54], which leverages a multi-layer perceptron (MLP) architecture trained on top of CLIP embeddings. This model's training data consists of 176,000 human image ratings, spanning a range from 1 to 10, with images achieving a score of 10 being considered art pieces.

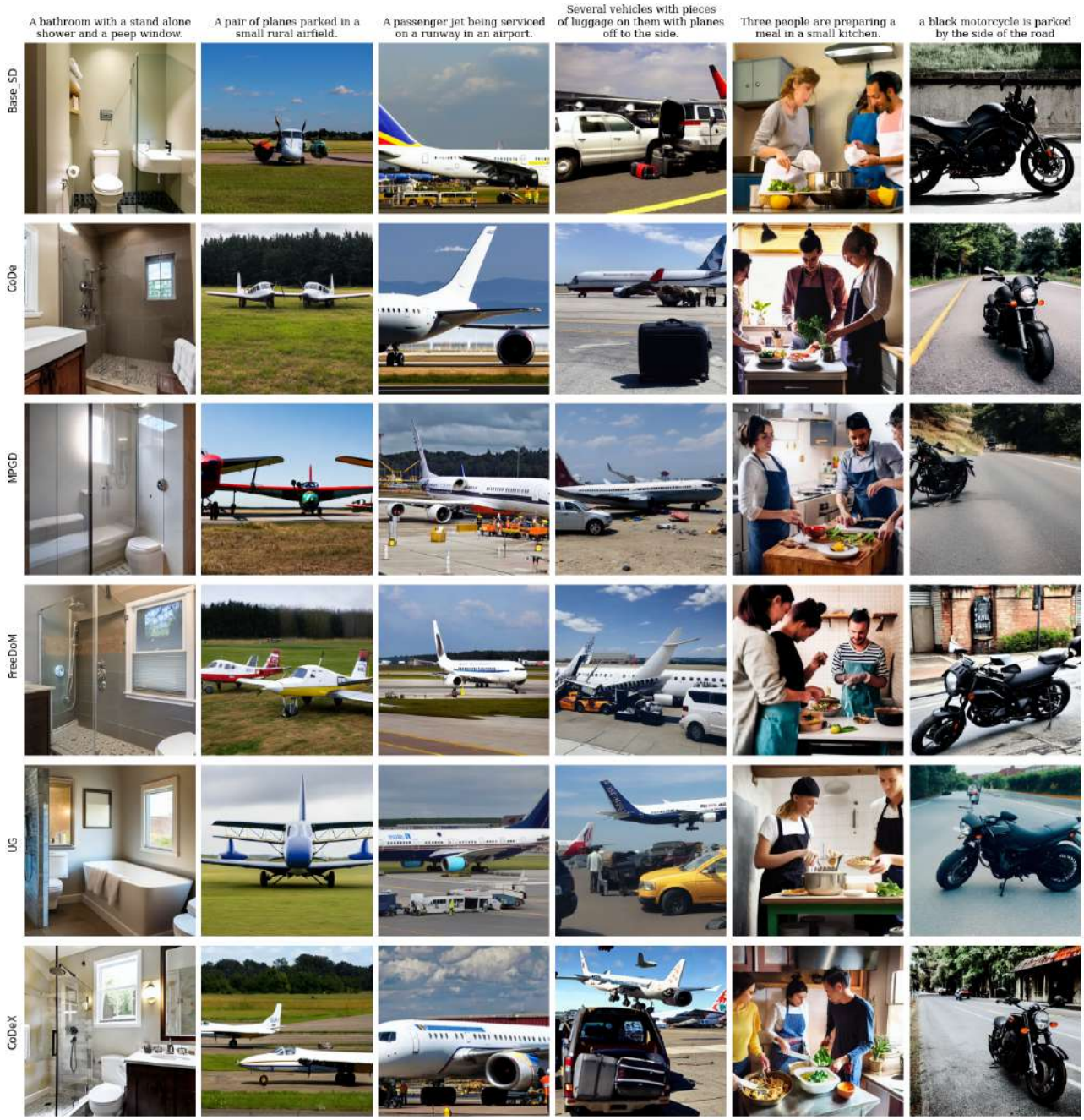

Figure 4. These qualitative examples illustrate how CoDeX consistently produces images that more accurately and richly reflect the nuances of the input prompts across various scenarios. This is quantitatively supported by higher pickscore and T-CLIP scores (Tables 3, 4).

We evaluate them on the evaluation dataset of animals from ImageNet [62] (containing 51 prompts).

The quantitative results are summarized in Table 1, we can see that CoDeX not only takes roughly $4-5$ times less time as compared to CoDe [1] but also gives much better rewards. This comes at the cost of higher divergence from the prior distribution which is shown by the increase in the CMMD, while the FID and the T-CLIP stay similar. We also ablate on the different settings in CoDeX, as evident from the results the major improvement comes from adding the gradients, and then we can observe little improvements from scheduling samples and making the sampling multino-

mial. Furthermore, using clustering we can further reduce the time taken at the cost of the rewards.

We further compare the performance of CoDeX against state-of-the-art gradient-based guidance methods, as shown in Table 2. While CoDeX achieves higher reward scores, this improvement is accompanied by a greater divergence from the unconditional prior. However, as demonstrated by the qualitative results in Fig. 3, where CoDeX is evaluated alongside all competing methods, this increased divergence does not come at the cost of semantic or perceptual alignment. The samples clearly show that CoDeX maintains high visual fidelity and relevance to the conditioning inputs, suggesting that its superior reward performance is not attributable to reward hacking.

5.2. Prompt Alignment Reward Models (T2I)

In this section, we guide the model to adhere better to the given text prompt. Although the Stable Diffusion v1.5 is trained on text input conditioning, the performance (adherence to prompt and output quality) starts degrading as the prompt complexity increases. Hence, we use **pickscore** [61] as the reward to generate good-quality images adhering to the prompts. pickscore is a CLIP-based scoring function trained on Pick-a-Pic a large dataset of text-to-image prompts and real user preferences over generated images. We evaluate the performance of the generated images with prompts taken from the HPDv2 [63] evaluation settings.

The quantitative results are given in Tables 3 and 4, and the qualitative examples are shown in Fig. 4. CoDeX significantly improves efficiency ($3-4\times$ faster), achieves higher rewards, and maintains less divergence from the prior (lower FID and CMMD). Moreover, CoDeX outperforms gradient guidance methods, providing better prompt alignment and image quality.

5.3. Multi-Reward

In this section, instead of guiding the generated image for a single reward, we consider an objective that balances multiple rewards. Specifically, we guide the generation using a weighted sum of the aesthetic score and the pickscore: $\gamma_1 \times$ aesthetic $+ \gamma_2 \times$ pickscore. The aesthetic score motivates aesthetically pleasing outputs, while the pickscore assures alignment with the prompt based on human evaluations. This grants users a more controlled trade-off between visual quality and prompt fidelity. By adjusting the weights γ_1 and γ_2, users can tailor the generation process to their specific preferences.

A plot of aesthetic score versus pickscore is shown in Fig. 5. In comparison to the base diffusion model operating unconditionally, our approach allows for controlled guidance by adjusting the parameters γ_1 and γ_2, enabling a flexible trade-off between multiple reward signals such as aesthetic quality and pickscore. This provides users with greater control to tailor generations based on specific objectives or use cases. The performance obtained by CoDeX demonstrates significantly improved rewards across multiple reward components. Specifically, CoDeX significantly improves performance in multi-reward scenarios, boosting the aesthetic reward by about 43% and the pickscore by 7% compared to CoDe on average. Against gradient guidance methods like FreeDoM, CoDeX still outperforms with increases of 29% (aesthetic) and 6% (pickscore).

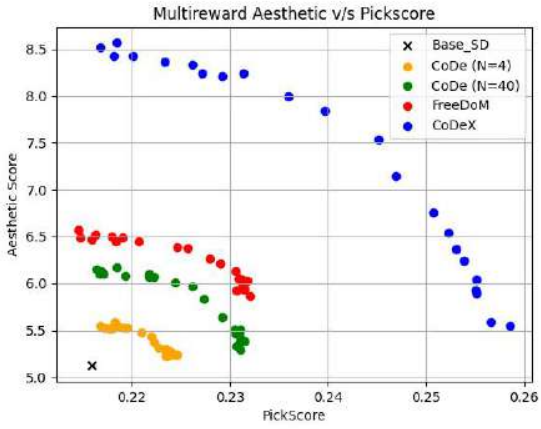

Figure 5. In a multi-reward guidance scenario, CoDeX consistently achieves higher aesthetic scores and pickscores, indicating superior performance in optimizing both rewards compared to other methods.

5.4. (T+I)2I (Text-and-Image-to-Image)

As we move through the experiments we keep on adding more control. This setting blends an additional reference image, enabling the generation of a new image that not only aligns with a given text prompt but also imitates the stylistic elements of the provided reference. We use SDEdit [53] for the style transfer and start the denoising from an intermediary point by adding noise to the reference image and then during the denoising using the pickscore as the reward maintain a strong adherence to the prompt.

This offers users more fine-grained control over multiple competing aspects of image generation, specifically balancing adherence to the text prompt with matching the stylistic elements of a reference image. In this case, for prompt adherence, we are guiding the model using the pickscore explicitly while for style transfer it starts the denoising from the noised reference image (r=0.6) [53]. Thus we want the final generated image to preserve the style and structure of the reference image and at the same time be able to adhere to the provided prompt. As seen in Table 5 and Figure 6 SDEdit prioritizes style transfer. However due to the absence of guidance its effectiveness in capturing prompt in-

Figure 6. (T+I)2I scenario with pickscore for prompt alignment, `SDEdit` shows a stronger emphasis on style at the expense of capturing detailed prompt intricacies. However, `CoDe` and `CoDeX` successfully manage to achieve both the style transfer while adhering to the prompt. `CoDeX` generates images that adhere to the prompt more while still capturing the stylistic elements (Table 5).

Method	Aesthetic Guidance (Image Based Guidance)				
	Rew. (↑)	FID (↓)	CMMD (↓)	T-CLIP (↑)	Time (↓)
Base-SD	5.17	88.213	0.203	0.769	0.32
CoDe (N=4)	5.59	90.51	0.277	0.770	6.93
CoDe (N=40)	6.16	91.52	**0.389**	0.772	57.53
CoDeX (N=4,greedy)	8.26	93.46	0.657	0.772	11.67
CoDeX (N=4,mn($\tau = 0.001$))	8.27	92.44	0.666	0.772	11.52
CoDeX (N schedule,greedy)	8.34	92.93	0.644	0.771	12.94
CoDeX (N schedule, mn($\tau = 0.001$))	**8.37**	**90.65**	0.634	0.772	13.20
CoDeX (N schedule, mn($\tau = 0.001$), KMeans)	8.19	91.23	0.602	**0.773**	**9.42**

Table 1. Aesthetic guidance results comparing CoDeX and CoDe on various metrics like reward (aesthetic score), FID, CMMD, T-CLIP, and time (in minutes per image), The schedule used is [2, 2, 2, 4, 4, 4, 4, 6, 6, 6], mn is multinomial sampling. CoDeX significantly reduces time by $5-6\times$ while improving aesthetic reward alignment. This comes with a higher divergence from the prior (increased CMMD), but without reward hacking (as shown in qualitative samples in Fig. 3 and maintaining alignment with text (T-CLIP)).

Method	Aesthetic Guidance			
	Rew (↑)	FID (↓)	CMMD (↓)	T-CLIP (↑)
MPGD	5.96	95.60	**0.472**	0.762
FreeDoM	6.56	104.61	0.637	0.756
UG	6.18	153.71	0.844	0.745
CoDeX	**8.37**	**90.65**	0.634	**0.772**

Table 2. Aesthetic guidance results comparing gradient guidance methods to CoDeX. CoDeX achieves the highest reward and better prompt alignment (T-CLIP score), while remaining closer to the prior than other methods. FreeDoM and UG, due to their recurrent steps, are further from the prior (higher CMMD and FID).

Method	Pickscore Guidance			
	Rew (↑)	FID (↓)	CMMD (↓)	T-CLIP (↑)
MPGD	0.2254	140.00	0.500	0.810
FreeDoM	0.2287	143.23	0.483	0.817
UG	0.2282	149.54	0.626	0.816
CoDeX	**0.2449**	**131.64**	**0.457**	**0.824**

Table 3. Pickscore guidance results comparing gradient guidance methods to CoDeX. It deviates less from the prior (lower CMMD) and offers better prompt alignment (higher pickscore and T-CLIP).

tricacies is limited. In comparison, both CoDe and CoDeX demonstrate the capacity to balance both style and prompt adherence. CoDeX achieves slightly better prompt adherence and enhanced style transfer, all while being nearly twice as fast as CoDe.

5.5. Non-Differentiable Rewards

One of the key advantages of CoDe is its ability to operate in settings with non-differentiable rewards, as it does not rely on gradient-based optimization. Motivated by this, we also extended our algorithm to such scenarios. We use zero-order optimization, which enables gradient approximation without explicitly requiring reward differentiability.

For the non-differentiable reward, we use image compressibility [40] which measures the size of the image in kilobytes. Therefore we guide the model to create lightweight compressible images. The quantitative results are summarized in Table 6. As seen from the performance degradation, approximating gradients in this setting is particularly challenging. The high dimensionality of image data, even when working in a latent space, requires a large number of particles to obtain a reliable gradient estimate. This leads to high computational overhead that renders the method inefficient in practice when compared to CoDe. This can be also seen in the qualitative results in Fig. 7.

Unconditional models produce highly detailed images, which are visually rich and are thus less compressible. CoDe effectively blurs backgrounds and removes fine details, like wolf hair in the qualitative examples, leading to improved compressibility. CoDeX despite using many particles (N'=50), suffers from noisy gradient estimates which leads to significantly increased computational cost and time. Therefore, as seen qualitatively, while CoDeX introduces some blurring, it struggles to remove fine details as effectively as CoDe, illustrating the inefficiencies of zero-order optimization in high-dimensional spaces.

6. Conclusion

We propose a unified framework that efficiently combines blockwise sampling with blockwise gradient-based guidance, reducing sampling overhead while maintaining strong performance trade-offs. This integration makes CoDe more effective for differentiable reward settings, where it outperforms purely sampling-based strategies in both efficiency and reward quality. However, for non-

Method	pickscore Guidance (T2I Based Guidance)			
	Rew. (↑)	CMMD (↓)	T-CLIP (↑)	Time (↓)
Base-SD	0.21901	0.3176	0.8115	0.33
CoDe (N=4)	0.23094	0.3873	0.8179	1.89
CoDe (N=40)	0.24244	0.5116	0.8246	16.89
CoDeX (N=4,greedy)	0.24444	0.4666	**0.8265**	4.45
CoDeX (N=4,mn(τ=0.0003))	0.24440	0.4598	0.8229	4.47
CoDeX (N schedule,greedy)	**0.24493**	0.4569	0.8242	4.45
CoDeX (N schedule,mn(τ=0.0003))	0.24413	0.4429	0.8230	4.44
CoDeX (N schedule,mn(τ=0.0003),KMeans)	0.24221	**0.4235**	0.8216	**3.44**

Table 4. Pickscore guidance results comparing CoDeX and CoDe on various metrics like reward (pickscore), CMMD, T-CLIP, and time (in minutes per image), The used schedule is [2, 6, 6, 2, 2, 2, 4, 4, 6, 6], mn is multinomial sampling. CoDeX improves guidance efficiency (reducing time by $3-4\times$) and achieves better prompt alignment (higher pickscore and T-CLIP), while also deviating less from the prior (lower CMMD).

Method	pickscore Guidance ((T+I)2I Based Guidance)				
	Rew. (↑)	I-Gram (↑)	CMMD (↓)	T-CLIP (↑)	Time (↓)
SDEdit	0.20385	46.35	1.0078	0.75618	0.06
CoDe (N=4)	0.21324	44.91	1.1162	0.77535	0.32
CoDe (N=40)	0.22210	43.69	**1.1984**	0.77993	2.87
CoDeX (N=4,greedy)	0.22229	**45.07**	1.2489	**0.78113**	1.12
CoDeX (N=4,mn(τ=0.0003))	**0.22263**	45.04	1.2210	0.77717	**1.05**

Table 5. (T+I)2I pickscore guidance results comparing CoDeX and CoDe on various metrics like reward (pickscore), I-Gram CMMD, T-CLIP, and time (in minutes per image), mn is multinomial sampling. CoDeX improves guidance efficiency (reducing the time by $2\times$) and achieves better prompt alignment (higher pickscore and T-CLIP), while also preserving the style better (I-Gram), which comes at higher deviation from the prior (higher CMMD).

Method	Compress Guidance (Non-Differential)				
	Rew. (↑)	FID (↓)	CMMD (↓)	T-CLIP (↑)	Time (↓)
Base-SD	-103.089	72.30	0.1870	0.7712	0.35
CoDe (N=4)	-61.480	81.86	0.3378	0.7744	1.74
CoDe (N=40)	**-32.015**	135.00	0.9901	0.7574	15.79
CoDeX ($N'=1$)	-64.524	**81.69**	**0.3390**	0.7721	**3.70**
CoDeX ($N'=10$)	-59.048	83.56	0.3183	**0.7770**	11.26
CoDeX ($N'=50$)	-57.170	105.13	0.3290	0.7667	45.40

Table 6. Compressibility guidance comparison between CoDe and CoDeX (N' represents the number of particles taken for the zero-order gradient estimations. High image dimensionality in latent space leads to noisy gradient estimates for CoDeX (larger N'), increasing computational cost (time) and diminishing performance compared to sampling-based methods like CoDe.

differentiable rewards, the gradient estimates become increasingly noisy, requiring a large number of particles to approximate them accurately. As a result, the performance degrades and falls short of the purely sampling-based approach in these scenarios.

7. Future Work

Despite significant advancements in conditional image generation, its inference time remains a considerable bottleneck compared to unconditional models, limiting real-world adoption. Future work should look into making this even more efficient and reduce the time close to uncondi-

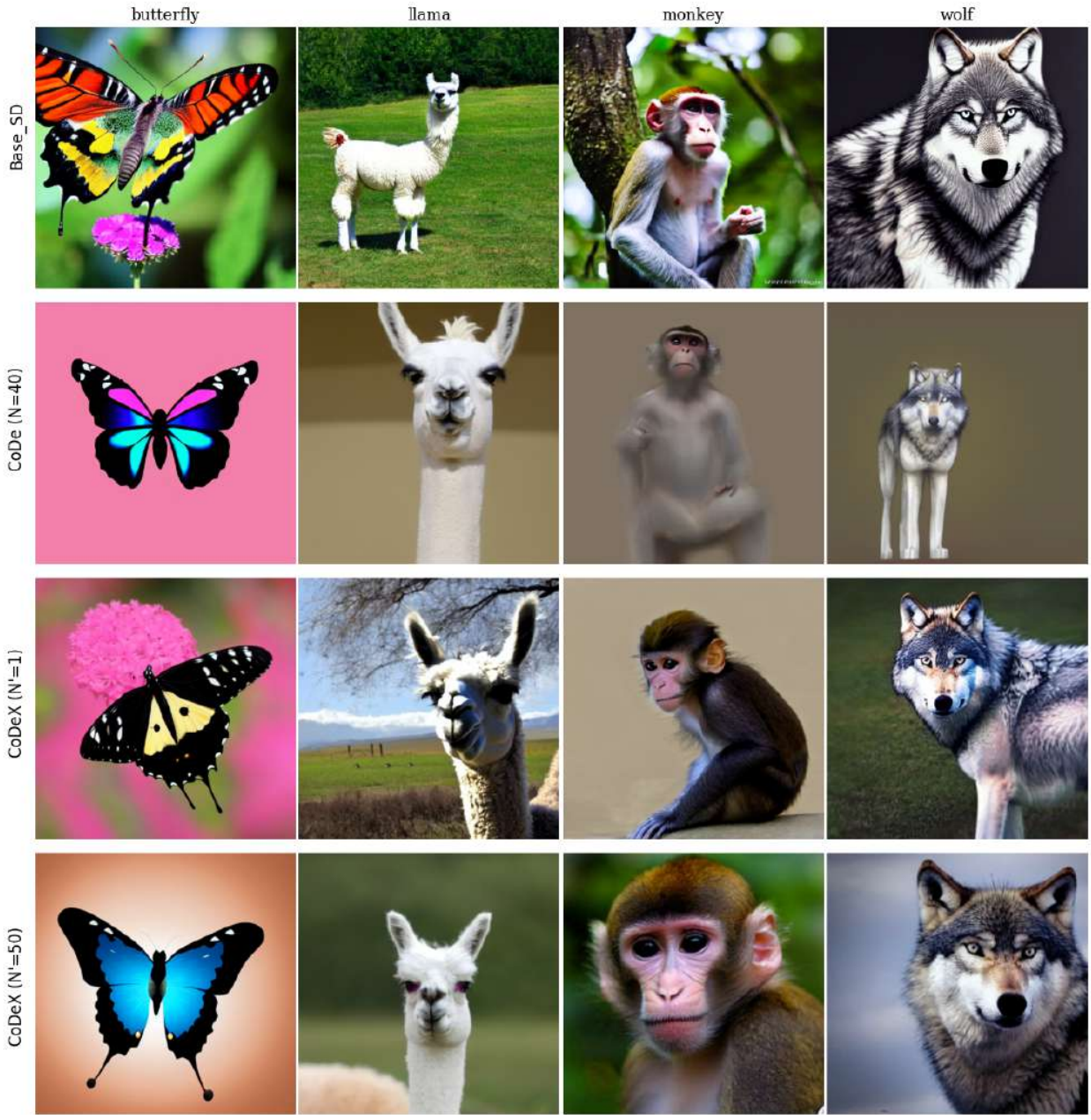

Figure 7. Qualitative samples comparing unconditional generation (Base SD) with compressible guidance methods (CoDe and CoDeX). Base SD generates detailed images, indicating low compressibility. CoDe blurs backgrounds and removes fine details (e.g., wolf hair). CoDeX even with $N' = 50$ struggles to remove subtle details due to the noisy gradient estimation.

tional diffusion. Additionally, due to time constraints, we did not explore the effects of dynamic temperature scheduling or adaptive particle allocation. Investigating these directions could yield further improvements in both generalization and efficiency, potentially enhancing reward outcomes across a broader range of tasks.

References

[1] Anuj Singh, Sayak Mukherjee, Ahmad Beirami, and Hadi Jamali-Rad. Code: Blockwise control for denoising diffusion models. *arXiv preprint arXiv:2502.00968*, 2025. 1, 2, 3, 4, 8

[2] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015. 1

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 3

[4] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1

[5] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 1, 2, 3, 4

[6] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021. 1

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 1, 2, 3, 4

[8] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1, 3, 5

[9] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 1

[10] Yuta Oshima, Masahiro Suzuki, Yutaka Matsuo, and Hiroki Furuta. Inference-time text-to-video alignment with diffusion latent beam search. *arXiv preprint arXiv:2501.19252*, 2025. 1, 3

[11] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020. 1

[12] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning*, pages 13916–13932. PMLR, 2023. 1

[13] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023. 1

[14] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *Advances in neural information processing systems*, 35:24240–24253, 2022. 1

[15] 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. 1

[16] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023. 2

[17] Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. 2023. 2, 3

[18] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023. 2, 3

[19] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Sergey Levine, and Tommaso Biancalani. Feedback efficient online fine-tuning of diffusion models. *arXiv preprint arXiv:2402.16359*, 2024. 2

[20] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022. 2, 3, 6

[21] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852, 2023. 2, 3, 6

[22] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23174–23184, 2023. 2, 3, 6

[23] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. *arXiv preprint arXiv:2311.16424*, 2023. 2, 3, 6

[24] Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Y Zou, and Stefano Ermon. Tfg: Unified training-free guidance for diffusion models. *Advances in Neural Information Processing Systems*, 37:22370–22417, 2024. 2, 3

[25] Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, et al. Derivative-free guidance

in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024. 2, 3, 4

[26] Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023. 2, 3

[27] Jie Ren, Xidong Feng, Bo Liu, Xuehai Pan, Yao Fu, Luo Mai, and Yaodong Yang. Torchopt: An efficient library for differentiable optimization. *arXiv preprint arXiv:2211.06934*, 2022. 2

[28] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2

[29] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013. 2

[30] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. 2

[31] Gustav Bredell, Kyriakos Flouris, Krishna Chaitanya, Ertunc Erdil, and Ender Konukoglu. Explicitly minimizing the blur error of variational autoencoders. *arXiv preprint arXiv:2304.05939*, 2023. 2

[32] Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al. Introvae: Introspective variational autoencoders for photographic image synthesis. *Advances in neural information processing systems*, 31, 2018. 2

[33] Sanchayan Vivekananthan. Comparative analysis of generative models: Enhancing image synthesis with vaes, gans, and stable diffusion. *arXiv preprint arXiv:2408.08751*, 2024. 2

[34] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *Advances in neural information processing systems*, 30, 2017. 2

[35] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 2

[36] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 3

[37] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019. 3

[38] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[39] Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. *arXiv preprint arXiv:2309.17400*, 2023. 3

[40] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023. 3, 6, 11

[41] Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*, 2024. 3

[42] Meihua Dang, Anikait Singh, Linqi Zhou, Stefano Ermon, and Jiaming Song. Personalized preference fine-tuning of diffusion models. *arXiv preprint arXiv:2501.06655*, 2025. 3

[43] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023. 3

[44] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pages 32483–32498. PMLR, 2023. 3

[45] Sunwoo Kim, Minkyu Kim, and Dongmin Park. Alignment without over-optimization: Training-free solution for diffusion models. *arXiv preprint arXiv:2501.05803*, 2025. 3

[46] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3

[47] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017. 3

[48] Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020. 3

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 4

[50] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022. 4

[51] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011. 4

[52] Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alex D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. 2024. *URL https://api. semanticscholar. org/CorpusID*, 266741736. 4

[53] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021. 4, 9

[54] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022. 5, 6, 7

[55] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[56] Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9307–9315, 2024. 6

[57] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 6

[58] Mao-Chuang Yeh, Shuai Tang, Anand Bhattad, Chuhang Zou, and David Forsyth. Improving style transfer with calibrated metrics. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3160–3168, 2020. 6

[59] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 6

[60] Christoph Schuhmann. Laion-aesthetics, 2025. Accessed: 2025-05-01. 6

[61] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:36652–36663, 2023. 6, 9

[62] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 8

[63] Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score: Better aligning text-to-image models with human preference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2096–2105, 2023. 9

3

Generative AI

Throughout the history, humans have learned to leverage machines to automate various tasks. We used machines to boost our limited physical strength, endurance, memory, and computing capability. With the emergence of **Generative AI**, this boundary is shifting rapidly. Generative models that can make text, images, code, and even strategic decisions are starting to help us think and be creative. As long as we can gather enough data to train an AI to act and make decisions like a person, we can now automate many tasks that used to be done by people [27]. This change has happened thanks to not only new model architectures like large language models (LLMs) or diffusion models but also better hardware power and the availability of much more training data that help with the needs of these complex, data-heavy systems.

Generative AI is reshaping the landscape of technology and creativity by enabling systems to autonomously generate text, images, audio, video, and other forms of content. This capability is revolutionizing content creation, problem-solving, and innovation across a wide range of domains [41]. Industries such as healthcare, logistics, entertainment, manufacturing, real estate, finance, and law are profoundly adopting generative AI for tasks like visual content generation, audio synthesis, video production, text and code creation, and intelligent data retrieval. These systems boost efficiency, creativity, and user control by also supporting back-and-forth collaboration and knowledge management through personalized and adaptive solutions.

As the amount and diversity of training data for generative models keeps growing, these models are showing more and more traits of foundational models, systems that can generalize widely, and display a deeper understanding of the world. This has been possible due to *emergence* and *homogenization*, which allow the behavior to be implicitly induced and build them across a wide range of applications respectively [9]. This increasing generality also brings challenges, especially in terms of controllability and alignment with user intent. In this work, we look into a subclass of these models (diffusion models) and attempt to make them more controllable, while our experiments are conducted in the image generation domain, the proposed approach is generalizable and can be easily adapted to other modalities.

4

Image Generation

AI Generated Content has revolutionized the way we retrieve and consume information. By automating the generation of text, images, and videos, AI enables quick, scalable content production tailored to individual needs and preferences. Generative AI aims to learn the underlying patterns of a dataset to generate new, original samples that resemble those drawn from the same distribution. Thus given enough samples, sampled from an underlying unknown distribution $p(x)$ the goal is to generate new samples z such that $z \sim p(x)$.

4.1. AI Based Image Generation

In recent years, image generation has made great progress as an important application area of artificial intelligence technology. Generative models based on deep learning have been able to create incredible visual content, which is widely used in various fields such as entertainment, scientific research, art, and commercial applications. The core of this technology is to simulate or reconstruct real-world image content [6, 29]. Early frameworks for image generation models were dominated by generative adversarial networks (GANs) [15] and variational autoencoders (VAEs) [25]. Nowadays, large-scale generative models based on diffusion models [52] have become state of the art, and have significantly improved the image fidelity and diversity.

4.2. Generative Models

In this section, we discuss the evolution of the generative models used for image generation. We discuss the previously used methods like VAEs and GANs and then discuss Diffusion Models. Figure 4.1 gives a high-level overview of these different models.

4.2.1. VAE: Variational Autoencoders

Variational autoencoders (VAE) [25, 44] are a class of generative models comprising of an encoder-decoder [10] structure that can reconstruct input data. Therefore they aim to generate new samples by learning the latent representation of the data. Unlike Autoencoders, VAE applies a probabilistic model to deep neural networks. Their major objective is to map the input to a data distribution rather than mapping it into a fixed vector [63]. Thus, sampling from the distribution increases the data diversity as well as the capability of noise handling.

The encoder maps each data element x_i to a latent space z_i that follows a Gaussian distribution (Figure 4.2).

$$\log q_\phi(z^{(i)}|x^{(i)}) = \log \mathcal{N}(z^{(i)}; \mu_\phi^{(i)}(x), \sigma_\phi^{2(i)}(x)I) \tag{4.1}$$

$$p(z) = \mathcal{N}(z; 0, I) \tag{4.2}$$

Thus, the encoder learns to predict the mean and the variance of the distribution. However, this process

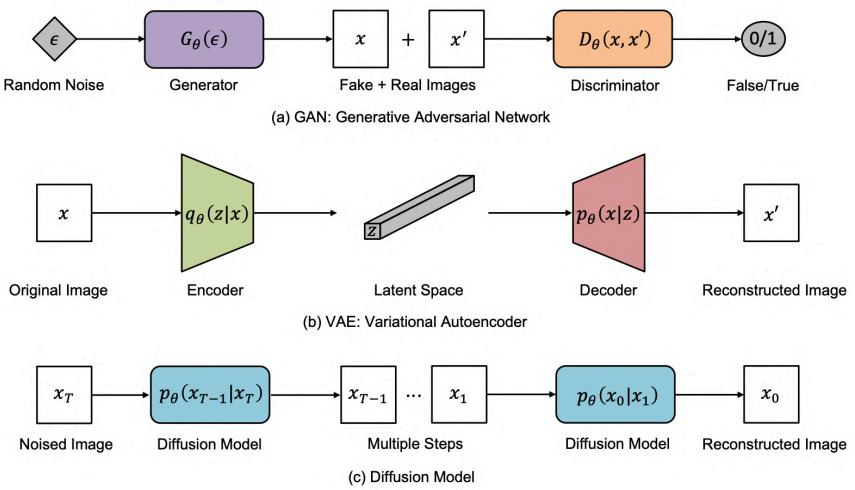

Figure 4.1: A sketch comparison of generative model architectures. (a) GAN model generates images using the generator that is trained by adversarial training. (b) VAE model reconstructs the image by maximizing the variational lower bound from the latent representation. (c) Diffusion model reconstructs image by denoising the Gaussian-noised image in an iterative process.

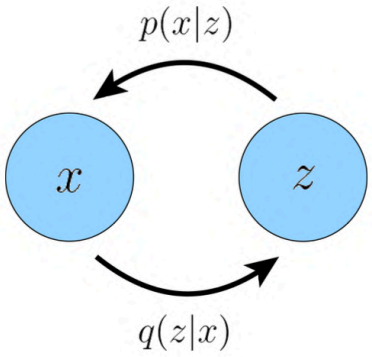

Figure 4.2: A Variational Autoencoder graphically represented. Here, encoder $q(z|x)$ defines a distribution over latent variables z for observations x, and $p(x|z)$ decodes latent variables into observations.

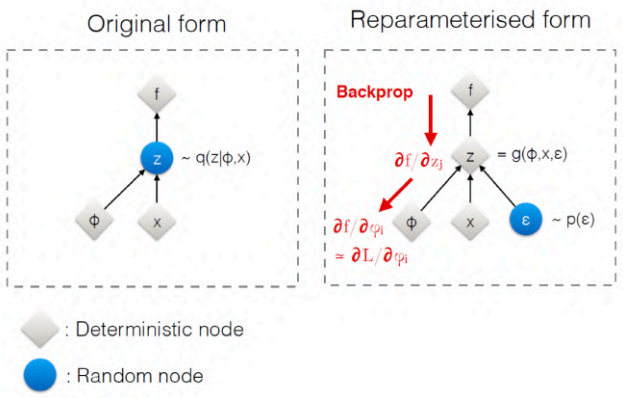

Figure 4.3: The reparameterization trick to make sampling from parameterized distribution differentiable [65].

is not differentiable as sampling is a stochastic process and therefore we cannot back-propagate the gradient. To make it trainable, the **reparameterization** trick (Figure 4.3) is used, which separates the sampling and the prediction of the mean and variance.

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \tag{4.3}$$

where ϵ is randomly sampled from normal Gaussian distribution. The decoder aims to reconstruct the input, which is trained by maximizing the probability of generating input data given the latent space variable $P(x|z)$. It is trained by maximizing the tractable lower bound of $P(x)$ [33]

$$L_b = \arg\max_{\phi,\theta} \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(x|z^{(l)}) - D_{KL}(q_\phi(z|x) \parallel p(z)). \tag{4.4}$$

VAEs are very useful because of their latent space structures and are thus used for reducing computational complexity by learning the representation in the latent space instead of the pixel space. However for image generation direct sampling from Gaussian distribution leads to blurred images, and information loss (due to them being projected on lower dimensional latent). Thus they have inferior performance to SoTA image generation architectures like GANs and Diffusion models. Figure 4.4 represents the overall structure of VAE for image generation.

4.2.2. GAN: Generative Adversarial Networks

Generative Adversarial Networks (GANs) [15] are inspired by the ideas of game theory. They have excellent performance in image generation tasks, are fast, and dominated the field in the early days. GANs introduced the idea of **adversarial training** and its model structure consists of two parts: the *Generator (G)* and the *Discriminator (D)*. Wherein the generator is responsible for generating fake data to trick the discriminator into believing that the generated data is close to the real data distribution. Whereas the discriminator's task is to determine whether the data is real or generated. Through adversarial training with **minmax algorithm**, the generator and discriminator learn together, thereby enabling the generator to generate more realistic data and the discriminator to more accurately distinguish between the real and the generated data.

The training objective either requires fixing the generator and taking gradient ascent on the discriminator (as it maximizes the value function):

$$V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))], \tag{4.5}$$

or fixing the discriminator and taking the gradient descent on the generator's loss:

$$L_G = \log(1 - D(G(z))). \tag{4.6}$$

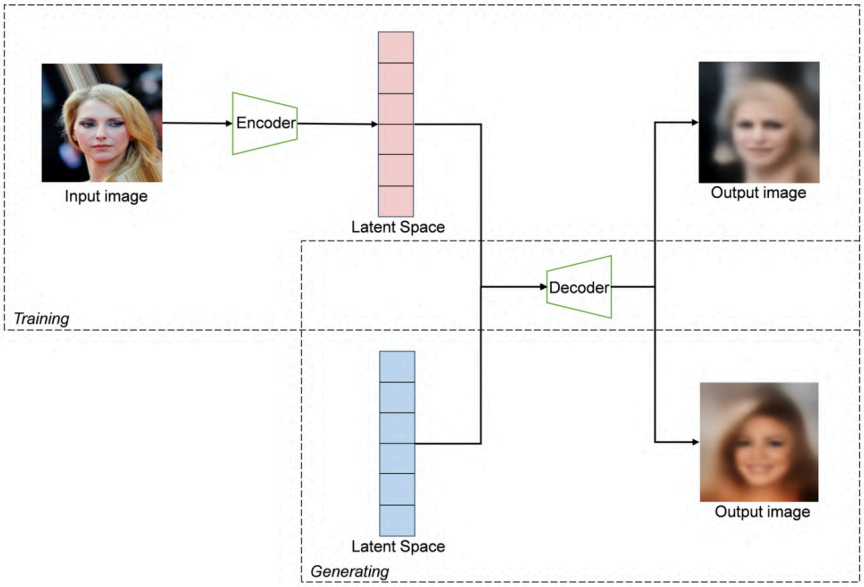

Figure 4.4: The overall model structure of VAE consists of an encoder, which maps the image to the latent space, and a decoder, which recovers the image from the latent space to the pixel space. During training, the VAE learns to recreate its inputs, while during generation, it samples from the latent distribution to produce novel images similar to the training data.

Thus it effectively becomes a min-max game with the objective:

$$\min_G \max_D V(D, G). \tag{4.7}$$

GANs can generate images with higher fidelity when compared to VAEs. They can also perform well on complex tasks like text-based image generation, image superresolution, object detection, etc. However, training a GAN is trickier, since there is no explicit representation of the data density $p(x)$ which leads to lower diversity. Furthermore, the need to balance the performance of the discriminator and the generator to avoid model collapse ends up generating the same image from the noise [48]. Figure 4.5 represents the overall architecture of GANs for image generation.

4.2.3. Diffusion Models

Sohl-Dickstein et al. [52] first proposed the application of diffusion models in the field of deep learning. The core idea of this model is to simulate the diffusion and restoration processes in physical systems to generate high-quality data samples.

They break down the process of converting the noise into the image into multiple iterative steps. Thus the neural network learns to reverse this process $p_\theta(x_{t-1}|x_t)$ since $q(x_t|x_{t-1})$ is known and deterministic. There are two popular diffusion model formulations, denoising diffusion probabilistic models (DDPM) [18] and score-based diffusion models [56].

DDPM is a fundamental type of diffusion model that uses discrete timesteps. It is based on Monte Carlo Markov Chain for both the forward and the reverse process. In the forward process which is deterministic, it goes from a clean sample at $t = 0$ to a complete random Gaussian noise at $t = T$ (for large T). This noise is scaled by a scheduler β_t and $\alpha_t = 1 - \beta_t$. The forward process is defined as:

$$q(x_{1:T}) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \tag{4.8}$$

$$q_\theta(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, \beta_t \mathrm{I}), \tag{4.9}$$

$$q_\theta(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_0, (1 - \bar{\alpha_t})\mathrm{I}). \tag{4.10}$$

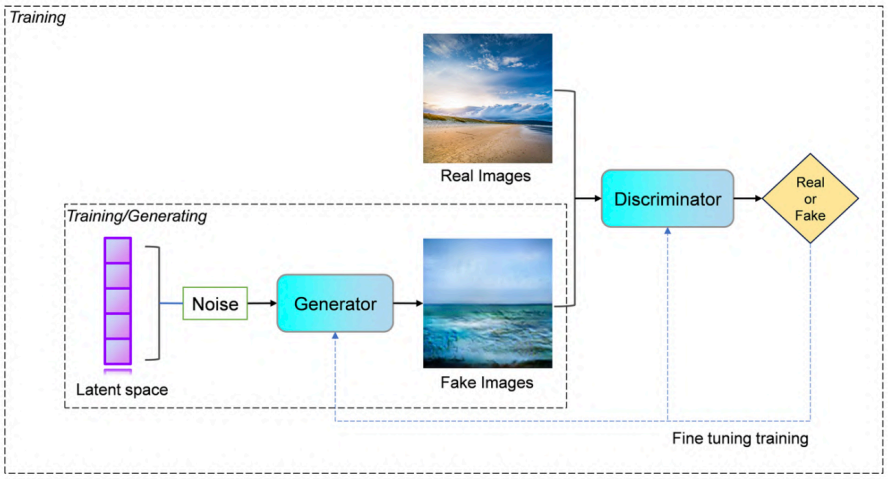

Figure 4.5: The overall structure of GAN consists of a generator and discriminator. The generator inputs a random noise vector to generate an image. The discriminator judges the real image and the generated image.

where $\bar{\alpha}_t = \prod_{t=1}^{T} \alpha_t$.

The reverse process is also a Markovian chain that samples the data from a learned Gaussian distribution at each timestep to remove this noise. Thus to generate a new image we just sample a random Gaussian noise and then iteratively apply the reverse process:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t, \Sigma_\theta(x_t, t)). \tag{4.11}$$

DDPM takes the variance as a fixed schedule thus $\Sigma_\theta(x_t, t) = \beta_t$ and calculates the mean value with the approximation of x_0 given x_t that is the posterior mean of the forward process, derived from Bayesian theorem :

$$\mu_\theta(x_t, t) = \tilde{\mu}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)). \tag{4.12}$$

Training is based on the optimization of the variational upper bound, which can be reparameterized as:

$$L = \mathbb{E}_{t,x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon, t)\right\|^2\right]. \tag{4.13}$$

The loss function is the MSE error between the predicted noise at a certain timestep and the noise that is sampled from the forward process. Therefore, DDPM uses a time-aware U-net [46] as the backbone network that takes the noise data x_t and the timestep t as the input.

Score-based diffusion models generalize the DDPM process using differential equations. Score matching refers to the use of a score function to approximate the gradient of the dense probability function $S(x_t, t) = \nabla_x \log p(x_t)$. In diffusion models, this is applied by using stochastic differential equations (SDEs). The forward process becomes:

$$dx = f_t(x)dt + g_t dw. \tag{4.14}$$

In the above equation $f_t(x)$ is vectored value function called the *drift* coefficient of x and g_t is a scalar function known as the *diffusion* coefficient of x, both of which are known. The reverse process is defined as:

$$dx = [f_t(x) - g_t^2 \nabla_x \log p_t(x)]dt + g(t)d\bar{\mathbf{w}}. \tag{4.15}$$

We learn the score function and the training objective is:

$$\theta^* = \arg\min_\theta \mathbb{E}_t\left\{\lambda(t)\mathbb{E}_{x(0)}\mathbb{E}_{x(t)|x(0)}[\left\|s_\theta(x(t), t) - \nabla_{x(t)} \log p_{0t}(x_t|x_0)\right\|_2^2]\right\}. \tag{4.16}$$

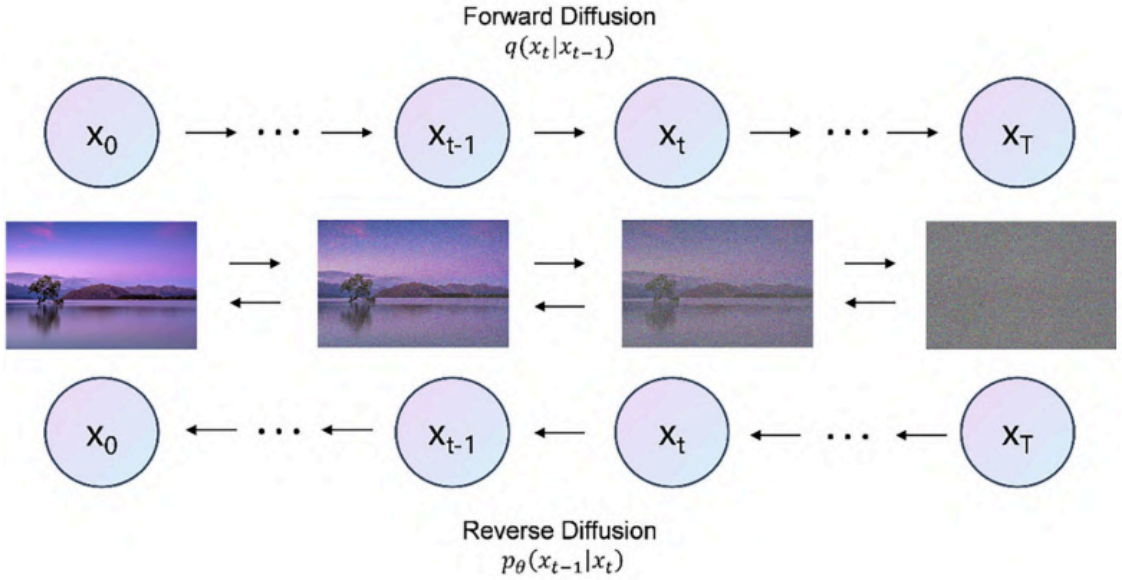

Figure 4.6: The overall structure of diffusion models; x_0 represents the true/clean natural image, x_T is pure Gaussian noise. $q(x_t, x_{t-1})$ is known and from this we learn $p_\theta(x_{t-1}|x_t)$.

Diffusion models show a much more stable training process as compared to GANs, preserving the high fidelity of the images and the diversity. However, it needs massive repeating steps to denoise the data in the correct direction. Figure 4.6 represents the overall structure of the diffusion models.

In the next section, we will dive deeper into the mathematical derivation for the diffusion model.

5

Diffusion Models

Diffusion models [52, 18, 54, 56] excel at modeling complex and realistic data distributions and demonstrate remarkable abilities across vision [55, 36, 45], small molecules [66, 67, 21], proteins [1, 62], audio [26, 32], 3D objects [34, 35] and a lot more. Diffusion models take inspiration from thermodynamics and propose an iterative Markov Chain process for image generation. They degrade the image gradually by adding Gaussian noise and learn to reverse this process, thereby generating images from sampled Gaussian noise. In this chapter, we aim to understand the underlying mathematical formulation behind the diffusion models and try to understand how they are trained. For a more deeper understanding, readers are encouraged to consult Luo et al. [33].

5.1. Mathematical Formulation

In this section, we will discuss the mathematical formulation behind training a diffusion model, the ELBO based objective, and the score matching formulation.

5.1.1. Evidence Lower Bound (ELBO)

Since it is a Markov Chain:

$$q(x_t|x_{t-1}) = q(x_t|x_{t-1}, x_0). \tag{5.1}$$

the extra conditioning term is superfluous due to the Markov property. Then with Bayes rule, we can rewrite this as:

$$q(x_t|x_{t-1}, x_0) = \frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)}. \tag{5.2}$$

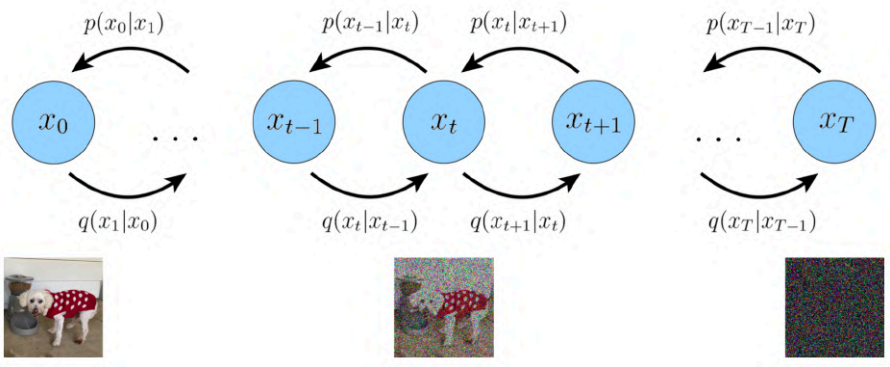

Figure 5.1: In diffusion model we know the noising process $q(x_t|x_{t-1})$ and need to learn $p(x_{t-1}|x_t)$. x_T is pure random Gaussian noise and x_0 is the clean image.

$$\log p(x) = \log \int p(x_{0:T})\, dx_{1:T} \tag{5.3}$$

$$= \log \int \frac{p(x_{0:T})q(x_{1:T}|x_0)}{q(x_{1:T}|x_0)}\, dx_{1:T} \qquad \text{(multiplying and dividing by } q(x_{1:T}|x_0)) \tag{5.4}$$

$$= \log \mathbb{E}_{q(x_{1:T}|x_0)} \left[\frac{p(x_{0:T})}{q(x_{1:T}|x_0)} \right] \tag{5.5}$$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_{0:T})}{q(x_{1:T}|x_0)} \right] \qquad \text{(using Jensen's Inequality)} \tag{5.6}$$

We get the ELBO term which is the lower bound thus, maximizing this term means we are maximizing $\log p(x)$. Hence, we use this as our proxy optimization term.

$$\log p(x) \geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^{T} q(x_t|x_{t-1})} \right] \tag{5.7}$$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) p_\theta(x_0|x_1) \prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)}{q(x_1|x_0) \prod_{t=2}^{T} q(x_t|x_{t-1})} \right] \tag{5.8}$$

Using Equation 5.1

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) p_\theta(x_0|x_1) \prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)}{q(x_1|x_0) \prod_{t=2}^{T} q(x_t|x_{t-1}, x_0)} \right] \tag{5.9}$$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1}, x_0)} \right] \tag{5.10}$$

Using Equation 5.2

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{\frac{q(x_{t-1}|x_t, x_0) q(x_t|x_0)}{q(x_{t-1}|x_0)}} \right] \tag{5.11}$$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{\frac{q(x_{t-1}|x_t, x_0) \cancel{q(x_t|x_0)}}{\cancel{q(x_{t-1}|x_0)}}} \right] \tag{5.12}$$

terms get canceled except for $\frac{q(x_T|x_0)}{q(x_1|x_0)}$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) p_\theta(x_0|x_1)}{\cancel{q(x_1|x_0)}} + \log \frac{\cancel{q(x_1|x_0)}}{q(x_T|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right] \tag{5.13}$$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) p_\theta(x_0|x_1)}{q(x_T|x_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right] \tag{5.14}$$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log p_\theta(x_0|x_1) \right] + \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T)}{q(x_T|x_0)} \right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right] \tag{5.15}$$

$$\geq \mathbb{E}_{q(x_1|x_0)} [\log p_\theta(x_0|x_1)] + \mathbb{E}_{q(x_T|x_0)} \left[\log \frac{p(x_T)}{q(x_T|x_0)} \right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_t, x_{t-1}|x_0)} \left[\log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1}, x_0)} \right] \tag{5.16}$$

$$\geq \underbrace{\mathbb{E}_{q(x_1|x_0)} [\log p_\theta(x_0|x_1)]}_{\text{reconstruction term}} - \underbrace{D_{KL}(q(x_T|x_0) \parallel p(x_T))}_{\text{prior matching term}}$$

$$-\sum_{t=2}^{T}\underbrace{\mathbb{E}_{q(x_t|x_0)}[D_{KL}(q(x_{t-1}|x_t,x_0)\parallel p_\theta(x_{t-1}|x_t))]}_{\text{denoising matching term}}. \tag{5.17}$$

Thus we have derived an interpretation for the ELBO, and each individual term here represents:

- $\mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)]$ can be interpreted as a reconstruction term, which predicts the log probability of the original data given the first step latent.
- $D_{KL}(q(x_T|x_0)\parallel p(x_T)$ The prior matching term represents how close the distribution of the final noisy input is to the standard Gaussian prior. It has no trainable parameters and for large enough T, can be approximated to zero.
- $\mathbb{E}_{q(x_t|x_0)}[D_{KL}(q(x_{t-1}|x_t,x_0)\parallel p_\theta(x_{t-1}|x_t))]$ the denoising matching term ensures that the ground truth denoising step $q(x_{t-1}|x_t,x_0)$ matches the transition step $p_\theta(x_{t-1}|x_t)$. This term dominates the reconstruction term and thus can be used as a proxy for optimizing.

5.1.2. Training Objective

Thus the objective becomes (as ELBO needs to be maximized and thus the negative of this term needs to be maximized which means it should be minimized):

$$\arg\min_\theta \mathbb{E}_{t\sim\{2,T\}}\left[\mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t,x_0)\parallel p_\theta(x_{t-1}|x_t))\right]\right]. \tag{5.18}$$

Thus we model the $p_\theta(x_{t-1}|x_t)$ also as a gaussian, with the same variance as $q(x_{t-1}|x_t,x_0)$ i.e. (refer Section 5.1.3):

$$\begin{aligned}
&\arg\min_\theta D_{KL}(q(x_{t-1}|x_t,x_0)\parallel p_\theta(x_{t-1}|x_t))\\
=&\arg\min_\theta D_{KL}(\mathcal{N}(x_{t-1};\mu_q,\Sigma_q(t))\parallel\mathcal{N}(x_{t-1};\mu_\theta,\Sigma_q(t)))\\
=&\arg\min_\theta \frac{1}{2\sigma_q^2(t)}\left[\|\mu_\theta(x_t,t)-\mu_q(x_t,x_0)\|_2^2\right].
\end{aligned} \tag{5.19}$$

This can be optimized in two major ways where the neural network learns different objectives which are both interrelated:

1. $\hat{\epsilon}_\theta(x_t,t)$ is a neural network that learns to predict the source noise $\epsilon_0\sim\mathcal{N}(\epsilon;0,\mathbf{I})$ which basically determines x_t from x_0. Learning to predict the noise is equivalent to learning to predict the original image x_0, but some prior works have found that predicting the noise resulted in better performance [18, 47]:

$$x_t=\sqrt{\bar{\alpha}_t}x_0+\sqrt{1-\bar{\alpha}_t}\epsilon_0. \tag{5.20}$$

$$\mu_q(x_t,x_0)=\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t} \tag{5.21}$$

Substituting eq 5.20 for x_0 in eq 5.2 (from subsection 5.1.3 eq 5.30) we get:

$$\mu_q(x_t,\epsilon_0)=\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\frac{x_t-\sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} \tag{5.22}$$

$$=\frac{1}{\sqrt{\alpha_t}}x_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_0. \tag{5.23}$$

Therefore, we approximate the denoising transition mean $\mu_\theta(x_t,t)$ as:

$$\mu_\theta(x_t,t)=\frac{1}{\sqrt{\alpha_t}}x_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\epsilon}_\theta(x_t,t). \tag{5.24}$$

Substituting eq 5.23, 5.24 in eq 5.19 we get the training objective for this formulation as:

$$\arg\min_{\theta} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \left[\|\epsilon_0 - \hat{\epsilon}_\theta(x_t, t)\|_2^2 \right]. \tag{5.25}$$

2. $s_\theta(x_t, t)$ is a neural network that learns to predict the score function $\nabla_{x_t} \log p(x_t)$ which is the gradient of x_t in the data space, for any random noise level t.

$$\sqrt{\bar{\alpha}_t} x_0 = x_t + (1-\bar{\alpha}_t) \nabla_{x_t} \log p(x_t) \tag{5.26}$$

Following similar steps as before the training objective becomes:

$$\arg\min_{\theta} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{\alpha_t} \left[\|s_\theta(x_t, t) - \nabla_{x_t} \log p(x_t)\|_2^2 \right]. \tag{5.27}$$

Thus we can either train the neural network to predict the noise or predict the score function. Generally, diffusion models are trained by randomly sampling timesteps t and minimizing the norm of the prediction with the ground truth target.

5.1.3. Determining $q(x_{t-1}|x_t, x_0)$

From the Bayes rule, we get that (after skipping some steps for detailed calculation refer to [33]):

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0) q(x_{t-1}|x_0)}{q(x_t|x_0)} \tag{5.28}$$

$$= \frac{\mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)\mathbf{I}) \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}} x_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(x_t; \sqrt{\bar{\alpha}} x_0, (1-\bar{\alpha}_t)\mathbf{I})} \tag{5.29}$$

$$\propto \mathcal{N}(x_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1}) x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t) x_0}{1-\bar{\alpha}_t}}_{\mu_q(x_t, x_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}_{\Sigma_q(t) = \sigma_q^2(t)}) \tag{5.30}$$

5.2. DDPM and DDIM

5.2.1. DDPM

A diffusion model trained using the Markov Chain and predicting the noise was proposed by DDPM [18]. The training and the sampling algorithms are shown in the Algorithm 1 and Algorithm 2.

Algorithm 1 Training

repeat
  $x_0 \sim q(x_0)$
  $t \sim$ Uniform$(\{1, \ldots, T\})$
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  Take gradient descent step on
  $\nabla_\theta \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t) \right\|^2$
until converged

Algorithm 2 Sampling

$x_T \sim \mathcal{N}(0, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  if $t > 1$ $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$,else $\mathbf{z} = 0$
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t \mathbf{z}$
end for
return x_0

Generating a high-fidelity image requires traversing the entire reverse diffusion chain, which typically comprises a large number of timesteps ($T \sim 1000$). This sequential, step-by-step nature of sampling leads to slow inference times and high computational expense, as each denoising step requires a forward pass through the neural network. This dependency on numerous iterations to ensure high sample quality poses a challenge for real-time applications and large-scale data generation. This computational and inefficient bottleneck served as a primary motivation for further research, leading to the development of methods that could accelerate the sampling process without compromising generation quality, notably through the introduction of Denoising Diffusion Implicit Models (DDIMs) [54].

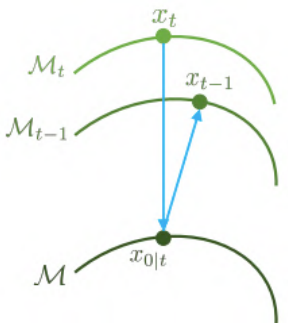

Figure 5.2: DDIM sampling explained, $\mathcal{M}_t$ represents the manifold for t timestep

5.2.2. DDIM

DDIM aimed to accelerate the generative process without sacrificing the high sample quality achieved by DDPMs. The key innovation through which they achieve this lies in generalizing the forward diffusion process to a **non-Markovian** one. This means that in DDPM's the forward process requires a strict dependence of x_t only on x_{t-1} DDIM allows it to depend on the initial data point x_0 aswell. Then formulating the reverse process deterministically as a solution to an ODE (Ordinary Differential Equation) allows for jumping in the denoising trajectory. Thus can directly estimate a less noisy sample x_{t-k} from x_t skipping $t-k-1$ steps. Thus by choosing a subset of total timesteps T (around 50 instead of 1000). DDIM can drastically reduce inference time, often by orders of magnitude ($10-50\times$ faster), while maintaining comparable or even superior sample quality in many cases.

Crucially, the training objective of DDIM remains identical to that of DDPM, thus we can use the neural network trained using DDPM and sample much faster from this. Figure 5.2 explains how it works thus instead of going directly from x_t to x_{t-1} in DDIM we go from x_t to $x_{0|t}$ using the Tweedies formula (predicted x_0) and then add noise to this predicted x_0 to go till x_{t-1}.

$$x_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta(x_t, t)}{x_t} \right)}_{\text{predicted } x_0} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\epsilon_\theta(x_t, t)}_{\text{direction pointing to } x_t} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}} \tag{5.31}$$

5.3. Latent Diffusion Models

Building on the advancements of DDPM and DDIM, a further critical step towards making diffusion models practical for high-resolution image generation was the introduction of Latent Diffusion Models (LDMs). The fundamental challenge with previous diffusion models, especially for large images, was that the diffusion process (both forward and reverse) operated directly on the raw pixel data. This meant that for a high-resolution image (e.g., 1024x1024 pixels), the neural network had to process an enormous number of dimensions at every single step, leading to extremely high computational costs for both training and inference.

LDMs address this computational burden by performing the diffusion process not in the high-dimensional pixel space, but in a much smaller, perceptually meaningful latent space. This is achieved by leveraging a pre-trained autoencoder, which is a neural network composed of two parts: an encoder and a decoder. The encoder takes a high-resolution image and compresses it into a lower-dimensional **latent representation**, capturing the most essential visual information while discarding redundant details. Whereas, the decoder learns to reconstruct the original high-resolution image from this compressed latent code.

Thus using the autoencoder we train and sample from the diffusion models in the latent space leading to a substantial reduction in computational requirements. This still retains enough information to reconstruct a high-quality image as the autoencoder is trained to preserve perceptual fidelity. During inference, a random noise vector is sampled from the latent space, and the diffusion model performs

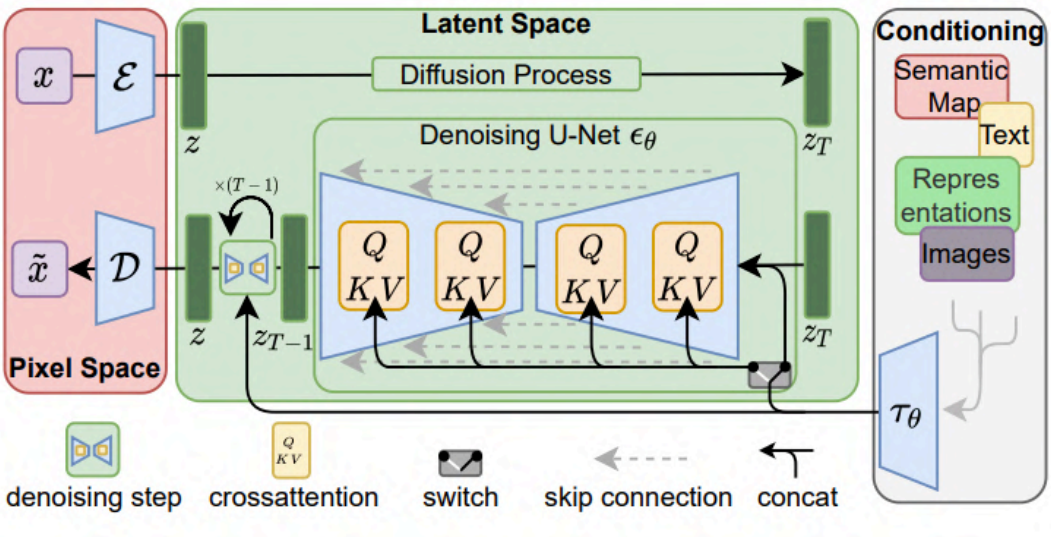

Figure 5.3: Stable Diffusion architecture.

its denoising steps on this low-dimensional representation. Once the denoising is complete, the resulting clean latent code is passed through the pre-trained decoder, which reconstructs the final, high-resolution image. This separation of concerns allows for much faster training and sampling of diffusion models, even on consumer-grade GPUs, making them far more accessible and enabling the creation of powerful models like Stable Diffusion. Furthermore, the latent space often allows for easier manipulation and conditioning (e.g., text-to-image generation) by connecting external information to this compressed representation via mechanisms like cross-attention.

This architectural efficiency, coupled with the ability to condition the generation process, led to the development of highly influential models. The subsequent section will delve into how these principles are basically applied in Stable Diffusion, a widely recognized example of a Latent Diffusion Model that has revolutionized text-to-image synthesis and is widely used.

5.3.1. Stable Diffusion

Stable Diffusion [45] is a **text-conditional latent diffusion model** which allows it to drastically reduce the computational resources required for both training and inference, allowing for high-quality image generation accessible on consumer-grade hardware like standard GPUs.

Text-conditional generation allows users to exert fine-grained control over the generated content by simply providing a textual prompt. Thus it provides users the ability to control the image generation using text prompts. The remarkable diversity and quality of Stable Diffusion is due to its extensive training. The model was mostly trained on massive datasets of image-text pairs, most notably subsets of the LAION-5B dataset [49]. LAION-5B is a publicly available dataset comprising of billions of image-text pairs scraped from the internet. This huge amount of diverse data allowed Stable Diffusion to learn a rich understanding of the relationship between visual concepts and their textual descriptions, covering a vast variety of objects, scenes, styles, and artistic movements. Thus, this comprehensive training enables it to generate a wide spectrum of images, from photorealistic scenes to abstract art, all guided by textual input. This large dataset allows it to serve as a foundational model for image generation.

The core components of Stable Diffusion, as depicted in the architectural diagram (Figure 5.3), work to translate textual prompts into stunning visual imagery. The architecture has two major parts:

1. **VAE** The VAEs are an important part of the stable diffusion architecture as using the encoder ($\mathcal{E}$) and the decoder ($\mathcal{D}$) they train the diffusion mode and perform the inference in the latent space (z) instead of doing it in the pixel space (x) which saves a lot of compute and time.

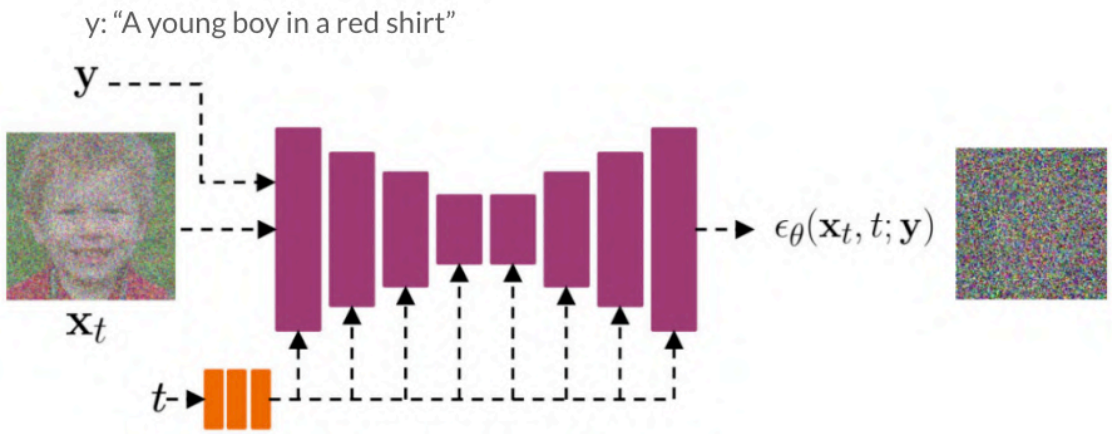

Figure 5.4: Stable Diffusion (Text conditioning) training

For stable diffusion models like SD 1.5, and SD 2.1 the VAE used is `kl-f8` (Kullback-Leibler f8 denotes 8x downsampling and upsampling which uses KL divergence regularization during the training). Thus for example a $512 \times 512 \times 3$ image is turned to $4 \times (\frac{512}{8}) \times (\frac{512}{8})$ that is $4 \times 64 \times 64$ in the latent space.

For the newer and larger models like SDXL [39], they use a retrained-from-scratch VAE which is specifically optimized for larger resolution and refined latent space characteristics.

2. **Text-Conditioned UNet** text conditioning allows users to guide the image generation process with natural language, transforming abstract noise into specific visual concepts. The mechanism for this conditioning primarily revolves around two key components: **a Text Encoder** and **Cross-Attention** layers embedded within the U-Net architecture.

   (a) **Text Encoder** Using pre-trained CLIP (Contrastive Language-Image Pre-training) [42] Text Encoder the text is converted into a numerical form which is machine understandable. CLIP itself was trained on a massive dataset of image-text pairs from the internet. Its objective was to learn a shared, multimodal embedding space where similar concepts, whether expressed in text or image, are mapped close together. The CLIP Text Encoder breaks it down into individual tokens (words or sub-word units). Each token is then converted into a high-dimensional vector (token embedding). These embeddings are not just arbitrary numbers, they capture the semantic meaning of each word in the context of a vast language and image understanding. The output of the Text Encoder is a sequence of these token embeddings, which essentially represents the numerical representation of the user's prompt.

   (b) **U-Net with cross attention** The UNet is trained to predict text conditioned $\epsilon_\theta(x_t, t; y)$ instead of predicting $\epsilon_\theta(x_t, t)$ where y represents the conditioning (text/prompt) and is trained similarly (Figure 5.4). The U-Net itself is a convolutional neural network designed for image-to-image translation, operating in the VAE's latent space. Its U-shape comes from its symmetrical encoder-decoder structure with skip connections, which helps preserve fine details during the denoising process. The U-Net in Stable Diffusion consists of multiple resolution blocks. As the noisy latent image passes through the encoder path, its spatial dimensions are progressively downsampled, capturing coarser features. The decoder path then upsamples these features to reconstruct the denoised latent. Skip connections connect corresponding resolution levels in the encoder and decoder to ensure information flow. This is then combined with a **cross-attention** mechanism to allow text conditioning.

   At various points within the U-Net, typically after convolutional blocks at different resolution levels, cross-attention layers are strategically inserted. These layers are the points where the text embeddings from the text encoder interact with the latent image features being processed by the U-Net. **Queries(Q)** are derived from the current latent image features within the U-Net. They represent *what the U-Net is currently looking for* in the image. **Keys(K)**

and **Values(V)** are derived from the text embeddings produced by the CLIP text encoder and represent the *available information*. The attention calculates a similarity score between the image feature queries (Q) and the text embedding keys (K). This score determines how much attention different parts of the image features should pay to different words in the prompt. The resulting attention weights are then used to combine the text embedding values (V), effectively injecting the semantic guidance from the text into the U-Net's image processing.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V, \tag{5.32}$$

$$Q = W_Q^{(i)} \cdot \varphi_i(z_t), K = W_K^{(i)} \cdot \tau_\theta(y), V = W_V^{(i)} \cdot \tau_\theta(y), \tag{5.33}$$

where $\varphi_i(z_t) \in \mathbb{R}^{N \times d_\epsilon^i}$ represents a flattened intermediate representation of the UNet implementing ϵ_θ and $W_V^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i}$, $W_Q^{(i)} \in \mathbb{R}^{d \times d_\tau^i}$ and $W_K^{(i)} \in \mathbb{R}^{d \times d_\tau^i}$ are learnable projection matrices.

Thus the conditional LDM is learned using:

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x), y, \epsilon \sim \mathcal{N}(0,1), t}\left[\| \epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y)) \|_2^2\right], \tag{5.34}$$

which is jointly optimized for both τ_θ and ϵ_θ [45].

On top of all this to enhance the influence of the text prompt and improve image quality, Stable Diffusion also uses **Classifier-Free Guidance** [19] during inference (explained in detail in Section 6.1.2) which involves running the U-Net for each denoising step twice once with text and once with an empty prompt. The final prediction used is then obtained by subtracting the null prompt noise from the conditioned one to push the model more strongly toward generating content with the specific text prompt, leading to more faithful and striking results. The guidance scale parameter allows users to control the strength of this conditioning.

6

Conditional Image Generation

Since image generation models are trained on large and diverse sets of images their training objectives don't always align with the human expectations and intentions [31]. For example, images generated by pre-trained T2I (text-to-image) models, like stable diffusion, may not accurately *align* to text prompts or may have a low aesthetic quality [7, 14, 28]. To address this and give users more control over the generation beyond simply modeling the training data distribution recent works have begun to optimize pre-trained diffusion models directly for human-preferred properties [31]. This idea is inspired by similar prior developments in the LLMs to align them with human intentions to enhance their capabilities. Common popular powerful large language models (LLMs) like GPT-4 [2] after the pretraining stage are fine-tuned to follow instructions and align well with human preferences. This post-training process usually involves supervised fine-tuning (SFT) followed by alignment with human feedback using techniques like reinforcement learning from human feedback (RLHF) [2, 38] and direct preference optimization (DPO) [43, 16].

These methods can be broadly categorized into two major categories training-based and training-free (inference-time alignment) techniques. Wherein the training-based methods involve either training a new model from scratch for this conditioning on paired examples of images and conditions like stable diffusion or fine-tuning the pre-trained models to *align* it with the new conditioning signals. Training-free alignment methods on the other hand utilize an off-the-shelf reward function to approximate the performance of the image and the conditioning signal and use this during the denoising to generate images offering better rewards and thereby better alignment to the conditioning signal.

6.1. Alignment Techniques

Recent research has now shifted towards leveraging the powerful representations learned by foundational models, which serve as a robust starting point. This saves the cost and time required for full model training by allowing efficient alignment to novel conditioning signals.

6.1.1. Finetuning Based Methods

These methods use the pre-trained foundational model and fine-tune it for conditioning using Supervised Fine-Tunning (SFT), Reinforcement Learning (RLHF) [11]. In RLHF the base policy remains the foundational model which is then optimized towards maximizing the reward provided by the reward model (which is trained to reflect human preferences) [31].

Thus the base policy p_θ is fine-tuned to maximize reward $r_\phi(c, x)$ while minimizing the KL divergence D_{KL} between the current policy p_θ and the initial reference policy p_{ref}. Basically, we want to still stay close to prior distribution (to avoid reward hacking) and get a higher reward that matches the human preference.

$$\max_{p_\theta} \mathbb{E}_{c\sim p, x\sim p_\theta(x|c)} \left[r_\phi(c, x) - \beta D_{KL}(p_\theta(x|c) \parallel p_{ref}(x|c)) \right], \tag{6.1}$$

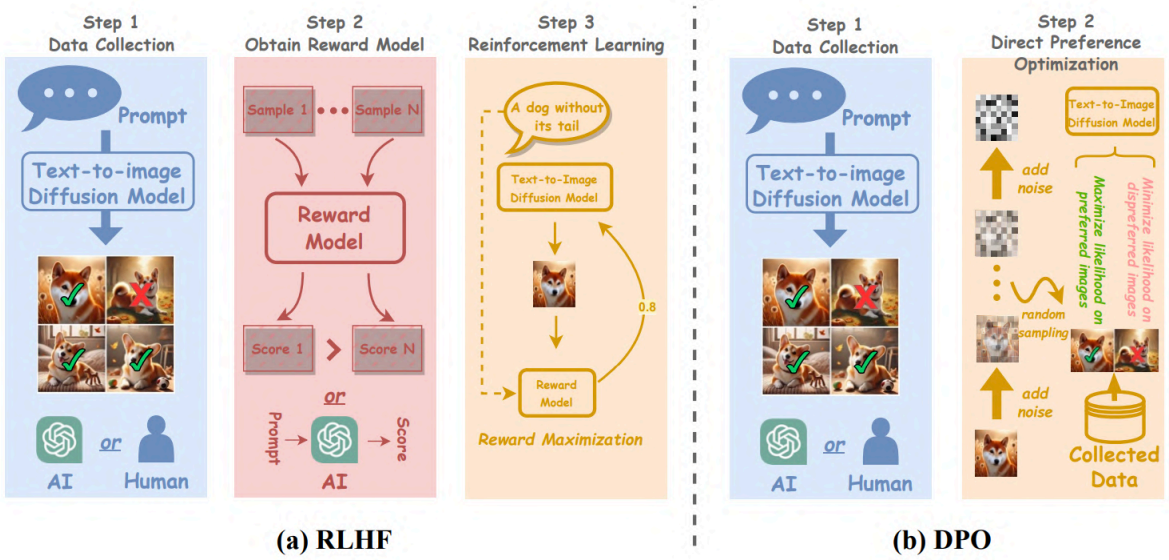

Figure 6.1: In RLHF we need to explicitly train a reward model that captures human preference which is then used to fine-tune the foundational model whereas in DPO using just a preferred and dispreferred output during the training of the model to align it towards human preference without requiring any explicit reward model [32].

β controls the extent of deviation from the prior [57]. Optimizing this objective is equivalent to maximizing the following KL-shaped reward in expectation:

$$r_\phi(c, x) - \beta \log \frac{p_\theta(x|c)}{p_{ref}(x|c)}. \tag{6.2}$$

Some ways of doing this are Proximal Policy Optimization (PPO) [50], Direct Preference Optimization (DPO) [43, 61], Odds ratio preference optimization (ORPO) [20], Kahneman-Tversky optimization (KTO) [58, 13, 30], and preference ranking optimization (PRO) [53], etc. To get a better understanding of these methods refer to their respective papers or this survey paper [31].

These methods work well for conditioning or aligning to a new control signal but require significant data and computation. Furthermore, for every new conditioning, a new model needs to be fine-tuned which is not very scalable. Thus we look into inference-time guidance models which use pre-trained off-the-shelf reward models and guide the images during the denoising to obtain higher rewards.

6.1.2. Inference-Time Guidance: Classifier Based Guidance and Classifier-Free Guidance

Inference-time guidance allows users to align the pre-trained foundational model to the downstream task in a training-free manner by using off-the-shelf reward functions. For more details readers are encouraged to refer to [60]. This is done by either modifying the denoising step to explicitly push the denoising towards better rewards or by sampling from it multiple times and then selecting the one expected to give the highest reward. We will now discuss some popular approaches to achieve this - Classifier Based Guidance (Gradient Guidance) [12, 56], Classifier Free Guidance [19] and Sampling Based Guidance [51].

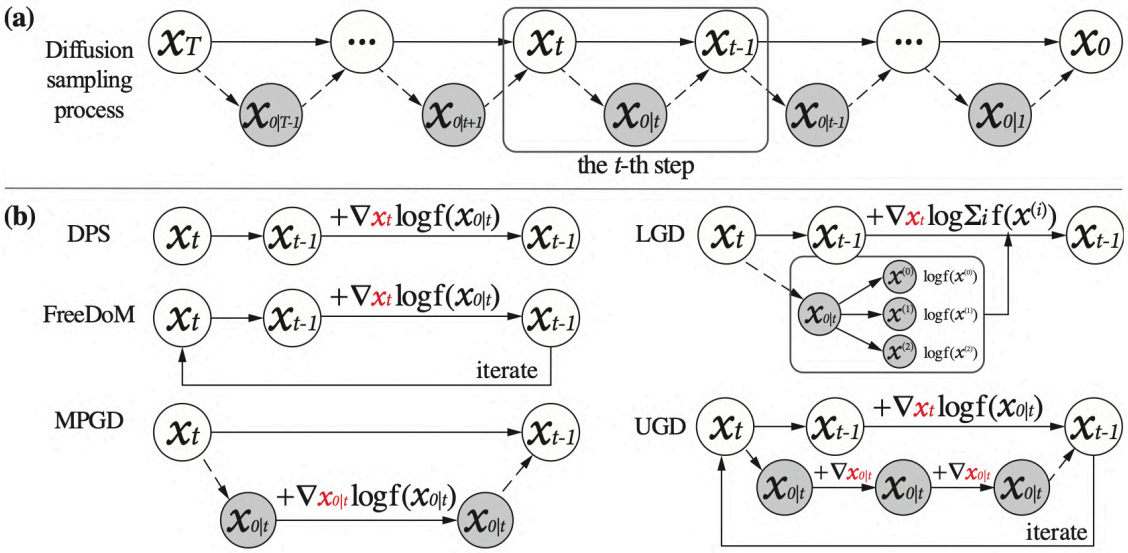

Figure 6.2: (a) shows the denoising process for DDIM scheduler, (b) Illustration of different most popular training-free gradient guidance algorithms. [68]

Classifier Based Guidance (Gradient Guidance)

For conditional generation[12], the goal is to sample from $p(x|y)$ instead of $p(x)$. Thus wrt to the score-based formulation we want the $\nabla_{x_t} \log p(x_t|y)$. Using the Bayes Rule [33]: ∇ represents ∇_{x_t}

$$\begin{aligned} \nabla \log p(x_t|y) &= \nabla \log \left(\frac{p(x_t)p(y|x_t)}{p(y)} \right) \\ &= \nabla \log p(x_t) + \nabla \log p(y|x_t) - \nabla \log p(y) \\ &= \underbrace{\nabla \log p(x_t)}_{\text{unconditional score}} + \underbrace{\nabla \log p(y|x_t)}_{\text{adversarial gradient}} \end{aligned} \tag{6.3}$$

In the above Eq. 6.3, we know the first-term unconditional score from the foundational diffusion model and we get the second term using an off-the-shelf reward model. Since we are utilizing the predicted clean image using the Tweedies formula for the gradient estimation this is a bit noisy. There are many different variations of circumventing this which either do this repeatedly [4, 69] or take the gradients wrt to the predicted image to stay on the manifold [17].

For comparing with gradient guidance we use the methods `MPGD` [17], `FreeDoM` [69] and `UG` [4]. For more details about these methods and their subtle details refer to their respective papers or `TFG` [68] which explains the differences very well. A general view of all the popular gradient guidance in contrast to DDIM sampling is shown in Figure 6.2.

`FreeDoM` [69] at the cost of additional compute introduces a *recurrent strategy* (also called time-travel strategy) which iteratively denoises x_{t-1} from x_t and adds noise to x_{t-1} to regenerate x_t back and forth. During each of these steps it does move it towards higher rewards using the gradients. In the algorithm 3 we can see that we do gradient ascent in line 12 and do it multiple times (r_t) and then again go from x_{t-1} to x_t (line 16).

`MPGD` [17] aims to keep the image on the manifold even after the gradient and does it by taking the gradient wrt to $x_{0|t}$ ($\nabla_{x_{0|t}} \log r(x_{0|t}, c)$) instead of x_t ($\nabla_{x_t} \log r(x_{0|t}, c)$) and adds it to the clean image and then goes to the timestep x_{t-1}. In algorithm 4 for stable diffusion we can see we predict the clean sample (line 5) using the Tweedies estimate and to this, we add the gradient (line 7) and then return to the $t-1$ timestep by adding noise.

`UG` [4] on top of the recurrent strategy solves the backward optimization problem $\Delta_0 = \arg\max_{\Delta} f(x_{0|t} + \Delta)$ which guides both $x_{0|t}$ and x_t simultaneously. Algorithm 5 shows the recurrent strategy for k steps

Algorithm 3 `FreeDoM` [69]

Require: condition c, unconditional score estimator $s(\cdot, t)$, time-independent distance measuring function $\mathcal{D}_\theta(\boldsymbol{c}, \cdot)$, pre-defined parameters β_t, $\bar{\alpha}_t$, learning rate ρ_t, and repeat times $\{r_1, \cdots, r_T\}$

Sample $\boldsymbol{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$
for $t = T$ to 1 **do**
  for $i = r_t$ to 1 **do**
    if $t > 1$ **then**
      $\boldsymbol{\epsilon}_1 \sim \mathcal{N}(0, \boldsymbol{I})$
    else
      $\boldsymbol{\epsilon}_1 = 0$
    end if
    $\boldsymbol{x}_{t-1} = \left(1 + \frac{1}{2}\beta_t\right)\boldsymbol{x}_t + \beta_t s(\boldsymbol{x}_t, t) + \sqrt{\beta_t}\boldsymbol{\epsilon}_1$
    $\boldsymbol{x}_{0|t} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{x}_t + (1 - \bar{\alpha}_t)s(\boldsymbol{x}_t, t)\right)$
    $\boldsymbol{g}_t = \nabla_{\boldsymbol{x}_t}\mathcal{D}_\theta(\boldsymbol{c}, \boldsymbol{x}_{0|t}(\boldsymbol{x}_t))$
    $\boldsymbol{x}_{t-1} = \boldsymbol{x}_{t-1} - \rho_t \boldsymbol{g}_t$
    if $i > 1$ **then**
      $\boldsymbol{\epsilon}_2 \sim \mathcal{N}(0, \boldsymbol{I})$
      $\boldsymbol{x}_t = \sqrt{1 - \beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_2$
    end if
  end for
end for
return $\boldsymbol{x}_0$

Algorithm 4 `MPGD` for latent diffusion models [17]

Sample $\boldsymbol{z}_T \sim \mathcal{N}(0, \boldsymbol{I})$
for $t = T, \ldots, 1$ **do**
  Sample $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \boldsymbol{I})$
  $\boldsymbol{z}_{0|t} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{z}_t - \sqrt{1 - \bar{\alpha}_t}\,\epsilon_\theta(\boldsymbol{z}_t, t)\right)$
  $\boldsymbol{z}_{0|t} = \boldsymbol{z}_{0|t} - c_t \nabla_{\boldsymbol{z}_{0|t}}\mathcal{L}\left(D(\boldsymbol{z}_{0|t}); \boldsymbol{y}\right)$
  $\boldsymbol{z}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{z}_{0|t}$
    $+\sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\,\epsilon_\theta(\boldsymbol{z}_t, t) + \sigma_t \boldsymbol{\epsilon}_t$
end for
return $\boldsymbol{x}_0 = D(\boldsymbol{z}_0)$

Algorithm 5 UG [4]

Parameter: Recurrent steps k, gradient steps m for backward guidance and guidance strength $s(t)$
Required: $\boldsymbol{z}_T$ sampled from $\mathcal{N}(0, \boldsymbol{I})$, diffusion model ϵ_θ, noise scales $\{\bar{\alpha}_t\}_{t=1}^T$, guidance function $\mathbf{f}$, loss function ℓ, and prompt c

1: **for** $t = T, T-1, \ldots, 1$ **do**
2: **for** $n = 1, 2, \ldots, k$ **do**
3: Calculate $\hat{\boldsymbol{z}}_0$ as

$$\hat{\boldsymbol{z}}_0 = \frac{\boldsymbol{z}_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(\boldsymbol{z}_t, t)}{\sqrt{\bar{\alpha}_t}}$$

4: $\hat{\epsilon}_\theta(\boldsymbol{z}_t, t) = \epsilon_\theta(\boldsymbol{z}_t, t) + s(t) \cdot \nabla_{\boldsymbol{z}_t} \ell(c, \mathbf{f}(\hat{\boldsymbol{z}}_0))$
5: **if** $m > 0$ **then**
6: Calculate $\Delta \boldsymbol{z}_0$ by minimizing $\ell(c, \mathbf{f}(\hat{\boldsymbol{z}}_0 + \Delta))$ with m steps of gradient descent
7: Perform backward universal guidance:

$$\hat{\epsilon}_\theta \leftarrow \hat{\epsilon}_\theta - \sqrt{\bar{\alpha}_t/(1-\bar{\alpha}_t)}\,\Delta \boldsymbol{z}_0$$

8: **end if**
9: $\boldsymbol{z}_{t-1} \leftarrow S(\boldsymbol{z}_t, \hat{\epsilon}_\theta, t)$
10: Sample $\boldsymbol{\epsilon}' \sim \mathcal{N}(0, \boldsymbol{I})$
11: $\boldsymbol{z}_t \leftarrow \sqrt{\bar{\alpha}_t/\bar{\alpha}_{t-1}}\boldsymbol{z}_{t-1} + \sqrt{1-\bar{\alpha}_t/\bar{\alpha}_{t-1}}\,\boldsymbol{\epsilon}'$
12: **end for**
13: **end for**

and shows the two gradients to optimize z_t and $z_{0|t}$ in lines 4 and 7 respectively. Also, it takes m steps to optimize the backward and get a better approximation of the $\hat{z}_0$.

Classifier-Free Guidance

Classifier-based guidance uses a pre-trained unconditional model and using an off-the-shelf reward model it is able to guide the generation towards generating samples for the downstream task the reward model follows. However, in some cases we can train a conditional model and want to make it abide better to the given conditioning for example we have stable diffusion models that are text-conditioned and we want them to generate images that follow the text prompts better. For these cases, we can use inference time methods like classifier-free guidance to guide it away from the ”null” prompt and more towards the given prompt.

∇ represents ∇_{x_t}:

$$\nabla \log p(y|x_t) = \nabla \log p(x_t|y) - \nabla \log p(x_t). \tag{6.4}$$

Substituting Eq. 6.4 to Eq 6.3 we get [33]:

$$\begin{aligned}
\nabla \log p(x_t|y) &= \nabla \log p(x_t) + \gamma\left(\nabla \log p(\boldsymbol{x}_t|y) - \nabla \log p(x_t)\right) && (6.5)\\
&= \nabla \log p(x_t) + \gamma \nabla \log p(x_t|y) - \gamma \nabla \log p(x_t) && (6.6)\\
&= \underbrace{\gamma \nabla \log p(x_t|y)}_{\text{conditional score}} + \underbrace{(1-\gamma)\nabla \log p(x_t)}_{\text{unconditional score.}} && (6.7)
\end{aligned}$$

Generally, γ is set greater than 1 which means that we are guiding the model to follow the given conditioning y more and steer it away from the ”null” conditioning. This null conditioning can also be replaced by *negative prompts* [3], the prompt we don’t want the image to adhere to. Subsequent research has also shown that replacing the unconditional score with a checkpoint diffusion model before it’s fully trained works well, essentially guiding the model away from a *less refined (bad) version of itself* [24].

6.1.3. Inference-Time Guidance: Sampling-Based Guidance

The shortcoming of the gradient-based guidance methods (classifier-based guidance) methods is that they require the reward models to be known and differentiable, which is not always the case. Thus sampling-based guidance methods which are built on top of the Best-of-N (BoN) approach evolved [51]. The idea of BoN is relatively simple and it just samples from the prior (unconditional) model multiple times and then selects whichever one aligns the best with the conditioning. Since it is sampling from the prior it doesn't deviate much from it and one way to think why it is very efficient is to think about estimating winrate. Thus when compared to the base model for n streams the winrate becomes $(n-1)/n$. Thus for a large enough n (around 100) it almost always beats the base policy.

However, BoN fails or the number of particles required becomes very large for complex rewards and large-foundational models. The bottleneck is that we have to unroll the whole diffusion model and then see which sample or stream is aligning the best with the reward. `CoDe` [51] makes it more efficient by unrolling till the blocksize and then using the Tweedies formula to estimate the rewards and then using greedy sampling to explore more in the directions expected to give higher rewards.

Algorithm 6 `CoDe` [51]

Require: p unconditional prior, T total timesteps, N number of parallel sampling streams, B blocksize, c conditioning

Sample initial noise: $x_T \sim \mathcal{N}(0, I)$
Initialize counter: $s = 1$
for $t \in [T-1, \ldots, 0]$ **do**
  if mod$(s, B) = 0$ **then**
    Sample N times over B steps:
    $\left\{x_{t-1}^{(n)}\right\}_{n=1}^{N} \underset{i.i.d.}{\sim} \prod_{i=t}^{t+B} p(x_{i-1}|x_i)$
    Compute values of all N samples:
    $\left\{V(x_{t-1}^{(n)})\right\}_{n=1}^{N} = \left\{r(\mathbb{E}[x_0|x_{t-1}^{(n)}])\right\}_{n=1}^{N}$
    Select the sample with maximum value:
    $x_{t-1} \leftarrow \underset{\left\{x_{t-1}^{(n)}\right\}_{n=1}^{N}}{\arg\max}\, V(x_{t-1}; p, c)$
  end if
  $s \leftarrow s + 1$
end for
Return: x_0

As seen by the Algorithm 6 we sample for B steps for N parallel streams and then select the one which aligns the most with the reward. For detailed mathematical proofs and the working refer to the paper.

7

Additional Discussion

In this chapter, we talk about pipeline details, modeling design choices that didn't work, and explain the reasoning behind some of the choices made and the alternatives considered.

7.1. (T+I)2I scenario affect of the r parameter

For the Image-Text to Image setting where we have a reference image and start the denoising from $r \cdot T$ instead of T by adding controlled noise to the reference image and then further guiding the denoising towards the pickscore. The hyperparameter r is crucial, balancing the preservation of the reference image's style and structure against allowing the model more creative freedom to follow the prompt. A higher r offers more creative freedom but can lead to the final image hardly resembling the reference, while a lower r heavily fixes the style and structure, thereby failing to incorporate the prompt. For a fair comparison and optimal performance, r should be adaptively tuned for each method, prompt, and reference image, as its optimal value varies based on prompt complexity and reference image characteristics but due to time and computational restraints we fixed this hyperparameter for all the methods.

This can be seen from Fig 7.1 as in this case for the same prompt and r it doesn't perform that well on a complex style like van Gogh and performs well on an abstract style. When $r = 0.6$, the model demonstrates a stronger ability to capture the complex brushstrokes and overall aesthetic of the Van Gogh style. However, for the abstract style (bottom row), while the colors are preserved, the distinctive stylistic elements of the abstract art are not fully transferred. Whereas at $r = 0.5$ for a different prompt, the model appears to over-capture the Van Gogh style, leading to an output that, while matching the style, doesn't adhere to the specified prompt at all. The model for the abstract style captures the style elements and applies them in a manner that aligns well with the given prompt. These observations empirically show that the optimal r value is not universal but should be adjusted dynamically based on both the complexity of the input prompt and the characteristics of the reference style image. Further research can look into making this adaptive by analyzing the reference image and the prompt.

7.2. Gradient-guidance on the greedy selected sample

The first extension we implemented was adding the gradient guidance to the greedily best-selected sample as this was much more efficient than adding it individually to all the steams. However, the increase in performance was not substantial. The reason for this was that by focusing only on the best path at each step, we were overlooking potentially better, unexplored paths that might have a lower immediate reward but could lead to a much higher overall outcome with the addition of gradient guidance. Thus we decided to add the gradients to all the streams which is a bit more computationally expensive but helps in achieving much better rewards.

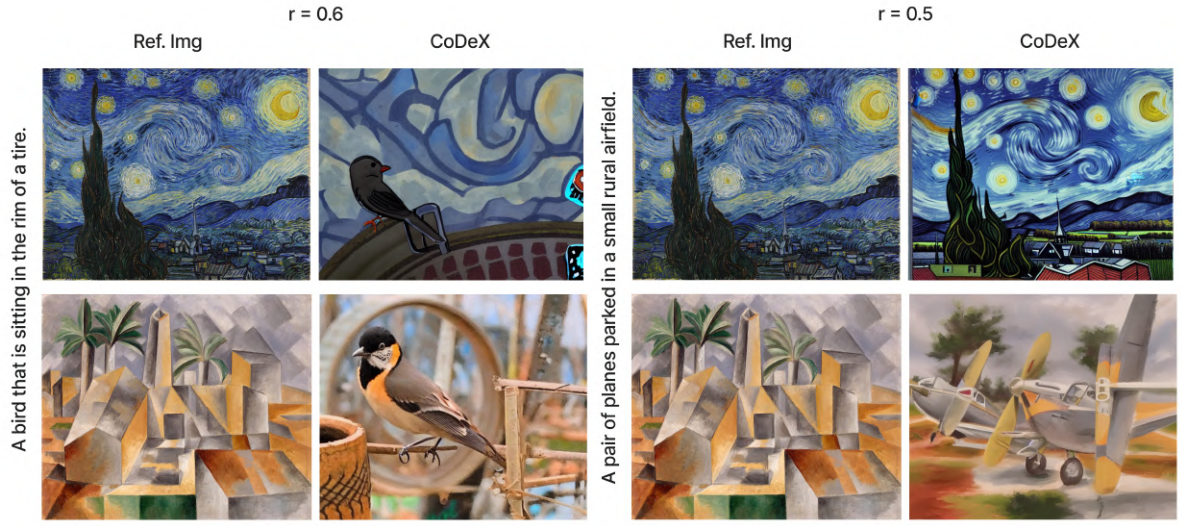

Figure 7.1: Performance of CoDeX for varying r values across different reference images. The left columns show results for $r = 0.6$, and the right columns for $r = 0.5$. The top row is for a more detailed and complex reference style (Van Gogh's Starry Night) while the bottom row is for a more abstract style

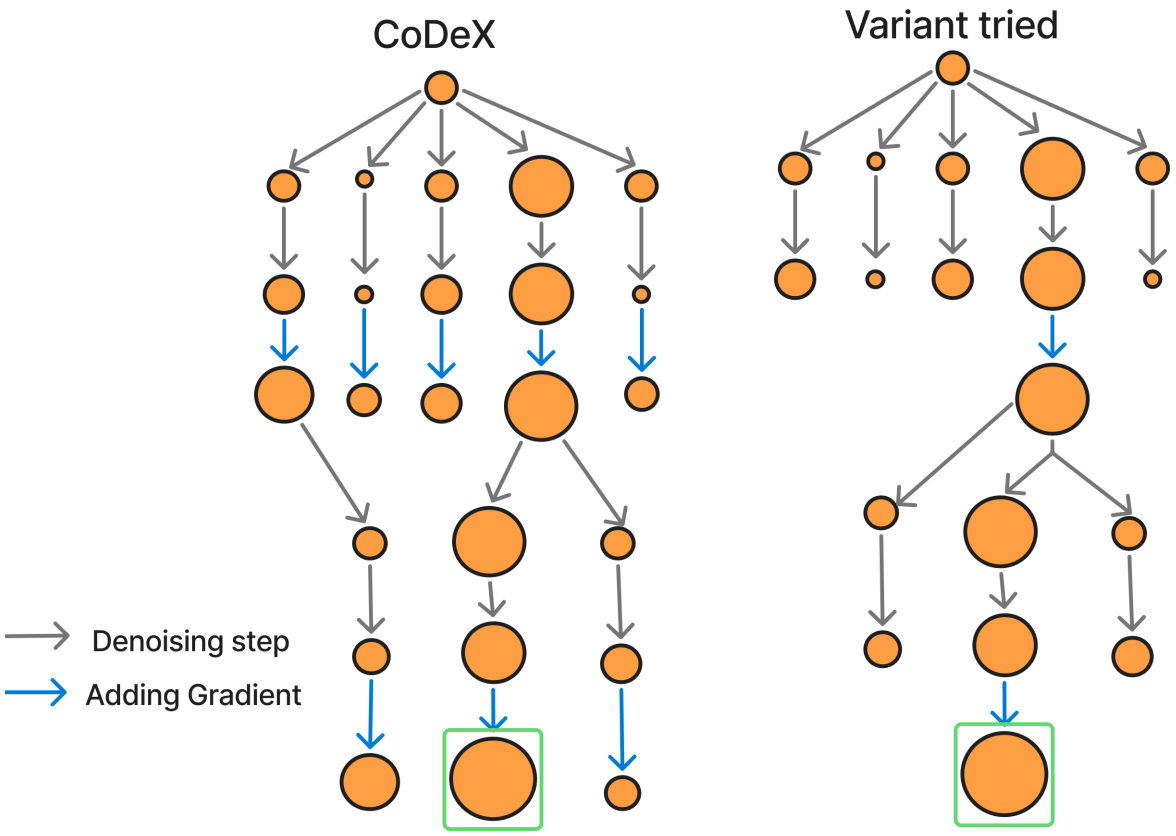

Figure 7.2: The schematic diagram representing the working of the variant tried and the CoDeX

8

Conclusion

By combining blockwise sampling with blockwise gradient descent we are not only able to achieve better rewards but are able to do so while deviating less from the prior all while reducing the sampling computational overhead significantly. Thus it makes the process much more efficient and can be used for high-quality conditioned image generation for any conditioning with an off-the-shelf reward model. This makes it much more general and easy to use and extend without requiring any training.

However, extending it to non-differentiable rewards didn't work primarily due to the highly noised estimates of the gradients in the multi-dimensional image space (even for latent diffusion models) which required many forward passes for it to work and therefore not only made it lose its efficiency but also performance. Further research on more efficient gradient approximation methods can be applied to make these work in non-differentiable reward scenarios.

Despite the notable progress in making conditional image generation through sampling more efficient, the considerable inference time compared to unconditional models continues to be a major barrier to real-world adoption. Additionally, due to the time constraints of the current study, a thorough investigation into the impact of dynamic adaptive temperature scheduling and adaptive particle schedules was not feasible. Dynamic temperature scheduling could allow for more subtle control over the exploration-exploitation trade-off during generation, while adaptive particle schedules could dynamically adjust computational resources based on the complexity of the generation task. Thorough experimentation in these areas could further increase the overall reward and generalization of our method.

References

[1] Josh Abramson et al. “Accurate structure prediction of biomolecular interactions with AlphaFold 3”. In: *Nature* 630.8016 (2024), pp. 493–500.

[2] Josh Achiam et al. “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (2023).

[3] Yuanhao Ban et al. “Understanding the Impact of Negative Prompts: When and How Do They Take Effect?” In: *European Conference on Computer Vision*. Springer. 2024, pp. 190–206.

[4] Arpit Bansal et al. “Universal guidance for diffusion models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 843–852.

[5] Omer Bar-Tal et al. “Lumiere: A space-time diffusion model for video generation”. In: *SIGGRAPH Asia 2024 Conference Papers*. 2024, pp. 1–11.

[6] Fengxiang Bie et al. “Renaissance: A survey into ai text-to-image generation in the era of large model”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[7] Kevin Black et al. “Training diffusion models with reinforcement learning”. In: *arXiv preprint arXiv:2305.13301* (2023).

[8] Andreas Blattmann et al. “Stable video diffusion: Scaling latent video diffusion models to large datasets”. In: *arXiv preprint arXiv:2311.15127* (2023).

[9] Rishi Bommasani et al. “On the opportunities and risks of foundation models”. In: *arXiv preprint arXiv:2108.07258* (2021).

[10] Kyunghyun Cho et al. “On the properties of neural machine translation: Encoder-decoder approaches”. In: *arXiv preprint arXiv:1409.1259* (2014).

[11] Paul F Christiano et al. “Deep reinforcement learning from human preferences”. In: *Advances in neural information processing systems* 30 (2017).

[12] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[13] Kawin Ethayarajh et al. “Model alignment as prospect theoretic optimization”. In: *Forty-first International Conference on Machine Learning*. 2024.

[14] Ying Fan et al. “Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 79858–79885.

[15] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[16] Aaron Grattafiori et al. “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (2024).

[17] Yutong He et al. “Manifold preserving guided diffusion”. In: *arXiv preprint arXiv:2311.16424* (2023).

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[19] Jonathan Ho and Tim Salimans. “Classifier-free diffusion guidance”. In: *arXiv preprint arXiv:2207.12598* (2022).

[20] Jiwoo Hong, Noah Lee, and James Thorne. “Orpo: Monolithic preference optimization without reference model”. In: *arXiv preprint arXiv:2403.07691* (2024).

[21] Emiel Hoogeboom et al. “Equivariant diffusion for molecule generation in 3d”. In: *International conference on machine learning*. PMLR. 2022, pp. 8867–8887.

[22] Rongjie Huang et al. “Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 13916–13932.

[23] Bowen Jing et al. “Torsional diffusion for molecular conformer generation”. In: *Advances in neural information processing systems* 35 (2022), pp. 24240–24253.

[24] Tero Karras et al. “Guiding a diffusion model with a bad version of itself”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 52996–53021.

[25] Diederik P Kingma, Max Welling, et al. *Auto-encoding variational bayes*. 2013.

[26] Zhifeng Kong et al. “Diffwave: A versatile diffusion model for audio synthesis”. In: *arXiv preprint arXiv:2009.09761* (2020).

[27] George Krasadakis. *The Impact of AI on Society and Everyday Life*. Medium post. Nov. 2023. URL: `https://medium.com/60-leaders/the-impact-of-ai-on-society-and-everyday-life-711307e06b87` (visited on 06/12/2025).

[28] Kimin Lee et al. “Aligning text-to-image models using human feedback”. In: *arXiv preprint arXiv:2302.12192* (2023).

[29] Jun Li et al. “A comprehensive survey of image generation models based on deep learning”. In: *Annals of Data Science* 12.1 (2025), pp. 141–170.

[30] Shufan Li et al. “Aligning diffusion models by optimizing human utility, 2024a”. In: *URL https://arxiv.org/abs/2404.04465* ().

[31] Buhua Liu et al. “Alignment of diffusion models: Fundamentals, challenges, and future”. In: *arXiv preprint arXiv:2409.07253* (2024).

[32] Haohe Liu et al. “Audioldm: Text-to-audio generation with latent diffusion models”. In: *arXiv preprint arXiv:2301.12503* (2023).

[33] Calvin Luo. “Understanding diffusion models: A unified perspective”. In: *arXiv preprint arXiv:2208.11970* (2022).

[34] Shitong Luo and Wei Hu. “Diffusion probabilistic models for 3d point cloud generation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 2837–2845.

[35] Zhaoyang Lyu et al. “A conditional point diffusion-refinement paradigm for 3d point cloud completion”. In: *arXiv preprint arXiv:2112.03530* (2021).

[36] Alexander Quinn Nichol and Prafulla Dhariwal. “Improved denoising diffusion probabilistic models”. In: *International conference on machine learning*. PMLR. 2021, pp. 8162–8171.

[37] Yuta Oshima et al. “Inference-Time Text-to-Video Alignment with Diffusion Latent Beam Search”. In: *arXiv preprint arXiv:2501.19252* (2025).

[38] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[39] Dustin Podell et al. “Sdxl: Improving latent diffusion models for high-resolution image synthesis”. In: *arXiv preprint arXiv:2307.01952* (2023).

[40] Mihir Prabhudesai et al. “Aligning text-to-image diffusion models with reward backpropagation”. In: (2023).

[41] Q3 Technologies. “How Generative AI Development is Transforming Diverse Industries”. In: *Q3Tech Blog* (June 2024). Updated June 6, 2024. URL: `https://www.q3tech.com/blogs/how-generative-ai-development-is-transforming-diverse-industries/` (visited on 06/12/2025).

[42] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PmLR. 2021, pp. 8748–8763.

[43] Rafael Rafailov et al. “Direct preference optimization: Your language model is secretly a reward model”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 53728–53741.

[44] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286.

[45] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer. 2015, pp. 234–241.

[47] Chitwan Saharia et al. “Photorealistic text-to-image diffusion models with deep language understanding”. In: *Advances in neural information processing systems* 35 (2022), pp. 36479–36494.

[48] Tim Salimans et al. “Improved techniques for training gans”. In: *Advances in neural information processing systems* 29 (2016).

[49] Christoph Schuhmann et al. “Laion-5b: An open large-scale dataset for training next generation image-text models”. In: *Advances in neural information processing systems* 35 (2022), pp. 25278–25294.

[50] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[51] Anuj Singh et al. “CoDe: Blockwise Control for Denoising Diffusion Models”. In: *arXiv preprint arXiv:2502.00968* (2025).

[52] Jascha Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International conference on machine learning*. pmlr. 2015, pp. 2256–2265.

[53] Feifan Song et al. “Preference ranking optimization for human alignment”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 17. 2024, pp. 18990–18998.

[54] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising diffusion implicit models”. In: *arXiv preprint arXiv:2010.02502* (2020).

[55] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in neural information processing systems* 32 (2019).

[56] Yang Song et al. “Score-based generative modeling through stochastic differential equations”. In: *arXiv preprint arXiv:2011.13456* (2020).

[57] Nisan Stiennon et al. “Learning to summarize with human feedback”. In: *Advances in neural information processing systems* 33 (2020), pp. 3008–3021.

[58] Amos Tversky and Daniel Kahneman. “Advances in prospect theory: Cumulative representation of uncertainty”. In: *Journal of Risk and uncertainty* 5 (1992), pp. 297–323.

[59] Masatoshi Uehara et al. “Feedback efficient online fine-tuning of diffusion models”. In: *arXiv preprint arXiv:2402.16359* (2024).

[60] Masatoshi Uehara et al. “Inference-Time Alignment in Diffusion Models with Reward-Guided Generation: Tutorial and Review”. In: *arXiv preprint arXiv:2501.09685* (2025).

[61] Bram Wallace et al. “Diffusion model alignment using direct preference optimization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 8228–8238.

[62] Joseph L Watson et al. “De novo design of protein structure and function with RFdiffusion”. In: *Nature* 620.7976 (2023), pp. 1089–1100.

[63] Lilian Weng. *From Autoencoder to Beta-VAE*. `https://lilianweng.github.io/posts/2018-08-12-vae/`. Updated on 2019-07-18 and 2019-07-26. 2018. (Visited on 05/26/2025).

[64] Xiaoshi Wu et al. “Human preference score: Better aligning text-to-image models with human preference”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 2096–2105.

[65] Ray Xiao. *Structured Encoders and Decoders*. `https://www.cs.toronto.edu/~duvenaud/courses/csc2541/slides/structured-encoders-decoders.pdf`. Lecture slides for CSC2541: Topics in Machine Learning, University of Toronto. 2016. (Visited on 05/26/2025).

[66] Minkai Xu et al. “Geodiff: A geometric diffusion model for molecular conformation generation”. In: *arXiv preprint arXiv:2203.02923* (2022).

[67] Minkai Xu et al. “Geometric latent diffusion models for 3d molecule generation”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 38592–38610.

[68] Haotian Ye et al. “Tfg: Unified training-free guidance for diffusion models”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 22370–22417.

[69] Jiwen Yu et al. “Freedom: Training-free energy-guided conditional diffusion model”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 23174–23184.

A

Appendix

A.1. Guidance Rescaling

We rescale the guidance scale using the same mechanism as the one used in `FreeDoM` [69]. What they do is that they basically guide the model guidance scale times in the direction of the gradient and rescale this scale based on the CFG guidance. Thus they guide the model even more if there is a big difference between the text conditional and the unconditional noise prediction.

$$\text{scale}_{\text{new}} = \frac{\| \text{ correction } \|_2 \cdot \text{scale}_{\text{CFG}} \cdot \text{scale}_{\text{grad}}}{\| \text{ grad } \|_2 + \varepsilon} \tag{A.1}$$

where correction is just the CFG [19] correction term:

$$\text{correction} = \hat{\epsilon}_\theta(x_t, prompt) - \hat{\epsilon}_\theta(x_t) \tag{A.2}$$

We found the dynamic rescaling strategy to be the most effective within our evaluation setting and thus adopted it in place of a fixed guidance scale. By normalizing the guidance using the gradient norm and scaling it proportionally to the correction norm it reduces the sensitivity to the choice of guidance scale. Therefore, a consistent range of values (generally between 0.2 and 0.6) performs reliably well across different reward models. This not only improved performance stability but also **saved considerable time**, as it decreased the need to manually tune the guidance scale for each individual reward function checking for a ton of different values based on the reward scale.

A.2. Image Based Guidance

For the aesthetic reward model, we use the ViT-L/14 as the backbone model for the CLIP. We use a blocksize of 5 for both the sampling and gradient guidance in this case and set the guidance scale to be 0.2. We do the denoising for 500 DDPM steps and the CFG guidance scale is also set to 5. Also, we start adding the gradients from the 0.6 noise ratio i.e. if $T = 1000$ we start adding the gradients from the 600 timestep. This preserves the image structure, doesn't reward hack, and guides it towards achieving a better reward. The evaluation set contains 51 prompts (from the ImageNet evaluation set) and we generate 10 images for each prompt.

For gradient guidance experiments for `MPGD` and `FreeDoM` we do the guidance only for noise ratios between 0.7 and 0.3. Thus for timesteps $70 - 30$ in a 100 step DDIM scheduler. The guidance scales used are 7.5 for `MPGD` and 0.2 for `FreeDoM`. Additionally, for `FreeDoM`, we perform 10 optimization iterations per denoising step. For the `UG` baseline, we also use 100 DDIM steps, with 6 optimization steps and a forward guidance weight of 30.

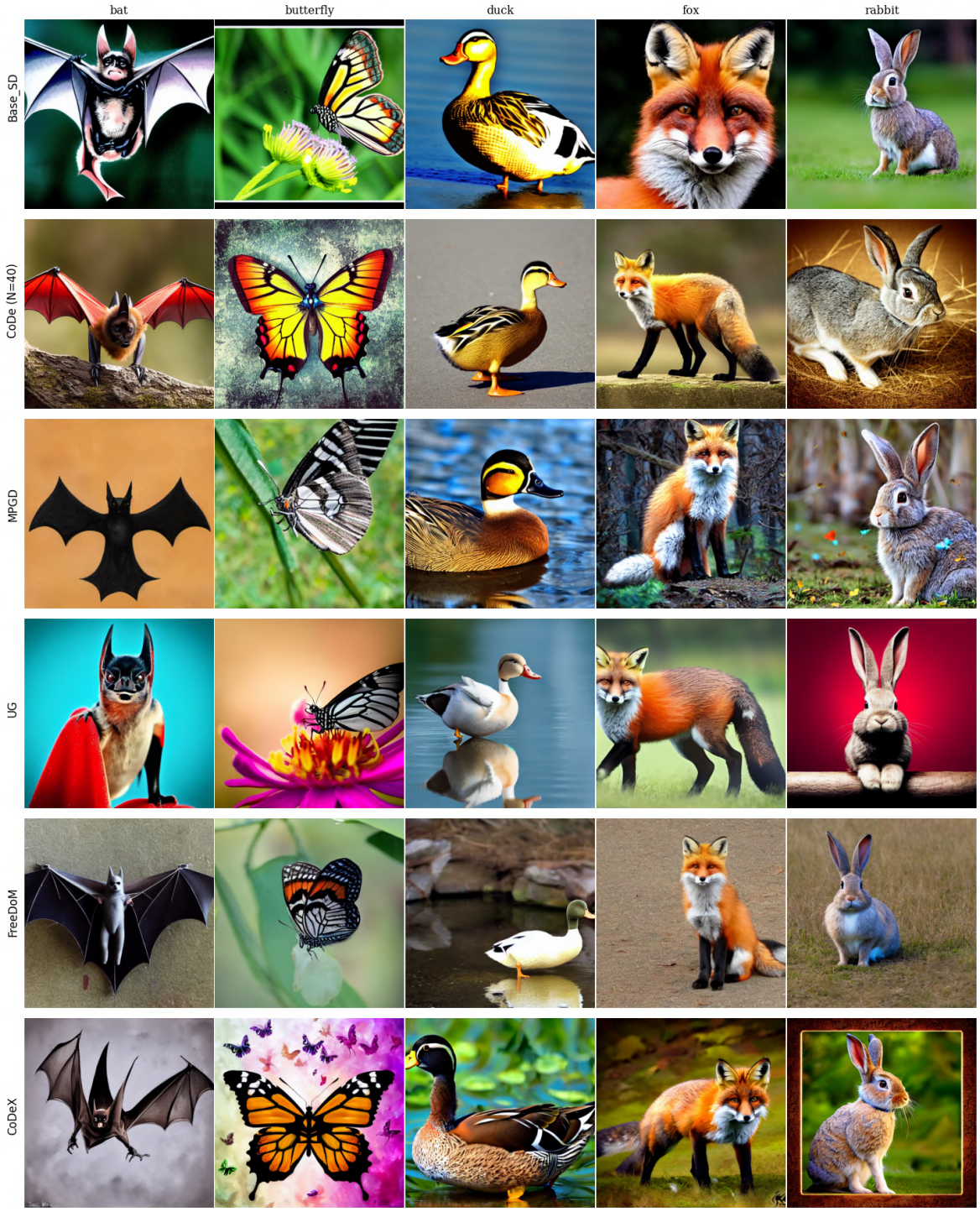

Figure A.1: More qualitative examples for aesthetic guidance

A.3. T2I Setting

For the pickscore reward model, we again use the ViT-L/14 as the backbone model for the CLIP. We use a blocksize of 5 for both the sampling and 4 gradient guidance in this case and set the guidance scale to be 0.2. We do the denoising for 500 DDPM steps and the CFG guidance scale is also set to 5. We do the gradient addition during the whole denoising as we also want to alter the structural properties for the alignment to complex prompts. This preserves the image structure, doesn't reward hack, and guides it towards achieving a better reward. The evaluation set contains 50 prompts (from the `HPD` [64] evaluation dataset) and we generate 10 images for each prompt.

The setting for gradient guidance remains the same except for `UG` where we increase the weight to 150.

A.4. Multireward Setting

In the multireward setting, we use the same prompts as the Aesthetic case but only a subset of 6 out of the 51 and generate 10 images for each as it takes longer due to the weighted addition of the two reward models. The rest of the settings are the same and we add the gradients during the whole denoising.

The values of γ_1 and γ_2 are taken as $(1, 0), (0, 1)$ and we set γ_1 as 1 and change γ_2 in $[2, 3, 5, 10, 15, 20, 25, 30, 50, 70, 100, 150, 200, 250, 300, 350, 400, 450, 500, 750, 1000]$

A.5. (T+I)2I Setting

For this, we use the same 50 prompts as the T2I setting, use the three style images as the reference images, and generate 10 images for each prompt and each style. We use the 100 step DDPM scheduler and we set r to be 0.6. The sampling blocksize is 5 and the gradient blocksize is 2 with a guidance scale of 0.4. The rest of the settings are the same.

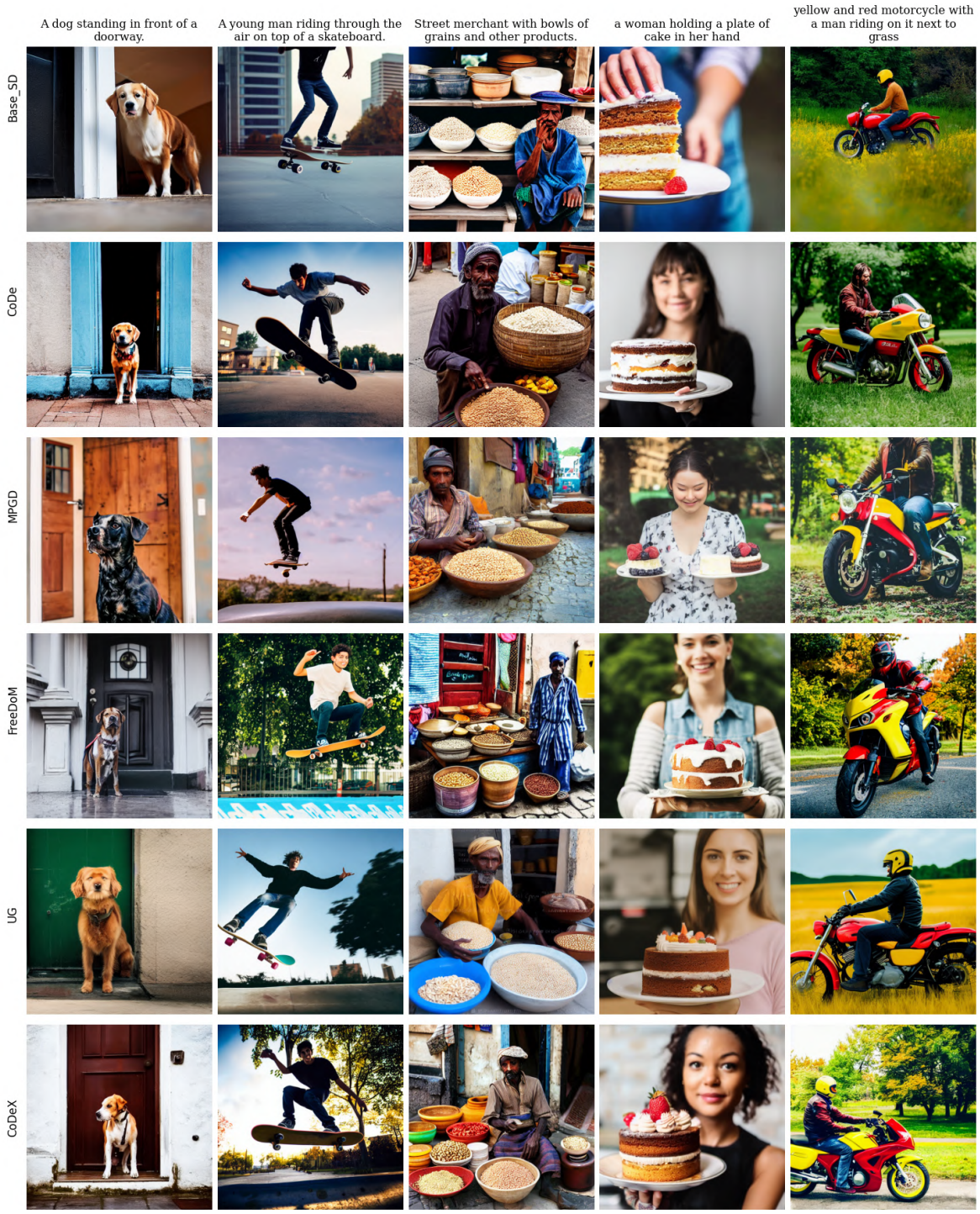

Figure A.2: More qualitative examples for T2I pickscore guidance

Figure A.3: Each column corresponds to a different prompt. Rows are grouped by $\gamma_2 \in \{0, 150, 1000\}$ (top to bottom), with γ_1 fixed at 1. Within each group, the three rows show results from `CoDe`, `FreeDoM`, and `CoDeX` (top to bottom). The top group focuses more on aesthetic quality, whereas the bottom group prefers Pickscore, and the middle group ($\gamma_2 = 150$) offers a balanced trade-off between the two.

Figure A.4: More qualitative samples for the (T+I)2I scenario