Diffusion Reading Group #24

5/27/2023

Tanishq Mathew Abraham

Training Diffusion Models with Reinforcement Learning

Preliminaries

3.1 Diffusion Models

In this work, we consider conditional diffusion probabilistic models [47, 19], which represent a distribution over data $\mathbf{x}_0$ conditioned on context $\mathbf{c}$ as the result of sequential denoising. The denoising procedure is trained to reverse a Markovian forward process $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, which iteratively adds noise to the data. Reversing the forward process can be accomplished by training a forward process posterior mean predictor $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{c})$ for all $t \in \{0, 1, \ldots, T\}$ with the following simplified objective:

$$\mathcal{L}_{\text{DDPM}}(\theta) = \mathbb{E}\left[\|\tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{c})\|^2\right] \tag{1}$$

where $\tilde{\boldsymbol{\mu}}$ is a weighted average of $\mathbf{x}_0$ and $\mathbf{x}_t$. This objective is justified as maximizing a variational lower bound on the model log-likelihood [19].

Sampling from a diffusion model begins with sampling $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and using the reverse process $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$ to produce a trajectory $\{\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_0\}$ ending with a sample $\mathbf{x}_0$. The reverse process depends not only on the predictor $\boldsymbol{\mu}_\theta$ but also the choice of sampler. Most popular samplers [19, 48] use an isotropic Gaussian reverse process with a fixed timestep-dependent variance:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t, \mathbf{c}\right), \sigma_t^2 \mathbf{I}). \tag{2}$$

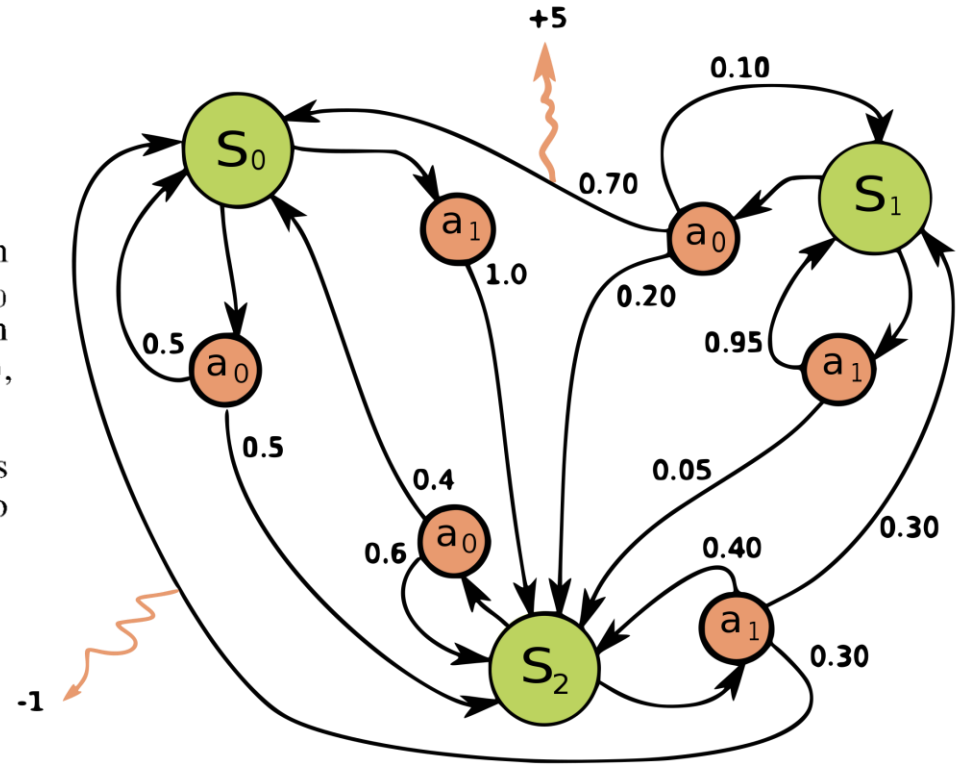
Markov Decision Processes – Markov Chains for RL

3.2 Markov Decision Processes and Reinforcement Learning

A Markov decision process (MDP) is a formalization of sequential decision-making problems. An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \rho_0, P, R)$, in which $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, ρ_0 is the distribution of initial states, P is the transition kernel, and R is the reward function. At each timestep t, the agent observes a state $\mathbf{s}_t \in \mathcal{S}$, takes an action $\mathbf{a}_t \in \mathcal{A}$, receives a reward $R(\mathbf{s}_t, \mathbf{a}_t)$, and transitions to a new state $\mathbf{s}_{t+1} \sim P(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$. An agent acts according to a policy $\pi(\mathbf{a} \mid \mathbf{s})$.

As the agent acts in the MDP, it produces trajectories, which are sequences of states and actions $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)$. The reinforcement learning (RL) objective for the agent is to maximize $\mathcal{J}_{\mathrm{RL}}(\pi)$, the expected cumulative reward over trajectories sampled from its policy:

$$\mathcal{J}_{\mathrm{RL}}(\pi) = \mathbb{E}_{\tau \sim p(\cdot|\pi)} \left[\sum_{t=0}^{T} R(\mathbf{s}_t, \mathbf{a}_t) \right].$$



What we are optimizing

We assume a pre-existing diffusion model, which may be pretrained or randomly initialized. If we choose a fixed sampler, the diffusion model induces a sample distribution $p_\theta(\mathbf{x}_0 \mid \mathbf{c})$. The denoising diffusion RL objective is to maximize a reward signal r defined on the samples and contexts:

$$\mathcal{J}_{\mathrm{DDRL}}(\theta) = \mathbb{E}_{\mathbf{c}\sim p(\mathbf{c}),\, \mathbf{x}_0 \sim p_\theta(\cdot|\mathbf{c})}\left[r(\mathbf{x}_0, \mathbf{c})\right]$$

for some context distribution $p(\mathbf{c})$ of our choosing.

Reward-weighted regression (RWR) – a simple RL baseline

- Train with standard DDPM objective, but weighted by the reward per sample

A standard weighting scheme uses exponentiated rewards to ensure nonnegativity,

$$w_{\text{RWR}}(\mathbf{x}_0, \mathbf{c}) = \frac{1}{Z} \exp\left(\beta R(\mathbf{x}_0, \mathbf{c})\right),$$

where β is an inverse temperature and Z is a normalization constant. We also consider a simplified weighting scheme that uses binary weights,

$$w_{\text{sparse}}(\mathbf{x}_0, \mathbf{c}) = \mathbb{1}\left[R(\mathbf{x}_0, \mathbf{c}) \geq C\right],$$

where C is a reward threshold determining which samples are used for training. The sparse weights may be desirable because they eliminate the need to retain every sample from the model.

RWR as a one-step MDP

Within the RL formalism, the RWR procedure corresponds to the following one-step MDP:

$$\mathbf{s} \triangleq \mathbf{c} \qquad \mathbf{a} \triangleq \mathbf{x}_0 \qquad \pi(\mathbf{a} \mid \mathbf{s}) \triangleq p_\theta(\mathbf{x}_0 \mid \mathbf{c}) \qquad \rho_0(\mathbf{s}) \triangleq p(\mathbf{c}) \qquad R(\mathbf{s}, \mathbf{a}) \triangleq r(\mathbf{x}_0, \mathbf{c})$$

with a transition kernel P that immediately leads to an absorbing termination state. Therefore, maximizing $\mathcal{J}_{\mathrm{DDRL}}(\theta)$ is equivalent to maximizing $\mathcal{J}_{\mathrm{RL}}(\pi)$ in this MDP.

Weighting a maximum likelihood objective by w_{RWR} approximately optimizes $\mathcal{J}_{\mathrm{RL}}(\pi)$ subject to a KL divergence constraint on the policy [30]. However, $\mathcal{L}_{\mathrm{DDPM}}$ is not an exact maximum likelihood objective, but is derived from a reweighted variational bound. Therefore, RWR algorithms applied to $\mathcal{L}_{\mathrm{DDPM}}$ optimize $\mathcal{J}_{\mathrm{DDRL}}$ via two levels of approximation. Thus, this methodology provides us with a starting point, but might underperform for complex objectives.

Denoising diffusion as a multi-step MDP

Denoising as a multi-step MDP. We map the iterative denoising procedure to the following MDP:

$$\mathbf{s}_t \triangleq (\mathbf{c}, t, \mathbf{x}_t) \quad \pi(\mathbf{a}_t \mid \mathbf{s}_t) \triangleq p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) \qquad P(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t) \triangleq \left(\delta_{\mathbf{c}}, \delta_{t-1}, \delta_{\mathbf{x}_{t-1}}\right)$$

$$\mathbf{a}_t \triangleq \mathbf{x}_{t-1} \qquad \rho_0(\mathbf{s}_0) \triangleq \left(p(\mathbf{c}), \delta_T, \mathcal{N}(\mathbf{0}, \mathbf{I})\right) \qquad R(\mathbf{s}_t, \mathbf{a}_t) \triangleq \begin{cases} r(\mathbf{x}_0, \mathbf{c}) & \text{if } t = 0 \\ 0 & \text{otherwise} \end{cases}$$

in which δ_y is the Dirac delta distribution with nonzero density only at y. Trajectories consist of T timesteps, after which P leads to a termination state. The cumulative reward of each trajectory is equal to $r(\mathbf{x}_0, \mathbf{c})$, so maximizing $\mathcal{J}_{\text{DDRL}}(\theta)$ is equivalent to maximizing $\mathcal{J}_{\text{RL}}(\pi)$ in this MDP.

The benefit of this formulation is that, if we use a standard sampler parameterized as in Equation 2, the policy π becomes an isotropic Gaussian as opposed to an arbitrarily complicated distribution

How to optimize

- We want to optimize the policy, which is our diffusion model $\pi(\mathbf{a}_t|\mathbf{s}_t) = p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to maximize the $\mathcal{J}_{\mathrm{RL}}(\pi)$

- Need to determine $\nabla_\theta \mathcal{J}_{\mathrm{DDRL}}$ but this would require us to backpropagate through the entire trajectory!

- A simple way → REINFORCE

REINFORCE algorithm - DDPO_{SF}

Example derivation of REINFORCE algorithm: https://mcneela.github.io/math/2018/04/18/A-Tutorial-on-the-REINFORCE-Algorithm.html

$$\hat{g}_{\text{SF}} = \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{c}, t, \mathbf{x}_t)\, r(\mathbf{x}_0, \mathbf{c})\right]$$

$$\theta \leftarrow \theta + \alpha \hat{g}_{SF}$$

one step of optimization per round of data collection

Importance sampling estimator - DDPO_{IS}

$$\hat{g}_{\text{IS}} = \mathbb{E}\left[\sum_{t=0}^{T} \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{c}, t, \mathbf{x}_t)}{p_{\theta_{\text{old}}}(\mathbf{x}_{t-1} \mid \mathbf{c}, t, \mathbf{x}_t)} \nabla_\theta \log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{c}, t, \mathbf{x}_t)\, r(\mathbf{x}_0, \mathbf{c})\right]$$

$$\theta \leftarrow \theta + \alpha \hat{g}_{SF}$$

Allows for multiple optimization steps for a single round of data collection

Becomes inaccurate if p_θ deviates too far from $p_{\theta\text{old}}$

The ratio measures divergence between old and new policies → clip it! (trust regions)

Implementation details

- Optimizing Stable Diffusion v1.4 U-net
- 256 samples per training iteration
- $DDPO_{SF}$ - accumulate gradients across all 256 samples and perform one gradient update
- $DDPO_{IS}$ - split the samples into 4 minibatches and perform 4 gradient updates
- Gradients are always accumulated across all denoising timesteps
- A very small clip range is needed – 1e-4
- Ancestral (Euler) sampler, 50 timesteps, guidance scale of 5

Reward functions examined

- Compressibility and incompressibility

The capabilities of text-to-image diffusion models are limited by the co-occurrences of text and images in their training distribution. For instance, images are rarely captioned with their file size, making it impossible to specify a desired file size via prompting. This limitation makes reward functions based on file size a convenient case study: they are simple to compute, but not controllable through the conventional workflow of likelihood maximization and prompt engineering.

We fix the resolution of diffusion model samples at 512x512, such that the file size is determined solely by the compressibility of the image. We define two tasks based on file size: compressibility, in which the file size of the image after JPEG compression is minimized, and incompressibility, in which the same measure is maximized.

Reward functions examined - RLHF

- Aesthetic quality

To capture a reward function that would be useful to a human user, we define a task based on perceived aesthetic quality. We use the LAION aesthetics predictor [43], which is trained on 176,000 human image ratings. The predictor is implemented as a linear model on top of CLIP embeddings [37]. Annotations range between 1 and 10, with the highest-rated images mostly containing artwork. Since the aesthetic quality predictor is trained on human judgments, this task constitutes reinforcement learning from human feedback [34, 7, 61].

Reward functions examined - RLAIF

- Prompt alignment

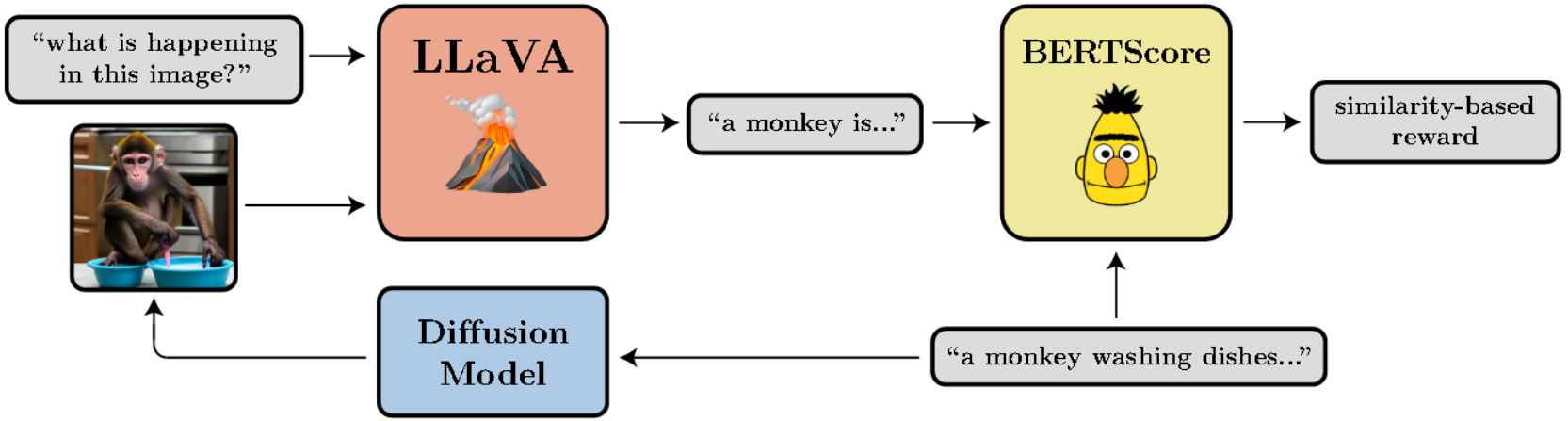


Figure 2 (VLM reward function) Illustration of the VLM-based reward function for prompt-image alignment. LLaVA [26] provides a short description of a generated image; the reward is the similarity between this description and the original prompt as measured by BERTScore [59].

Experimental results - samples

- Compressibility and incompressibility prompts are sampled uniformly from all 398 animals in the ImageNet categories. Aesthetic quality prompts are sampled uniformly from a smaller set of 45 common animals.

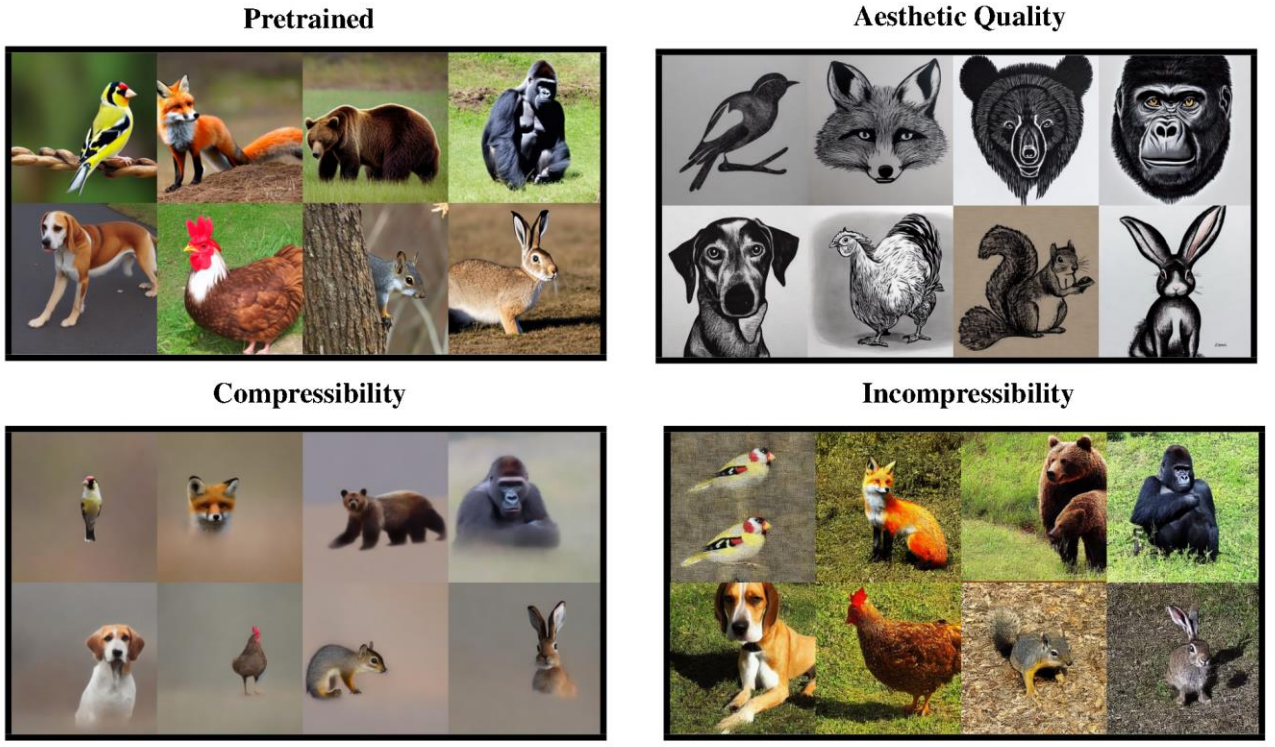


Figure 3 (DDPO samples) Qualitative depiction of the effects of RL fine-tuning on different reward functions. DDPO transforms naturalistic images into stylized line drawings to maximize predicted aesthetic quality, removes background content and applies a foreground blur to maximize compressibility, and adds artifacts and high-frequency noise to maximize incompressibility.

Experimental results – comparison to RWR

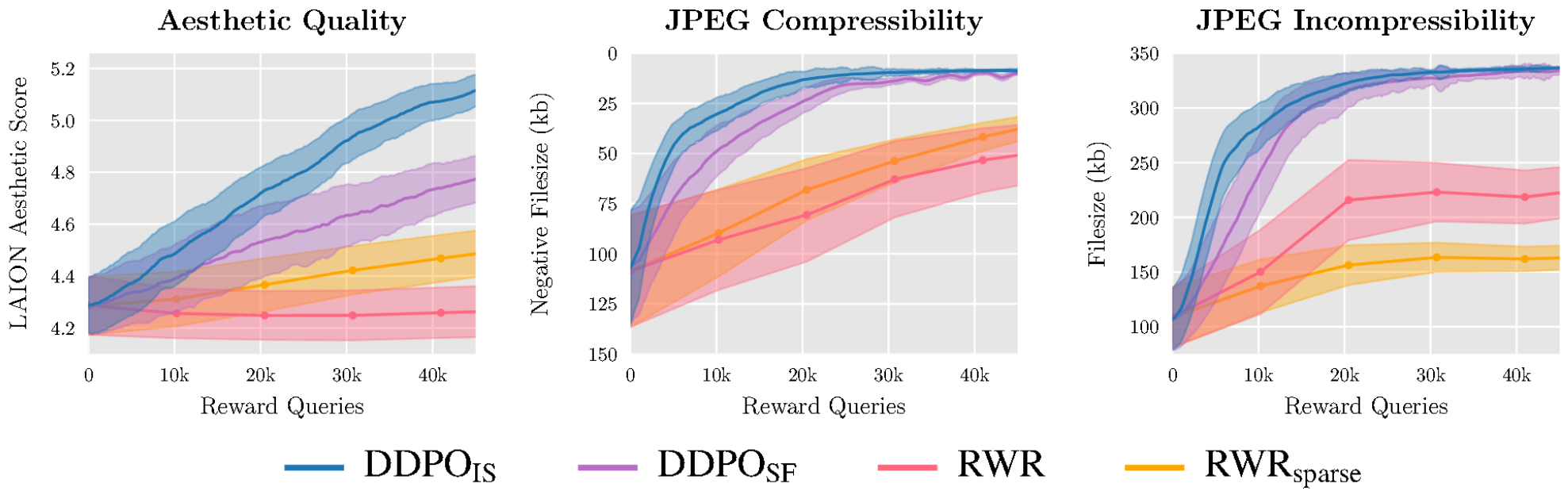
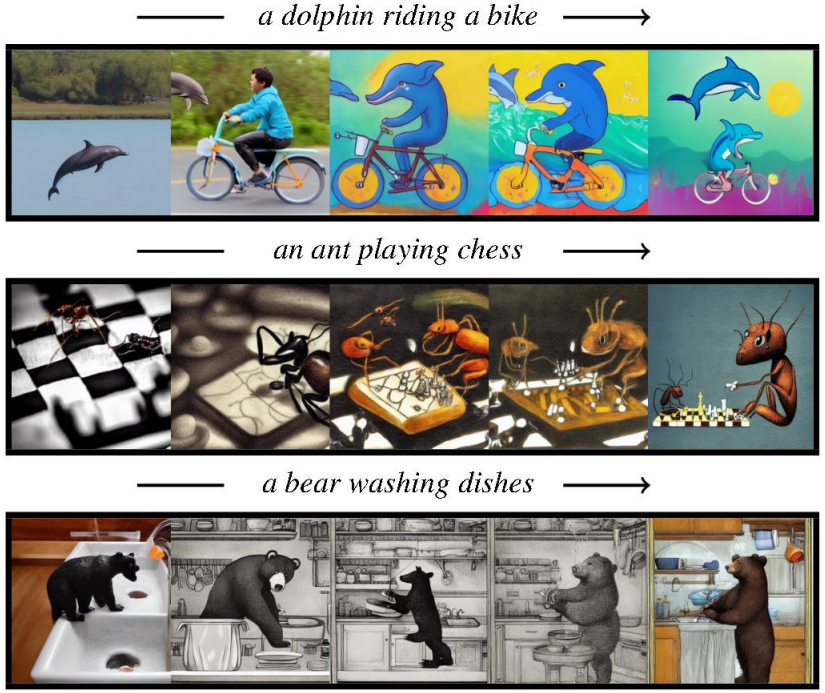


Figure 4 (Finetuning effectiveness) The relative effectiveness of different RL algorithms on three reward functions. We find that the policy gradient variants, denoted DDPO, are more effective optimizers than both RWR variants.

Experimental results – prompt alignment

- The prompts for this task all have the form “a(n) [animal] [activity] ”, where the animal comes from the same list of 45 common animals used in Section 6.1 and the activity is chosen from a list of 3 activities: “riding a bike”, “playing chess”, and “washing dishes”.



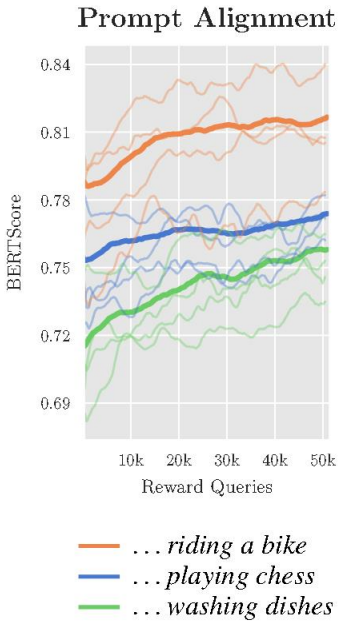


Figure 5 (Prompt alignment) (L) Progression of samples for the same prompt and random seed over the course of training. The images become significantly more faithful to the prompt. The samples also adopt a cartoon-like style, which we hypothesize is because the prompts are more likely depicted as illustrations than realistic photographs in the pretraining distribution. **(R)** Quantitative improvement of prompt alignment. Each thick line is the average score for an activity, while the faint lines show average scores for a few randomly selected individual prompts.

Experimental results - generalization

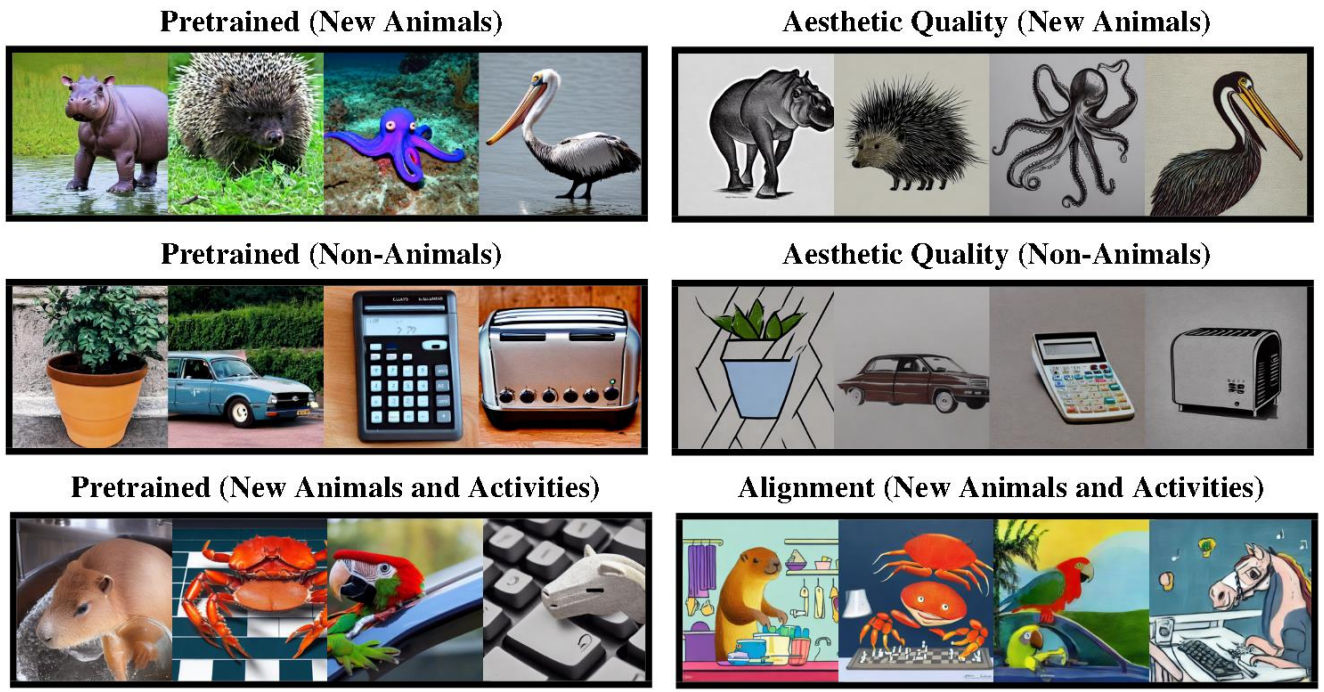


Figure 6 (Generalization) For aesthetic quality, finetuning on a limited set of 45 animals generalizes to both new animals and non-animal everyday objects. For prompt alignment, finetuning on the same set of animals and only three activities generalizes to both new animals, new activities, and even combinations of the two. The prompts for the bottom row (left to right) are: "*a capybara washing dishes*", "*a crab playing chess*", "*a parrot driving a car*", and "*a horse typing on a keyboard*". More samples are provided in Appendix C.

Experimental results – reward model overoptimization

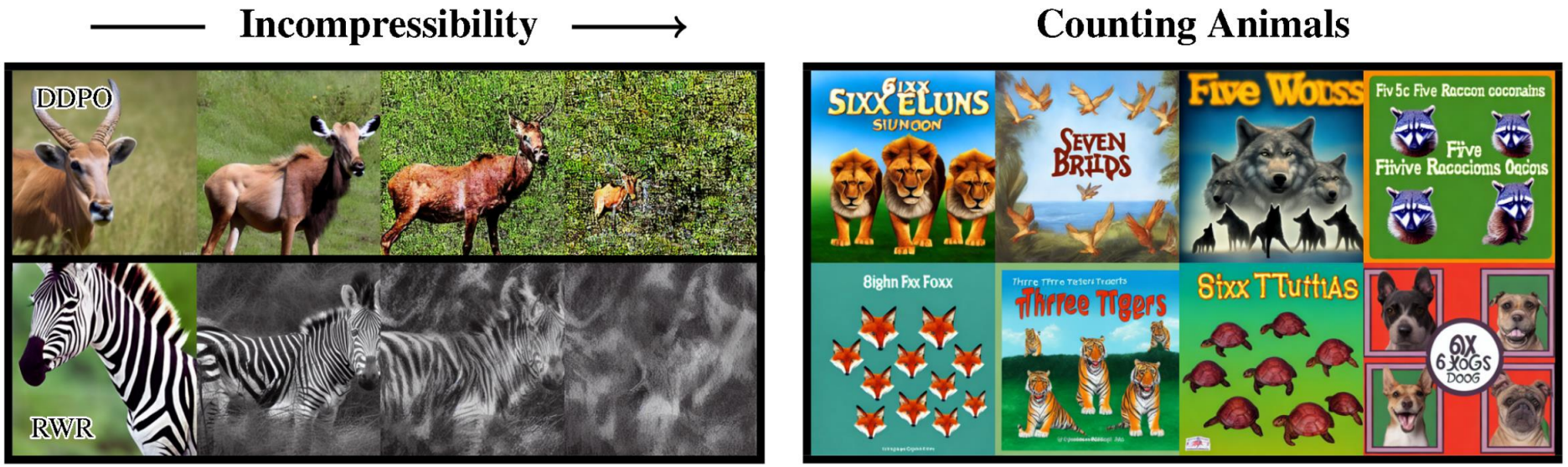


Figure 7 (Reward model overoptimization) Examples of RL overoptimizing reward functions. **(L)** The diffusion model eventually loses all recognizable semantic content and produces noise when optimizing for incompressibility. **(R)** When optimized for prompts of the form "*n animals*", the diffusion model exploits the VLM with a typographic attack [14], writing text that is interpreted as the specified number n instead of generating the correct number of animals.