Algorithms for Optimal Adaptation of Diffusion Models to Reward Functions

Krishnamurthy (Dj) Dvijtoham [*1] Shayegan Omidshafei [*2] Katherine Collins [1 3] Deepak Ramachandran [2]
Kimin Lee [2] Ying Fan [4 2] Jeremiah Liu [2] Milad Nasr [1] Mohammad Ghavamzadeh [2]

Abstract

We develop algorithms for adapting pretrained diffusion models to optimize reward functions while retaining fidelity to the pretrained model. We propose a general framework for this adaptation that trades off fidelity to a pretrained diffusion model and achieving high reward. Our algorithms take advantage of the continuous nature of diffusion processes to pose reward-based learning either as a trajectory optimization or continuous state reinforcement learning problem. We demonstrate the efficacy of our approach across several application domains, including the generation of time series of household power consumption and images satisfying specific constraints like the absence of memorized images or corruptions.

1. Introduction

Diffusion models have become the de-facto state of the art in image, video and audio generation, following recent breakthroughs (Song et al., 2020a;b; Ramesh et al., 2021; Ho et al., 2022). This has opened the door to several compelling applications of diffusion models including image editing and super resolution, time series forecasting and medical image reconstruction.

In many application domains, there are constraints or preferences one may want to express regarding the samples generated by a diffusion model. A convenient way to capture these is using a reward function that evaluates the quality of the generated sample with respect to a specific consideration. For example, for image generation, since diffusion models are typically trained on images scraped from the web (for example, (Schuhmann et al., 2022)), one may want to avoid generating samples containing image artifacts like timestamps or logos, or inappropriate content. While progress has been made towards these goals by using simple filtering techniques, the optimal adaptation of a pretrained diffusion model towards maximizing a reward while maintaining its ability to generate high quality samples remains an open question. We address this problem by posing reward-based adaptation as either a continuous state reinforcement learning or stochastic control problem, using off-the-shelf reinforcement learning on trajectory optimization algorithms to accomplish optimal adaptation.

Our formulation is inspired by analogous work on reward-based fine tuning for language models, which has seen great success as noted in recent breakthroughs on reinforcement learning with human feedback (Stiennon et al., 2020; Ouyang et al., 2022). These works have an additional step of learning the reward function from human preferences. In this work, we do not study reward functions learned from human feedback; rather, we focus on developing the right algorithms for adapting diffusion models given a reward function. However, our framework is *compatible with any reward function*, including those which are handcrafted, learned by solving a classification task (for example classifying images with artifacts versus images with no artifacts), or learned from human feedback.

1.1. Contributions

1 We develop a novel theoretical formulation of stochastic control of diffusion processes to optimize a combination of a fidelity term that penalizes KL divergence to a pretrained diffusion model and a reward function capturing desired behavior. We show that a special case of our framework reduces to a deterministic optimal control problem that can be solved using gradient based optimization, as opposed to general purpose reinforcement learning (RL) algorithms.

2 We demonstrate how off-the-shelf actor critic policy gradient algorithms can be tuned towards being successful in the high dimensional continuous state spaces encountered in RL based fine tuning of diffusion mod-

*Equal contribution [1]Google DeepMind [2]Google Research [3]University of Cambridge [4]University of Wisconsin, Madison. Correspondence to: Krishnamurthy (Dj) Dvijotham <dvij@cs.washington.edu>.

Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023. Copyright 2023 by the author(s).

els. We also show that our custom gradient based trajectory optimization algorithm can outperform RL on some tasks.

3 We conduct experiments demonstrating the effectiveness of our methods across a variety of domains and modeling tasks:

– Modeling the time series of power consumption in a household adapted to preferentially sample time series patterns with large ramps up or down ("spikes") in power consumption.
– Modeling images from the MNIST dataset corrupted with time stamps, adapted to prevent the generation of images containing time stamps.
– Modeling images from the MNIST dataset with a recurring canary image that is memorized by the diffusion model, adapted to prevent generation of the memorized canary images.

2. Background and Theoretical Formulation

We work with diffusion models that describe probability distributions over a random variable taking values in $\mathbb{R}^n$. The random variable is assumed to come from an underlying distribution $\mathbb{P}_{\text{data}}$. We use $\mathcal{U}(\cdot)$ and $\mathcal{N}(\cdot,\cdot)$ to refer to uniform and normal distributions, and I to be the identity matrix.

Forward diffusion process: We consider the diffusion process that perturbs this data distribution to Gaussian noise, according to the following stochastic differential equation (SDE)

$$dx = f\left(t\right) x dt + g\left(t\right) d\omega, \qquad t \in [0,1],\ x\left(0\right) \sim \mathbb{P}_{\text{data}}, \tag{1}$$

where $f : [0,1] \mapsto \mathbb{R}$ and $g : [0,1] \mapsto \mathbb{R}$ are functions determined by pre-set schedules meant to diffuse $\mathbb{P}_{\text{data}}$ to the standard normal distribution $\mathcal{N}\left(0, I\right)$. The marginal distribution of $x\left(t\right)$ evolving according to (1) conditioned on $x\left(0\right)$ is

$$P\left(x\left(t\right)|x\left(0\right)\right) = \mathcal{N}\left(\alpha\left(t\right) x\left(t\right); \sigma^2\left(t\right) I\right),$$

where $\alpha : [0,1] \mapsto \mathbb{R}_+$ and $\sigma : [0,1] \mapsto \mathbb{R}_+$ are related to f and g via

$$f\left(t\right) = \frac{d\log\left(\alpha\left(t\right)\right)}{dt},$$
$$g\left(t\right) = \sqrt{\frac{d\sigma\left(t\right)^2}{dt} - 2\frac{d\log\left(\alpha\left(t\right)\right)}{dt}\sigma\left(t\right)^2}.$$

We denote by $p_t\left(x\right)$ the *unconditional* marginal probability of $x\left(t\right)$ evolving according to (1).

Reverse diffusion process: Beginning with the work of Song et al. (2020b), the above SDE formalism has been used to derive successful training objectives for diffusion models. Diffusion models are trained to approximate the score function, i.e., the gradient of the log of the density $p_t\left(x\right)$, which we denote it by $s\left(x,t\right) \triangleq \nabla_x \log\left(p_t\left(x\right)\right)$. Once the score function is learned, samples can be generated by reversing time in the SDE in (1). The exact time reversal for the above SDE can be written as (Anderson, 1982; Song et al., 2020b)

$$dx = \left(f\left(t\right) x - g\left(t\right)^2 s(x,t)\right) dt + g\left(t\right) d\omega \tag{2a}$$
$$t \in [0,1],\ x\left(1\right) \sim \mathcal{N}\left(0, I\right), \tag{2b}$$

which is solved backward in time starting at $t = 1$. Then $x\left(0\right)$ is guaranteed to be distributed as $\mathbb{P}_{\text{data}}$, and thus, solving the reverse SDE, one can generate samples from $\mathbb{P}_{\text{data}}$.

It can be shown (Zhang et al., 2022) that the following family of SDEs preserve the marginals $p_t(x),\ \forall t \in [0,1]$:

$$dx = \left(f\left(t\right) x - \left(\frac{1+\lambda^2}{2}\right) g\left(t\right)^2 s\left(x,t\right)\right) dt + \lambda g\left(t\right) d\omega \tag{3a}$$
$$x\left(1\right) \sim \mathcal{N}\left(0, I\right), \tag{3b}$$

for any $\lambda \geq 0$. For convenience, we consider the SDE (3) in reversed-time, i.e.,

$$dx = -\left(f\left(t\right) x - \left(\frac{1+\lambda^2}{2}\right) g\left(t\right)^2 s\left(x,t\right)\right) dt + \lambda g\left(t\right) d\omega \tag{4a}$$
$$x\left(0\right) \sim \mathcal{N}\left(0, I\right). \tag{4b}$$

Distribution over trajectories: A trajectory $\boldsymbol{x} : [0,1] \mapsto \mathbb{R}^n$ is a mapping from time $t \in [0,1]$ to data-points $x \in \mathbb{R}^n$. We denote by P^s_λ the probability measure over trajectories induced by the SDE in (4).

Training the score function: In practice, the score function s is unknown but has to be learned from data. This is usually done by training a neural network $\epsilon_\theta\left(x,t\right)$ to approximate $\frac{-s(x,t)}{\sigma(t)}$. It can be shown that this can be done by training a neural network with the following objective

$$\min_\theta \underset{\substack{x\sim\mathbb{P}_{\text{data}}\\ t\sim\mathcal{U}[0,1]\\ \omega\sim\mathcal{N}(0,I)}}{\mathbb{E}} \left[\left\|\epsilon_\theta\left(x + \sigma\left(t\right)\omega, t\right) - \omega\right\|^2 \frac{\delta\left(t\right)}{\sigma^2\left(t\right)}\right],$$

where $\delta : [0,1] \mapsto \mathbb{R}_+$ is a schedule chosen to balance sample quality with data likelihood (Song et al., 2020b).

3. Theoretical Formulation of Reward-based Adaptation

Our goal is to develop a procedure to adapt the parameters or generative process of a diffusion model in order to optimize a reward model, while retaining fidelity to the original diffusion model.

Defining the adaptation problem requires the following components:

- ***Reward function:*** This measures the desirability of the samples generated by the diffusion model w.r.t. a criterion of interest (e.g., non-toxicity or absence of artifacts). We denote the reward function as $r : \mathbb{R}^n \mapsto \mathbb{R}$.

- ***Pretrained diffusion model:*** This is parameterized as a score function $s_\theta(x, t)$, where θ represents the parameters of a parametric model (neural network) trained to approximate $s_\theta(x, t)$.

- ***Fidelity penalty:*** This measures the penalty we incur for deviating from the original diffusion model $\mathbb{KL}(P^u_\lambda \parallel P^{s_\theta}_\lambda)$, where P^u_λ and $P^{s_\theta}_\lambda$ are the probability measures over trajectories induced by the SDE in (4) for the new and pretrained diffusion models. Noting that the KL divergence blows up as $\lambda \to 0$, we scale the KL divergence by λ^2 to ensure a finite limit. However, the "correct" weighting between the fidelity penalty and the reward function is a hyperparameter that is tuned to achieve the right balance of sample quality and high reward in our experiments.

The following theorem provides the theoretical basis for our reward-based adaptation algorithms.

Theorem 3.1. *Consider the objective function*

$$\max_u \; -\lambda^2\, \mathbb{KL}(P^u_\lambda \parallel P^{s_\theta}_\lambda) + \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P^u}[r(x(1))],$$

where the maximization is over all measurable functions $u : \mathbb{R}^n \times [0,1] \mapsto \mathbb{R}^n$. The above objective can be rewritten as the following stochastic optimal control problem:

$$\max_u \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P^u_\lambda}\left[\frac{-\left(1+\lambda^2\right)^2 g(t)^2}{8} CC + r(\boldsymbol{x}(1))\right] \tag{5a}$$

$$\text{where } CC = \int_{t=0}^{1} \left\| u(\boldsymbol{x}(t), t) - s_\theta(\boldsymbol{x}(t), t)\right\|^2 dt \tag{5b}$$

where we denote by $u^\star$ its optimal solution.

Moreover, a sample $\boldsymbol{x}$ from the optimally controlled distribution $P^{u^\star}_0$ can be generated by sampling $z \sim \mathcal{N}(0, I)$ and solving the following trajectory optimization problem:

$$\max_{\substack{\boldsymbol{x}:[0,1]\mapsto\mathbb{R}^n \\ \boldsymbol{x}(0)=z}} \int_{t=0}^{1} \frac{-O}{8g(t)^2} + r(\boldsymbol{x}(1)). \tag{6a}$$

$$\text{where } O = \left\| \dot{\boldsymbol{x}}(t) + f(t)\boldsymbol{x}(t) - \frac{1}{2} g(t)^2 s_\theta(\boldsymbol{x}(t), t)\right\|^2 dt \tag{6b}$$

Proof. (5) can be derived by applying Girsanov's theorem to compute the Radon-Nikodym deriverive of $\frac{dP^u_\lambda}{dP^{s_\theta}_\lambda}$. The expectation of the logrithm of this quantity is (5). Taking limits as $\lambda \to 0$, we obtain an expectation where the only stochasticity is in $\boldsymbol{x}(0)$. This enables us to rewrite the optimization in a pathwise-form, where the optimal u can be determined purely based on $\boldsymbol{x}(0)$, by solving a problem that can be shown to be equivalent to (6). Details can be found in Appendix E. □

Theorem 3.1 suggests two algorithmic solutions for reward-based adaptation: **1)** a general purpose RL algorithm applied to (5) and **2)** a more specialized trajectory optimization algorithm (for the special case $\lambda = 0$) to solve (6). We explore both algorithms in this paper, show their effectiveness and offer different advantages and trade-offs for each of them (see Section 6). We present a theoretical comparison of the two approaches in the next section.

4. Algorithms

In order to be implementable, our practical algorithms for solving the formulations in Theorem 3.1 operate in discrete time. We primarily work with the standard Euler-Mayurama discretization of (3), although our approach can be adapted easily to other discretization schemes. Concretely, we consider a discretization with a fixed timestep $h = \frac{1}{T}$ where $T \in \mathbb{N}$ is an integer denoting the number of discrete time steps. We denote by $\mathcal{T} = \{0, h, 2h, \ldots, 1\}$ the set of discrete timesteps and by $\boldsymbol{x} = (x_0, x_h, \ldots, x_1)$ a trajectory. As a result, for each $t \in \mathcal{T}$ we have $\boldsymbol{x}(t) = x_t$. We use $P^{s_\theta}_\lambda$ and P^θ_λ interchangeably as the probability measure over trajectories induced by the SDE. Given this discretization, we may write (3) as

$$x_0 \sim \mathcal{N}(0, I) \tag{7a}$$

$$x_{t+h} \sim \pi_\theta(\cdot \mid x_t, t) \triangleq \mathcal{N}\left(x_t - hF^\lambda_\theta(x_t, t), hg(t)^2 \lambda^2 I\right), \tag{7b}$$

$$F^\lambda_\theta(x, t) \triangleq f(t)x - \left(\frac{1+\lambda^2}{2}\right) g(t)^2 s_\theta(x, t) \tag{7c}$$

$$\forall x \in \mathbb{R}^n,\; t \in [0, 1]. \tag{7d}$$

Note that in (7) we use $\pi_\theta\left(\cdot \mid \boldsymbol{x}\left(t\right), t\right)$ to refer to the distribution of $\boldsymbol{x}\left(t+h\right)$ conditioned on $\boldsymbol{x}\left(t\right)$, as opposed to the likelihood of a specific transition from $\boldsymbol{x}\left(t\right)$ to $\boldsymbol{x}\left(t+h\right)$.

4.1. Trajectory Optimization

Given the above time discretization, the objective in (6) can be approximated as

$$\max_{\substack{x_h,\ldots,x_1\\ x_0=z}} \sum_{t\in\mathcal{T}\setminus\{1\}} \frac{-1}{8g\left(t\right)^2}\left\|\frac{x_{t+h}-x_t}{h}+F_\theta^0\left(x,t\right)\right\|^2 h+\frac{1}{\beta}\cdot r\left(x_1\right), \tag{8}$$

where we introduce an additional scaling factor β that we sweep over in our experiments to control the tradeoff between the fidelity to the original diffusion model and optimizing the control cost.

This can easily be solved by gradient-based optimization approaches as long as r is differentiable. The overall algorithm to generate a sample using trajectory optimization is in Algorithm 2 where we can use any (stateful) optimizer like Adam (Kingma and Ba, 2014) with internal optimizer state opt-s and that updates the decision variables $\boldsymbol{x}$ given access to the optimizer state and gradients of the objective function.

4.2. Reinforcement Learning

We cast the diffusion model adaptation problem as a continuous-state continuous-action Markov Decision Process (MDP), where we tune the parameters θ of the estimated score function $s_\theta(x,t)$ via policy gradient (PG) methods. We denote the adapted parameters by ϕ. Algorithm 1 presents the pseudo-code of our method, which we describe its details below.

Any off-the-shelf PG algorithm can be used to optimize ϕ. We use Proximal Policy Optimization (PPO) (Schulman et al., 2017), which is an actor-critic method in our experiments. However, we modify PPO to only estimate the PG due to the rewards. For the KL penalty term, we estimate the gradient directly since we have direct access to both the controlled and uncontrolled diffusion models as well as their gradients.

Reward-based objective: Suppose the current policy parameters are ϕ^{old}. The PPO objective is:

$$J_{\phi^{\text{old}}}^{\text{CLIP}}\left(\phi,t,\boldsymbol{x}\right)= \tag{9a}$$

$$\min\left\{\rho_t(\phi,\boldsymbol{x})\hat{A}_t\ ,\ \text{clip}\big(\rho_t(\phi,\boldsymbol{x}),1-\epsilon,1+\epsilon\big)\hat{A}_t\right\} \tag{9b}$$

$$t\sim\mathcal{U}\left(\mathcal{T}\setminus\{1\}\right),\ \boldsymbol{x}\sim P_\lambda^{\phi^{\text{old}}}, \tag{9c}$$

where $\rho_t(\phi,\boldsymbol{x})=\frac{\pi_\phi(\boldsymbol{x}(t+h)|\boldsymbol{x}(t),t)}{\pi_{\phi old}(\boldsymbol{x}(t+h)|\boldsymbol{x}(t),t)}$ is the policy ratio between the current and previous iteration policy, $\hat{A}_t$ denotes an estimator of the advantage function at timestep t, and clip$(\cdot)$ clips the probability ratio to be within the interval $(1-\epsilon,1+\epsilon)$ (see Schulman et al. 2017 for details).

KL objective: While it is feasible to compute and backprop through the KL divergence $\mathbb{KL}\left(P_\lambda^\phi \parallel P_\lambda^\theta\right)$ this incurs significant computational burden in practice as it requires extensive sampling to compute an accurate estimate of the expectation. Instead, we use

$$\hat{\mathbb{KL}}_{\phi^{\text{old}}}\left(P_\lambda^\phi \parallel P_\lambda^\theta\right)\triangleq \tag{10a}$$

$$\mathbb{E}_{\boldsymbol{x}\sim P_\lambda^{\phi^{\text{old}}}}\left[\sum_{t\in\mathcal{T}\setminus\{1\}}\log\left(\frac{\pi_\phi\left(\boldsymbol{x}\left(t+h\right)|\boldsymbol{x}\left(t\right),t\right)}{\pi_\theta\left(\boldsymbol{x}\left(t+h\right)|\boldsymbol{x}\left(t\right),t\right)}\right)\right] \tag{10b}$$

$$=\frac{1}{h}\mathbb{E}_{\substack{\boldsymbol{x}\sim P_\lambda^{\phi^{\text{old}}}\\ t\sim\mathcal{U}(\mathcal{T}\setminus\{1\})}}\left[\mathbb{KL}\left(\pi_\phi\left(\cdot|\boldsymbol{x}\left(t\right),t\right)\parallel\pi_\theta\left(\cdot|\boldsymbol{x}\left(t\right),t\right)\right)\right], \tag{10c}$$

which simplifies the task, as the measure used to compute the expectation no longer depends on the updated parameters ϕ (the detailed derivation of the above equations can be found in Appendix E). So long as ϕ is close to ϕ^{old}, this is a reasonable estimate of the KL divergence; nicely, such a condition is often achievable in practice by choosing a small learning rate for the policy and only using samples from the most recent policy (rather than from a replay buffer).

***Overall objective*:** Given all the above, the overall policy objective to be maximized at each step of the policy gradient method is

$$J_{\phi^{\text{old}}}^{\text{Policy}}(\phi,t,\boldsymbol{x})=J_{\phi^{\text{old}}}^{\text{CLIP}}\left(\phi,t,\boldsymbol{x}\right)- \frac{\beta\lambda^2}{h}\hat{\mathbb{KL}}_{\phi^{\text{old}}}\left(\pi_\phi\left(\cdot|\boldsymbol{x}\left(t\right),t\right)\parallel\pi_\theta\left(\cdot|\boldsymbol{x}\left(t\right),t\right)\right), \tag{11}$$

where $t\sim\mathcal{U}\left(\mathcal{T}\setminus\{1\}\right)$, $\boldsymbol{x}\sim P_\lambda^{\phi^{\text{old}}}$, and β is a hyperparameter used to control the weighing between the PPO objective and the KL penalty term.

Algorithmic and architecture details: In our experiments, advantage function estimation $\hat{A}_t$ occurs as in vanilla PPO (Schulman et al., 2017), which involves training a value network using the raw terminal rewards r. Here, we use a value network with a torso architecture identical to the uncontrolled diffusion model, and an MLP head to generate scalar values. Policy and value network torso parameters are warm-started by the uncontrolled model parameters and trained independently via PPO. The value function MLP head is randomly initialized.

Algorithm 1 Reinforcement learning based adaptation of diffusion models (RL)

```
    Input: $\theta, r, \lambda$
    Output: Fine-tuned parameters $\phi$
 1: procedure RL($\theta, r, \lambda$)
 2:     Initialize $\phi \leftarrow \theta,\ \phi^{\text{old}} \leftarrow \theta$
 3:     Initialize torso parameters of value network using $\theta$
 4:     for $k = 0, 1, \ldots$ do
 5:         Sample $\boldsymbol{x} \sim P_{\lambda}^{\phi^{\text{old}}}$
 6:         Sample $t \sim \mathcal{U}(\mathcal{T} \setminus \{1\})$
 7:         Set $r_t = 0$ if $t < 1,\ r_1 = r(x_1)$
 8:         Update value function using $\hat{A}_t$ as in Schulman et al. (2017)
 9:         Compute policy update
                $$\phi \leftarrow \arg\max_{\phi, \boldsymbol{x}, t} J_{\phi^{\text{old}}}^{\text{Policy}}(\phi^{\text{old}}, \boldsymbol{x}, t)$$
10:     end for
11:     return $\phi$
12: end procedure
```

Algorithm 2 Trajectory optimization based online adaptation of diffusion models (TrajOpt)

```
    Input: $\theta, r, \lambda$
    Output: Sample $x$ from the adapted diffusion process
  procedure TrajOpt($\theta, r, \lambda$)
      Generate trajectory $\boldsymbol{x} \sim P_0^{\theta}$.
      Define $o$ as the objective function in (8)
      Initialize optimizer state opt-s
      for $i \in 1, \ldots, N_{\text{iter}}$ do
          $\mathbf{g} \leftarrow \nabla o(\boldsymbol{x})$
          $\boldsymbol{x}$, opt-s $\leftarrow$ optupdate $(\boldsymbol{x}, \mathbf{g}, \text{opt-s})$
      end for
      Return $\boldsymbol{x}_1$
  end procedure
```

4.3. Comparison between RL and Trajectory Optimization Formulations

A qualitative comparison of the two approaches on various dimensions is presented in Table 1. The general purpose formulation (5) is a challenging high-dimensional stochastic control problem. We show that it can be handled by leveraging a state-of-the-art deep RL algorithm for continuous control and conducting extensive hyperparameter sweeps. However, the need for extensive hyperparameter sweeps highlights the advantages of the alternative formulation (6), which can be solved using any off-the-shelf gradient-based optimizer as long as the reward function is differentiable. It only requires tuning parameters of the optimizer and the β parameter trading off control cost and reward.

Furthermore, trajectory optimization is "free" in terms of train compute, as the parameters of the pretrained model are left unchanged; only the generation process is modified by the trajectory optimization. Finally, trajectory optimization leads to a globally optimal policy assuming gradient descent converges to the global optimum of (6). This is true, for example, if the score function $s(x, t)$ is linear in x, or, more realistically, if the globally optimal solution $\boldsymbol{x}^{\star}$ is in the vicinity of the initial uncontrolled trajectory $\boldsymbol{x}^0 \sim P_{\lambda}^{\theta}$, ensuring that a linear approximation of s around $\boldsymbol{x}$ is accurate.

On the downside, TrajOpt requires access to the reward model and its gradients, and is expensive at the evaluation (or sample generation) time, since the optimization problem (6) needs to be solved every time a sample is generated.

5. Related work

RL based fine tuning of diffusion models: Fan and Lee (2023) propose to fine-tune pretrained diffusion models with policy gradient and GAN training to reduce the number of steps required by DDPM (Ho et al., 2020) sampling. They train a critic network to distinguish real and fake distributions, use it to provide reward signals, and then update the diffusion model via policy gradient with the advantage function. They demonstrate that fine-tuned model can generate realistic samples with few steps with DDPM sampling. However, they mainly focus on improving the quality of generation and do not explore the best way to maximize the reward. In our work, we explore both trajectory optimization and RL methods to maximize a pre-defined reward function and demonstrate the advantages of fine-tuning methods for diverse purposes, such as artifact removal or addressing memorization issues.

Learning reward functions for evaluating text-to-image models: Even though several evaluation metrics, such as CLIP score (Radford et al., 2021) and FID score (Heusel et al., 2017), exist for assessing the quality of text-to-image models, they do not perfectly align with human judgment (Hu et al., 2023; Kirstain et al., 2023). To overcome this limitation, significant progress has been made in reward learning from human feedback recently (Lee et al., 2023; Kirstain et al., 2023; Xu et al., 2023; Wu et al., 2023). Such methods first collect human feedback that assesses model output over a set of diverse text prompts and then train a reward function to predict human feedback. They show that learned rewards are better-aligned with human evaluation than existing score functions. However, optimizing text-to-image models using this reward function remains a challenging problem. This work investigates a trajectory optimization and RL algorithm for the purpose of fine-tuning.

Stochastic control and diffusion models: Several papers have

Table 1. Comparison of algorithmic approaches for adaptation of diffusion models. RL refers to an actor critic method applied to (5) and TrajOpt to a gradient-based optimization method applied to (6).

	Train compute	Train stability	Eval compute	Reward	Optimality
RL	High	Low	Low	Black box	Local
TrajOpt	Zero	High	High	White box gradient	Global

studied control theoretic aspects of diffusion models Liu et al. (2023); Zhang and Chen (2021). However, their focus was on directly improving training or sampling from diffusion models using control theoretic ideas, while our focus here is on . We also believe that our results in theorem 3.1 are novel relative to the theoretical results in these papers.

6. Experiments

In this section, we compare the proposed trajectory optimization and RL-based approaches over several domains. Hyperparameter sweep, computation resources, and additional experiment details and results are presented in Appendix B.

6.1. Approaches Considered

We compare the following approaches:

- Uncontrolled: This denotes the uncontrolled diffusion model with parameters θ
- Rejection: This is a simple baseline where we generate samples from the uncontrolled model until we obtain $K = 512$ samples (per class for conditional sampling) that achieve a reward above a threshold.
- RL: This denotes the diffusion model adapted using the RL algorithm 1. The RL algorithm updates the parameters of the diffusion model to a new set of parameters ϕ.
- TrajOpt: This denotes the diffusion model adapted using the trajectory optimization algorithm 2.

Details of RL algorithm: As mentioned in Section 4.2, the policy and value network torsos have the same architecture as the uncontrolled diffusion model, and are warm-started from the uncontrolled model checkpoint. For the value network head, we use a randomly-initialized 3-layer MLP with hidden units (256, 128, 1), thus yielding scalar-value outputs. We use the Acme library (Hoffman et al., 2020) for all RL experiments. Reward and KL-divergence curves for all RL experiments are presented in Appendix C.2.

Details of TrajOpt algorithm: We initialize TrajOpt at a trajectory $\boldsymbol{x}$ generated by the uncontrolled diffusion model, and perform gradient-based optimization using the Adam optimizer with a piecewise constant schedule (details in Appendix C.2).

6.2. Metrics

Sample quality metrics: we consider the following evaluation metrics (Kynkäänniemi et al., 2019; Naeem et al., 2020) to compare the quality of generated samples:[1]

- Fidelity: we compute the *precision* and *density* metrics to check whether generated samples are realistic.
- Diversity: we compute the *recall* and *coverage* metrics to measure the ratio of real samples covered by generated samples.

For class-conditional generation, we compute the evaluation metrics per each class and then report the average value across all classes. We remark that these metrics can provide more detailed and reliable information than the Frechet Inception Distance (FID) score (Heusel et al., 2017). We defer the details of each metric to the original papers.

Reward metrics: We report the mean[2] and standard deviation or 95% confidence intervals for (for cases where the data is skewed) the reward obtained by each approach across all the 512 samples (per class for class conditional generation) generated.

6.3. Power Consumption Time Series

We first consider time series data. We utilize the POWER dataset from the the UCI ML data hub (Dua and Graff, 2017). The dataset represents the power consumption for a single household.

We craft a reward function which favors spikes in consumption. Spikes in power have the potential to severely disrupt energy networks; yet, such events are typically rare (Pinson et al., 2014; CAISO, 2016; Muratori et al., 2014; Silva et al., 2020). As such, it may be advisable for purposes such as stress-testing to generate more instances of these safety-critical events – while maintaining realism, with re-

[1]We used the publicly released implementation repository: https://github.com/clovaai/generative-evaluation-prdc/tree/master.

[2]We report median for the power consumption domain due to skew by outliers in TrajOpt.

spect to the distribution over observed events. Spikiness is computed as the average derivative over the samples.

Mathematically, our reward function can be expressed as: $\min\left(0.1, \sum_{i=1}^{23}(x_{i+1}-x_i)^2\right)$, where we saturate the reward at .1 to avoid creating overly spiky samples. We include further details about reward construction and trajectory optimization in the Supplement.

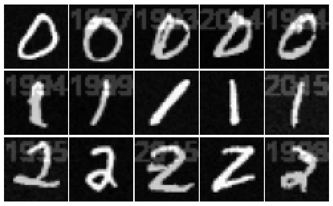
We find in Table 2 that while the RL fine-tuning method boasts samples with consistently high reward (as further demonstrated by the spikyness of generated samples, see Supplement). However, the generated samples achiever lower fidelity to the data distribution and diversity than TrajOpt. In this domain, TrajOpt struggles with optimization stability; we observe high variance in reward. Careful choice of the control cost can ameliorate high variance concerns, albeit at the cost of either high-fidelity or high reward (see Appendix), but underscores important pragmatic considerations when weighing method selection.

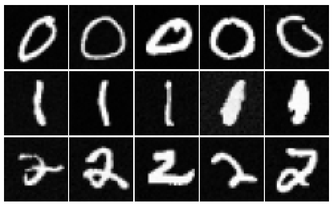
6.4. Removing artifacts and canaries in image generation

For both tasks in this section, we train a diffusion model with a UNet architecture with with time embedding dimension 256, 3 resnet blocks and attention resolutions 16, 8. The diffusion models are trained on MNIST datasets corrupted with artifacts or canaries as described in the following paragraphs for 100 training epochs.

Handling artifacts: timestamp removal Images sourced from the web may have artifacts; in particular, many classical cameras emblazon images with a timestamp. However, when generating new images, we may not want such artifacts to appear. To mimic the presence of timestamp artifacts, we construct a series of insignia of years (i.e., 32 years (ranging from 1992 to 2023). Timestamps are rendered in large font to ensure visibility. We blend these timestamps with images from the MNIST dataset via linear interpolation with a factor of .8 (i.e., 50% images used for training are .8 * MNIST image + .2 * Timestamp).

When training the diffusion model, 50% of each batch is corrupted with a timestamp artifact. All 32 different timestamps are included per batch, and wrapped to corrupted portion. We take our reward model to be a convolutional neural network (CNN; 2 convolutional layers and 2 full layers with an average pooling layer in between) trained to perform binary classification between timestamped and non-timestamped examples; reward is the classifier's predicted probability (as given by the sigmoid function applied to the logit output by the binary classifier) that an example is not time-stamped.


(a) Uncontrolled model


(b) RL-tuned model

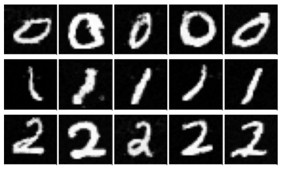

(c) Trajectory-Optimized

Figure 1. Generated samples for MNIST timestamp removal dataset.

Figure 1b shows the generated samples from the uncontrolled diffusion model, the RL fine-tuned model and the Trajectory Optimization approach. Both the RL and TrajOpt approaches successfully achieve removal of the timestamp while generating high quality samples (with the exception of one miss of timestamp removal via the RL approach). Overall, TrajOpt outperforms RL on both sample quality and reward. Unsurprisingly, the rejection sampling method achieves higher fidelity (since the accepted samples all come from the original diffusion model which has high fidelity to the training distribution), but lower diversity than our alternative approaches.

Handling memorized inputs: canary detection It has been shown that diffusion models can memorize images in the training data, particularly ones that have been duplicated sufficiently often (Carlini et al., 2023). Uncurated datasets scraped from the web often have duplicate or near-duplicate images, and deduplication is not always easy especially if the images are not exact but near duplicates. In this situation, it is problematic if the diffusion model generates memorized images.

We simulate such a situation by inserting a canary image (the second image in the top panel of figure (Appendix figure 9a) repeatedly into the training data of the MNIST dataset. Specifically, we replace 13% of each batch of images with the canary. Training the diffusion model thus results in a

Table 2. Inducing spikes in power consumption time series

Methods	Fidelity		Diversity		Reward (95% CI)
	Precision	Density	Recall	Coverage	
Uncontrolled	0.71	**0.93**	0.43	0.48	0.0021 (0.0021, 0.0022)
Rejection	**0.90**	0.87	0.65	0.49	0.0022 (0.0022, 0.0023)
TrajOpt	**0.85**	0.86	**0.68**	**0.62**	0.0019 (0.0014, 0.0048)
RL fine-tuned	0.55	0.84	0.23	0.2	**0.0027 (0.0027, 0.0029)**

Table 3. MNIST Artifact Removal

Methods	Fidelity		Diversity		Reward
	Precision	Density	Recall	Coverage	
Uncontrolled	0.56	0.34	0.77	0.40	.21 $\pm$.4
Rejection	**0.85**	**0.73**	0.87	0.67	0.92 $\pm$0.15
TrajOpt	0.84	0.69	**0.90**	**0.68**	**.99** $\pm 10^{-7}$
RL fine-tuned	0.83	0.67	0.87	0.63	.94 $\pm$.25

Table 4. MNIST Canary Removal

Methods	Fidelity		Diversity		Reward
	Precision	Density	Recall	Coverage	
Uncontrolled	0.37	0.34	0.62	0.44	.65 $\pm$ .17
Rejection	**0.92**	**0.93**	**0.89**	**0.78**	0.74 $\pm$ .01
TrajOpt	0.83	0.75	0.88	0.45	**0.76** $\pm$.05
RL fine-tuned	0.32	0.16	0.19	0.10	.75 $\pm$ .008

model that reproduces the canary frequently (3/32 or 10% of the time in Appendix Figure 9a). We train a linear classifier that achieves 100% accuracy on validation data to discriminate between all the other images of an 8 in the MNIST dataset vs the canary image and use this as a reward function.

The results in Table 4 and the output samples in Appendix Figure 9 demonstrate that we can effectively prevent the model from generating the canary, while still preserving the quality of outputs. Overall, on this task though, we find that the trajectory rejection sampling is a very strong method, as the number of samples that need to be rejected (that match the canary) are relatively small, and despite rejection, the method can preserve strong fidelity and diversity.

7. Limitations & Broader Impacts

Our method relies on access to a high-quality reward function. However, in practice, reward functions may be imperfect and fail to generalize under distribution shift, e.g., if trained on only sparse human feedback (Stiennon et al., 2020). Even here, we needed to apply heuristic clipping onto our reward functions to support generalization. More fundamentally, achieving the right balance between fidelity to the pretrained diffusion model and optimizing the reward is challenging. Finding principled ways to resolve these issues are key directions for future work. Yet, it is worth possible ramifications of facilitating ease of adaption of diffusion models via reward functions; e.g., it is conceivable that our methods could be deployed to hide that a model was trained on copyrighted images, or that bad actors could design a reward function to nudge a model to produce more explicit or violent content.

8. Conclusion

As diffusion models are increasingly deployed in real-world applications, it is critical to tune them away from undesirable behavior such as regurgitating memorized images or producing inappropriate content. Reward-based adaptation is a flexible framework for achieving better control. We have developed a theoretical framework for optimally trading off fidelity to an initial generative diffusion model with maximizing a user specified reward function. The algorithms we develop are effective in practice across a range of domains. Scaling our algorithms to state-of-the-art text-to-image models, and investigating the interplay of reward model uncertainty/robustness and adaptation are interesting directions for future work.

References

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

CAISO. What the duck curve tells us about managing a green grid. https://www.caiso.com/documents/flexibleresourceshelprenewables_fastfacts.pdf, 2016. Accessed: 2023-05-15.

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Ying Fan and Kangwook Lee. Optimizing ddpm sampling with shortcut fine-tuning. *arXiv preprint arXiv:2301.13362*, 2023.

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

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.

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. *arXiv preprint arXiv:2303.11897*, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *arXiv preprint arXiv:2305.01569*, 2023.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Guan-Horng Liu, Arash Vahdat, De-An Huang, Evangelos A Theodorou, Weili Nie, and Anima Anandkumar. I^{2}sb: Image-to-image schrödinger bridge. *arXiv*, 2023.

Ravi Montenegro and Prasad Tetali. Mathematical aspects of mixing times in markov chains. *Foundations and Trends® in Theoretical Computer Science*, 1(3):237–354, 2006. ISSN 1551-305X. doi: 10.1561/0400000003. URL http://dx.doi.org/10.1561/0400000003.

Matteo Muratori, Beth-Anne Schuelke-Leech, and Giorgio Rizzoni. Role of residential demand response in modern electricity markets. *Renewable and Sustainable Energy Reviews*, 33:546–553, 2014.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

Pierre Pinson, Henrik Madsen, et al. Benefits and challenges of electrical demand response: A critical review. *Renewable and Sustainable Energy Reviews*, 39:686–699, 2014.

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*, 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*, pages 8821–8831. PMLR, 2021.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

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

Bhagya Nathali Silva, Murad Khan, and Kijun Han. Futuristic sustainable energy management in smart environments: A review of peak load shaving and demand response strategies, challenges, and opportunities. *Sustainability*, 12(14):5561, 2020.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

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*, 2020b.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pages 3084–3114. PMLR, 2019.

Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Better aligning text-to-image models with human preference. *arXiv preprint arXiv:2303.14420*, 2023.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *arXiv preprint arXiv:2304.05977*, 2023.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

Qinsheng Zhang, Molei Tao, and Yongxin Chen. gddim: Generalized denoising diffusion implicit models. *arXiv preprint arXiv:2206.05564*, 2022.

Appendix: Algorithms for Optimal Adaptation of Diffusion Models to Reward Functions

A. Outline

In section B, we provide details on hyperparameter sweeps, archictecture details and compute resources used for running our experiments. In section C, we provide results on another domain (generating Markov chains with specific spectral properties), provide additional details on domains used for experiments and sample images for the MNIST domains. In section D, we provide additional details on the trajectory optimization approach and in section E, we provide a proof of theorem 3.1 and the derivations related to the PPO objective.

B. Experiment Details and Hyperparameters

This section provides an overview of the details for our experiments.

Hyperparameter sweeps Details of hyperparameter sweeps are summarized in Tables 5 and 6.

Resources RL experiments were conducted on a cluster of TPUs.

C. Additional Results

C.1. Generating Markov Chains with Mixing Properties

We include further experiments investigating adaptation into an additional domain; i.e., the problem of generating discrete Markov chains over graph structures with desirable properties related to their equilibrium distributions and mixing times. Specifically, we wish to sample Markov chains from a fixed distribution of transition probabilities over a fixed graph, such that either: 1. The resulting equilibrium distributions (of each chain) are close to uniform, or 2. The spectral gap (difference between the largest and second-largest eigenvalue of the transition matrix) is maximized. Controlling the spectral gap in particular, is a very well-studied problem with numerous applications e.g. construction of rapidly mixing processes, ensuring convergence of random walks etc. Generally however, results are obtained by bounding the spectral gap for specific families of graphs with regular properties. Here, we investigate how optimal adaptation algorithms may be applied to generate graphs satisfying these properties from general distributions.

In both cases, we consider Markov chains over a fixed graph with $N = 10$ nodes and $E = 57$ edges. The distribution over transition probabilities between nodes is sampled from a non-uniform distribution determined by a small set of affinity parameters between nodes. This distribution is learned by the pre-trained diffusion model (an MLP) before the adaptation algorithms are applied. During adaptation, we apply a cost that is proportional to the variance of the equilibrium distribution in the first case and to the spectral gap of the *reversibilization* (See Montenegro and Tetali (2006)) of the chain in the second.

The results (shown in tables 7 and 8) indicate some degree of difference in the performance of each. Rejection sampling is quite competitive in both settings, achieving low cost, though with a significant loss of fidelity in the equilibrium case. TrajOpt achieves the best cost in the spectral gap setting but with a large penalty for fidelity. For the uniform case, it achieves a moderate reduction in cost with high diversity and fidelity to the pre-trained distribution. RL training was observed to be extremely unstable despite extensive hyper-parameter search and barely moved the cost in both settings, showing that this optimization problem is non-trivial. The single-best trajectory observed over a large sample had cost 2.42×10^{-4} in the equilibrium case and 0.645 in the spectral gap case. This may explain some of the differences in the results, since all methods find trajectories with average cost far from optimal for the uniform case, but close to optimal for the spectral gap, perhaps indicating that "good" trajectories are much sparser for the uniform cost setting than for the spectral gap.

C.2. RL training curves

Here, we present reward curves for RL results. For each series, errorbars correspond to standard deviations over all other hyperparameters included in the sweep in Table 6 (e.g., seeds, learning rates, etc.). Figures 2 to 7 illustrate reward and KL divergence curves throughout training for various hyperparameter values.

C.3. Additional Details on Power Consumption Experiments

We include further details on the data and model setup for the power consumption domain.

Power consumption is measured every minute. We focus on a single measure per timestep; i.e., global average power consumed (in kilowatts). We follow (Papamakarios et al., 2017) in in-filling missing data with the entry from the previous time's measurement. The samples $x \in \mathbb{R}^{24}$ represent the time-series of power consumption averaged over each hour of the day. While training and sampling from the diffusion model, x is normalized such that each dimension is between 0 and 1. We employ an MLP-based architecture for the score function, with time embedding of dimension 128 and 2 hidden layers of 100 neurons each. We train for

Table 5. Hyperparameters used for trajectory optimization experiments. Values in braces indicate sweeps. The learning rate drops by a factor of 10 at the iteration number indicated in the row (Drops in learning rate)

Parameter	Power Time Series	Markov Chain	MNIST
Initial learning rate	1e-3	1e-3	1e-3
Adam #steps	10000	10000	40000
Drops in learning rate	-	-	20000
Reward scaling	1	1	10, 20, . . . 90

Table 6. Hyperparameters used for RL experiments. Values in braces indicate sweeps.

Parameter	Power Time Series	Markov Chain	MNIST
Training steps	1e7	5e6	2e6
Adam learning rate	{3e-6, 3e-5, 3e-4}	{3e-7, 3e-6}	{3e-7, 3e-6}
Adam epsilon	1e-7	1e-7	1e-7
KL penalty β	{1e-2, 1}	{5e-4, 5e-3, 7e-3, 1e-2}	{3e-4, 5e-4, 1e-2, 1}
Diffusion λ	{0.1, 1.0}	{0.1, 1.0}	{0.1, 1.0}
Diffusion # timesteps	40	100	40
Batch size	128	128	128
PPO unroll length	2	2	2
PPO # minibatches	8	8	8
PPO # epochs	2	2	2
PPO clipping ϵ	0.1	0.1	0.1
# seeds	2	2	2

150,000 iterations.

C.4. Additional Details on MNIST experiments

We include additional information on the MNIST artificat and canary removal domains. Additional class-conditional generations for artifact removal are depicted in Figure 8 and samples for canary removal are in 9.

D. Details of TrajOpt

We use the Adam optimizer (Kingma and Ba, 2014) with details on learning rate schedules as outlined in tabel 5. We initalize TrajOpt with $\boldsymbol{x}$ generated by the uncontrolled model and then optimize towards the controlled model. Interestingly, the fidelity penalty term rises initially in order to produces samples with high reward, but eventually falls so that the trajectories remain close to those produced by the original denoising steps of the diffusion model. Investigating this behavior more closely might provide avenues for developing more efficient trajectory optimization algorithms in the future.

E. Proofs and derivations

E.1. Proof of theorem 3.1

Proof. By Girsanov's theorem (as used in the proof of theorem 2.1 of (Tzen and Raginsky, 2019)), we find that the Radon-Nikodym derivative of P_λ^u with respect to $P_\lambda^{s_\theta}$ is given by

$$\frac{dP_\lambda^u}{dP_\lambda^{s_\theta}} = \int_0^1 \frac{1}{2}\frac{\left(\left(\frac{1+\lambda^2}{2}\right)g\left(t\right)^2\right)^2\left\|u\left(x,t\right)-s_\theta\left(x,t\right)\right\|^2}{\lambda^2 g\left(t\right)^2}dt$$

Thus, the KL divergence between them is

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_\lambda^u}\left[\int_0^1 \frac{\left(1+\lambda^2\right)^2}{8\lambda^2}g\left(t\right)^2\left\|u\left(\boldsymbol{x}\left(t\right),t\right)-s_\theta\left(\boldsymbol{x}\left(t\right),t\right)\right\|^2 dt\right]$$

□

Thus, the objective evaluates to

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_\lambda^u}\left[\frac{-\left(1+\lambda^2\right)^2 g\left(t\right)^2}{8}\mathrm{CC}+r\left(\boldsymbol{x}\left(1\right)\right)\right]$$

which coincides with (5).

Furthermore, if we take the limit as $\lambda \to 0$, samples from

Table 7. Controlling for Uniform Equilibrium Distributions of Markov Chains.

Methods	Fidelity		Diversity		Cost $\times 10^3$
	Precision	Density	Recall	Coverage	(Var. of eq. dist.)
Uncontrolled	0.87	0.98	0.98	0.99	1.683 $\pm$0.08
Rejection Sampling	0.67	0.56	**0.90**	0.88	**0.82** $\pm$0.006
TrajOpt	**0.87**	**0.98**	0.84	**0.95**	1.524 $\pm$0.07
RL fine-tuned	0.77	0.74	0.85	0.93	1.642 $\pm$0.03

Table 8. Controlling Spectral Gaps of Markov Chains.

Methods	Fidelity		Diversity		Cost
	Precision	Density	Recall	Coverage	$(1-\lambda_2)$
Uncontrolled	0.87	0.98	0.98	0.99	0.665 $\pm 7 \times 10^{-4}$
Rejection Sampling	**0.87**	**0.85**	0.83	**0.92**	0.658 $\pm 4 \times 10^{-5}$
TrajOpt	0.27	0.08	**0.95**	0.24	**0.653** $\pm 6 \times 10^{-4}$
RL fine-tuned	0.20	0.047	0.002	0.014	0.664 $\pm 1 \times 10^{-3}$

P_0^u can be generated as (taking the limit of (4) as $\lambda \to 0$)

$$\dot{\boldsymbol{x}}(t) = -\left(f(t)\,\boldsymbol{x}(t) - \frac{1}{2}g(t)^2 u(\boldsymbol{x}(t), t)\right)$$
$$t \in [0,1], \boldsymbol{x}(0) \sim \mathcal{N}(0, I)$$

and the objective reduces to

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_0^u}\left[\frac{-g(t)^2}{8}\text{CC} + r(\boldsymbol{x}(1))\right]$$

Note that the stochasticity here is only in $\boldsymbol{x}(0)$ since the rest of the trajectory can be generated by following the ODE above starting at $\boldsymbol{x}(0)$. Hence, the expectation above reduces to

$$\mathop{\mathbb{E}}_{\boldsymbol{x}(0)\sim\mathcal{N}(0,I)}\left[\frac{-g(t)^2}{8}\text{CC} + r(\boldsymbol{x}(1))\right]$$

Since we allow for arbitrary measurable functions $u : \mathbb{R}^n \times [0,1] \mapsto \mathbb{R}^n$, once $\boldsymbol{x}0$, we can reprhase the optimal control problem (5) as

$$\min_u \mathop{\mathbb{E}}_{\boldsymbol{x}(0)\sim\mathcal{N}(0,I)}\left[\frac{-g(t)^2}{8}\text{CC} + r(\boldsymbol{x}(1))\right]$$

where

$$\dot{\boldsymbol{x}}(t) = -\left(f(t)\,\boldsymbol{x}(t) - \frac{1}{2}g(t)^2 u(\boldsymbol{x}(t), t)\right)$$

Substituting this value into the above equation, we obtain

$$\min_u \mathop{\mathbb{E}}_{\boldsymbol{x}(0)\sim\mathcal{N}(0,I)}\left[\frac{-1}{2g(t)^2}\text{OO} + r(\boldsymbol{x}(1))\right]$$

where

$$\int_{t=0}^{1}\left\|\dot{\boldsymbol{x}}(t) + f(t)\,\boldsymbol{x}(t) - \frac{1}{2}g(t)^2 s_\theta(\boldsymbol{x}(t), t)\right\|^2 dt \tag{12}$$

$$u(\boldsymbol{x}(t), t) = \frac{\dot{\boldsymbol{x}}(t) + f(t)\,\boldsymbol{x}(t)}{\frac{1}{2}g(t)^2} \tag{13}$$

Thus, there is a one to one mapping between trajectories $\boldsymbol{x}(t)$ and policies u. Further, the optimal sequence of values $u(\boldsymbol{x}(t), t)$ is determined given the value of $\boldsymbol{x}(0)$.

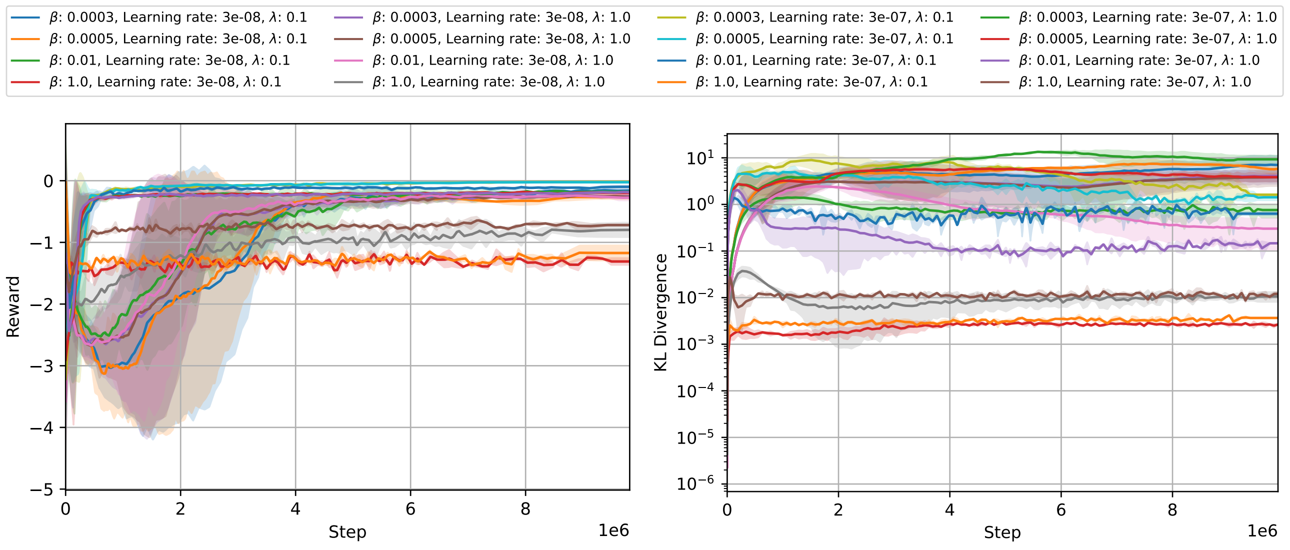
Furthermore, the optimal $u(\boldsymbol{x}(t), t)$ can be determined by solving

$$\min_{\boldsymbol{x}:\boldsymbol{x}(0)=z} \frac{-1}{2g(t)^2}\text{OO} + r(\boldsymbol{x}(1))$$

where $z \sim \mathcal{N}(0, I)$ and setting $u^\star(\boldsymbol{x}(t), t)$ based on (13). Hence the theorem.

E.2. Detailed derivation of PPO KL term

We note that 10a in the original submission has a typo, and that $\hat{\mathbb{KL}}_{\phi^{\text{old}}}\left(P_\lambda^\phi \parallel P_\lambda^\theta\right)$ should be directly defined via



(a) Rewards (b) KL Divergence

Figure 2. Training curves for Toy Dataset.

equation 10b.

$$\mathbb{KL}\left(P_\lambda^\phi \parallel P_\lambda^\theta\right) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_\lambda^\phi}\left[\log\left(\frac{P_\lambda^\phi(\boldsymbol{x})}{P_\lambda^\theta(\boldsymbol{x})}\right)\right] \tag{14a}$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_\lambda^\phi}\left[\log\left(\frac{P_\lambda^\phi(\boldsymbol{x}(0))}{P_\lambda^\theta(\boldsymbol{x}(0))}\right) + \sum_{t\in\mathcal{T}\setminus\{1\}}\log\left(\frac{\pi_\phi(\boldsymbol{x}(t+h)\,|\boldsymbol{x}(t),t)}{\pi_\theta(\boldsymbol{x}(t+h)\,|\boldsymbol{x}(t),t)}\right)\right] \tag{14b}$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_\lambda^\phi}\left[\sum_{t\in\mathcal{T}\setminus\{1\}}\log\left(\frac{\pi_\phi(\boldsymbol{x}(t+h)\,|\boldsymbol{x}(t),t)}{\pi_\theta(\boldsymbol{x}(t+h)\,|\boldsymbol{x}(t),t)}\right)\right] \tag{14c}$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_\lambda^\phi}\left[\sum_{t\in\mathcal{T}\setminus\{1\}}\mathbb{KL}\left(\pi_\phi(\cdot|\boldsymbol{x}(t),t) \parallel \pi_\theta(\cdot|\boldsymbol{x}(t),t)\right)\right] \tag{14d}$$

$$= \frac{1}{h}\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_\lambda^\phi,t\sim\mathcal{U}(\mathcal{T}\setminus\{1\})}\left[\mathbb{KL}\left(\pi_\phi(\cdot|\boldsymbol{x}(t),t) \parallel \pi_\theta(\cdot|\boldsymbol{x}(t),t)\right)\right] \tag{14e}$$

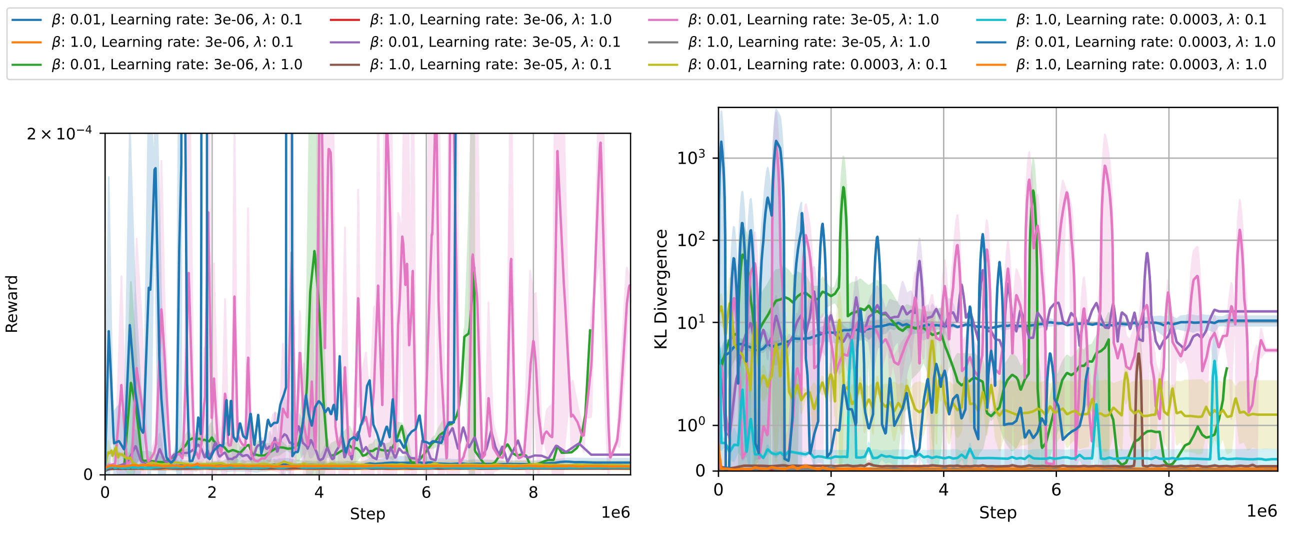
where (14d) follows by observing that

$$\mathbb{E}\left[\log\left(\frac{\pi_\phi(\boldsymbol{x}(t+h)\,|\boldsymbol{x}(t),t)}{\pi_\theta(\boldsymbol{x}(t+h)\,|\boldsymbol{x}(t),t)}\right)\right] = \mathbb{E}\left[\mathop{\mathbb{E}}_{\boldsymbol{x}(t+1)\sim\pi_\phi(\cdot|\boldsymbol{x}(t),t)}\left[\log\left(\frac{\pi_\phi(\boldsymbol{x}(t+h)\,|\boldsymbol{x}(t),t)}{\pi_\theta(\boldsymbol{x}(t+h)\,|\boldsymbol{x}(t),t)}\right)\middle|\,\boldsymbol{x}(t)\right]\right]$$

$$= \mathbb{E}\left[\mathbb{KL}\left(\pi_\phi(\cdot|\boldsymbol{x}(t),t) \parallel \pi_\theta(\cdot|\boldsymbol{x}(t),t)\right)\right]$$

and (14e) follows by observing that there is an expectation of average over $T = \frac{1}{h}$ terms which can be replaced by an expaction where t is sampled uniformly.

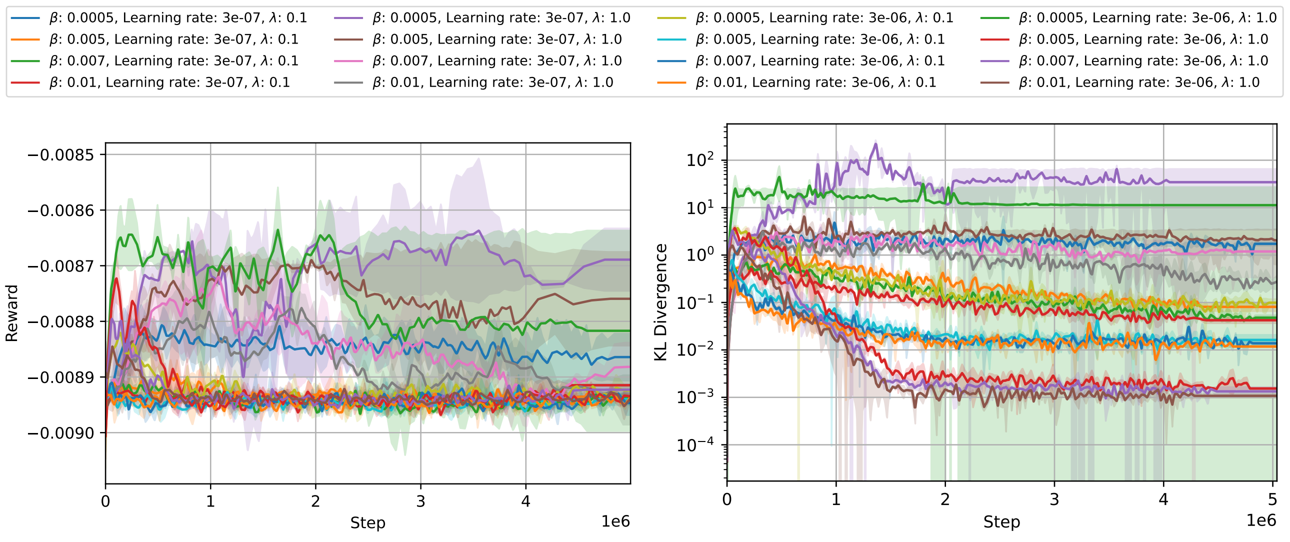
Replacing the expectation with an expectation over $\boldsymbol{x} \sim P_\lambda^{\phi^{\text{old}}}$ yields $\hat{\mathbb{KL}}_{\phi^{\text{old}}}\left(P_\lambda^\phi \parallel P_\lambda^\theta\right)$.



(a) Rewards

(b) KL Divergence

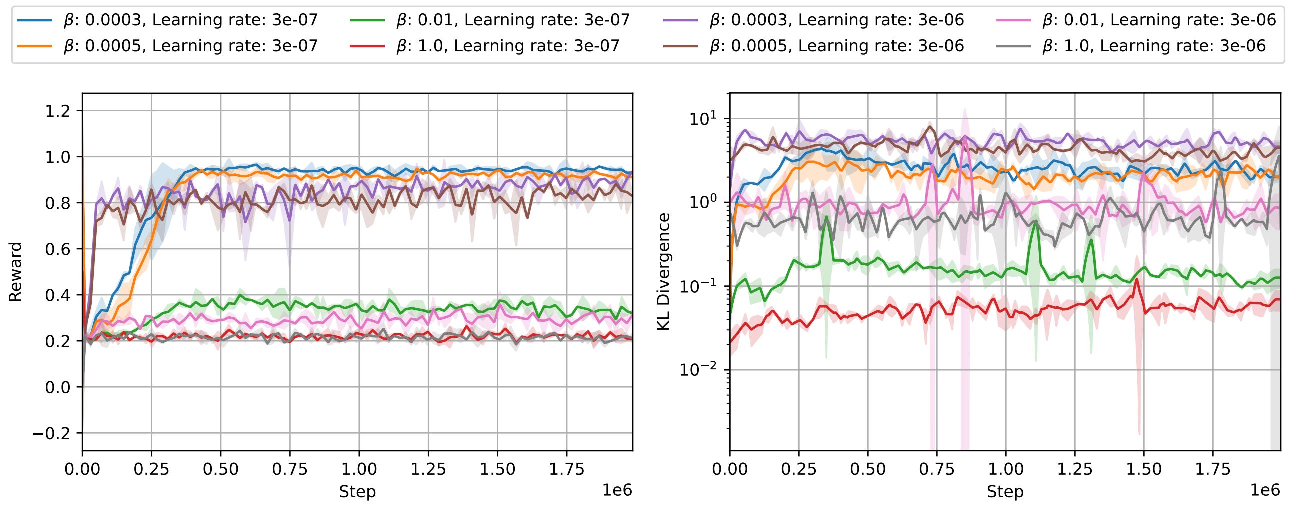
Figure 3. Training curves for Power Consumption Time Series dataset.



(a) Rewards

(b) KL Divergence

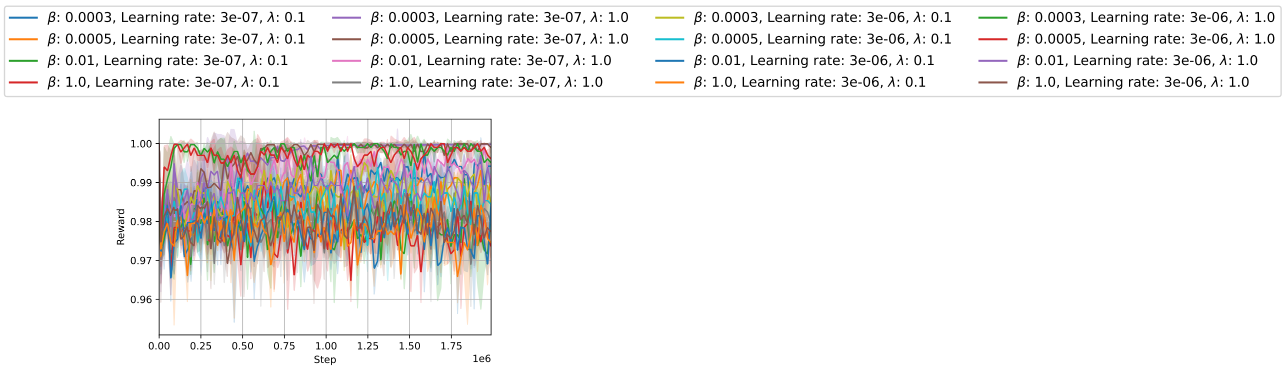
Figure 4. Training curves for Graph dataset.



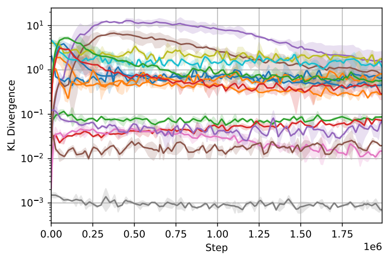
(a) Rewards

(b) KL Divergence

Figure 5. Training curves for MNIST Timestamp dataset.

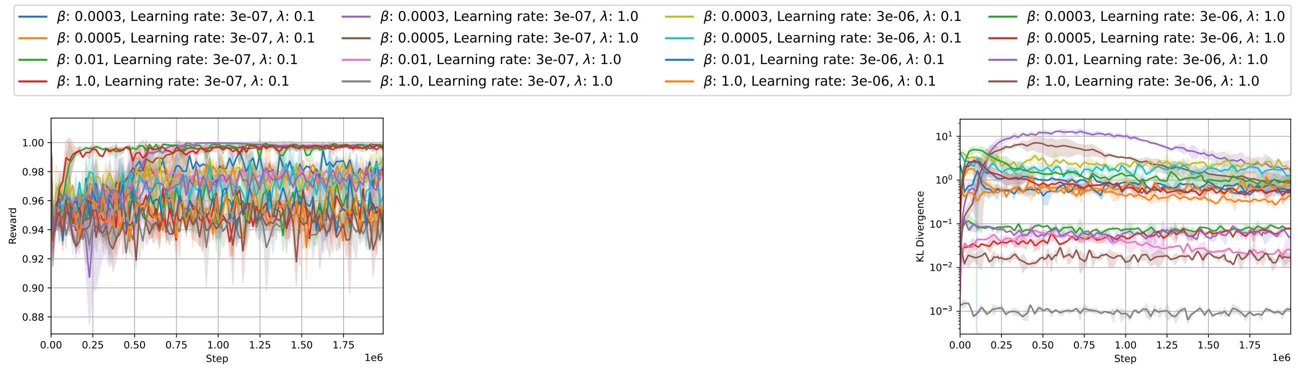


(a) Rewards



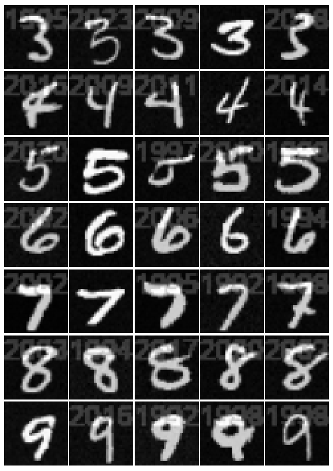
(b) KL Divergence

Figure 6. Training curves for MNIST Canary dataset.

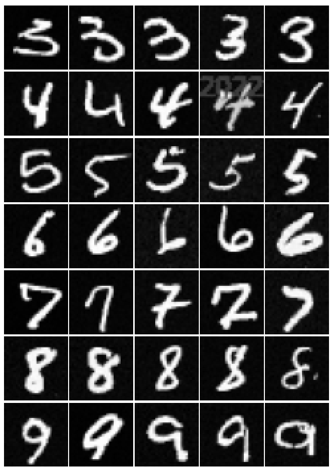


(a) Rewards (b) KL Divergence

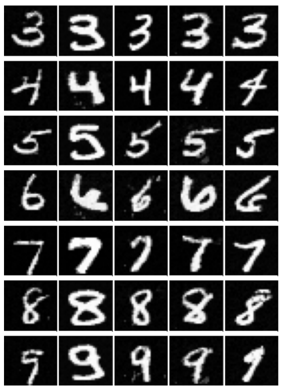
Figure 7. Training curves for MNIST Canary (linear reward) dataset.



(a) Uncontrolled model

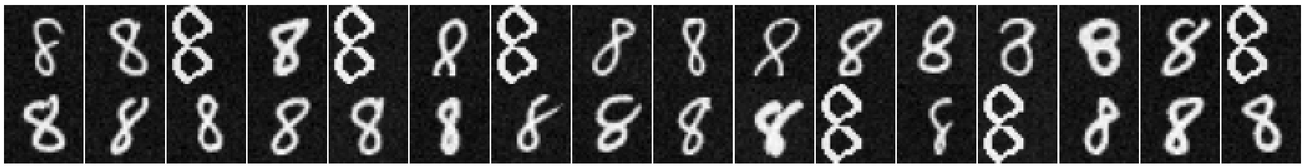


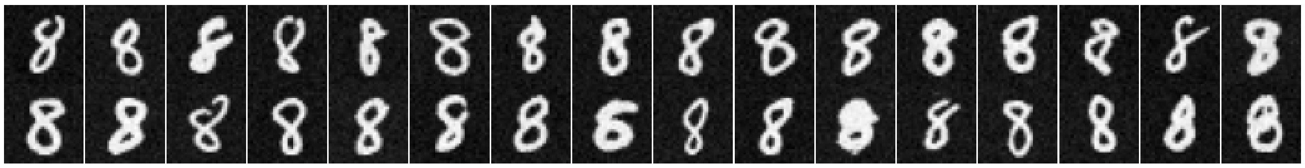
(b) RL-tuned model



(c) Trajectory-Optimized

Figure 8. Generated samples for MNIST timestamp removal dataset.


(a) Uncontrolled model.


(b) RL-tuned model. Canary images for digit class 8 are rarely generated ;

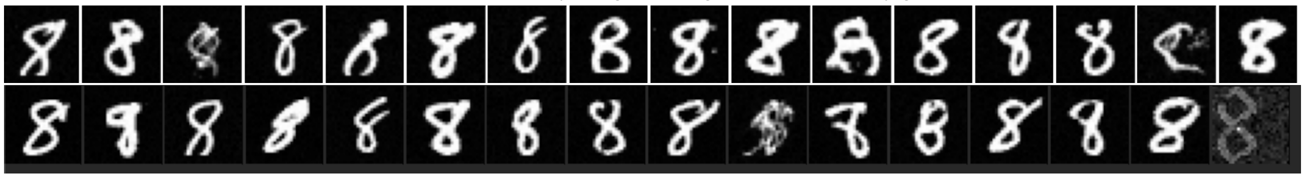

(c) TrajOpt. Canary images for digit class 8 are rarely generated ;

Figure 9. Generated samples for MNIST canary dataset.