

ADDING CONDITIONAL CONTROL TO DIFFUSION MODELS WITH REINFORCEMENT LEARNING

Anonymous authors

Paper under double-blind review

ABSTRACT

Diffusion models are powerful generative models that allow for precise control over the characteristics of the generated samples. While these diffusion models trained on large datasets have achieved success, there is often a need to introduce additional controls in downstream fine-tuning processes, treating these powerful models as pre-trained diffusion models. This work presents a novel method based on reinforcement learning (RL) to add such controls using an offline dataset comprising inputs and labels. We formulate this task as an RL problem, with the classifier learned from the offline dataset and the KL divergence against pre-trained models serving as the reward functions. Our method, **CTRL** (Conditioning pre-Trained diffusion models with **R**einforcement **L**earning), produces soft-optimal policies that maximize the abovementioned reward functions. We formally demonstrate that our method enables sampling from the conditional distribution with additional controls during inference. Our RL-based approach offers several advantages over existing methods. Compared to classifier-free guidance, it improves sample efficiency and can greatly simplify dataset construction by leveraging conditional independence between the inputs and additional controls. Additionally, unlike classifier guidance, it eliminates the need to train classifiers from intermediate states to additional controls.

1 INTRODUCTION

Diffusion models have emerged as effective generative models for capturing intricate distributions (Sohl-Dickstein et al., 2015; Ho et al., 2020). Their capabilities are further enhanced by building conditional diffusion models $p(x|c)$. For instance, in text-to-image generative models like DALL-E (Ramesh et al., 2021) and Stable Diffusion (Rombach et al., 2022), $c \in \mathcal{C}$ is a prompt, and $x \in \mathcal{X}$ is the image generated according to this prompt. While diffusion models trained on extensive datasets have shown remarkable success, additional controls often need to be incorporated during downstream fine-tuning when treating these powerful models as pre-trained diffusion models.

In this work, our goal is to incorporate new conditional controls into pre-trained diffusion models. Specifically, given access to a large pre-trained model capable of modeling $p(x|c)$ trained on extensive datasets, we aim to condition it on an additional random variable $y \in \mathcal{Y}$, thereby creating a generative model $p(x|c, y)$. To accomplish this, we utilize the pre-trained model and an offline dataset consisting of triplets $\{c, x, y\}$. This scenario is important, as highlighted in the existing literature on computer vision (e.g., Zhang et al. (2023)), because it enables the extension of generative capabilities with new conditional variables without requiring retraining from scratch. Currently, classifier-free guidance (Ho et al., 2020) is a prevailing approach for incorporating conditional controls into diffusion models, and it has proven successful in computer vision (Zhang et al., 2023; Zhao et al., 2024). However, its effectiveness may not extend well to other challenging problems, especially when large offline datasets are unavailable. Indeed, the success of training conditional diffusion models via classifier-free guidance heavily relies on such datasets (Brooks et al., 2023), which are often impractical to obtain. In these scenarios, this method tends to struggle.

In our work, we present a new approach for adding new conditional controls via reinforcement learning (RL) to further improve sample efficiency. Inspired by recent progress in RL-based fine-tuning (Black et al., 2023; Fan et al., 2023), we frame the conditional generation as an RL problem within a Markov Decision Process (MDP). In this formulation, the reward, which we want to maximize, is the

Table 1: Comparison with existing approaches. Unlike classifier guidance and its variations, our method involves directly fine-tuning pre-trained models. We avoid the need to learn a mapping from $x_t \rightarrow y$ or rely on heuristic approximations. Additionally, while classifier-free guidance always demands triplets $\{c, x, y\}$, our approach can leverage conditional independence ($y \perp c|x$) and only necessitate pairs $\{x, y\}$, simplifying the construction of the offline dataset.

Methods	Fine-tuning	Need to learn $x_t \rightarrow y$	Leveraging conditional independence
Classifier guidance (Dhariwal and Nichol, 2021)	No	Yes	Yes
Reconstruction guidance (e.g. Ho et al. (2022); Chung et al. (2022); Han et al. (2022))	No	No	Yes
Classifier-free guidance (Ho and Salimans, 2022)	Yes	No	No
CTRL (Ours)	Yes	No	Yes

(conditional) log-likelihood function $\log p(y|x, c)$, and the policy, conditioned on (c, y) , corresponds to the denoising process at each time step in a diffusion model. We formally demonstrate that, by executing the soft-optimal policy, which maximizes the reward $\log p(y|x, c)$ with KL penalty against the pre-trained model, we can sample from the target conditional distribution $p(x|c, y)$ during inference. Hence, our proposed algorithm, **CTRL (Conditioning pre-Trained diffusion models with Reinforcement Learning)** consists of three main steps: (1) learning a classifier $\log p(y|x, c)$ (which will serve as our reward function in the MDP) from the offline dataset, (2) constructing an augmented diffusion model by adding (trainable) parameters to the pre-trained model in order to accommodate an additional label y , and (3) learning soft-optimal policy within the aforementioned MDP during fine-tuning. Our approach is novel as it significantly diverges from classifier-free guidance and distinguishes itself from existing RL-based fine-tuning methods by integrating an augmented model in the fine-tuning process to support additional controls.

Our novel RL-based approach offers several advantages over existing methods for adding additional controls. Firstly, in contrast to classifier-free guidance, which uses offline data to directly model $p(x|y, c)$, our method leverages offline data to model the simpler distribution $p(y|x, c)$, improving sample efficiency (in typical scenarios where y is lower-dimensional than x). Secondly, in typical scenarios where the additional label y depends solely on x (e.g., the compressibility of an image depends only on the image, not the prompt), our fine-tuning method only requires pairs $\{x, y\}$, whereas classifier-free guidance still necessitates triplets $\{c, x, y\}$ from the offline dataset. This is because the reward function simplifies to $\log p(y|x)$ due to the conditional independence $y \perp c|x$, which gives $\log p(y|x, c) = \log p(y|x)$. Furthermore, when the goal is to simultaneously add conditioning controls on two labels, y_1 and y_2 , and both labels only depend on x , our method requires only pairs $\{x, y_1\}$ and $\{x, y_2\}$. In contrast, classifier-free guidance requires quadruples $\{c, x, y_1, y_2\}$. Therefore, in this manner, **CTRL** can also leverage the *compositional nature* of the mapping between inputs and additional labels.

Our contributions can be summarized as follows. We propose an RL-based fine-tuning approach for conditioning pre-trained diffusion models on additional labels. In comparison to classifier-free guidance, our method uses the offline dataset in a sample-efficient manner and enables leveraging the conditional independence assumption, which significantly simplifies the construction of the offline dataset. Additionally, we establish a close connection to classifier guidance (Dhariwal and Nichol, 2021; Song et al., 2020), showing that it provides an alternative method for obtaining the aforementioned soft-optimal policies (in ideal cases where there are no statistical/model-misspecification errors in the algorithms). Despite this connection, our algorithm addresses common challenges in classifier guidance, such as the need to learn classifiers at multiple noise scales in standard classifier guidance and the use of fundamental approximations in some variants to avoid learning these noisy classifiers (Chung et al., 2022; Song et al., 2022). Experimentally, we validate the superiority of **CTRL** over baselines in both single-task and multi-task conditional image generation, such as generating highly aesthetic yet compressible images, where existing methods often struggle. Table 1 summarizes the main features of the proposed algorithm compared to existing methods.

2 RELATED WORKS

Classifier guidance. Dhariwal and Nichol (2021); Song et al. (2020) introduced classifier guidance, a method that entails training a classifier and incorporating its gradients to guide inference (while freezing pre-trained models). However, a notable drawback of this technique lies in the classifier’s accuracy in predicting y from intermediate x_t , resulting in cumulative errors during the diffusion process. To mitigate this issue, several studies propose methods to circumvent it through reconstruction, referred to as **reconstruction guidance** in this work. Specifically, they employ certain approximations that map intermediate states x_t back to the original input space x_0 , allowing the classifier to be learned solely from x_0 to y (Ho et al., 2022; Han et al., 2022; Chung et al., 2022; Finzi et al., 2023; Bansal et al., 2023). In contrast to these works, our approach focuses on fine-tuning the diffusion model itself rather than relying on an inference-time technique. While the strict comparison between model fine-tuning and inference-time techniques is not feasible, we theoretically elucidate the distinctions and connections of our approach with classifier guidance in Section 5.1.

Classifier-free guidance. Classifier-free guidance (Ho and Salimans, 2022) is a method that directly conditions the generative process on both data and context, bypassing the need for explicit classifiers. This methodology has been widely and effectively applied, for example, in text-to-image models (Nichol et al., 2021; Saharia et al., 2022; Rombach et al., 2022). While the original research does not explore classifier-free guidance within the scope of fine-tuning pre-trained diffusion models, several subsequent studies address fine-tuning scenarios Zhang et al. (2023); Xie et al. (2023). As elucidated in Section 5.2, compared to classifier-free guidance, our approach can improve sample efficiency and leverage conditional independence to facilitate offline dataset construction.

Fine-tuning via RL. Several previous studies have addressed the fine-tuning of diffusion models by optimizing relevant reward functions. Methodologically, these approaches encompass supervised learning (Lee et al., 2023; Wu et al., 2023), reinforcement learning (Black et al., 2023; Fan et al., 2023; Uehara et al., 2024), and control-based techniques (Clark et al., 2023; Xu et al., 2023; Prabhudesai et al., 2023; Uehara et al., 2024). While our proposal draws inspiration from these works, our objective for fine-tuning is to tackle a distinct goal: incorporating *additional* controls. To achieve this, unlike previous approaches, we employ policies with augmented parameters rather than merely fine-tuning pre-trained models without adding any new parameters.

3 PRELIMINARIES

In this section, we introduce the problem setting, review the existing methods addressing this problem, and discuss their disadvantages.

3.1 GOAL: CONDITIONING WITH ADDITIONAL LABELS USING OFFLINE DATA

We first define our main setting and main objective. Throughout this paper, we use \mathcal{Y} and \mathcal{C} to represent condition spaces and \mathcal{X} to denote the (Euclidean) sample space. Given the pre-trained model, which enables us to sample from $p^{\text{pre}}(x|c) : \mathcal{C} \rightarrow \Delta(\mathcal{X})$, our goal is to add new conditional controls $y \in \mathcal{Y}$ such that we can sample from $p(x|c, y)$.

Pre-trained model and offline dataset. A (continuous-time) pre-trained conditional diffusion model is characterized by the following SDE¹:

$$dx_t = f^{\text{pre}}(t, c, x_t; \theta^{\text{pre}})dt + \sigma(t)dw_t, \quad x_0 = x_{\text{ini}}, \quad (1)$$

where $f^{\text{pre}} : [0, T] \times \mathcal{C} \times \mathcal{X} \rightarrow \mathbb{R}^d$ is a model with parameter θ .

In training diffusion models, the parameter θ^{pre} is derived by optimizing a specific loss function on large datasets². We refer interested readers to Appendix B for more details on constructing these loss

¹For simplicity, we consider the case where the initial distribution is a Dirac delta, as in bridge matching. The extension of our proposal for stochastic distributions remains straightforward, as shown in (Uehara et al., 2024).

²For notational simplicity, throughout this work, we would often drop θ^{pre} .

162 functions. Using the pre-trained model and following the above SDE (1) from 0 to T , we can sample
 163 from $p^{\text{pre}}(\cdot|c)$ for any condition $c \in \mathcal{C}$.

164 To add additional control to a pre-trained model, as in many recent works (Dhariwal and Nichol, 2021;
 165 Bansal et al., 2023; Epstein et al., 2023), we assume access to offline data: $\mathcal{D} = \{c^{(i)}, x^{(i)}, y^{(i)}\}_{i=1}^n \in$
 166 $\mathcal{C} \times \mathcal{X} \times \mathcal{Y}$. We denote the conditional distribution of y given x and c by $p^\diamond(y|x, c)$.

168 **Target distribution.** Using the pre-trained model and the offline dataset, our goal is to obtain a
 169 diffusion model such that we can sample from a distribution over $\mathcal{C} \times \mathcal{Y} \rightarrow \Delta(\mathcal{X})$ as below:

$$171 p_\gamma(\cdot|c, y) := \frac{\{p^\diamond(y|\cdot, c)\}^\gamma p^{\text{pre}}(\cdot|c)}{\int \{p^\diamond(y|x, c)\}^\gamma p^{\text{pre}}(x|c) \mu(dx)}, \quad (2)$$

172 where $\gamma \in \mathbb{R}^+$ denotes the strength of additional guidance and μ is the Lebsgue measure.

175 Such target distribution is extensively explored in the literature on classifier guidance and classifier-
 176 free guidance (Dhariwal and Nichol, 2021; Ho and Salimans, 2022; Nichol et al., 2021; Saharia et al.,
 177 2022; Rombach et al., 2022). Specifically, when $\gamma = 1$, this distribution corresponds to the standard
 178 conditional distribution $p(x|c, y)$, which is a fundamental objective of many conditional generative
 179 models (Dhariwal and Nichol, 2021; Ho and Salimans, 2022). Moreover, for a general γ , p_γ can be
 180 formulated via the following optimization problem:

$$181 p_\gamma(\cdot|c, y) = \underset{q: \mathcal{C} \times \mathcal{Y} \rightarrow \Delta(\mathcal{X})}{\operatorname{argmin}} \mathbb{E}_{x \sim q(\cdot|c, y)} [-\gamma \log p^\diamond(y|x, c)] + \text{KL}(q(\cdot|c, y) \| p^{\text{pre}}(\cdot|c)).$$

183 This relation is clear as the objective function equals $\text{KL}(q(\cdot|c, y) \| p_\gamma(\cdot|c, y))$ up to a constant.

185 **Goal.** As discussed, the primary goal of this research is to train a generative model capable of
 186 simulating $p_\gamma(\cdot|c, y)$. To achieve this, we introduce the following SDE:

$$187 dx_t = g(t, c, y, x_t) dt + \sigma(t) dw_t, \quad x_0 = x_{\text{ini}}, \quad (3)$$

189 where $g: [0, T] \times \mathcal{C} \times \mathcal{Y} \times \mathcal{X} \rightarrow \mathbb{R}^d$ is an augmented model to add additional controls into pre-trained
 190 models. The primary challenge involves leveraging both offline data and pre-trained model weights
 191 to train the term g , ensuring that the marginal distribution of x_T induced by the SDE (3) accurately
 192 approximates p_γ .

194 **Notation.** Let the space of trajectories $x_{0:T}$ be \mathcal{K} . Conditional on c and y , we denote the measure
 195 induced by the SDE (3) over \mathcal{K} by $\mathbb{P}^g(\cdot|c, y)$. Similarly, we use $\mathbb{P}_t^g(\cdot|c, y)$ and $p_t^g(\cdot|c, y)$ to represent
 196 the marginal distribution of x_t and density $d\mathbb{P}_t^g(\tau|c, y)/d\mu$.

197 **Remark 1** (Extension to the non-Euclidean space). *To streamline the notation, we focus on scenarios*
 198 *where \mathcal{X} is Euclidian. However, when dealing with discrete spaces, we can still extend our discussion*
 199 *by examining the discretized version from the beginning (Uehara et al., 2024, Theorem 1).*

201 3.2 EXISTING METHODS

202 In this subsection, we describe two existing methods that are applicable in our context to achieve the
 203 aforementioned goal.

206 3.2.1 CLASSIFIER-FREE GUIDANCE

207 Recent works have studied fine-tuning pre-trained models with classifier-free guidance (Brooks et al.,
 208 2023; Zhang et al., 2023; Xie et al., 2023). These methods introduce an augmented model g as
 209 described in (3), where the weights are initialized from the pre-trained model. Fine-tuning is via
 210 minimizing classifier-free guidance loss on the offline dataset. While successful in many applications,
 211 these methods may struggle in scenarios where offline datasets for new conditions are limited (Huang
 212 et al., 2021; Yellapragada et al., 2024; Giannone et al., 2024).

214 3.2.2 CLASSIFIER GUIDANCE

215 Classifier guidance (Dhariwal and Nichol, 2021; Song et al., 2020) is based on the following result.

Lemma 1 (Doob’s h-transforms (Rogers and Williams, 2000)). *For any $c \in \mathcal{C}$ and $y \in \mathcal{Y}$, by evolving according to the following SDE from 0 to T :*

$$dx_t = \{f^{\text{pre}}(t, c, x_t) + \sigma^2(t) \underbrace{\nabla_{x_t} \log \mathbb{E}_{x_t, T \sim \mathbb{P}^{\text{pre}}(\cdot | x_t, c)}[\mathbb{I}(y = y') | x_T, c]}_{\text{Additional Drift} := \nabla_{x_t} \log p(y | x_t, c)}\} dt + \sigma(t) dw_t, \quad (4)$$

the marginal distribution of x_T , i.e., $p(x_T | c, y)$, is equal to the target distribution $p_{\gamma=1}(\cdot | c, y)$ (2). Here, \mathbb{P}^{pre} denotes the distribution induced by the pre-trained diffusion model (1).

This lemma suggests that to simulate the target distribution (2), we only need to construct SDE (4). However, practical issues arise: first, training classifier $p(y | x_t, c)$ requires extensive data at each timestep, which is cumbersome with large pre-trained models. Furthermore, accumulated inaccuracies in drift estimates may lead to poor performance (Li and van der Schaar, 2023).

Reconstruction guidance. To mitigate these issues, several studies propose to approximate $p(y | x_t, c)$ directly via reconstruction (Ho et al., 2022; Han et al., 2022; Chung et al., 2022; Guo et al., 2024)³, specifically by $p(y | x_t, c) = \int p^\diamond(y | x_T, c) p(x_T | x_t, c) dx_T \approx p(y | \hat{x}_T(x_t, c), c)$, where $\hat{x}_T(x_t, c)$ is the (expected) denoised sample given x_t, c , i.e., $\hat{x}_T(x_t, c) = \mathbb{E}[x_T | x_t, c]$. Given such an approximation, we only need to learn $p^\diamond(y | x_T, c)$ from data. However, this approximation may become imprecise when $\mathbb{P}(x_T | x_t, c)$ is noisy or is difficult to estimate reliably (Chung et al., 2022).

4 CONDITIONING PRE-TRAINED DIFFUSION MODELS WITH RL

This section provides details on how our method solves the aforementioned goal with methodological motivations. We begin with a key observation: the conditioning problem can be effectively conceptualized as an RL problem. Building upon this insight, we illustrate our main algorithm.

4.1 CONDITIONING AS RL

Recall that our objective is to learn a drift term g in (3) so that the induced marginal distribution at T (i.e., p_T^g) closely matches our target distribution p_γ . To achieve this, we first formulate the problem via the following minimization:

$$\operatorname{argmin}_g \text{KL}(p_T^g(\cdot | c, y) \| p_\gamma(\cdot | c, y)).$$

With some algebra, we can show that the above optimization problem is equivalent to the following:

$$\operatorname{argmin}_g \mathbb{E}_{x_0, T \sim \mathbb{P}^g(\cdot | c, y)} \left[-\gamma \log p^\diamond(y | x, c) + \frac{1}{2} \int_0^T \frac{\|f^{\text{pre}}(s, c, x_s) - g(s, c, y, x_s)\|^2}{\sigma^2(s)} ds \right].$$

Here, recall that \mathbb{P}^g is the measure induced by SDE (3) with a drift coefficient g . Based on this observation, we derive the following theorem.

Theorem 1 (Conditioning as RL). *Consider the following RL problem:*

$$g^* := \operatorname{argmax}_g \mathbb{E}_{\substack{(c, y) \sim \Pi(c, y) \\ x_0, T \sim \mathbb{P}^g(\cdot | c, y)}} \left[\gamma \log p^\diamond(y | x_T, c) - \frac{1}{2} \int_0^T \frac{\|f^{\text{pre}}(s, c, x_s) - g(s, c, y, x_s)\|^2}{\sigma^2(s)} ds \right], \quad (5)$$

where $\Pi \in \Delta(\mathcal{C} \times \mathcal{Y})$. Significantly, the marginal distribution $p_T^{g^}$ matches our target distribution:*

$$\forall (c, y) \in \text{Supp}(\Pi); p_T^{g^*}(\cdot | c, y) = p^\gamma(\cdot | c, y).$$

Proofs are deferred to Appendix D. This theorem demonstrates that, after obtaining the optimal drift g^* by solving the RL problem in (5), we can sample from the target distribution $p^\gamma(\cdot | c, y)$ by following SDE (3) from time 0 to T . In the next section, we explain how to solve (5) in practice.

Algorithm 1 Conditioning pre-Trained diffusion models with **Reinforcement Learning (CTRL)**

- 1: **Input:** Pre-trained model with a drift coefficient f^{pre} , Offline data $\mathcal{D} = \{c^{(i)}, x^{(i)}, y^{(i)}\}$, Exploratory distribution $\Pi \in \Delta(\mathcal{C} \times \mathcal{Y})$
- 2: Construct an augmented model $g(t, c, y, x; \psi)$.
- 3: Train a classifier $\hat{p}(y|x, c)$ to approximate $p^\diamond(y|x, c)$ from the offline data \mathcal{D}
- 4: Fine-tune the diffusion model by solving the following RL problem (e.g. using Algorithm 2):

$$\hat{\psi} = \underset{\psi}{\operatorname{argmax}} \mathbb{E}_{\substack{(c, y) \sim \Pi(c, y) \\ x_{0:T} \sim \mathbb{P}^g(\cdot|c, y; \psi)}} \left[\gamma \log \hat{p}(y|x_T, c) - \frac{1}{2} \int_0^T \frac{\|f^{\text{pre}}(s, c, x_s) - g(s, c, y, x_s; \psi)\|^2}{\sigma^2(s)} ds \right]$$

where $\mathbb{P}^g(\cdot|c, y; \psi)$ is an distribution induced by the SDE with a parameter ψ .

- 5: **Output:** $dx_t = g(t, c, y, x_t; \hat{\psi})dt + \sigma(t)dw_t$

4.2 ALGORITHM

Theoretically inspired by Theorem 1, we introduce Algorithm 1. It consists of three steps.

Step 1: Constructing the augmented model (Line 2). To add additional conditioning to the pre-trained diffusion model, it is necessary to enhance the pre-trained model $f^{\text{pre}}(t, c, x; \theta)$. We introduce an augmented model $g(t, c, y, x; \psi)$ with parameters $\psi = [\theta^\top, \phi^\top]^\top$, initialized at $\psi^{\text{ini}} = [\theta^{\text{pre}\top}, \mathbf{0}^\top]$. Here, ψ is structured as a combination of the existing parameters θ and new parameters ϕ .

Determining the specific architecture of the augmented model involves a tradeoff: adding more new parameters enhances expressiveness but raises computational costs. In scenarios where \mathcal{Y} is discrete with cardinality $|\mathcal{Y}|$, the most straightforward solution is to instantiate ϕ with a simple linear embedding layer that maps each $y \in \mathcal{Y}$ to its corresponding embedding. These embeddings are then added to every intermediate output in the diffusion SDE (i.e., x_t in (3)). This method preserves the original structure to the fullest extent while ensuring that all pre-trained weights are fully utilized. Experimentally, we observe that this lightweight modification leads to accurate conditional generations for complex conditioning tasks, as shown in Section 6.

Step 2: Training a calibrated classifier with offline data (Line 3). Using a function class $\mathcal{F} \subset [\mathcal{C} \times \mathcal{X} \rightarrow \Delta(\mathcal{Y})]$, such as a neural network, we perform maximum likelihood estimation (MLE):

$$\hat{p}(\cdot|x, c) := \underset{r \in \mathcal{F}}{\operatorname{argmax}} \sum_{i=1}^n \log r(y^{(i)}|x^{(i)}, c^{(i)}). \quad (6)$$

For instance, when \mathcal{Y} is discrete, this loss reduces to the standard cross-entropy loss. When \mathcal{Y} is continuous, assuming Gaussian noise, it reduces to a regression loss.

Step 3: Planning (Line 4). Equipped with a classifier, we proceed to solve the RL problem (5), which constitutes the core of the proposed algorithm. As noted by Black et al. (2023); Fan et al. (2023), the diffusion model can be regarded as a special Markov Decision Process (MDP) with known transition dynamics. Thus, it is technically feasible to employ any off-the-shelf RL algorithms. In this work, inspired by (Clark et al., 2023; Prabhudesai et al., 2023), we employ direct back-propagation, which requires differentiable. If a classifier is not differentiable, we recommend using PPO-based methods (Black et al., 2023; Fan et al., 2023). Please refer to Appendix A for more details.

Below, we make several remarks regarding implementing **CTRL** in practice.

Remark 2 (Using classifier-free guidance to adjust guidance strength). *Throughout the fine-tuning process demonstrated in Algorithm 1, the guidance strength for the additional conditional control (i.e., y) is fixed at a specific γ (see the target conditional distribution (2)). However, we note that during inference, this guidance strength γ can be adjusted—either increased or decreased—using the classifier-free guidance technique. Details are deferred to Appendix C.*

³We categorize them as reconstruction guidance methods for simplicity. We note that there are many variants.

Remark 3 (Choice of exploratory distribution Π). *According to Theorem 1, it is desired to improve the coverage over $\mathcal{C} \times \mathcal{Y}$ during fine-tuning. For example, in practice, if \mathcal{Y} only takes several discrete values, we can sample $y \in \mathcal{Y}$ uniformly from these values as done in Section 6.*

4.3 SOURCE OF ERRORS IN CTRL

We discuss the potential sources of error that CTRL may encounter, which will be useful for comparison with existing methods in the next section. Additional limitations, such as computational cost, memory complexity, and the choice of guidance strength γ , are discussed in Appendix G.

Statistical error. Statistical errors arise during the training of a classifier $\hat{p}(y|x, c)$ from offline data while learning $p^\diamond(y|x, c)$. A typical statistical error is given by:

$$\mathbb{E}_{(x,c) \sim l^{\text{off}}} [\|\hat{p}(\cdot|x, c) - p^\diamond(\cdot|x, c)\|_1^2] = O(\text{Cap}(\mathcal{F})/n), \quad (7)$$

where $l^{\text{off}} \in \Delta(\mathcal{X} \times \mathcal{C})$ represents the distribution of offline data, and $\text{Cap}(\mathcal{F})$ denotes the size of the function class \mathcal{F} (Wainwright, 2019).

Model-misspecification error. Model-misspecification errors may occur during the learning of the classifier $p^\diamond(y|x, c)$ and in the augmented model if it fails to capture the optimal drift g^* .

Optimization error. Optimization errors may occur during both the classifier training step and the planning step.

5 ADDITIONAL COMPARISONS WITH EXISTING CONDITIONING METHODS

In this section, we further clarify the connections and comparisons between our algorithm and the existing methods.

5.1 COMPARISON TO CLASSIFIER GUIDANCE

We explore the advantages of CTRL over classifier guidance (Dhariwal and Nichol, 2021), while also offering theoretical insights that link the two approaches. Despite their distinct goals – classifier guidance is an inference-time technique, whereas our method fine-tunes an augmented diffusion model, there is a deep theoretical connection. This link is highlighted by our derivation of the analytical expression for the optimal drift in the RL problem (5) as below.

Lemma 2 (Bridging RL-based conditioning with classifier guidance). *The optimal drift term g^* for RL problem (5) has the following explicit solution:*

$$g^*(t, c, y, x_t) = f^{\text{pre}}(t, c, x_t) + \sigma^2(t) \nabla_{x_t} \log \mathbb{E}_{\mathbb{P}^{\text{pre}}(\cdot|x_t, c)} [(p^\diamond(y|x_T, c))^\gamma | x_t, c], \quad \forall t \in [0, T]$$

The proof of Lemma 2 is deferred to Appendix D.3. This lemma indicates that when $\gamma = 1$, the optimal drift g^* corresponds to the drift term obtained from Doob’s h-transform (i.e., Lemma 1), which is a precise used formula in classifier guidance. Despite the link to classifier guidance through Lemma 2, our algorithm is fundamentally different. Classifier guidance requires learning a predictor from x_t to y for every $t \in T$, leading to accumulated inaccuracies. In contrast, our algorithm directly solves the RL problem (5), avoiding the need for such predictors.

In Section 3.2.2, we explore **reconstruction guidance** methods that also aim to circumvent predicting y from x_t . These methods propose first mapping x_t to a denoised estimate $\hat{x}_T(x_t)$ and using this for further computations. However, this approximation can be imprecise, especially over longer time horizons. As shown in (Chung et al., 2022, Theorem 1), inherent approximation errors persist even without statistical, model-misspecification, or optimization errors. In contrast, our algorithm avoids such approximation errors.

5.2 COMPARISON TO CLASSIFIER-FREE GUIDANCE

We first show how CTRL leverages *conditional independence* to ease implementation, a feature absent in classifier-free guidance. Finally, we discuss the improvements regarding sample (statistical) efficiency.

5.2.1 LEVERAGING CONDITIONAL INDEPENDENCE, COMPOSITIONALLY VIA CTRL

We discuss two scenarios where our method outperforms the classifier-free approach by exploiting the conditional independence between inputs and additional controls.

Example 1 (Scenario $Y \perp C|X$). *If a new condition Y is conditionally independent of an existing condition C given X , meaning $p^\circ(y|x, c) = p(y|x)$. This allows CTRL to operate efficiently with just (x, y) pairs, avoiding the need for (c, x, y) triplets in the offline dataset.*

This scenario is common in practice. For example, when using the Stable Diffusion pre-trained model (Rombach et al., 2022), where X is an image and C is a text prompt, we may also want to condition the generations on Y , such as score functions like compressibility, aesthetic score, or color (Black et al., 2023). These scores depend solely on the image and are independent of the prompt, meaning $Y \perp C|X$. We further explore this scenario in our experimental analysis in Section 6.1.

Multi-task conditional generation. Multi-task conditional generation is a significant challenge, requiring the integration of multiple controls into pre-trained models. In the following example, we show how our method can be extended to handle this.

Example 2 (Scenario $Y_1 \perp Y_2|X, C$). *If two conditions, Y_1 and Y_2 , exhibit conditional independence given X and C , such that $\log p(y_1, y_2|x, c) = \log p(y_1|x, c) + \log p(y_2|x, c)$, the two classifiers can be trained separately using (c, x, y_1) and (c, x, y_2) triplets. Furthermore, if Y_1 and Y_2 are also independent of C given X (as in Example 1), the classifiers can be trained solely with (x, y_1) and (x, y_2) pairs, significantly simplifying dataset construction.*

This scenario is also common in practice. For instance, with the Stable Diffusion pre-trained model, where X is an image and C is a text prompt, additional attributes like Y_1 (compressibility) and Y_2 (color) depend only on the image, not the prompt. Thus, we can leverage the conditional independence of Y_1 and Y_2 from C given X to simplify the implementation of CTRL. The effectiveness of CTRL in this context is further validated experimentally in Section 6.2.

Can classifier-free guidance leverage conditional independence? The applicability of conditional independence in classifier-free guidance, which directly models $p_\gamma(\cdot|c, y)$, is uncertain. For instance, when $Y \perp C|X$ as in Example 1, our method only requires (x, y) pairs, while classifier-free guidance typically needs (c, x, y) triplets. when $Y_1 \perp Y_2|C, X$ as in Example 2, our approach utilizes triplets (c, x, y_1) and (c, x, y_2) . However, as far as we are concerned, quadruples (c, x, y_1, y_2) are necessary for classifier-free guidance, and acquiring such data at scale could pose a bottleneck.

5.2.2 STATISTICAL EFFICIENCY

We present the rationale for our approach being more sample-efficient than classifier-free guidance. Most importantly, we leverage a pre-trained model to sample from $p^{\text{pre}}(x|c)$, which is already trained on large datasets. This allows us to focus only on learning the classifier $p^\circ(y|x, c)$ from offline data. As a result, any statistical errors from the offline data affect only the classifier learning step (6). In contrast, classifier-free guidance attempts to model the entire distribution $p_\gamma(\cdot|c, y)$ directly from offline data. Therefore, our method is more sample-efficient by learning only the necessary components from the offline data.

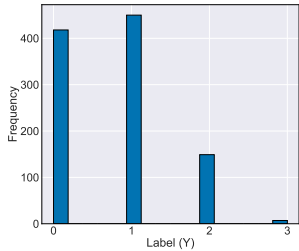
6 EXPERIMENTS

We compare CTRL with two baselines: (1) **Reconstruction Guidance**. It attempts to alleviate the approximation error of classifier guidance via reconstruction. (2) **Classifier-Free** guidance (Ho and Salimans, 2022)⁴. Both methods are widely used for conditioning generative models and are reviewed in detail in Section 3.2. For more detailed information on each experiment, such as dataset, architecture, and baselines, please refer to Appendix F.

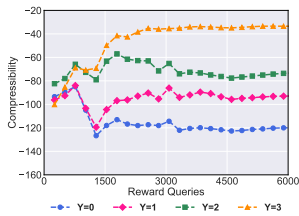
⁴Implementing the **Classifier-Free** baseline in our setting would require using the pre-trained diffusion model to augment x on certain c . Please refer to Appendix F.1 for details.

Experimental setup. For image experiments (Section 6.1, Section 6.2), we use Stable Diffusion v1.5 (Rombach et al., 2022) as the pre-trained model $p^{\text{pre}}(x|c)$, here c is a text prompt (e.g., “cat” or “dog”) and x is the corresponding image. For the additional control y , we validate compressibilities and aesthetic scores. We defer experiments for designing DNA enhancers to Appendix E.

6.1 IMAGE: CONDITIONAL GENERATION ON COMPRESSIBILITY



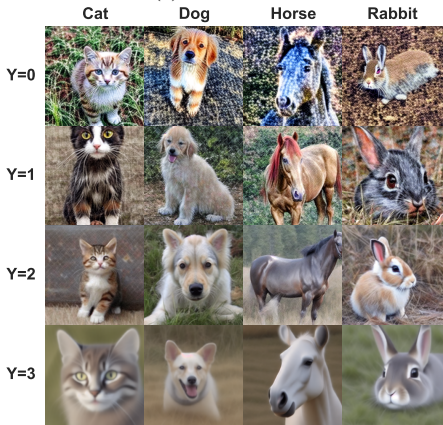
(a) Histogram.



(b) Training curves.

	Accuracy \uparrow	Macro F1 score \uparrow
Classifier-Free	0.33 ± 0.04	0.28
Reconstruction Guidance	0.45 ± 0.04	0.45
CTRL	1.0 ± 0.0	1.0

(c) Evaluation.



(d) Images generated by CTRL.

Figure 1: Results for conditioning on compressibility. Figure a plots the histogram of samples generated by the pre-trained diffusion model. Figure b shows the mean compressibility curves during fine-tuning, with four distinct lines representing each condition. It is evident that CTRL effectively aligns the generated samples with their target compressibility levels via fine-tuning. Table c provides evaluation metrics. Figure d shows images generated by a single model fine-tuned with CTRL.

We start by conditioning generations on their file sizes, specifically focusing on **compressibility**⁵. Denoting compressibility as **CP**, we define 4 compressibility labels as follows: $Y = 0 : \text{CP} < -110.0$; $Y = 1 : -110.0 \leq \text{CP} < -85.0$; $Y = 2 : -85.0 \leq \text{CP} < -60.0$; $Y = 3 : \text{CP} \geq -60.0$. Particularly, as depicted in Figure 1a, generating samples conditioned on $Y = 3$ is challenging due to the infrequent occurrence of such samples from the pre-trained model.

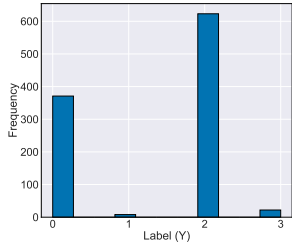
Results. We evaluate performance across four compressibility levels using the following steps: (1) generating samples conditioned on each $Y \in [0, 1, 2, 3]$; (2) verifying alignment between the generated samples and their conditions; and (3) calculating classification accuracy and macro F1 score. Table 1c presents evaluation statistics. Our results show that CTRL accurately generates samples for each condition, including the rare $Y = 3$ case from the pre-trained model (see Figure 1a), notably outperforming the baselines. Figure 1d showcases diverse images with correct compressibility levels for various prompts. Additional visualizations are available in Appendix F.6.

6.2 IMAGE: MULTI-TASK CONDITIONAL GENERATION

A more challenging setting involves multi-task conditional generation. In this experiment, in addition to compressibility, we *simultaneously* aim to condition the generations on their aesthetic pleasingness. Following prior research (Black et al., 2023; Fan et al., 2023; Uehara et al., 2024), we employ an aesthetic scorer implemented as a linear MLP on top of the CLIP embeddings (Radford et al., 2021), which is trained on more than 400k human evaluations.

⁵Unlike standard tasks in classifier/reconstruction guidance (Chung et al., 2022), this score is non-differentiable w.r.t. images. This score function is only dependent on the image itself.

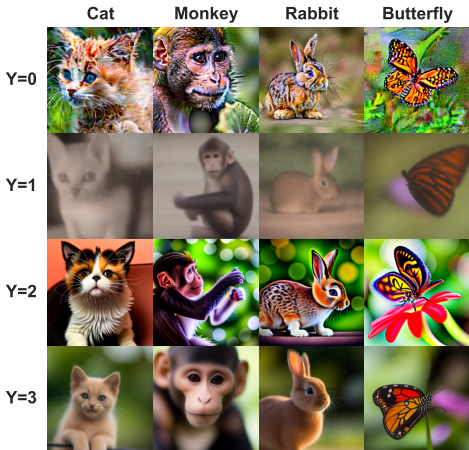
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539



(a) Histogram.

	Task	Accuracy \uparrow	Macro F1 \uparrow
Classifier-Free	CP	0.52 ± 0.04	0.39
	AS	0.59 ± 0.04	0.55
Reconstruction Guidance	CP	0.61 ± 0.04	0.55
	AS	0.66 ± 0.04	0.62
CTRL	CP	0.94 ± 0.02	0.94
	AS	0.93 ± 0.02	0.93

(b) Evaluation.



(c) Images generated by CTRL.

Figure 2: Results for multi-task conditioning. Figure a plots the histogram of samples generated by the pre-trained diffusion model. Table b presents the evaluation statistics. Figure c displays images generated by a **single** model fine-tuned with CTRL.

In this experiment, by leveraging conditional independence of Y_1 and Y_2 given X (see Example 2), we aim to fine-tune the diffusion model to generate samples with *compositional* conditions. Specifically, denoting compressibility as **CP** and aesthetic score as **AS**, we define four compositional conditions as follows: $Y = 0 : \mathbf{AS} < 5.7, \mathbf{CP} < -70$; $Y = 1 : \mathbf{AS} < 5.7, \mathbf{CP} \geq -70$; $Y = 2 : \mathbf{AS} \geq 5.7, \mathbf{CP} < -70$; $Y = 3 : \mathbf{AS} \geq 5.7, \mathbf{CP} \geq -70$. Particularly, as depicted in Figure 2a, generating samples conditioned on $Y = 1$ or $Y = 3$ is challenging due to the infrequent occurrence of such samples from the pre-trained model.

Results. We follow the evaluation procedure outlined in Section 6.1, with results summarized in Table 2b, demonstrating that CTRL outperforms all baselines across both tasks by a big margin. Notably, CTRL can generate samples rarely produced by the pre-trained model with over 90% accuracy, particularly for the desired class $Y = 3$ (highly aesthetic images with high compressibility). Producing such images is challenging as aesthetically pleasing images typically require more storage and thus have low compressibility. Generated images are displayed in Figure 2c. More visualizations are provided in Appendix F.6.

7 CONCLUSION

We introduce a provable RL-based fine-tuning approach for conditioning pre-trained diffusion models on additional controls. Compared to classifier-free guidance, our proposed method uses the offline dataset more efficiently and is able to leverage the conditional independence assumption, thereby greatly simplifying the construction of the offline dataset. Our approach is empirically validated across three settings: image generation conditioned on a new task, image generation conditioned on the composition of two new tasks, and biological sequence design.

Reproducibility Statement. We submit the code for our image experiments as supplementary materials. Complete proofs of our theoretical results are provided in Appendix D. Detailed information about our experiments, including dataset descriptions, model architecture, and baseline implementations, can be found in Appendix F.

REFERENCES

Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications* 12(3), 313–326.

- 540 Avdeyev, P., C. Shi, Y. Tan, K. Dudnyk, and J. Zhou (2023). Dirichlet diffusion score model for
541 biological sequence generation. In *International Conference on Machine Learning*, pp. 1276–1301.
542 PMLR.
- 543 Avsec, Ž., V. Agarwal, D. Visentin, J. R. Ledsam, A. Grabska-Barwinska, K. R. Taylor, Y. Assael,
544 J. Jumper, P. Kohli, and D. R. Kelley (2021). Effective gene expression prediction from sequence
545 by integrating long-range interactions. *Nature methods* 18(10), 1196–1203.
546
- 547 Bansal, A., H.-M. Chu, A. Schwarzschild, S. Sengupta, M. Goldblum, J. Geiping, and T. Goldstein
548 (2023). Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on*
549 *Computer Vision and Pattern Recognition*, pp. 843–852.
- 550 Black, K., M. Janner, Y. Du, I. Kostrikov, and S. Levine (2023). Training diffusion models with
551 reinforcement learning.
552
- 553 Brooks, T., A. Holynski, and A. A. Efros (2023). Instructpix2pix: Learning to follow image editing
554 instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern*
555 *Recognition*, pp. 18392–18402.
- 556 Castillo-Hair, S. M. and G. Seelig (2021). Machine learning for designing next-generation mrna
557 therapeutics. *Accounts of Chemical Research* 55(1), 24–34.
558
- 559 Chen, T., B. Xu, C. Zhang, and C. Guestrin (2016). Training deep nets with sublinear memory cost.
560 *arXiv preprint arXiv:1604.06174*.
- 561 Chung, H., J. Kim, M. T. McCann, M. L. Klasky, and J. C. Ye (2022). Diffusion posterior sampling
562 for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*.
- 563 Clark, K., P. Vicol, K. Swersky, and D. J. Fleet (2023). Directly fine-tuning diffusion models on
564 differentiable rewards. *arXiv preprint arXiv:2309.17400*.
565
- 566 Company, C., M. J. Schmitt, Y. Dramaretska, M. Serresi, S. Kertalli, B. Jiang, J.-A. Yin, A. Aguzzi,
567 I. Barozzi, and G. Gargiulo (2024). Logical design of synthetic cis-regulatory dna for genetic
568 tracing of cell identities and state changes. *Nature Communications* 15(1), 897.
- 569 de Almeida, B. P., C. Schaub, M. Pagani, S. Secchia, E. E. Furlong, and A. Stark (2024). Targeted
570 design of synthetic enhancers for selected tissues in the drosophila embryo. *Nature* 626(7997),
571 207–211.
572
- 573 Dhariwal, P. and A. Nichol (2021). Diffusion models beat gans on image synthesis. *Advances in*
574 *neural information processing systems* 34, 8780–8794.
- 575 Epstein, D., A. Jabri, B. Poole, A. Efros, and A. Holynski (2023). Diffusion self-guidance for
576 controllable image generation. *Advances in Neural Information Processing Systems* 36, 16222–
577 16239.
- 578 Fan, Y., O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and
579 K. Lee (2023). Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models.
580 *arXiv preprint arXiv:2305.16381*.
581
- 582 Finzi, M. A., A. Boral, A. G. Wilson, F. Sha, and L. Zepeda-Núñez (2023). User-defined event
583 sampling and uncertainty quantification in diffusion models for physical dynamical systems. In
584 *International Conference on Machine Learning*, pp. 10136–10152. PMLR.
- 585 Ghari, P. M., A. Tseng, G. Eraslan, R. Lopez, T. Biancalani, G. Scalia, and E. Hajiramezani (2023).
586 Generative flow networks assisted biological sequence editing. In *NeurIPS 2023 Generative AI*
587 *and Biology (GenBio) Workshop*.
- 588 Giannone, G., A. Srivastava, O. Winther, and F. Ahmed (2024). Aligning optimization trajectories with
589 diffusion models for constrained design generation. *Advances in Neural Information Processing*
590 *Systems* 36.
591
- 592 Gosai, S. J., R. I. Castro, N. Fuentes, J. C. Butts, S. Kales, R. R. Noche, K. Mouri, P. C. Sabeti, S. K.
593 Reilly, and R. Tewhey (2023). Machine-guided design of synthetic cell type-specific cis-regulatory
elements. *bioRxiv*.

- 594 Gruslys, A., R. Munos, I. Danihelka, M. Lanctot, and A. Graves (2016). Memory-efficient backprop-
595 agation through time. *Advances in neural information processing systems* 29.
- 596
- 597 Guo, C., G. Pleiss, Y. Sun, and K. Q. Weinberger (2017). On calibration of modern neural networks.
598 In *International conference on machine learning*, pp. 1321–1330. PMLR.
- 599
- 600 Guo, Y., H. Yuan, Y. Yang, M. Chen, and M. Wang (2024). Gradient guidance for diffusion models:
601 An optimization perspective. *arXiv preprint arXiv:2404.14743*.
- 602
- 603 Han, X., H. Zheng, and M. Zhou (2022). Card: Classification and regression diffusion models.
604 *Advances in Neural Information Processing Systems* 35, 18100–18115.
- 605
- 606 Ho, J., A. Jain, and P. Abbeel (2020). Denoising diffusion probabilistic models. *Advances in neural
607 information processing systems* 33, 6840–6851.
- 608
- 609 Ho, J. and T. Salimans (2022). Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.
- 610
- 611 Ho, J., T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet (2022). Video diffusion
612 models. *Advances in Neural Information Processing Systems* 35, 8633–8646.
- 613
- 614 Hu, E. J., Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen (2021). Lora:
615 Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- 616
- 617 Huang, K., T. Fu, W. Gao, Y. Zhao, Y. H. Roohani, J. Leskovec, C. W. Coley, C. Xiao, J. Sun, and
618 M. Zitnik (2021). Therapeutics data commons: Machine learning datasets and tasks for drug
619 discovery and development. In *Thirty-fifth Conference on Neural Information Processing Systems
620 Datasets and Benchmarks Track (Round 1)*.
- 621
- 622 Kendall, M. G. (1948). Rank correlation methods.
- 623
- 624 Lal, A., D. Garfield, T. Biancalani, and G. Eraslan (2024). reglm: Designing realistic regulatory dna
625 with autoregressive language models. *bioRxiv*, 2024–02.
- 626
- 627 Lee, K., H. Liu, M. Ryu, O. Watkins, Y. Du, C. Boutilier, P. Abbeel, M. Ghavamzadeh, and S. S. Gu
628 (2023). Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*.
- 629
- 630 Li, Y. and M. van der Schaar (2023). On error propagation of diffusion models. In *The Twelfth
631 International Conference on Learning Representations*.
- 632
- 633 Loshchilov, I. and F. Hutter (2019). Decoupled weight decay regularization. In *International
634 Conference on Learning Representations*.
- 635
- 636 Murray, N., L. Marchesotti, and F. Perronnin (2012). Ava: A large-scale database for aesthetic visual
637 analysis. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 2408–2415.
638 IEEE.
- 639
- 640 Nichol, A., P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen
641 (2021). Glide: Towards photorealistic image generation and editing with text-guided diffusion
642 models. *arXiv preprint arXiv:2112.10741*.
- 643
- 644 Prabhudesai, M., A. Goyal, D. Pathak, and K. Fragkiadaki (2023). Aligning text-to-image diffusion
645 models with reward backpropagation. *arXiv preprint arXiv:2310.03739*.
- 646
- 647 Radford, A., J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin,
648 J. Clark, G. Krueger, and I. Sutskever (2021). Learning transferable visual models from natural
649 language supervision. *arXiv preprint arXiv:2103.00020*.
- 650
- 651 Ramesh, A., M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever (2021). Zero-
652 shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831.
653 Pmlr.
- 654
- 655 Rogers, L. C. G. and D. Williams (2000). *Diffusions, Markov processes and martingales: Volume 2,
656 Itô calculus*, Volume 2. Cambridge university press.

- 648 Rombach, R., A. Blattmann, D. Lorenz, P. Esser, and B. Ommer (2022). High-resolution image
649 synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer
650 vision and pattern recognition*, pp. 10684–10695.
- 651 Saharia, C., W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes,
652 B. Karagol Ayan, T. Salimans, et al. (2022). Photorealistic text-to-image diffusion models with deep
653 language understanding. *Advances in neural information processing systems* 35, 36479–36494.
- 654 Shreve, S. E. et al. (2004). *Stochastic calculus for finance II: Continuous-time models*, Volume 11.
655 Springer.
- 656 Sohl-Dickstein, J., E. Weiss, N. Maheswaranathan, and S. Ganguli (2015). Deep unsupervised
657 learning using nonequilibrium thermodynamics. In *International conference on machine learning*,
658 pp. 2256–2265. PMLR.
- 659 Song, J., C. Meng, and S. Ermon (2020). Denoising diffusion implicit models. *arXiv preprint
660 arXiv:2010.02502*.
- 661 Song, J., A. Vahdat, M. Mardani, and J. Kautz (2022). Pseudoinverse-guided diffusion models for
662 inverse problems. In *International Conference on Learning Representations*.
- 663 Song, Y., J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole (2020). Score-based
664 generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.
- 665 Taskiran, I. I., K. I. Spanier, H. Dickmanken, N. Kempynck, A. Panıkova, E. C. Eksi, G. Hulselmans,
666 J. N. Ismail, K. Theunis, R. Vandepoel, et al. (2024). Cell-type-directed design of synthetic
667 enhancers. *Nature* 626(7997), 212–220.
- 668 Uehara, M., Y. Zhao, K. Black, E. Hajiramezanali, G. Scalia, N. L. Diamant, A. M. Tseng, T. Biancalani,
669 and S. Levine (2024). Fine-tuning of continuous-time diffusion models as entropy-
670 regularized control. *arXiv preprint arXiv:2402.15194*.
- 671 Uehara, M., Y. Zhao, K. Black, E. Hajiramezanali, G. Scalia, N. L. Diamant, A. M. Tseng, S. Levine,
672 and T. Biancalani (2024). Feedback efficient online fine-tuning of diffusion models. *arXiv preprint
673 arXiv:2402.16359*.
- 674 Uehara, M., Y. Zhao, E. Hajiramezanali, G. Scalia, G. Eraslan, A. Lal, S. Levine, and T. Biancalani
675 (2024). Bridging model-based optimization and generative modeling via conservative fine-tuning
676 of diffusion models. *arXiv preprint arXiv:2405.19673*.
- 677 Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint*, Volume 48.
678 Cambridge university press.
- 679 Wu, X., K. Sun, F. Zhu, R. Zhao, and H. Li (2023). Better aligning text-to-image models with human
680 preference. *arXiv preprint arXiv:2303.14420*.
- 681 Xie, E., L. Yao, H. Shi, Z. Liu, D. Zhou, Z. Liu, J. Li, and Z. Li (2023). DiffFit: Unlocking
682 transferability of large diffusion models via simple parameter-efficient fine-tuning. In *Proceedings
683 of the IEEE/CVF International Conference on Computer Vision*, pp. 4230–4239.
- 684 Xu, J., X. Liu, Y. Wu, Y. Tong, Q. Li, M. Ding, J. Tang, and Y. Dong (2023). Imagereward: Learning
685 and evaluating human preferences for text-to-image generation. *arXiv preprint arXiv:2304.05977*.
- 686 Yellapragada, S., A. Graikos, P. Prasanna, T. Kurc, J. Saltz, and D. Samaras (2024). Pathldm: Text
687 conditioned latent diffusion model for histopathology. In *Proceedings of the IEEE/CVF Winter
688 Conference on Applications of Computer Vision*, pp. 5182–5191.
- 689 Zhang, K., J. D. Hocker, M. Miller, X. Hou, J. Chiou, O. B. Poirion, Y. Qiu, Y. E. Li, K. J. Gaulton,
690 A. Wang, et al. (2021). A single-cell atlas of chromatin accessibility in the human genome.
691 *Cell* 184(24), 5985–6001.
- 692 Zhang, L., A. Rao, and M. Agrawala (2023). Adding conditional control to text-to-image diffusion
693 models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp.
694 3836–3847.

702 Zhang, Z., L. Han, A. Ghosh, D. N. Metaxas, and J. Ren (2023). Sine: Single image editing with
703 text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision*
704 *and Pattern Recognition*, pp. 6027–6037.

705
706 Zhao, S., D. Chen, Y.-C. Chen, J. Bao, S. Hao, L. Yuan, and K.-Y. K. Wong (2024). Uni-controlnet:
707 All-in-one control to text-to-image diffusion models. *Advances in Neural Information Processing*
708 *Systems* 36.

709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755

A PLANNING ALGORITHM

Algorithm 2 Direct back-propagation for conditioning

- 1: **Input** Batch size n , Learning rate η , Discretization step Δt , Exploratory distribution $\Pi \in \Delta(\mathcal{C} \times \mathcal{Y})$.
- 2: **Initialize:** $\psi = [\{\theta^{\text{pre}}\}^\top, \mathbf{0}^\top]$
- 3: **for** $i \leftarrow 1$ to S **do**
- 4: We obtain n trajectories

$$\{X_0^{(k)}, \dots, X_T^{(k)}\}_{k=1}^n, \{Z_0^{(k)}, \dots, Z_T^{(k)}\}_{k=1}^n.$$

following $(C^{(k)}, Y^{(k)}) \sim \Pi(\cdot)$, $X_0^{(k)} \sim \mathcal{N}(0, I_d)$, $Z_0 = 0$, and

$$\begin{aligned} X_t^{(k)} &= X_{t-1}^{(k)} + g(t-1, C^{(k)}, Y^{(k)}, X_{t-1}^{(k)}; \psi_i) \Delta t + \sigma(t) (\Delta w_t), \quad \Delta w_t \sim \mathcal{N}(0, (\Delta t)^2), \\ Z_t^{(k)} &= Z_{t-1}^{(k)} + \frac{\|g(t-1, C^{(k)}, Y^{(k)}, X_{t-1}^{(k)}; \psi_i) - f^{\text{pre}}(t-1, C^{(k)}, X_{t-1}^{(k)}; \theta^{(i)})\|^2}{2\sigma^2(t-1)} \Delta t. \end{aligned}$$

- 5: Update a parameter:

$$\psi_{i+1} = \psi_i + \eta \nabla_\psi \left\{ \frac{1}{n} \sum_{k=1}^n \left[\gamma \log \hat{p}(Y^{(k)} | X_T^{(k)}, C^{(k)}) - Z_T^{(k)} \right] \right\} \Big|_{\psi=\psi_i},$$

- 6: **end for**

- 7: **Output:** Parameter ψ_S
-

Inspired by (Clark et al., 2023; Prabhudesai et al., 2023), our planning algorithm, listed in Algorithm 2, is based on direct back-propagation. This method is iterative in nature. During each iteration, we: (1) compute the expectation over trajectories ($\mathbb{E}_{x_0, T \sim \mathbb{P}^{g(\cdot; \psi)}}$) using discretization techniques such as Euler-Maruyama; (2) directly optimize the KL-regularized objective function with respect to parameters of the augmented model (i.e., ψ).

In practice, such computation might be memory-intensive when there are numerous discretization steps and the diffusion models have a large number of parameters. This is because gradients would need to be back-propagated through the diffusion process. To improve computational efficiency, we recommend employing specific techniques, including (a) only fine-tuning LoRA (Hu et al., 2021) modules instead of the full diffusion weights, (b) employing gradient checkpointing (Gruslys et al., 2016; Chen et al., 2016) to conserve memory, and (c) randomly truncating gradient back-propagation to avoid computing through all diffusion steps (Clark et al., 2023; Prabhudesai et al., 2023).

Remark 4 (PPO). *In Algorithm 1, we employ direct back-propagation (i.e., Algorithm 2) for planning (i.e., solving the RL problem (5)), which necessarily demands the differentiability of the classifier. If the classifier is non-differentiable, we suggest using Proximal Policy Optimization (PPO) for planning, such as Black et al. (2023); Fan et al. (2023). Other parts remain unchanged.*

B TRAINING DIFFUSION MODELS

In standard diffusion models, given a training dataset $\{x^{(j)}\} \sim p_{\text{data}}(\cdot)$, the goal is to construct a transport that maps noise distribution and data distribution $p_{\text{data}} \in \Delta(\mathcal{X})$ ($\mathcal{X} = \mathbb{R}^d$). More specifically, suppose that we have an SDE ⁶:

$$dx_t = f(t, x_t; \theta) dt + \sigma(t) dw_t, \quad (8)$$

where $f : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a drift coefficient, $\sigma : [0, T] \rightarrow \mathbb{R}$ is a diffusion coefficient, w_t is d -dimensional Brownian motion, and initial state $x_0 \sim p_{\text{ini}}$ where $p_{\text{ini}} \in \Delta(\mathcal{X})$ denotes the

⁶In standard diffusion models, the direction is reversed, i.e., x_T corresponds to the noise distribution.

initial distribution. By denoting the marginal distribution at time T by $p_T^\theta(x)$, a standard goal in training diffusion models is to learn the parameter θ so that $p_T^\theta(x) \approx p_{\text{data}}$. This means we can (approximately) sample from p_{data} by following the SDE (8) from 0 to T .

To train diffusion models, we first introduce a (fixed) *forward* reference SDE, which gradually adds noise to p_{data} :

$$dz_t = \bar{f}(t, z_t)dt + \bar{\sigma}(t)dw_t, \quad z_0 \sim p_{\text{data}}, \quad (9)$$

where $\bar{f} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a drift coefficient, $\bar{\sigma} : [0, T] \rightarrow \mathbb{R}$ is a diffusion coefficient. An example is the classical denoising diffusion model (Ho et al., 2020), also known as the variance-preserving (VP) process, which sets $\bar{f} = -0.5z_t$, $\bar{\sigma} = 1$.

Now, we consider the time-reversal SDE (Anderson, 1982), which reverses the direction of SDE while keeping the marginal distribution, as follows:

$$dx_t = \{-\bar{f}(T-t, x_t) + \nabla \log q_{T-t}(x_t)\} dt + \bar{\sigma}(T-t)dw_t, \quad x_0 \sim \mathcal{N}(0, I_d). \quad (10)$$

Here, $q_t(\cdot)$ denotes the marginal distribution at time t for the distribution induced by the reference SDE, and $\nabla \log q_{T-t}(x_t)$ means a derivative w.r.t. x_t , which is often referred to as the *score function*. Furthermore, when the time horizon T is sufficiently large, z_t follows Gaussian noise distribution $\mathcal{N}(0, I_d)$. Hence, if we could learn the score function, by following the SDE (10) starting from Gaussian noise, we can sample from the data distribution.

Then, we aim to learn the score function from the data. By comparing the time-reversal SDE with the original SDE, a natural parameterization is:

$$f(t, x_t; \theta) = -\bar{f}(T-t, x_t) + s(T-t, x_t; \theta), \quad \sigma(t) = \bar{\sigma}(T-t),$$

where $s(T-t, x_t; \theta)$ is the parametrized neural network introduced to approximate the score function $\nabla \log q_{T-t}(x_t)$. Here, we can leverage the analytical form of the conditional distribution $q_{T-t|0}(\cdot|\cdot)$ (which is a Gaussian distribution derived from the reference SDE). This approach enables us to tackle the approximation problem via regression:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \mathbb{E}_{t \in [0, T], z_0 \sim p_{\text{data}}, z_t \sim q_{t|0}(z_0)} \left[\lambda(t) \left\| s(t, z_t; \theta) - \nabla_{z_t} \log q_{t|0}(z_t|z_0) \right\|^2 \right], \quad (11)$$

where $\lambda : [0, T] \rightarrow \mathbb{R}$ is a weighting function.

C INFERENCE TECHNIQUE IN CLASSIFIER-FREE GUIDANCE

Although the fine-tuning process sets the guidance level for the additional conditioning (i.e., y) at a specific γ , classifier-free guidance makes it possible to adjust the guidance strength freely during inference. Recall that the augmented model is constructed as: $g(t, c, y, x; \psi)$ where $\psi = [\theta^\top, \phi^\top]^\top$. Suppose we have obtained a drift term \hat{g} , parametrized by $\hat{\psi} = [\hat{\theta}^\top, \hat{\phi}^\top]^\top$ from running Algorithm 1. In inference, we may alter the guidance levels by using the following drift term in the SDE (3)

$$g_{\gamma_1, \gamma_2}(t, c, y, x_t) = \underbrace{g(t, \emptyset, \emptyset, x_t; \hat{\psi}) + \gamma_1(g(t, c, \emptyset, x_t; \hat{\psi}) - g(t, \emptyset, \emptyset, x_t; \hat{\psi}))}_{\text{Term 1: pre-trained diffusion model conditioned on } \mathcal{C}} + \underbrace{\gamma_2(g(t, c, y, x_t; \hat{\psi}) - g(t, c, \emptyset, x_t; \hat{\psi}))}_{\text{Term 2: additional conditioning on } \mathcal{Y}}$$

where \emptyset indicates the unconditional on \mathcal{Y} or on \mathcal{C} . In the above, both γ_1 and γ_2 do not necessarily need to equal γ . They can be adjusted respectively to reflect guidance strength levels for two conditions.

D PROOFS

D.1 IMPORTANT LEMMAS

We first introduce several important lemmas to prove our main statement.

864 First, recall that $\mathbb{P}^g(\cdot|c, y)$ is the induced distribution by the SDE:

$$865 \quad dx_t = g(t, c, y, x_t)dt + \sigma(t)dw_t, \quad x_0 = x_{\text{ini}}$$

867 over \mathcal{K} conditioning on c and y . Similarly, denote $\mathbb{P}^{\text{pre}}(\cdot|c)$ by the induced distribution by the SDE:

$$868 \quad dx_t = f^{\text{pre}}(t, c, x_t)dt + \sigma(t)dw_t, \quad x_0 = x_{\text{ini}}$$

870 over \mathcal{K} conditioning on c .

871 **Lemma 3** (KL-constrained reward). *The objective function in (5) is equivalent to*

$$872 \quad \text{obj} = \mathbb{E}_{(c,y) \sim \Pi, \mathbb{P}^g(\cdot|c,y)} [\gamma \log p^\diamond(y|x_T, c) - \text{KL}(\mathbb{P}^g(\cdot|c, y) \parallel \mathbb{P}^{\text{pre}}(\cdot|c))]. \quad (12)$$

875 *Proof.* We calculate the KL divergence of \mathbb{P}^g and \mathbb{P}^{pre} as below

$$876 \quad \text{KL}(\mathbb{P}^g(\cdot|c, y) \parallel \mathbb{P}^{\text{pre}}(\cdot|c)) = \mathbb{E}_{x_0:T \sim \mathbb{P}^g(\cdot|c,y)} \left[\int_0^T \frac{1}{2} \frac{\|g(t, c, y, x_t) - f^{\text{pre}}(t, c, x_t)\|^2}{\sigma^2(t)} dt \right]. \quad (13)$$

880 This is because

$$881 \quad \begin{aligned} & \text{KL}(\mathbb{P}^g(\cdot|c, y) \parallel \mathbb{P}^{\text{pre}}(\cdot|c)) \\ &= \mathbb{E}_{\mathbb{P}^g(\cdot|c,y)} \left[\frac{d\mathbb{P}^g(\cdot|c, y)}{d\mathbb{P}^{\text{pre}}(\cdot|c)} \right] \\ &= \mathbb{E}_{\mathbb{P}^g(\cdot|c,y)} \left[\int_0^T \frac{1}{2} \frac{\|g(t, c, y, x_t) - f^{\text{pre}}(t, c, x_t)\|^2}{\sigma^2(t)} dt + \int_0^T \{g(t, c, y, x_t) - f^{\text{pre}}(t, c, x_t)\} dw_t \right] \\ & \hspace{15em} \text{(Girsanov theorem)} \\ &= \mathbb{E}_{\mathbb{P}^g(\cdot|c,y)} \left[\int_0^T \frac{1}{2} \frac{\|g(t, c, y, x_t) - f^{\text{pre}}(t, c, x_t)\|^2}{\sigma^2(t)} dt \right]. \quad \text{(Martingale property of It\^o integral)} \end{aligned}$$

892 Therefore, the objective function in (5) is equivalent to

$$893 \quad \text{obj} = \mathbb{E}_{(c,y) \sim \Pi, \mathbb{P}^g(\cdot|c,y)} [\gamma \log p^\diamond(y|x_T, c) - \text{KL}(\mathbb{P}^g \parallel \mathbb{P}^{\text{pre}})]. \quad (14)$$

896 \square

897 **Optimal value function.** For the RL problem (5), it is beneficial to introduce the optimal value function $v_t^*(x|c, y)$ at any time $t \in [0, T]$, given $x_t = x$, conditioned on parameters c and y defined as:

$$901 \quad v_t^*(x|c, y) = \max_g \mathbb{E} \left[\gamma \log p^\diamond(y|x_T, c) - \frac{1}{2} \int_t^T \frac{\|f^{\text{pre}}(s, c, x_s) - g(s, c, y, x_s)\|^2}{\sigma^2(s)} ds \mid x_t = x, c, y \right]. \quad (15)$$

905 Specifically, we note that $v_T(x|c, y) = \gamma \log p^\diamond(y|x, c)$ represents the terminal reward function (i.e., a loglikelihood in our MDP), while v_0 represents the original objective function (5) that integrates the entire trajectory's KL divergence along with the terminal reward.

909 Below we derive the optimal value function in analytical form.

910 **Lemma 4** (Feynman–Kac Formulation). *At any time $t \in [0, T]$, given $x_t = x$, and conditioned on c and y , we have the optimal value function $v_t^*(x|c, y)$ (induced by the optimal drift term g^*) as follows*

$$911 \quad \exp(v_t^*(x|c, y)) = \mathbb{E}_{\mathbb{P}^{\text{pre}}(\cdot|c)} [(p^\diamond(y|x_T, c))^\gamma | x_t = x, c].$$

914 *Proof.* From the Hamilton–Jacobi–Bellman (HJB) equation, we have

$$915 \quad \max_u \left\{ \frac{\sigma^2(t)}{2} \sum_i \frac{d^2 v_t^*(x|c, y)}{dx^{[i]} dx^{[i]}} + g \cdot \nabla v_t^*(x|c, y) + \frac{dv_t^*(x|c, y)}{dt} - \frac{\|g - f^{\text{pre}}\|_2^2}{2\sigma^2(t)} \right\} = 0. \quad (16)$$

where $x^{[i]}$ is a i -th element in x . Hence, by simple algebra, we can prove that the optimal drift term satisfies

$$g^*(t, c, y, x) = f^{\text{pre}}(t, c, x) + \sigma^2(t) \nabla v_t^*(x|c, y).$$

By plugging the above into the HJB equation (16), we get

$$\frac{\sigma^2(t)}{2} \sum_i \frac{d^2 v_t^*(x|c, y)}{dx^{[i]} dx^{[i]}} + f^{\text{pre}} \cdot \nabla v_t^*(x|c, y) + \frac{dv_t^*(x|c, y)}{dt} + \frac{\sigma^2(t) \|\nabla v_t^*(x|c, y)\|_2^2}{2} = 0, \quad (17)$$

which characterizes the optimal value function. Now, using (17), we can show

$$\begin{aligned} & \frac{\sigma^2(t)}{2} \sum_i \frac{d^2 \exp(v_t^*(x|c, y))}{dx^{[i]} dx^{[i]}} + f^{\text{pre}} \cdot \nabla \exp(v_t^*(x|c, y)) + \frac{d \exp(v_t^*(x|c, y))}{dt} \\ &= \exp(v_t^*(x|c, y)) \times \left\{ \frac{\sigma^2(t)}{2} \sum_i \frac{d^2 v_t^*(x|c, y)}{dx^{[i]} dx^{[i]}} + f^{\text{pre}} \cdot \nabla v_t^*(x|c, y) + \frac{dv_t^*(x|c, y)}{dt} + \frac{\sigma^2(t) \|\nabla v_t^*(x|c, y)\|_2^2}{2} \right\} \\ &= 0. \end{aligned}$$

Therefore, to summarize, we have

$$\frac{\sigma^2(t)}{2} \sum_i \frac{d^2 \exp(v_t^*(x|c, y))}{dx^{[i]} dx^{[i]}} + f^{\text{pre}} \cdot \nabla \exp(v_t^*(x|c, y)) + \frac{d \exp(v_t^*(x|c, y))}{dt} = 0, \quad (18)$$

$$v_T^*(x|c, y) = \gamma \log p^\diamond(y|x, c). \quad (19)$$

Finally, by invoking the Feynman-Kac formula (Shreve et al., 2004), we obtain the conclusion:

$$\exp(v_t^*(x|c, y)) = \mathbb{E}_{\mathbb{P}^{\text{pre}}(\cdot|x_t, c)} [(p^\diamond(y|x_T, c))^\gamma | x_t = x, c].$$

□

D.2 PROOF OF THEOREM 1

Firstly, we aim to show that the optimal conditional distribution over \mathcal{K} on c and y (i.e., $\mathbb{P}^{g^*}(\tau|c, y)$) is equivalent to

$$\frac{\mathbb{P}^{\text{pre}}(\tau|c)(p^\diamond(y|x_T, c))^\gamma}{C(c, y)}, \quad C(c, y) := \exp(v_0^*(x_0|c, y)).$$

To do that, we need to check that the above is a valid distribution first. This is indeed valid because the above is decomposed into

$$\underbrace{\frac{(p^\diamond(y|x_T, c))^\gamma \cdot \mathbb{P}^{\text{pre}}(x_T|c)}{C(c, y)}}_{(\alpha 1)} \times \underbrace{\mathbb{P}^{\text{pre}}(\tau|c, x_T)}_{(\alpha 2)}, \quad (20)$$

and both $(\alpha 1)$, $(\alpha 2)$ are valid distributions. Especially, for the term $(\alpha 1)$, we observe

$$C(c, y) = \int (p^\diamond(y|x_T, c))^\gamma d\mathbb{P}^{\text{pre}}(x_T|c) = \mathbb{E}_{\mathbb{P}^{\text{pre}}(\cdot|c)} [(p^\diamond(y|x_T, c))^\gamma] = \exp(v_0^*(x_0|c, y)).$$

(cf. Lemma 4)

Now, after checking (20) is a valid distribution, we calculate the KL divergence:

$$\begin{aligned} & \text{KL} \left(\mathbb{P}^{g^*}(\tau|c, y) \left\| \frac{\mathbb{P}^{\text{pre}}(\tau|c)(p^\diamond(y|x_T, c))^\gamma}{C(c, y)} \right. \right) \\ &= \text{KL}(\mathbb{P}^{g^*}(\tau|c, y) \| \mathbb{P}^{\text{pre}}(\tau|c)) - \mathbb{E}_{\mathbb{P}^{g^*}(\cdot|c, y)} [\gamma \log p^\diamond(y|x_T, c) - \log C(c, y)] \\ &= \mathbb{E}_{\mathbb{P}^{g^*}(\cdot|c, y)} \left[\left\{ \int_0^T \frac{1}{2} \frac{\|g^*(t, c, y, x_t) - f^{\text{pre}}(t, c, x_t)\|^2}{\sigma^2(t)} dt - \gamma \log p^\diamond(y|x_T, c) + \log C(c, y) \right\} \right] \\ & \hspace{15em} \text{(cf. KL divergence (13))} \\ &= -v_0^*(x_0|c, y) + \log C(c, y). \hspace{10em} \text{(Definition of optimal value function)} \end{aligned}$$

Therefore,

$$\text{KL} \left(\mathbb{P}^{g^*}(\tau|c, y) \parallel \frac{\mathbb{P}^{\text{pre}}(\tau|c)(p^\diamond(y|x_T, c))^\gamma}{C(c, y)} \right) = -v_0^*(x_0|c, y) + \log C(c, y) = 0.$$

Hence,

$$\mathbb{P}^{g^*}(\tau|c, y) = \frac{\mathbb{P}^{\text{pre}}(\tau|c)(p^\diamond(y|x_T, c))^\gamma}{C(c, y)}.$$

Marginal distribution at t . Finally, consider the marginal distribution at t . By marginalizing before t , we get

$$\mathbb{P}^{\text{pre}}(\tau_{[t, T]}|c) \times (p^\diamond(y|x_T, c))^\gamma / C(c, y).$$

Next, by marginalizing after t ,

$$\mathbb{P}_t^{\text{pre}}(x|c) / C(c, y) \times \mathbb{E}_{\mathbb{P}^{\text{pre}}(\cdot|c)}[(p^\diamond(y|x_T, c))^\gamma | x_t = x, c].$$

Using Feynman–Kac formulation in Lemma 4, this is equivalent to

$$\mathbb{P}_t^{\text{pre}}(x|c) \exp(v_t^*(x|c, y)) / C(c, y).$$

Marginal distribution at T . We marginalize before T . We have the following

$$\mathbb{P}_T^{\text{pre}}(x|c)(p^\diamond(y|x_T, c))^\gamma / C(c, y).$$

D.3 PROOF OF LEMMA 2

Recall $g^*(t, c, y, x) = f^{\text{pre}}(t, c, x) + \sigma^2(t) \times \nabla_x v_t^*(x|c, y)$ from the proof of Lemma 4, we have

$$g^*(t, c, y, x) = f^{\text{pre}}(t, c, x) + \sigma^2(t) \times \nabla_x \log \mathbb{E}_{\mathbb{P}^{\text{pre}}(\cdot|c)} [(p^\diamond(y|x_T, c))^\gamma | x_t = x, c].$$

E ADDITIONAL EXPERIMENT: DESIGN DNA ENHANCERS CONDITIONED ON TWO CELL LINES

In this section, we focus on the design of cell-type specific DNA enhancers, a crucial task in the field of genomics. DNA is the molecule that carries the genetic instructions used in the growth, development, and functioning of all living organisms. Within DNA, there are regions called cis-regulatory elements (CREs), which regulate the expression of genes. One type of CREs is DNA enhancers, which are short sequences that can increase the transcription of target genes, often acting at a distance from the gene itself.

The task of cell-type specific DNA enhancer design aims to create DNA enhancers that exhibit specific accessibility or activity in targeted cell types (Taskiran et al., 2024). This task is crucial for precisely controlling gene expression in a cell-type-specific manner, which has broad implications for understanding tissue-specific gene regulation and developing therapies such as gene therapy, tissue engineering, and treatments for genetic diseases (de Almeida et al., 2024; Company et al., 2024; Gosai et al., 2023).

Experimental setup. For x , we use the enhancer dataset ($n \approx 700k$) from (Gosai et al., 2023) which consists of DNA sequences with a length of 200. This dataset has been extensively used in sequence optimization (Castillo-Hair and Seelig, 2021; Ghari et al., 2023; Lal et al., 2024). To create c and y for any sequence x , we query a binarized ATAC reward model following Lal et al. (2024)⁷. This reward model was trained on binarized CATLAS scATAC (single-cell Assay for Transposase Accessible Chromatin) data (Zhang et al., 2021). Finally, in our dataset, $y \in (0, 1)$ and $c \in (0, 1)$ represent the probabilities of accessibility of the enhancer measured in two cell types.

⁷We utilize an alternative reward model to label sequences because the original HepG2/K562/SKNSH scores from the dataset (Gosai et al., 2023) are highly correlated, making it nearly impossible to conditionally generate sequences for two distinct cell lines.

Setting of oracles and diffusion model. We use discrete diffusion models customized for sequences over simplex space (Avdeyev et al., 2023). To obtain the pre-trained (conditional) diffusion model, we employ classifier-free guidance to train a conditional diffusion model $p(x|c)$, where c is the accessibility in cell line HepG2 and is discretized into 10 classes. For the additional control, we set y (also discretized into 10 classes) as the accessibility in cell line SKNSH. We train the classifier $p(y|x)$ using an Enformer-based model (Avsec et al., 2021). For evaluation, we also query the binarized ATAC reward model (Lal et al., 2024).

In cell type-specific enhancer design, the goal is to generate enhancers that are active in one cell type but inactive in another (Gosai et al., 2023). In this experiment, we fine-tune a diffusion model using HepG2 and SKNSH cell lines to generate enhancers with (1) high c and low y or (2) low c and high y , focusing on $Y = 0$ and $Y = 9$ during inference.

Results. We evaluate performance as follows: (1) generate sequences by setting C uniformly from $[0, 1, \dots, 9]$ and using $Y = 0$ and $Y = 9$; (2) To assess the effectiveness of additional control on Y , we calculate the SKNSH score difference between $Y = 0$ and $Y = 9$, averaging the result across all values of C . (3) For each Y , we apply Kendall’s tau test (Kendall, 1948) on the samples’ HepG2 scores and their conditions to assess the monotonic increasing trend with respect to C . We report the tau value averaged across all values of Y .

The results are shown in Table 3. From the table, it is clear that **CTRL** achieves a stronger response to the additional control y . Moreover, it retains the pre-trained model’s ability to sample based on the old condition c .

This highlights a clear improvement over the baselines. Figure 4 presents boxplots of HepG2 and SKNSH scores for sequences generated by our method, illustrating its ability to conditionally sample based on both SKNSH and HepG2.

	SKNSH difference \uparrow	Kendall’s tau \uparrow
Classifier-Free	0.090 \pm 0.007	0.41
Reconstruction Guidance	0.210 \pm 0.007	0.45
CTRL	0.299 \pm 0.010	0.51

Figure 3: Evaluation of generated enhancers.

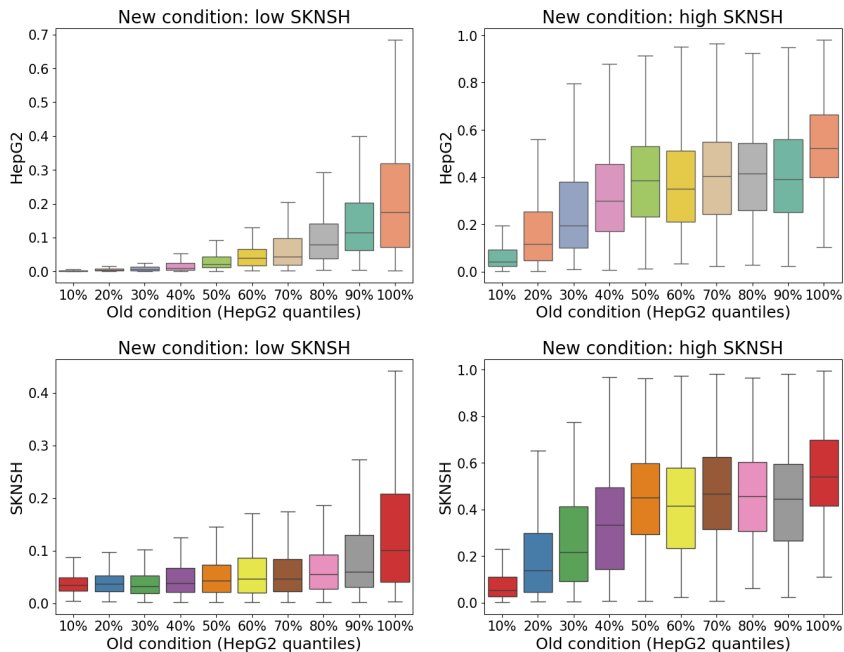


Figure 4: HepG2 and SKNSH scores of sequences generated by **CTRL**.

F DETAILS OF EXPERIMENTS

F.1 IMPLEMENTATION OF CLASSIFIER-FREE BASELINE

The effectiveness of classifier-free guidance often relies on a sufficiently large offline dataset (c, x, y) . However, in our experiments (Section 6.1 and Section 6.2), we only have access to offline datasets (x, y) and (x, y_1, y_2) respectively. Thus, to implement a classifier-free guidance baseline in this context, we leverage the pre-trained diffusion model for data augmentation. The procedure for Section 6.1 is outlined below. The procedure for Section 6.2 is similar.

Data augmentation. Consider the scenario where $Y \perp C|X$ and we have access to p^{pre} and offline dataset $\{(x, y)\}$. First, we use the offline data $\{(x, y)\}$ to train a classifier $\hat{p} : \mathcal{X} \rightarrow \Delta(\mathcal{Y})$. We then use \hat{p} to generate triplets $(x, c, y) \sim p^{\text{pre}}(x|c)\hat{p}(y|x)$ for given c . In practice, for text-to-image diffusion models, c can be uniformly sampled from a set of prompts, such as animals. However, this process becomes computationally demanding when applied to large pre-trained models like Stable Diffusion (Rombach et al., 2022).

Potential limitations. Given the data augmentation strategy, several limitations arise for classifier-free guidance. A primary concern is the accuracy of the trained classifier $\hat{p}(y|x)$. If the classifier is not sufficiently accurate, the generated y values may be unreliable, compromising the quality of the augmented triplets (x, c, y) . Additionally, selecting the condition c presents challenges. While models like Stable Diffusion (Rombach et al., 2022) are pre-trained on vast and diverse datasets with a wide range of prompts, we are constrained to a smaller, more limited set of prompts for c in this context. This lack of diversity reduces the representativeness of the augmented data and may lead to mode collapse during fine-tuning—a common issue observed in fine-tuning of diffusion models (Uehara et al., 2024).

F.2 IMPLEMENTATION OF RECONSTRUCTION GUIDANCE BASELINE

As reviewed in Section 3.2.2, reconstruction guidance baseline employs the following approximation

$$p(y|x_t, c) = \int p^\diamond(y|x_T, c)p(x_T|x_t, c)dx_T \approx p(y|\hat{x}_T(x_t, c), c),$$

where $\hat{x}_T(x_t, c)$ is the (expected) denoised sample given x_t, c , i.e., $\hat{x}_T(x_t, c) = \mathbb{E}[x_T|x_t, c]$. We note that such approximation is often readily available from diffusion noise schedulers, such as DDIM scheduler (Song et al., 2020).

Given such an approximation, we only need to learn $p^\diamond(y|x_T, c)$ from offline data. Accordingly, we can leverage the trained classifiers from Algorithm 1 (see Section 4.2, **Step 2**).

As an inference-time technique, the choice of guidance strength is often subtle. We present ablation studies on guidance strength in Appendix F.5. For reporting classification metrics, as shown in Table 1c and Table 2b, we consistently select the optimal configuration for each baseline.

F.3 IMAGES

In this subsection, we provide details of experiments in Section 6. We first explain the training details and list hyperparameters in Table 2.

We use 4 A100 GPUs for all the image tasks. We use the AdamW optimizer (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9, \beta_2 = 0.999$ and weight decay of 0.1. To ensure consistency with previous research, in fine-tuning, we also employ training prompts that are uniformly sampled from 50 common animals (Black et al., 2023; Prabhudesai et al., 2023).

Construction of the augmented score model. An important engineering aspect is how to craft the augmented score model architecture. For most of the diffusion models, the most natural and direct technique of adding another conditioning control is (1) augmenting the score prediction networks by incorporating additional linear embeddings, while using the existing neural network architecture and weights for all other parts. In our setting, we introduce a linear embedding layer that maps $|\mathcal{Y}| + 1$

Table 2: Training hyperparameters.

Hyperparameter	compressibility (Section 6.1)	multi-task (Section 6.2)
Classifier-free guidance weight on prompts (i.e., c)	7.5	7.5
γ (i.e., strength of the additional guidance on y)	10	10
DDIM steps	50	50
Truncated back-propagation step	$K \sim \text{Uniform}(0, 50)$	$K \sim \text{Uniform}(0, 50)$
Learning rate for LoRA modules	$1e^{-3}$	$3e^{-4}$
Learning rate for the linear embeddings	$1e^{-2}$	$1e^{-2}$
Batch size (per gradient update)	256	512
Number of gradient updates per epoch	2	2
Epochs	15	60

class labels to embeddings in \mathbb{R}^d , where d is the same dimension as intermediate diffusion states. Among all embeddings, the first $|\mathcal{Y}|$ embeddings correspond to $|\mathcal{Y}|$ conditions of our interest, whereas the last one represents the unconditional category (i.e., NULL conditioning) (2) for any $y \in \mathcal{Y}$, the corresponding embedding is added to the predicted score in the forward pass. During fine-tuning, the embeddings are initialized as zeros. We only fine-tune the first $|\mathcal{Y}|$ embeddings, and freeze the last one at zero as it is the unconditional label.

We note that, while it is possible to add additional conditioning by reconstructing the score networks like ControlNet (Zhang et al., 2023), in practice it is often desired to make minimal changes to the architecture of large diffusion models, e.g., Stable Diffusion (Rombach et al., 2022) to avoid the burdensome re-training. It is especially important to leverage pre-trained diffusion models in our setting where the offline dataset is limited, therefore a total retraining of model parameters can be struggling.

Sampling. We use the DDIM sampler with 50 diffusion steps (Song et al., 2020). Since we need to back-propagate the gradient of rewards through both the sampling process producing the latent representation and the VAE decoder used to obtain the image, memory becomes a bottleneck. We employ two designs to alleviate memory usage following Clark et al. (2023); Prabhudesai et al. (2023): (1) Fine-tuning low-rank adapter (LoRA) modules (Hu et al., 2021) instead of tuning the original diffusion weights, and (2) Gradient checkpointing for computing partial derivatives on demand (Gruslys et al., 2016; Chen et al., 2016). The two designs make it possible to back-propagate gradients through all 50 diffusing steps in terms of hardware.

Table 3: Architecture of compressibility classifier.

#	Input Dimension	Output Dimension	Layer
1	$C \times H \times W$	$64 \times H \times W$	ResidualBlock (Conv2d(3, 64, 3x3), BN, ReLU)
2	$64 \times H \times W$	$128 \times \frac{H}{2} \times \frac{W}{2}$	ResidualBlock (Conv2d(64, 128, 3x3), BN, ReLU)
3	$128 \times \frac{H}{2} \times \frac{W}{2}$	$256 \times \frac{H}{4} \times \frac{W}{4}$	ResidualBlock (Conv2d(128, 256, 3x3), BN, ReLU)
4	$256 \times \frac{H}{4} \times \frac{W}{4}$	$256 \times 1 \times 1$	AdaptiveAvgPool2d (1, 1)
5	$256 \times 1 \times 1$	256	Flatten
6	256	num_classes	Linear

Table 4: Architecture of aesthetic score classifier.

#	Layer	Input Dimension	Output Dimension
1	Linear	768	1024
2	Dropout	-	-
3	Linear	1024	128
4	Dropout	-	-
5	Linear	128	64
6	Dropout	-	-
7	Linear	64	16
8	Linear	16	num_classes

Classifiers. In our experiments, we leverage conditional independence for both compressibility and aesthetic scores tasks. Therefore, we only demand data samples $\{x_i, y_i\}$ in order to approximate conditional classifier $p(y|x, c)$. Specifically,

- compressibility: the classifier is implemented as a 3-layer convolutional neural network (CNN) with residual connections and batch normalizations on top of the raw image space. The offline dataset is constructed by labeling a subset of 10k images of the AVA dataset (Murray et al., 2012), employing JPEG compression. We train the network using Adam optimizer for 100 epochs. Detailed architecture of the oracle can be found in Table 3.
- aesthetic scores: the classifier is implemented as an MLP on top of CLIP embeddings (Radford et al., 2021). To train the classifier, we use the full AVA dataset (Murray et al., 2012) which includes more than 250k human evaluations. The specific neural network instruction is listed in Table 4.

Note that in training both classifiers, we split the dataset with 80% for training and 20% for validation. After training, we use the validation set to perform temperature scaling calibration Guo et al. (2017).

F.4 DNA ENHANCERS

In this subsection, we add the details of experiments in Section E. In this experiment, we used A100 GPUs and Adam optimizers. Training hyperparameters are summarized in Table 6.

Construction of the augmented score model. We augment the diffusion model tailored to biological sequences over the simple space (Avdeyev et al., 2023) (i.e., DDSM) in the same manner as in the image experiments: we introduce additional control through a linear embedding layer and preserve all other components of the existing neural network architecture.

Classifier. Following Lal et al. (2024), we implement an Enformer-based classifier. Detailed architecture of the oracle can be found in Table 5.

Table 5: Architecture of sequence classifier.

#	Layer
1-12	(Conv1D, GELU, BatchNorm, AttentionPool, Dropout)
13	TransformerBlock (Multi-head Attention, LayerNorm, Dropout)*3
14	Pointwise Conv1D, GELU, BatchNorm
15	ConvHead (Conv1D, AdaptiveAvgPool)

Table 6: Training hyperparameters.

Type	Value
Batch size (per gradient update)	1024
Guidance strength on old condition c	1.0
γ (Guidance strength on new condition y)	1.0
# of diffusion steps during fine-tuning	50
# of diffusion steps during inference	100
Learning rate for UNet modules	$3e^{-4}$
Learning rate for the linear embeddings	$1e^{-1}$
Sampling to neural SDE	Euler Maruyama
Epochs	100

F.5 ADDITIONAL RESULTS OF RECONSTRUCTION GUIDANCE BASELINE

In this subsection, we provide more results of **Reconstruction Guidance** for conditioning images on compressibility (see Section 6.1).

1242 In Figure 5, we plot the confusion matrix for samples generated by **Reconstruction Guidance**. For
 1243 each condition, 128 samples are generated and are evaluated. We find that this method struggles to
 1244 generate samples accurately when conditioned on intermediate labels.
 1245

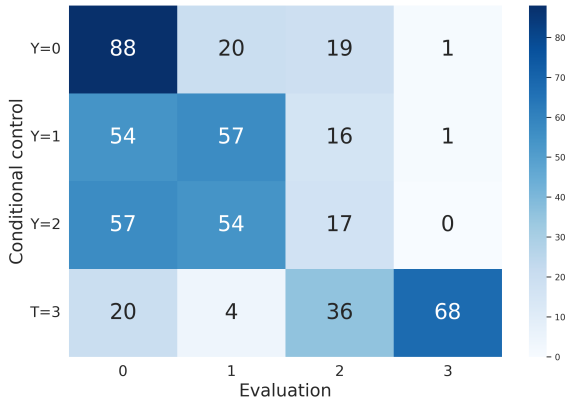


Figure 5: Confusion matrix for **Reconstruction Guidance**.

1261
 1262
 1263
 1264 Table 7: Results of **Reconstruction Guidance** conditioned on compressibility. Essentially, guidance
 1265 level= 0 indicates that the generations are unconditional on Y .
 1266

Conditional control (Y)	Guidance level (γ)	Accuracy \uparrow	Mean score
0	0	0.43	-110
0	5	0.64	-157.4
0	7.5	0.62	-148.2
0	10	0.52	-156.5
0	20	0.66	-152.5
0	50	0.69	-151.6
1	0	0.45	-110
1	5	0.14	-152.0
1	7.5	0.12	-189.7
1	10	0.08	-163.1
1	20	0.06	-169.5
1	50	0	-194.0
2	0	0.13	-110
2	5	0.02	-104.9
2	7.5	0.10	-122.1
2	10	0.12	-111.3
2	20	0.08	-157.6
2	50	0.08	-173.5
3	0	0	-110
3	5	0.46	-71.7
3	7.5	0.46	-67.6
3	10	0.53	-65.6
3	20	0.26	-112.4
3	50	0.32	-121.5

1290
 1291 For completeness, below we provide ablation studies on its hyper-parameters. In Table 7, we
 1292 present the classification statistics for generations across different conditions (y) and guidance levels
 1293 (γ). Recall that the four conditions are defined as follows: $Y = 0 : \mathbf{CP} < -110.0$, $Y = 1 :$
 1294 $-110.0 \leq \mathbf{CP} < -85.0$, $Y = 2 : -85.0 \leq \mathbf{CP} < -60.0$, $Y = 3 : \mathbf{CP} \geq -60.0$.
 1295

Our analysis reveals several key insights:

- 1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
- For $Y = 0$ and $Y = 3$, the accuracy of the generations improves as the guidance signal strength increases. This indicates a clear positive correlation between the guidance level and the accuracy of generation.
 - Conversely, for intermediate $Y = 1$ and $Y = 2$, guidance signals decrease generation accuracy compared to the pre-trained model, suggesting difficulty in maintaining accuracy within these specific compressibility intervals. The challenge in generating samples with medium compressibility scores lies in hand-picking the guidance strength. For instance, generating samples conditioned on $Y = 2$ requires compressibility scores between -85 and -60 , making it difficult to apply optimal guidance without overshooting or undershooting the target values.
 - Regarding the mean scores, distinct patterns are observed across different conditions and guidance levels:
 - For $Y = 0$, mean scores become more negative with increasing guidance levels.
 - For $Y = 1$, mean scores consistently drop with increasing guidance levels.
 - For $Y = 2$, mean scores initially improve slightly with increasing guidance levels but show a marked decline at $\gamma = 20$ and $\gamma = 50$, indicating a challenge in achieving the desired compressibility range.
 - For $Y = 3$, mean scores improve significantly with increased guidance, showing the best results at $\gamma = 10$, but then become more negative at higher guidance levels.

1316
1317
1318
1319

In summary, these observations suggest that while guidance can be beneficial for improving accuracy in extreme compressibility levels ($Y = 0$ and $Y = 3$), this method struggles with intermediate conditions ($Y = 1$ and $Y = 2$) due to the narrow range of acceptable scores and the non-linear effects of guidance strength on generation quality.

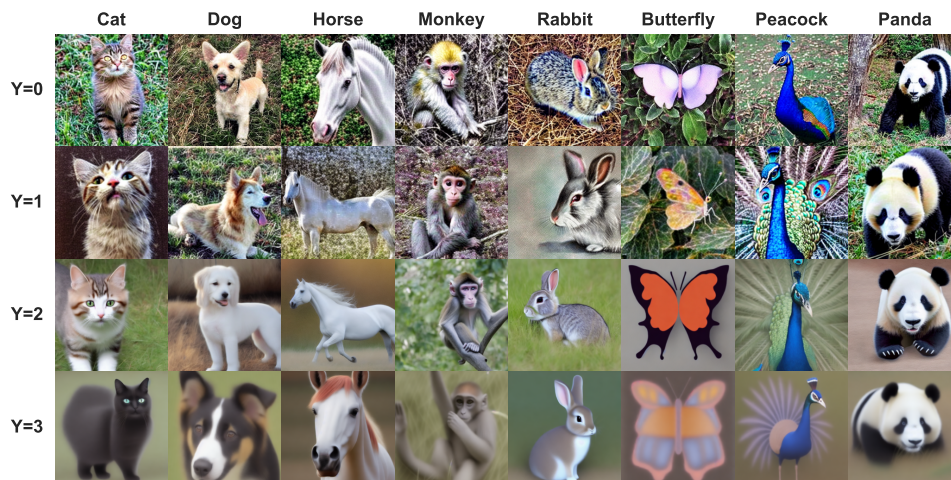
1320
1321
1322

For each conditional control, samples are generated by choosing the best γ according to Table 7. We report the evaluation statistics in Table 1c, and provide the confusion matrix in Figure 5

1323 F.6 ADDITIONAL VISUALIZATIONS

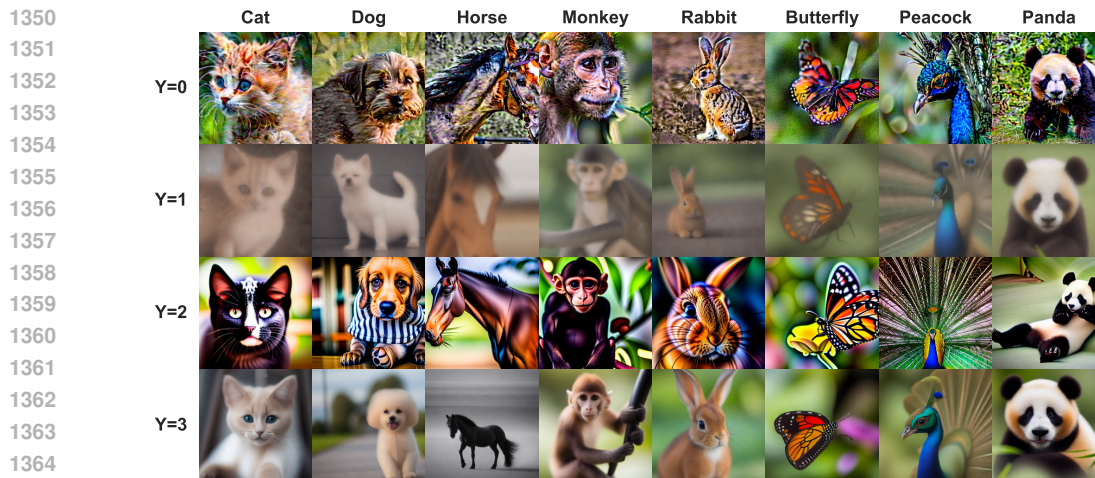
1324
1325

We provide more generated samples to illustrate the performances of **CTRL** in Figure 6 and Figure 7.



1342
1343
1344
1345
1346
1347
1348
1349

Figure 6: More images generated by **CTRL** in the compressibility task.



1366 Figure 7: More images generated by CTRL in the multi-task conditional generation.

1368 G LIMITATIONS AND MITIGATION STRATEGIES

1370 G.1 COMPUTATIONAL COST

1371

1372

1373 Fine-tuning large pre-trained diffusion models is computationally intensive, especially in multi-task

1374 settings. For instance, even in simpler cases, such as fine-tuning a model with a single reward

1375 function, the process can be slow—DDPO (Black et al., 2023) requires approximately 60 A100

1376 GPU hours for fine-tuning Stable Diffusion (Rombach et al., 2022). In contrast, our single-task

1377 experiment Section 6.1, which conditions on four compressibility labels (a more complex task than

1378 optimizing a single reward), is more computationally efficient, requiring only around 20 A100

1379 GPU hours. This demonstrates that our method is computationally efficient even when handling more

1380 challenging conditioning tasks.

1381 Multi-task experiments Section 6.2 are considerably more demanding, requiring around 200 A100

1382 GPU hours. This is due to the added complexity of aligning the model with multiple control signals,

1383 often necessitating larger batch sizes and careful balancing between tasks. As a result, multi-task

1384 fine-tuning requires not only more GPU time but also careful balancing to prevent overfitting to

1385 specific tasks.

1386 Below, we discuss promising ways to help reduce the computational burden without sacrificing

1387 performance.

1388 **Mitigation strategies.** To further reduce computational costs, an effective strategy is to truncate

1389 backpropagation to a small fixed number of steps, such as 3 or 5. As noted by (Clark et al., 2023),

1390 truncating the gradient flow in direct backpropagation to fewer than 10 steps not only significantly

1391 reduces computational overhead but also improves optimization stability by mitigating gradient

1392 explosion. Interestingly, performance begins to degrade when the number of steps exceeds 10,

1393 suggesting that shorter truncation steps (even as few as 1) can be more computationally efficient

1394 while maintaining or even improving model performance.

1395 Additionally, mixed precision training can be employed to further accelerate training.

1397 G.2 MEMORY COMPLEXITY

1398

1399 As we have clarified in Section 4, any off-the-shelf RL algorithm can be used for planning. We

1400 recommend using direct back-propagation (Clark et al., 2023) or PPO (Black et al., 2023).

1401

1402 For direct backpropagation, updating a single gradient requires $O(L)$ memory, where L is the number

1403 of discretizations. To reduce memory usage, our experiments employed techniques such as (a)

fine-tuning only LoRA (Hu et al., 2021) modules instead of the full diffusion model, (b) applying

1404 gradient checkpointing (Gruslys et al., 2016; Chen et al., 2016), and (c) randomly truncating gradient
1405 backpropagation. A detailed discussion of these techniques is provided in Appendix A.

1406
1407 If memory constraints persist, we recommend using PPO for planning, as it requires only $O(1)$
1408 memory per gradient update. Employing mixed precision training can also reduce memory usage.

1409 G.3 CHOICE OF GUIDANCE STRENGTH γ 1410

1411 First, note that $1/\gamma$ can be interpreted as the KL weight parameter in standard diffusion model
1412 fine-tuning works (Fan et al., 2023; Uehara et al., 2024), where selecting an optimal KL weight
1413 remains an open problem. As observed in these works, fine-tuning without entropy regularization
1414 often leads to over-optimization. Therefore, introducing a KL weight is beneficial as long as it is
1415 neither too small nor too large.

1416 In this work, a larger γ strengthens the guidance signal of the additional control (see Section 4) but
1417 can cause the fine-tuned model to deviate more from the pre-trained model, which is also undesired.
1418 For both image experiments, we set $\gamma = 10$ (see Table 2), which provides a good balance. In practice,
1419 we find that values between 5 and 20 are generally effective. Additionally, even if a smaller γ is used
1420 during fine-tuning, we note that it is possible to freely adjust (strengthen or weaken) the guidance
1421 strength during inference. Details can be found in Appendix C.

1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457