RL for Consistency Models: Faster Reward Guided Text-to-Image Generation

Owen Oertell
Department of Computer Science
Cornell University
ojo2@cornell.edu

Jonathan D. Chang
Department of Computer Science
Cornell University
jdc396@cornell.edu

Yiyi Zhang
Department of Computer Science
Cornell University
yz2364@cornell.edu

Kianté Brantley
Department of Computer Science
Cornell University
kdb82@cornell.edu

Wen Sun
Department of Computer Science
Cornell University
ws455@cornell.edu

Abstract

Reinforcement learning (RL) has improved guided image generation with diffusion models by directly optimizing rewards that capture image quality, aesthetics, and instruction following capabilities. However, the resulting generative policies inherit the same iterative sampling process of diffusion models that causes slow generation. To overcome this limitation, consistency models proposed learning a new class of generative models that directly map noise to data, resulting in a model that can generate an image in as few as one sampling iteration. In this work, to optimize text-to-image generative models for task specific rewards and enable fast training and inference, we propose a framework for fine-tuning consistency models via RL. Our framework, called Reinforcement Learning for Consistency Model (RLCM), frames the iterative inference process of a consistency model as an RL procedure. Comparing to RL finetuned diffusion models, RLCM trains significantly faster, improves the quality of the generation measured under the reward objectives, and speeds up the inference procedure by generating high quality images with as few as two inference steps. Experimentally, we show that RLCM can adapt text-to-image consistency models to objectives that are challenging to express with prompting, such as image compressibility, and those derived from human feedback, such as aesthetic quality. Our code is available at https://rlcm.owenoertell.com.

1 Introduction

Diffusion models have gained widespread recognition for their high performance in various tasks, including drug design (Xu et al., 2022) and control (Janner et al., 2022). In the text-to-image generation community, diffusion models have gained significant popularity due to their ability to output realistic images via prompting. Despite their success, diffusion models in text-to-image tasks face two key challenges. First, generating the desired images can be difficult for downstream tasks whose goals are hard to specify via prompting. Second, the slow inference speed of diffusion models poses a barrier, making the iterative process of prompt tuning computationally intensive.

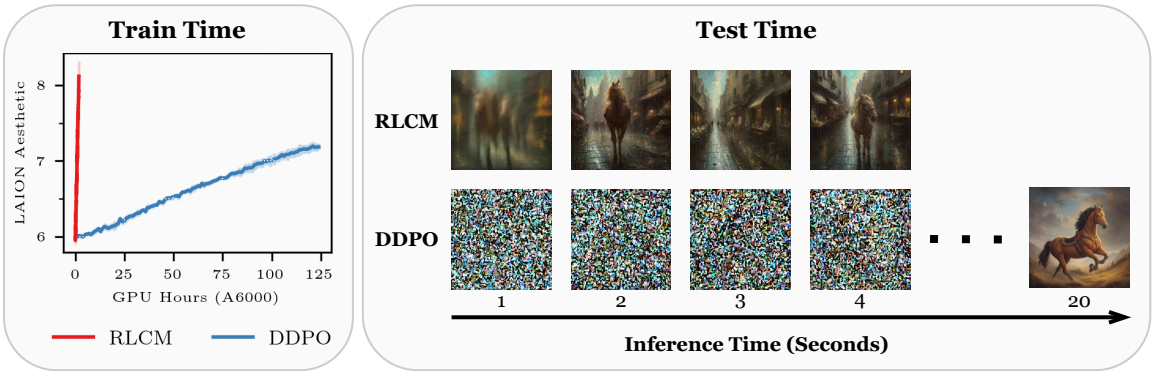

Figure 1: **Reinforcement Learning for Consistency Models (RLCM)**. We propose a new framework for finetuning consistency models using RL. On the task of optimizing aesthetic scores of a generated image, comparing to a baseline which uses RL to fine-tune diffusion models (DDPO), RLCM trains (left) and generates images (right) significantly faster, with higher image quality measured under the aesthetic score. Images generated with a batch size of 8 and RLCM horizon set to 8.

To enhance the generation alignment with specific prompts, diffusion model inference can be framed as sequential decision-making processes, permitting the application of reinforcement learning (RL) methods to image generation (Black et al., 2024; Fan et al., 2023). The objective of RL-based diffusion training is to fine-tune a diffusion model to maximize a reward function directly that corresponds to the desirable property.

Diffusion models also suffer from slow inference since they must take many steps to produce competitive results. This leads to slow inference time and even slower training time. Even further, as a result of the number of steps we must take, the resultant Markov decision process (MDP) possesses a long time horizon which can be hard for RL algorithms optimize. To resolve this, we look to consistency models. These models directly map noise to data and typically require only a few steps to produce good looking results. Although these models can be used for single step inference, to generate high quality samples, there exits a few step iterative inference process which we focus on. Framing consistency model inference, instead of diffusion model inference, as an MDP admits a much shorter horizon. This enables faster RL training and allows for generating high quality images with just few step inference.

More formally, we propose a framework **R**einforcement **L**earning for **C**onsistency **M**odels (RLCM), a framework that models the inference procedure of a consistency model as a multi-step Markov Decision Process, allowing one to fine-tune consistency models toward a downstream task using just a reward function. Algorithmically, we instantiate RLCM using a policy gradient algorithm as this allows for optimizing general reward functions that may not be differentiable. In experiments, we compare to the current more general method, DDPO (Black et al., 2024) which uses policy gradient methods to optimize a diffusion model. In particular, we show that on an array of tasks (compressibility, incompressibility, prompt image alignment, and LAION aesthetic score) proposed by DDPO, RLCM outperforms DDPO in tested compression, incompression, and aesthetic tasks in training time, inference time, and sample complexity (i.e., total reward of the learned policy versus number of reward model queries used in training) (Section 5).

Our contributions in this work are as follows:

- In our experiments, we find that RLCM has ***faster training*** and ***faster inference*** than existing methods.

- Further, that RLCM, in our experiments, enjoys ***better performance*** on most tasks under the tested reward models than existing methods.

2 Related Works

Diffusion Models Diffusion models are a popular family of image generative models which progressively map noise to data (Sohl-Dickstein et al., 2015). Such models generate high quality images (Ramesh et al., 2021; Saharia et al., 2022) and videos (Ho et al., 2022; Singer et al., 2022). Recent work with diffusion models has also shown promising directions in harnessing their power for other types of data such as robot trajectories and 3d shapes (Janner et al., 2022; Zhou et al., 2021). However, the iterative inference procedure of progressively removing noise yields slow generation time.

Consistency Models Consistency models (Song et al., 2023) are another family of generative models which directly map noise to data via the consistency function . Such a function provides faster inference generation as one can predict the image from randomly generated noise in a single step. Consistency models also offer a more fine-tuned trade-off between inference time and generation quality as one can run the multistep inference process (Algorithm 2, in Appendix A) which is described in detail in Section 3.2. Prior works have also focused on training the consistency function in latent space (Luo et al., 2023) which allows for large, high quality text-to-image consistency model generations. Sometimes, such generations are not aligned with the downstream for which they will be used. The remainder of this work will focus on aligning consistency models to fit downstream preferences, given a reward function.

RL for Diffusion Models Popularized by Black et al. (2024); Fan et al. (2023), training diffusion models with reinforcement learning requires treating the diffusion model inference sequence as a Markov decision process. Then, by treating the score function as the policy and updating it with a modified PPO algorithm (Schulman et al., 2017), one can learn a policy (which in this case is a diffusion model) that optimizes for a given downstream reward. Further work on RL fine-tuning has looked into entropy regularized control to avoid reward hacking and maintain high quality images (Uehara et al., 2024). Another line of work uses deterministic policy gradient methods to directly optimize the reward function when the reward function is differentiable (Prabhudesai et al., 2023). Note that when reward function is differentiable, we can instantiate a deterministic policy gradient method in RLCM as well. We focus on REINFORCE (Williams, 1992) style policy gradient methods for the purpose of optimizing a black-box, non-differentiable reward functions.

3 Preliminaries

We provide some preliminary information on reinforcement learning, diffusion and consistency models, and discuss the application of reinforcement learning to diffusion models. Also note that we abuse notation and use t to mean one of two things: the timestep along the diffusion trajectory or the timestep corresponding to the RL trajectory.

3.1 Reinforcement Learning

We model our sequential decision process as a finite horizon Markov Decision Process (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mu, H)$. In this tuple, we define our state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, initial state distribution μ, and horizon H. At each timestep t, the agent observes a state $s_t \in \mathcal{S}$, takes an action according to the policy $a_t \sim \pi(a_t|s_t)$ and transitions to the next state $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$. After H timesteps, the agent produces a trajectory as a sequence of states and actions $\tau = (s_0, a_0, s_1, a_1, \ldots, s_H, a_H)$. Our objective is to learn a policy π that maximizes the expected cumulative reward over trajectories sampled from π,

$$\mathcal{J}_{RL}(\pi) = \mathbb{E}_{\tau \sim p(\cdot|\pi)} \left[\sum_{t=0}^{H} R(s_t, a_t) \right].$$

3.2 Diffusion and Consistency Models

Generative models are designed to match a model with the data distribution, such that we can synthesize new data points at will by sampling from the distribution. Diffusion models belong to a novel type of generative model that characterizes the probability distribution using a score function rather than a density function. Specifically, it produces data by gradually modifying the data distribution and subsequently generating samples from noise through a sequential denoising step. More formally, we start with a distribution of data $p_{\text{data}}(\boldsymbol{x})$ and noise it according to the stochastic differential equation (SDE) (Song et al., 2020):

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{\mu}(\boldsymbol{x}_t, t)\mathrm{d}t + \boldsymbol{\sigma}(t)\mathrm{d}\boldsymbol{w}$$

for a given $t \in [0, T]$, fixed constant $T > 0$, and with the drift coefficient $\boldsymbol{\mu}(\cdot, \cdot)$, diffusion coefficient $\boldsymbol{\sigma}(\cdot)$, and $\{\boldsymbol{w}\}_{t\in[0,T]}$ being a Brownian motion. Letting $p_0(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x})$ and $p_t(x)$ be the marginal distribution at time t induced by the above SDE, as shown in Song et al. (2020), there exists an ODE (also called a *probability flow*) whose induced distribution at time t is also $p_t(\boldsymbol{x})$. In particular:

$$\mathrm{d}x_t = \left[\boldsymbol{\mu}(\boldsymbol{x}_t, t) - \frac{1}{2}\boldsymbol{\sigma}(t)^2 \nabla \log p_t(\boldsymbol{x}_t)\right] \mathrm{d}t.$$

The term $\nabla \log p_t(\boldsymbol{x}_t)$ is also known as the *score function* (Song & Ermon, 2019; Song et al., 2020). When training a diffusion models in such a setting, one uses a technique called *score matching* (Dinh et al., 2016; Vincent, 2011) in which one trains a network to approximate the score function and then samples a trajectory with an ODE solver. Once we learn such a neural network that approximates the score function, we can generate images by integrating the above ODE backward in time from T to 0, with $\boldsymbol{x}_T \sim p_T$ which is typically a tractable distribution (e.g., Gaussian in most diffusion model formulations).

This technique is clearly bottle-necked by the fact that during generation, one must run a ODE solver backward in time (from T to 0) for a large number of steps in order to obtain competitive samples (Song et al., 2023). To alleviate this issue, Song et al. (2023) proposed *consistency models* which aim to directly map noisy samples to data. The goal becomes instead to learn a *consistency function* on a given probability flow. The aim of this function is that for any two $t, t' \in [\epsilon, T]$, the two samples along the probability flow ODE, they are mapped to the same image by the consistency function: $f_\theta(\boldsymbol{x}_t, t) = f_\theta(\boldsymbol{x}_{t'}, t') = \boldsymbol{x}_\epsilon$ where $\boldsymbol{x}_\epsilon$ is the solution of the ODE at time ϵ. At a high level, this consistency function is trained by taking two adjacent timesteps and minimizing the consistency loss $d(f_\theta(\boldsymbol{x}_t, t), f_\theta(\boldsymbol{x}_{t'}, t'))$, under some image distance metric $d(\cdot, \cdot)$. To avoid the trivial solution of a constant, we also set the initial condition to $f_\theta(\boldsymbol{x}_\epsilon, \epsilon) = \boldsymbol{x}_\epsilon$.

Inference in consistency models After a model is trained, one can then trade inference time for generation quality with the multi-step inference process given in Appendix A, Algorithm 2. At a high level, the multistep consistency sampling algorithm first partitions the probability flow into $H + 1$ points ($T = \tau_0 > \tau_1 > \tau_2 \ldots > \tau_H = \epsilon$). Given a sample $x_T \sim p_T$, it then applies the consistency function f_θ at (x_T, T) yielding $\widehat{\boldsymbol{x}}_0$. To further improve the quality of $\widehat{\boldsymbol{x}}_0$, one can add noise ($\boldsymbol{x} \sim \mathcal{N}(0, 1)$) back to $\widehat{\boldsymbol{x}}_0$ using the equation $\widehat{\boldsymbol{x}}_{\tau_n} \leftarrow \widehat{\boldsymbol{x}}_0 + \sqrt{\tau_n^2 - \tau_H^2}\boldsymbol{z}$, and then again apply the consistency function to $(\widehat{\boldsymbol{x}}_{\tau_n}, \tau_n)$, getting $\widehat{\boldsymbol{x}}_0$. One can repeat this process for a few more steps until the quality of the generation is satisfied. For the remainder of this work, we will be referring to sampling with the multi-step procedure. We also provide more details when we introduce RLCM later.

3.3 Reinforcement Learning for Diffusion Models

Black et al. (2024) and Fan et al. (2023) formulated the training and fine-tuning of conditional diffusion probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020) as an MDP. Black et al. (2024) defined a class of algorithms, Denoising Diffusion Policy Optimization (DDPO), that optimizes for arbitrary reward functions to improve guided fine-tuning of diffusion models with RL.

Diffusion Model Denoising as MDP Conditional diffusion probabilistic models condition on a context $\boldsymbol{c}$ (in the case of text-to-image generation, a prompt). As introduced for DDPO, we map the iterative denoising procedure to the following MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mu, H)$. Let $r(\boldsymbol{s}, \boldsymbol{c})$ be the task reward function. Also, note that the probability flow proceeds from $x_T \to x_0$. Let $T = \tau_0 > \tau_1 > \tau_2 \ldots > \tau_H = \epsilon$ be a partition of the probability flow into intervals:

$$\boldsymbol{s}_t \triangleq (\boldsymbol{c}, \tau_t, x_{\tau_t}) \qquad \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) \triangleq p_\theta\left(x_{\tau_{t+1}}|x_{\tau_t}, \boldsymbol{c}\right) \qquad P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) \triangleq (\delta_{\boldsymbol{c}}, \delta_{\tau_{t+1}}, \delta_{x_{\tau_{t+1}}})$$

$$\boldsymbol{a}_t \triangleq x_{\tau_{t+1}} \qquad \mu \triangleq (p(\boldsymbol{c}), \delta_{\tau_0}, \mathcal{N}(0, I)) \qquad R(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} r(\boldsymbol{s}_t, \boldsymbol{c}) & \text{if } t = H \\ 0 & \text{otherwise} \end{cases}$$

where δ_y is the Dirac delta distribution with non-zero density at y. In other words, we are mapping images to be states, and the prediction of the next state in the denosing flow to be actions. Further, we can think of the deterministic dynamics as letting the next state be the action selected by the policy. Finally, we can think of the reward for each state being 0 until the end of the trajectory when we then evaluate the final image under the task reward function.

This formulation permits the following loss term:

$$\mathcal{L}_{\text{DDPO}} = \mathbb{E}_{\mathcal{D}} \sum_{t=1}^{T} \left[\min \left\{ r(\boldsymbol{x}_0, \boldsymbol{c}) \frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{c})}{p_{\theta_{\text{old}}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{c})}, r(\boldsymbol{x}_0, \boldsymbol{c}) \texttt{clip} \left(\frac{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{c})}{p_{\theta_{\text{old}}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{c})}, 1 - \varepsilon, 1 + \varepsilon \right) \right\} \right]$$

where clipping is used to ensure that when we optimize p_θ, the new policy stay close to $p_{\theta_{old}}$, a trick popularized by the well known algorithm Proximal Policy Optimization (PPO) (Schulman et al., 2017). However, one could easily replace this with other policy gradient optimizers like Gao et al. (2024).

In diffusion models (and in our experiments for DDPO), horizon H is usually set as 50 or greater and time T is set as 1000. A small step size is chosen for the ODE solver to minimize error, ensuring the generation of high-quality images as demonstrated by Ho et al. (2020). Due to the long horizon and sparse rewards, training diffusion models using reinforcement learning can be challenging.

4 Reinforcement Learning for Consistency Models

To remedy the long inference horizon that occurs during the MDP formulation of diffusion models, we instead frame consistency models as an MDP. We let H also represent the horizon of this MDP. Just as we do for DDPO, we partition the entire probability flow ($[0, T]$) into segments, $T = \tau_0 > \tau_1 > \ldots > \tau_H = \epsilon$. In this section, we denote t as the discrete time step in the MDP, i.e., $t \in \{0, 1, \ldots, H\}$, and τ_t is the corresponding time in the continuous time interval $[0, T]$. We now present the consistency model MDP formulation.

Consistency Model Inference as MDP We reformulate the multi-step inference process in a consistency model (Algorithm 2) as an MDP:

$$\boldsymbol{s}_t \triangleq (\boldsymbol{x}_{\tau_t}, \tau_t, \boldsymbol{c}) \qquad \pi(\boldsymbol{a}_t|\boldsymbol{s}_t) \triangleq f_\theta\left(\boldsymbol{x}_{\tau_t}, \tau_t, \boldsymbol{c}\right) + Z \qquad P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) \triangleq (\delta_{\boldsymbol{x}_{\tau_{t+1}}}, \delta_{\tau_{t+1}}, \delta_{\boldsymbol{c}})$$

$$\boldsymbol{a}_t \triangleq \boldsymbol{x}_{\tau_{t+1}} \qquad \mu \triangleq (\mathcal{N}(0, I), \delta_{\tau_0}, p(\boldsymbol{c})) \qquad R_H(\boldsymbol{s}_H) = r(f_\theta(\boldsymbol{x}_{\tau_H}, \tau_H, \boldsymbol{c}), \boldsymbol{c})$$

where is $Z = \sqrt{\tau_t^2 - \tau_H^2}\boldsymbol{z}$ which is noise from Line 5 of Algorithm 2. Further, where $r(\cdot, \cdot)$ is the reward function that we are using to align the model and R_H is the reward at timestep H. At other timesteps, we let the reward be 0. We can visualize this conversion from the multistep inference in Fig. 2.

Modeling the MDP such that the policy $\pi(s) \triangleq f_\theta(\boldsymbol{x}_{\tau_t}, \tau_t, \boldsymbol{c}) + Z$ instead of defining $\pi(\cdot)$ to be the consistency function itself has a major benefit in the fact that this gives us a stochastic policy instead

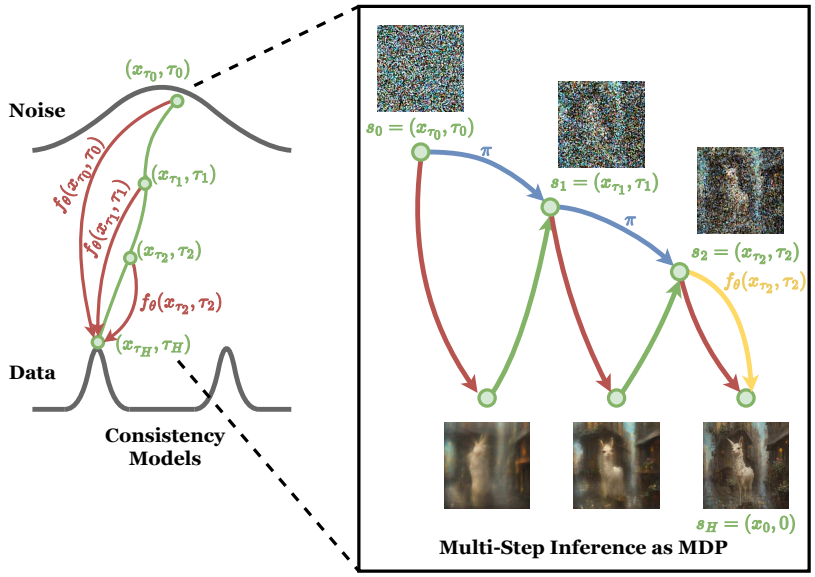

Figure 2: **Consistency Model As MDP:** In this instance, $H = 3$. Here we first start at a randomly sampled noised state $s_0 \sim (\mathcal{N}(0, I), \delta_{\tau_0}, p(\boldsymbol{c}))$. We then follow the policy by first plugging the state into the the consistency model (red line) and then noising the image back to τ_1 (green line). This gives us a_0 which, based off of the transition dynamics becomes s_1 (green circle). We then transition from s_1 by applying $\pi(\cdot)$, which applies the consistency function to $\widehat{x}_0$ and then noises up to τ_2. To calculate the end of trajectory reward, we apply the consistency function one more time (yellow line) to get a final approximation of $\widehat{x}_0$ and apply the given reward function to this image. Note that the red and green lines on both sides of the diagram represent the same thing.

of a deterministic one. This allows us to use a form of clipped importance sampling like Black et al. (2024) instead of a deterministic algorithm (e.g. DPG (Silver et al., 2014)) which we found to be unstable and in general is not unbiased. Thus a policy is made up of two parts: the consistency function and noising with Gaussian noises. The consistency function takes the form of the red arrows in Fig. 2 whereas the noise is the green arrows. In other words, our policy is a Gaussian policy whose mean is modeled by the consistency function f_θ, and covariance being $(\tau_t^2 - \epsilon^2)\mathbf{I}$ (here $\mathbf{I}$ is an identity matrix). Notice that in accordance with the sampling procedure in Algorithm 2, we only noise part of the trajectory. Note that the final step of the trajectory is slightly different. In particular, to calculate the final reward, we just apply the consistency function (red/yellow arrow) and obtain the final reward.

Policy Gradient RLCM We can then instantiate RLCM with a policy gradient optimizer, in the spirit of Black et al. (2024); Fan et al. (2023). Our algorithm is described in Algorithm 1. In practice we normalize the reward per prompt. That is, we create a running mean and standard deviation for each prompt and use that as the normalizer instead of calculating this per batch. This is because under certain reward models, the average score by prompt can vary drastically.

5 Experiments

In this section, we hope to investigate the performance and speed improvements of training consistency models rather than diffusion models with reinforcement learning. We compare our method to DDPO (Black et al., 2024), a state-of-the-art policy gradient method for finetuning diffusion models. First, we test how well RLCM is able to both efficiently optimize the reward score and maintain the qualitative integrity of the pretrained generative model. We show both learning curves and representative qualitative examples of the generated images on tasks defined by Black et al. (2024).

Algorithm 1 Policy Gradient Version of RLCM

Input: Consistency model policy $\pi_\theta = f_\theta(\cdot,\cdot) + Z$, finetune horizon H, prompt set $\mathcal{P}$, batch size b, inference pipeline P
for $i = 1$ **to** M **do**
  Sample b contexts from $\mathcal{C}$, $\boldsymbol{c} \sim \mathcal{C}$.
  $\boldsymbol{x}_0 \leftarrow P(f_\theta, H, \boldsymbol{c})$ ▷ where $\boldsymbol{x}_0$ is the batch of images
  Normalize rewards $r(\boldsymbol{x}_0, \boldsymbol{c})$ per context
  Split $\boldsymbol{x}_0$ into k minibatches.
  for minibatch $m = 0$ to `ceil(length(`$\boldsymbol{x}_0$`)/minibatch_size)` **do**
    for $t = 0$ to H **do**
      Update θ using rule:

$$\nabla_\theta \left[\min \left\{ r(\boldsymbol{x}_{0,m}, \boldsymbol{c}) \cdot \frac{\pi_{\theta_{m+1}}(a_t|s_t)}{\pi_{\theta_m}(a_t|s_t)}, r(\boldsymbol{x}_{0,m}, \boldsymbol{c}) \cdot \texttt{clip}\left(\frac{\pi_{\theta_{m+1}}(a_t|s_t)}{\pi_{\theta_m}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon \right) \right\} \right]$$

    end for
  end for
end for
Output trained consistency model $f_\theta(\cdot,\cdot)$

Next we show the speed and compute needs for both train and test time of each finetuned model to test whether RLCM is able to maintain a consistency model's benefit of having a faster inference time. We then conduct an ablation study, incrementally decreasing the inference horizon to study RLCM's tradeoff for faster train/test time and reward score maximization. Finally, we qualitatively evaluate RLCM's ability to generalize to text prompts and subjects not seen at test time to showcase that the RL finetuning procedure did not destroy the base pretrained model's capabilities.

For fair comparison, both DDPO and RLCM finetune the Dreamshaper v7[1] and its latent consistency model counterpart respectively[2] (Luo et al., 2023). Dreamshaper v7 is a finetune of stable diffusion (Rombach et al., 2022). For DDPO, we used the same hyperparameters and source code[3](Black et al., 2024) provided by the authors. We found that the default parameters performed best when testing various hyperparamters. Please see Appendix B.2 for more details on the parameters we tested.

Compression The goal of compression is to minimize the filesize of the image. Thus, the reward received is equal to the negative of the filesize when compressed and saved as a JPEG image. The highest rated images for this task are images of solid colors. The prompt space consisted of 398 animal categories.

Incompression Incompression has the opposite goal of compression: to make the filesize as large as possible. The reward function here is just the filesize of the saved image. The highest rated mages for this task are random noise. Similar to the comparison task, this task's prompt space consisted of 398 animal categories.

Aesthetic The aesthetic task is based off of the LAION Aesthetic predictor (Schumman, 2022) which was trained on 176,000 human labels of aesthetic quality of images. This aesthetic predictor is a MLP on top of CLIP embeddings (Radford et al., 2021). The images which produce the highest reward are typically artwork. This task has a smaller set of 45 animals as prompts.

Prompt Image Alignment We use the same task as Black et al. (2024) in which the goal is to align the prompt and the image more closely without human intervention. This is done through a

[1] https://huggingface.co/Lykon/dreamshaper-7
[2] https://huggingface.co/SimianLuo/LCM_Dreamshaper_v7
[3] https://github.com/kvablack/ddpo-pytorch

procedure of first querying a LLaVA model (Liu et al., 2023) to determine what is going on in the image and taking that response and computing the BERT score (Zhang et al., 2019) similarity to determine how similar it is to the original prompt. This values is then used as the reward for the policy gradient algorithm.

5.1 RLCM vs. DDPO Performance Comparisons

Following the sample complexity evaluation proposed in Black et al. (2024), we first compare DDPO and RLCM by measuring how fast they can learn based on the number of reward model queries. As shown in Fig. 4, RLCM has better performance on three out of four of our tested tasks in terms of number of reward queries. Note that for the prompt-to-image alignment task, the initial consistency model finetuned by RLCM has lower performance than the initial diffusion model trained by DDPO. RLCM is able to close the performance gap between the consistency and diffusion model through RL finetuning[4]. Fig. 3 demonstrates that similar to DDPO, RLCM is able to train its respective generative model to adapt to various styles just with a reward signal without any additional data curation or supervised finetuning.

5.2 Train and Test Time Analysis

To show faster training advantage of the proposed RLCM, we compare to DDPO in terms of training time in Fig. 5. Here we experimentally find that RLCM has a significant advantage to DDPO in terms of the number of GPU hours required in order to achieve similar performance. On all tested tasks RLCM reaches the same or greater performance than DDPO, notably achieving a x17 speedup in

[4]It is possible that this performance difference on the compression and incompression tasks are due to the consistency models default image being larger. However, in the prompt image alignment and aesthetic tasks, we resized the images before reward calculation.

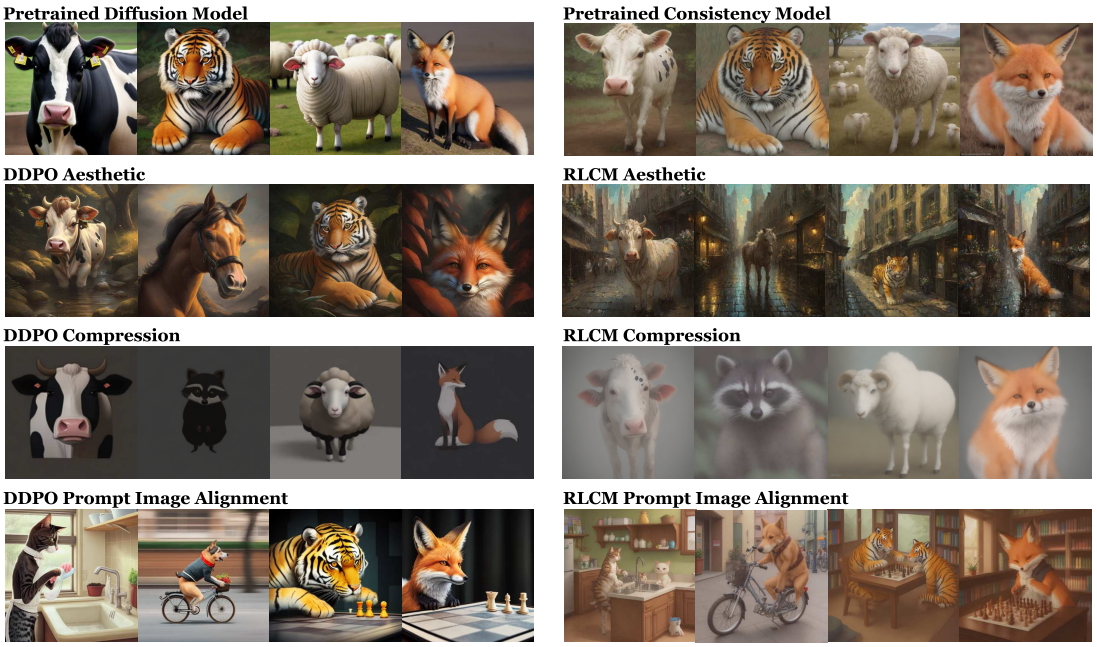

Figure 3: **Qualitative Generations:** Representative generations from the pretrained models, DDPO, and RLCM. Across all tasks, we see that RLCM is able to finetune output of the model to fit specific reward functions. Due to the lack of regularization to the pretrained model, some artifacts (seen in the compression task) and significant similarity in output are indeed seen).

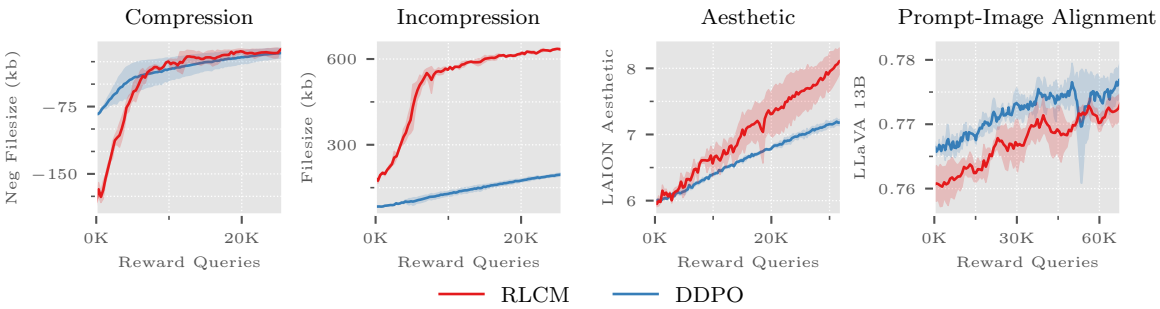

Figure 4: **Learning Curves:** Training curves for RLCM and DDPO by number of reward queries on compressibility, incompressibility, aesthetic, and prompt image alignment. We plot three random seeds for each algorithm and plot the mean and standard deviation across those seeds. RLCM seems to produce either comparable or better reward optimization performance across these tasks.

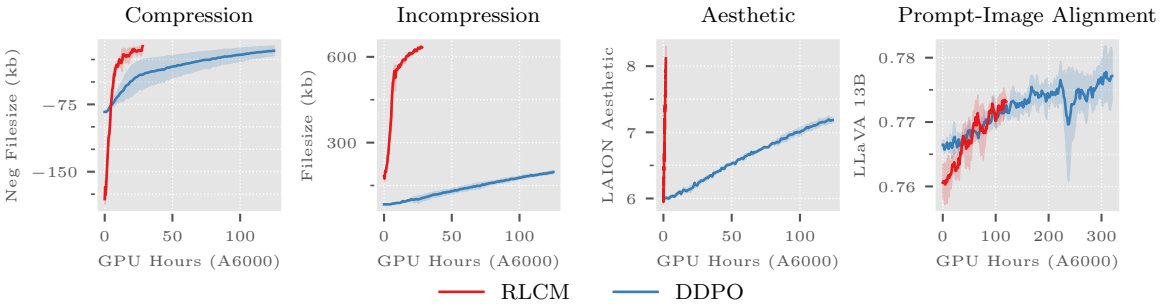

Figure 5: **Training Time:** Plots of performance by runtime measured by GPU hours. We report the runtime on four NVIDIA RTX A6000 across three random seeds and plot the mean and standard deviation. We observe that in all tasks RLCM noticeably reduces the training time while achieving comparable or better reward score performance.

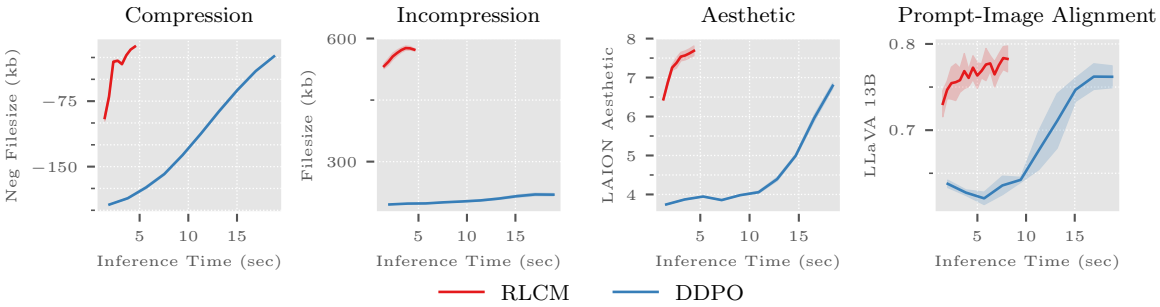

Figure 6: **Inference Time:** Plots showing the inference performance as a function of time taken to generate. For each task, we evaluated the final checkpoint obtained after training and measured the average score across 100 trajectories at a given time budget on 1 NVIDIA RTX A6000 GPU. We report the mean and std across three seeds for every run. Note that for RLCM, we are able to achieve high scoring trajectories with a smaller inference time budget than DDPO. Final reward values may differ from previous plots due to random selection of prompts used for measurement.

training time on the Aesthetic task. This is most likely due to a combination of factors – the shorter horizon in RLCM leads to faster online data generation (rollouts in the RL training procedure) and policy optimization (e.g., less number of backpropagations for training the networks).

Fig. 6 compares the inference time between RLCM and DDPO. For this experiment, we measured the average reward score obtained by a trajectory given a fixed time budget for inference. Similar to

training, RLCM is able to achieve a higher reward score with less time, demonstrating that RLCM retains the computational benefits of consistency models compared to diffusion models. Note that a full rollout with RLCM takes roughly a quarter of the time for a full rollout with DDPO.

5.3 Ablation of Inference Horizon for RLCM

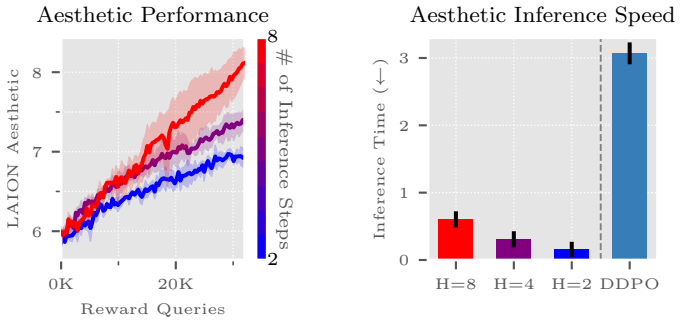

Figure 7: **Inference time vs Generation Quality:** We measure the performance of the policy gradient instantiation of RLCM on the aesthetic task at 3 different values for the number of inference steps (left) in addition to measuring the inference speed in seconds with varied horizons (right). We report the mean and std across three seeds.

We further explore the effect of finetuning a consistency model with different inference horizons. That is we aimed to test RLCM's sensitivity to H. As shown in Fig. 7 (left), increasing the number of inference steps leads to a greater possible gain in the reward. However, Fig. 7 (right) shows that this reward gain comes at the cost of slower inference time. This highlights the *inference time vs generation quality* tradeoff that becomes available by using RLCM. Nevertheless, regardless of the number of inference steps chosen, RLCM enjoys faster inference time than diffusion model based baselines.

5.4 Qualitative Effects on Generalization

We now test our trained models on new text prompts that do not appear in the training set. Specifically, we evaluated our trained models on the aesthetic task. As seen in Fig. 8 which consists of images of prompts that are not in the training dataset, the RL finetuning does not influence the ability of the model to generalize. We see this through testing a series of prompts ("bike", "fridge", "waterfall", and "tractor") unseen during training.

Figure 8: **Prompt Generalization:** We observe that RLCM is able to generalize to other prompts without substantial decrease in aesthetic quality. The prompts used to test generalization are "bike", "fridge", "waterfall", and "tractor".

5.5 Convergence Results of Tasks

To compare fairly to Black et al. (2024), we only train for only the same number of reward queries which means that in two tasks (Aesthetic and Prompt Image Alignment) convergence of the tasks is not shown.

We trained DDPO and RLCM for longer on the aesthetic task and observed that RLCM asymptotically arrived at the approximate maximum reward value (value 10 is the maximum reward available in the training dataset for the reward model). For DDPO, when it runs longer (after 72 hours), it reaches a reward around 9.5, but unfortunately crashes.

We also attempted to run the text-image alignment task longer for DDPO, unfortunately we observed the same crashing behavior. We suspect that it is due to the fixed learning rate schedule used in the original DDPO codebase (note that for fair comparison, we use the original DDPO codebase with the default hyperparameters proposed by the authors of DDPO). Applying strategies like learning rate decay may stabilize DDPO, but it would require additional hyperparameter tuning for DDPO.

5.6 Known Limitations

The main known limitation observed throughout the use of RLCM is overfitting to the reward function. Indeed, as seen in Fig. 3, unrealistic generations as seen in the compression task or extremely similar backgrounds like in the aesthetic task do arise. In cases where such overfitting is undesirable, a KL regularized loss which incorporates some measure of divergence between the currently trained model and the initial model will improve generations. However, this was not a focus of this work.

6 Conclusion and Future Directions

We present RLCM, a fast and efficient RL framework to directly optimize a variety of rewards to train consistency models. We empirically show that RLCM achieves better performance than a diffusion model RL baseline, DDPO, on most tasks while enjoying the fast train and inference time benefits of consistency models. Finally, we provide qualitative results of the finetuned models and test their downstream generalization capabilities.

There remain a few directions unexplored which we leave to future work. In particular, the specific policy gradient method presented uses a sparse reward. It may be possible to use a dense reward using the property that a consistency model always predicts to x_0. Another future direction is the possibility of creating a loss that further reinforces the consistency property, further improving the inference time capabilities of RLCM policies.

7 Social Impact

We believe that it is important to urge caution when using such fine-tuning methods. In particular, these methods can be easily misused by designing a malicious reward function. We therefore urge this technology be used for good and with utmost care.

Code References

We use the following open source libraries for this work: NumPy (Harris et al., 2020), diffusers (von Platen et al., 2022), and PyTorch (Paszke et al., 2017)

Acknowledgements

WS acknowledges funding from NSF IIS-2154711 and NSF CAREER 2339395. We would also like to acknowledge Yijia Dai and Dhruv Sreenivas for their helpful technical conversations.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning, 2024.

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

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *arXiv preprint arXiv:2305.16381*, 2023.

Zhaolin Gao, Jonathan D Chang, Wenhao Zhan, Owen Oertell, Gokul Swamy, Kianté Brantley, Thorsten Joachims, J Andrew Bagnell, Jason D Lee, and Wen Sun. Rebel: Reinforcement learning via regressing relative rewards. *arXiv preprint arXiv:2404.16767*, 2024.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

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

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

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

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Chrisoph Schumman. Laion aesthetics. https://laion.ai/blog/laion-aesthetics/, 2022.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. Pmlr, 2014.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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

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

Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control, 2024.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. 8(3):229–256, 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5826–5835, 2021.

A Consistency Models

We reproduce the consistency model algorithm from Song et al. (2023).

Algorithm 2 Consistency Model Multi-step Sampling Procedure (Song et al., 2023)

Input: Consistency model $\pi = f_\theta(\cdot, \cdot)$, sequence of time points $\tau_1 > \tau_2 > \ldots > \tau_{N-1}$, initial noise $\widehat{\boldsymbol{x}}_T$
$\boldsymbol{x} \leftarrow f(\widehat{\boldsymbol{x}}_T, T)$
for $n = 1$ **to** N-1 **do**
    $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\widehat{\boldsymbol{x}}_{\tau_n} \leftarrow \boldsymbol{x} + \sqrt{\tau_n^2 - \epsilon^2}\boldsymbol{z}$
    $x \leftarrow f(\widehat{\boldsymbol{x}}_{\tau_n}, \tau_n)$
end for
Output: $\boldsymbol{x}$

B Experiment Details

B.1 Hyperparameters

Parameters	Compression	Incompression	Aesthetic	Prompt Image Alignment
Advantage Clip Maximum	10	10	10	10
Batches Per Epoch	10	10	10	6
Clip Range	0.0001	0.0001	0.0001	0.0001
Gradient Accumulation Steps	2	2	4	20
Learning Rate	0.0001	0.0001	0.0001	0.0001
Max Grad Norm	5	5	5	5
Pretrained Model	Dreamshaper v7	Dreamshaper v7	Dreamshaper v7	Dreamshaper v7
Number of Epochs	100	100	100	118
Horizon (Number of inference steps)	8	8	8	16
Number of Sample Inner Epochs	1	1	1	5
Sample Batch Size (per GPU)	4	4	8	8
Rolling Statistics Buffer Size	16	16	32	32
Rolling Statistics Min Count	16	16	16	16
Train Batch Size (per GPU)	2	2	2	2
Number of GPUs	4	4	4	3
LoRA rank	16	16	8	16
LoRA α	32	32	8	32
Consistency Model Time Horizon	1000	1000	1000	1000

Table 1: Hyperparameters for all tasks (Compression, Incompression, Aesthetic, Prompt Image Alignment)

We note that a 4th gpu was used for Prompt Image Alignment as a sever for the LLaVA (Liu et al., 2023) and BERT models (Zhang et al., 2019) to form the reward function.

B.2 Hyperparameter Sweep Ranges

These hyperparameters were found via a sweep. In particular we swept the learning rate for values in the range [1e-5,3e-4]. Likewise we also swept the number of batches per epoch and gradient accumulation steps but found that increasing both of these values led to greater performance, at the cost of sample complexity. We also swept the hyperparameters for DDPO, our baseline, but found that the provided hyperparameters provided the best results. In particular we tried lower batch size to increase the sample complexity of DDPO but found that this made the algorithm unstable. Likewise, we found that increasing the number of inner epochs did not help performance. In fact, it had quite the opposite effect.

B.3 Details on Task Prompts

We followed (Black et al., 2024) in forming the prompts for each of the tasks. The prompts for incompression, compression, and aesthetic took the form of `[animal]`. For the prompt image alignment task, the prompt took the form of `a [animal] [task]` where the `a` was conjugated depending on the animal. The prompts for compression and incompression were the animal classes of Imagenet (Deng et al., 2009). Aesthetic was a set of simple animals, and prompt image alignment used the animals from the aesthetic task and chose from the tasks: `riding a bike`, `washing the dishes`, `playing chess`.

C Statistical Testing on Results

Following Agarwal et al. (2021), we compute 95% stratified bootstrap confidence intervals of the IQM, Mean, Median, and Optimality gap over the 4 tasks tested. We find that there is a statistically significant difference in rewards favoring RLCM for the mean, median, and optimality gap. There is slight overlap in the confidence intervals for the IQM.

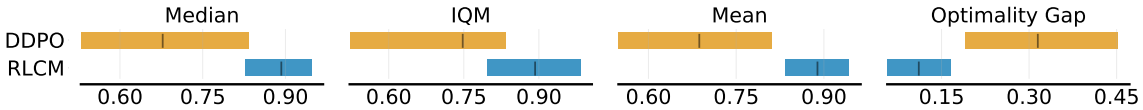

Figure 9: **Statistical Tests**: Stratified bootstrap confidence intervals and establish statistically significant difference in reward favoring RLCM.

D Additional Samples from RLCM

We provide random samples from RLCM at the end of training on aesthetic and prompt image alignment. Images from converged compression and incompression are relatively uninteresting and thus omitted.

D.1 Aesthetic Task

D.2 Prompt Image Alignment