10-703 Recitation: Reinforcement Learning with Diffusion Models

Li Cao

Carnegie Mellon University

Fall 2025

Content

Why RL with Diffusion?

Diffusion models (e.g., Stable Diffusion) are excellent density estimators ($p_\theta(x)$), but we often care about metrics beyond likelihood:

- **Aesthetics:** Does the image look artistic? (Subjective)
- **Alignment:** Does the image actually match the text prompt?
- **Functionality:** (Robotics) Does the generated trajectory solve the task?
- **Compressibility:** Is the file size small? (JPEG size optimization)

The Problem

These "rewards" are often non-differentiable or hard to optimize directly via standard loss functions (MSE). We need Reinforcement Learning.

Diffusion Model: The Forward Process

The forward process q is a fixed Markov chain that gradually adds Gaussian noise to the data x_0 until it becomes pure noise x_T.

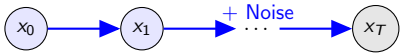
- **Transition Kernel:**

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

- α_t: Variance schedule $(0 < \alpha_t < 1)$.
- **Marginal Probability:** A key property allows us to sample x_t directly from x_0 without iterating steps:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

where $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.



Diffusion Model: The Reverse Process

The reverse process p_θ is a learned Markov chain that removes noise to generate data.

- **Reverse Transition:**
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$
- Instead of predicting μ_θ directly, we usually predict the noise $\epsilon_\theta(x_t, t)$ added at step t.
- **Training Objective (Simplified):** We train to predict the noise using simple Mean Squared Error (MSE):
$$\mathcal{L}_{simple} = \mathbb{E}_{t,x_0,\epsilon}\left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2\right]$$

Key Insight: If we can predict the noise, we can subtract it to recover the mean of the previous step.

Diffusion Model: Sampling Methods

How do we use the trained model to generate images?

DDPM (Denoising Diffusion Probabilistic Model)

- Stochastic Sampling.
- $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)) + \sigma_t z$
- $z \sim \mathcal{N}(0, I)$ adds randomness at every step.

DDIM (Denoising Diffusion Implicit Model)

- Deterministic Sampling
- A non-Markovian process with the *same* marginals.
- $\sigma_t = 0$ (No injected noise z).
- **Significance:** Mapping from $x_T \to x_0$ becomes deterministic.
- Crucial for differentiable optimization (e.g., AlignProp) as it allows backpropagation through the ODE chain.

Method 1: Reward-Weighted Regression (RWR)

The simplest approach to align diffusion models with a reward $r(x_0, c)$.

Concept

Treat the generation process as a bandit problem or simple likelihood maximization weighted by success.

1. Generate a batch of images $\{x_0^i\}$.
2. Calculate reward $r(x_0^i, c)$ (e.g., Aesthetic Score).
3. Compute weights:
$$w_i = \exp(\beta r(x_0^i, c))$$
4. Fine-tune the model to maximize likelihood of high-reward samples:
$$\mathcal{L}_{RWR} = -\sum w_i \log p_\theta(x_0^i|c)$$

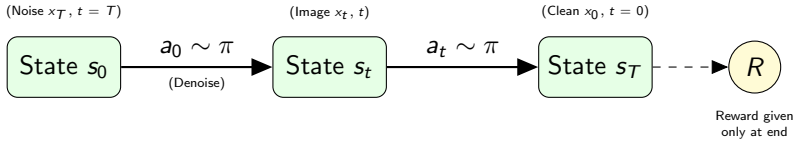
Pros: Simple implementation. **Cons:** Can suffer from mode collapse; doesn't exploit temporal structure.

Method 2: Denoising Diffusion Policy Optimization (DDPO)

DDPO

Paper: Black et al., "Training Diffusion Models with Reinforcement Learning"

Denoising as an MDP: We can map the iterative denoising process to a standard Reinforcement Learning Multi-Step MDP.



DDPO: The MDP Formulation

MDP Definition:

- **State (s_t):** (c, t, x_t)
- **Action (a_t):** The predicted x_{t-1} (or predicted noise ϵ)
- **Policy (π_θ):** The diffusion model $p_\theta(x_{t-1}|x_t, c)$
- **Reward:** $r(x_0, c)$ at the final step; 0 otherwise.

Policy Gradient: We can now apply PPO (Proximal Policy Optimization)

$$\nabla_\theta J = \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log p_\theta(x_{t-1}|x_t) \cdot R\right]$$

Since the "policy" is Gaussian, the log-probability gradient is straightforward (mean matching).

DDPO Algorithm Details

- **Importance Sampling:** Since collecting data is slow (diffusion is slow), DDPO uses Importance Sampling to reuse data collected from an old policy π_{old}.

$$\frac{p_\theta(x_{t-1}|x_t)}{p_{\theta_{old}}(x_{t-1}|x_t)}$$

- **Exact Likelihood:** Unlike GANs, Diffusion models provide exact likelihoods, making the PPO ratio calculation precise.
- **Results:**
  - Improves aesthetic scores significantly.
  - Better "Sim-to-Real" transfer in Robotics (fine-tune on simulator success).

What if the Reward is Differentiable?

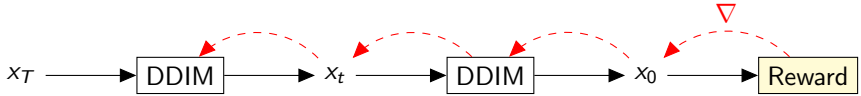
If the reward function (e.g., a neural network classifier, CLIP score) is differentiable, we don't necessarily need Policy Gradients.

AlignProp / DDIM Backprop

Paper: Prabhudesai et al., "Aligning Text-to-Image Diffusion Models with Reward Backpropagation"

We can treat the DDIM (deterministic sampling) chain as a deep Recurrent Neural Network (RNN).

AlignProp: Concept



- **Mechanism:** Backpropagate the gradient of the reward $\nabla_x R(x_0)$ all the way through the denoising steps to update model weights θ.
- **Pros:** Lower variance than RL (uses analytic gradients). Very sample efficient.
- **Cons:** High memory cost (must store computation graph for 50+ steps). Requires gradient checkpointing.

Extension: Video Diffusion Alignment

Paper: Prabhudesai, Mendonca et al.
Applying these concepts to Video Generation.

- **Problem:** Video models often ignore the prompt dynamics (e.g., "A shark riding a bicycle" $->$ generates just a shark).
- **Solution:** Use reward gradients to enforce temporal consistency and object-action alignment.
- **Result:** Lecture slides showed examples of "Lion riding bike" and "Shark riding bike" where baseline failed but alignment succeeded.

Diffusion-QL and Steering

Diffusion-QL

- Instead of policy gradients, learn a Q-function $Q(s, a)$.
- Backpropagate the Q-value through the denoising chain to update the action.
- Used heavily in Offline RL.

Latent-Noise Steering (DSRL-NA)

- Instead of changing the model weights, learn to pick the *initial noise* x_T.
- Learn a policy $\pi^w(s)$ that outputs the optimal starting noise w to result in a high-reward trajectory.
- **Benefit:** No expensive fine-tuning of the massive diffusion backbone.

Summary: The Landscape

Method	Reward Access	Mechanism	Pros
RWR	Black-Box	Re-weighting	Simple
DDPO	Black-Box	Policy Gradient	Stable, exact log-p
AlignProp	Differentiable	BPTT	Sample Efficient
Diffusion QL	Differentiable	Q-Learning	Offline RL

Takeaway: Diffusion models are just multi-step distributions. We can optimize them using the full toolkit of Reinforcement Learning (PG, Q-Learning, Bandit methods) to achieve objectives beyond simple data likelihood.