Reinforcement Learning with Discrete Diffusion Policies

Google Research

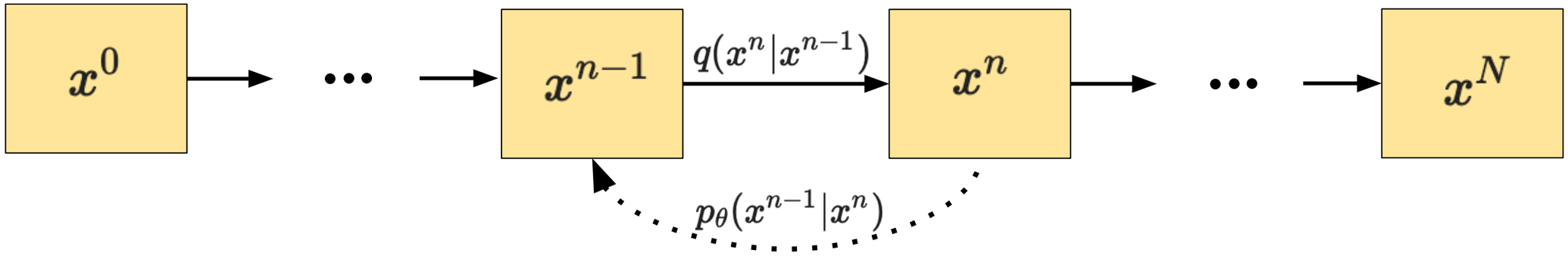
Ofir Nabati

Discrete Diffusion Models

- Clean sample:

$$x^0 = x^0_{1:K} \in \mathcal{A}^K \qquad \mathcal{A} = \{1, 2, 3, \ldots, V\}$$

- Forward process: $q(x^n|x^{n-1})$
- (Parameterized) backward process: $p_\theta(x^{n-1}|x^n)$



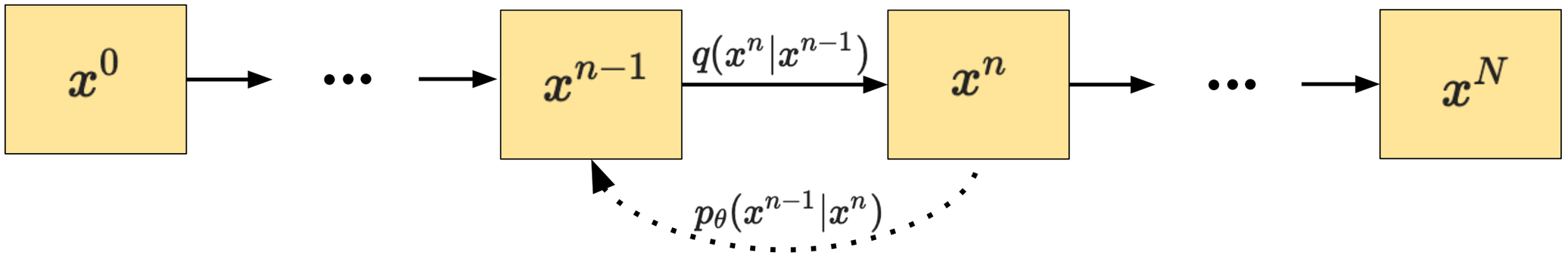
$$q(x^n|x^{n-1}) = Cat(x^n; Q_n^\top x^{n-1})$$

Q_n is a probability matrix

treated as one-hot vector $e_{x^{n-1}}$

Google Research

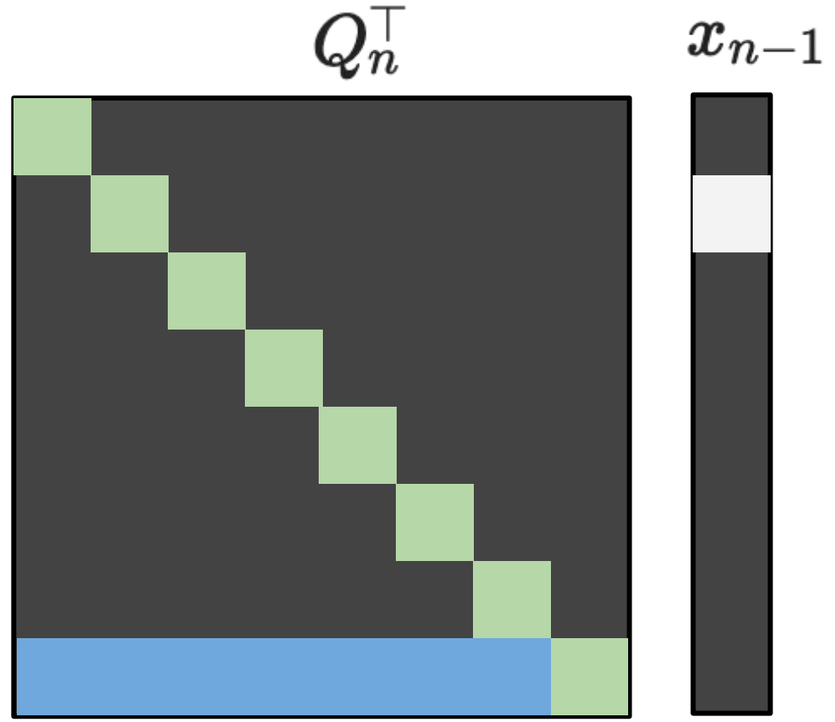
Masked Discrete Diffusion Models



We add an “absorbing” state: **m**(ask)

$$Q_n = \alpha_n I + (1 - \alpha_n) 1 e_m^\top$$

$$x_n \sim$$



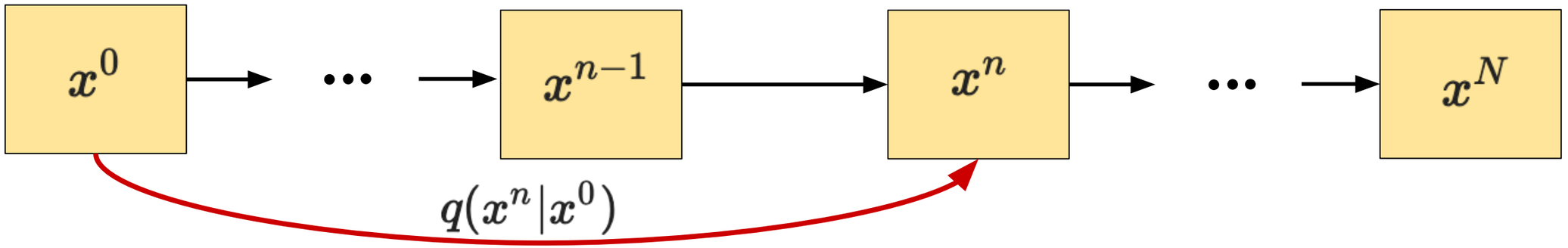
The [MASK] sat on the mat ⟵ The [MASK] sat on [MASK] [MASK] ⟵ [MASK] [MASK] [MASK] [MASK] [MASK] [MASK]

$q(x^n|x^{n-1})$ ⟶

$p_\theta(x^{n-1}|x^n)$ ⟵

Google Research

Masked Discrete Diffusion Models



The marginal distribution at time n:

$$q(x^n|x^0) = Cat(x^n; \bar{Q}_n^\top x^0) = Cat(x^n; \bar{\alpha}_n x^0 + (1-\bar{\alpha}_n)e_m)$$

$$\bar{Q}_n = \prod_{i=1}^{n} Q_i = \bar{\alpha}_n I + (1-\bar{\alpha}_n)1e_m^\top$$

$\{\bar{\alpha}_n\}_{n=1}^N$ -noise schedule e.g. linear: $\bar{\alpha}_n = 1 - \frac{n}{N}$

Masked Discrete Diffusion Models

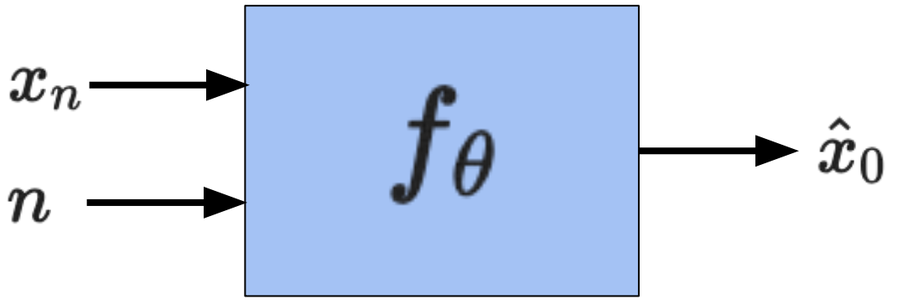
The diffusion objective (ELBO):

$$\log p_\theta(x^0) \geq \mathbb{E}_{q(x^1|x^0)}\left[\log p_\theta(x^0|x^1)\right] - \sum_{n>1} \mathbb{E}_{q(x^n|x^0)}\left[d_{KL}\left(q(x^{n-1}|x^n, x^0) \| p_\theta(x^{n-1}|x^n)\right)\right] - d_{KL}\left(q(x^N|x^0)\|p(x^N)\right)$$

posterior

Usually the backward process is parameterized as the posterior with a **denoiser**

$$p_\theta(x^{n-1}|x^n) = q(x^{n-1}|x^n, f_\theta(x^n, n))$$



a distribution over values!

Google Research

Masked Discrete Diffusion Models

For masked discrete diffusion..

$$d_{KL}\left(q(x^{n-1}|x^n, x^0)||p_\theta(x^{n-1}|x^n)\right) = -\frac{\bar{\alpha}_{n-1} - \bar{\alpha}_n}{1 - \bar{\alpha}_n}\delta_{x_n,m} \cdot (x^0)^\top \log f_\theta(x^n, n)_{x^0}$$

Estimating the (negative) ELBO

$$\mathcal{L}_{ELBO}(x^0;\theta) \approx \frac{1}{M}\sum_{i=1}^{M} d_{KL}\left(q(x^{n_i-1}|x^{n_i}, x^0)||p_\theta(x^{n_i-1}|x^{n_i})\right)] \quad \text{where} \quad n_i \sim \mathcal{U}\{1, 2, \ldots, N\}$$

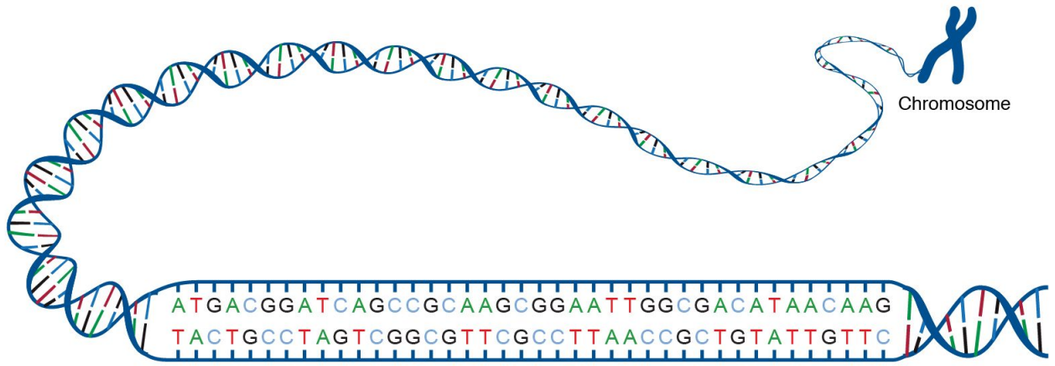
$$x^{n_i} \sim q(x^{n_i}|x^0)$$

Google Research

Why using discrete diffusion policies?

Combinatorial action spaces:

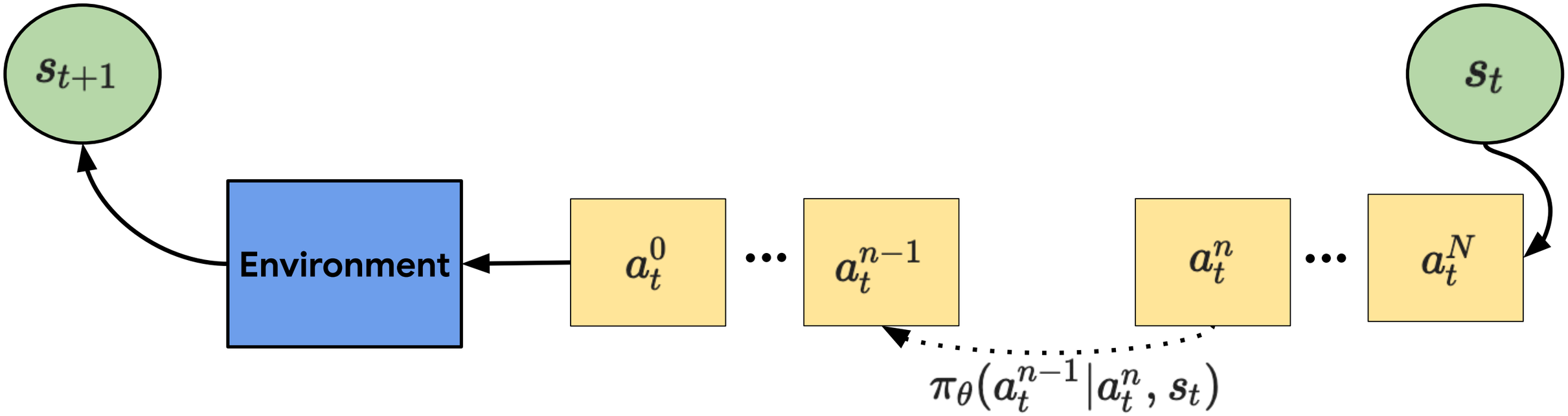
- Sequence generation
- Action sequences / planning
- Multi agent



How to model a diffusion policy π_θ ?

Option 1: Augmented MDP

There is only a **backward process**
and it's the **policy** $\pi_\theta(a^{n-1}|a^n, s) = p_\theta(a^{n-1}|a^n, s)$
We create an **augmented MDP** with augmented states: (s_t, a_t^n)

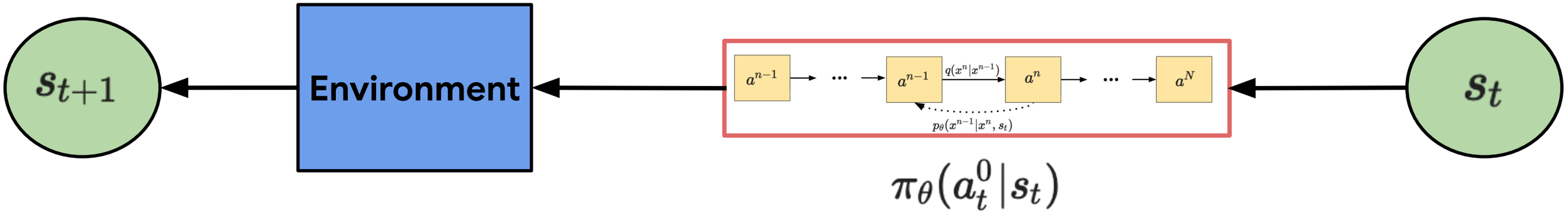


How to model a diffusion policy π_θ ?

Option 2: The diffusion model

The policy is the **diffusion model**

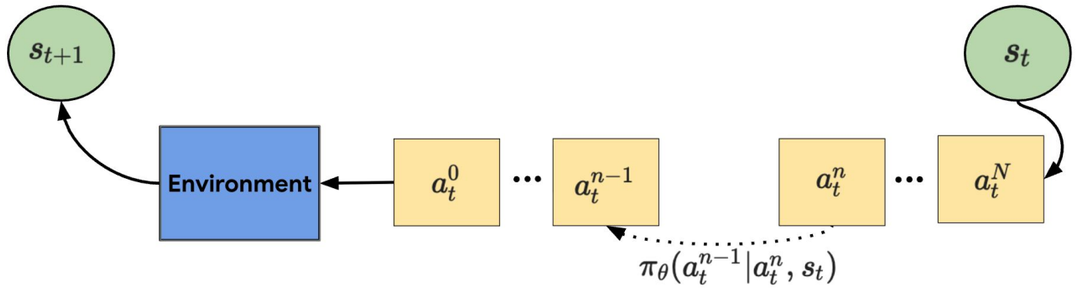
$$\pi_\theta(a^0|s) = p_\theta(a^0|s) = \mathbb{E}_{a^{1:N}}\left[\prod_{n=1}^{N} p_\theta(a^{n-1}|a^n, s)\right]$$



Google Research

Which approach is better?

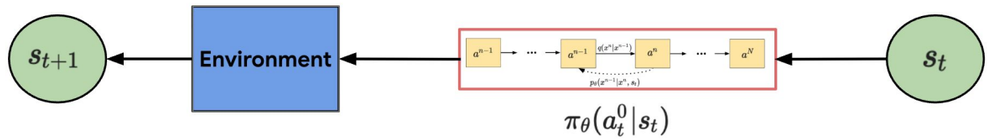
Augmented MDP



- Unbiased
- Noisy - high variance
- Sample inefficient

e.g: FlowGRPO, DDPO, PPO-variants, DRAKES

Policy as the diffusion model



- Biased
- Low variance

- e.g: DiffusionNFT, Self Imitation, BoNC, RWR

Google Research

Policy Mirror Decent

Provably convergent, regularized policy optimization step:

$$\pi(\cdot|s) \in \arg\max_{\pi \in \Pi} \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|s)}[A^{\pi_{\text{old}}}(s,\mathbf{a})] - \lambda d_{KL}(\pi, \pi_{\text{old}}; s) \quad \forall s \in \mathcal{S}$$

$$A^{\pi_{\text{old}}}(s,\mathbf{a}) := q^{\pi_{\text{old}}}(s,\mathbf{a}) - v^{\pi_{\text{old}}}(s)$$

previous turn policy

Closed form solution:

$$\pi_{\text{MD}}(\mathbf{a}|s) = \pi_{\text{old}}(\mathbf{a}|s) \exp(A^{\pi_{\text{old}}}(s,\mathbf{a})/\lambda) \;/\; Z(s)$$

Shani et al. “Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps”

Google Research

Policy Mirror Decent

Policy optimization in RL is done in two steps:

Start with initial policy π_0
For k=1,2,3,...:

- **Policy evaluation:** estimate the Q-function of π_k
- **Policy Improvement:** compute the next-step policy π_{k+1}:

$$\pi_{k+1} \in \arg\min_{\pi_\theta \in \Pi} \mathbb{E}_{s\sim\mathcal{D}}\left[d(\pi_\theta, \pi^k_{\mathrm{MD}}; s)\right]$$

$$\pi_{\mathrm{MD}}(\mathbf{a}|s) = \pi_{\mathrm{old}}(\mathbf{a}|s)\exp(A^{\pi_{\mathrm{old}}}(s,\mathbf{a})/\lambda) \;/\; Z(s)$$

Google Research

Forward KL+ Diffusion Policy (Self Imitation)

The choice of the distance determine the objective:

Forward KL: $d_{KL}(\pi_{MD}^k \| \pi_\theta; s)$

\+ policy as the diffusion model: $-\log \pi_\theta(a^0|s) \leq \mathcal{L}_{ELBO}(a^0, s; \theta)$

$$\mathcal{L}_{\text{FKL}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}, \hat{\mathcal{A}}_s \sim \pi_k}\left[\sum_{\mathbf{a}^0 \in \hat{\mathcal{A}}_s} \left(\text{softmax}_{\mathbf{a}\in\hat{\mathcal{A}}_s}\left(A^{\pi_k}(s, \mathbf{a}^0)/\lambda\right)\right) \cdot \mathcal{L}_{\text{ELBO}}(\mathbf{a}^0, s; \theta)\right]$$

sampled action set

Softmax weights

temperature

Reverse KL+ Diffusion Policy

Reverse KL: $d_{KL}(\pi_\theta \| \pi^k_{MD}; s)$

+ policy as the backward process (augmented MDP):

$$-\log \pi_\theta(a^{n-1}|a^n, s) = -\log p_\theta(a^{n-1}|a^n, s)$$

$$\mathcal{L}_{\mathrm{RKL}}(\theta) = \mathbb{E}_{s\sim\mathcal{D},\mathbf{a}\sim\pi_k}[-\eta(s,\mathbf{a};\theta)A^{\pi_k}(s,\mathbf{a}) + \lambda d_{KL}(\pi_\theta, \pi_k; s)]$$

Importance sampling: $\eta((s,\mathbf{a}^n),\mathbf{a}^{n-1};\theta) = \dfrac{p_\theta(\mathbf{a}^{n-1}|\mathbf{a}^n, s)}{p_k(\mathbf{a}^{n-1}|\mathbf{a}^n, s)}$

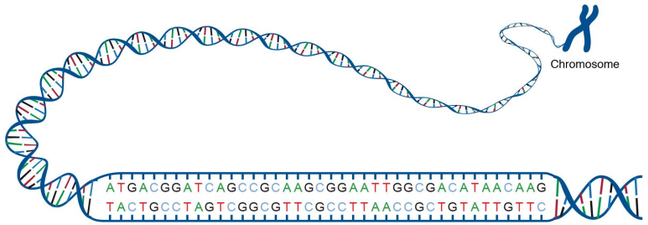
Experiments

Goals

- Better performance (high return)
- Better efficiency compared to AR:

    Number of diffusion steps (N) < sequence length (T)

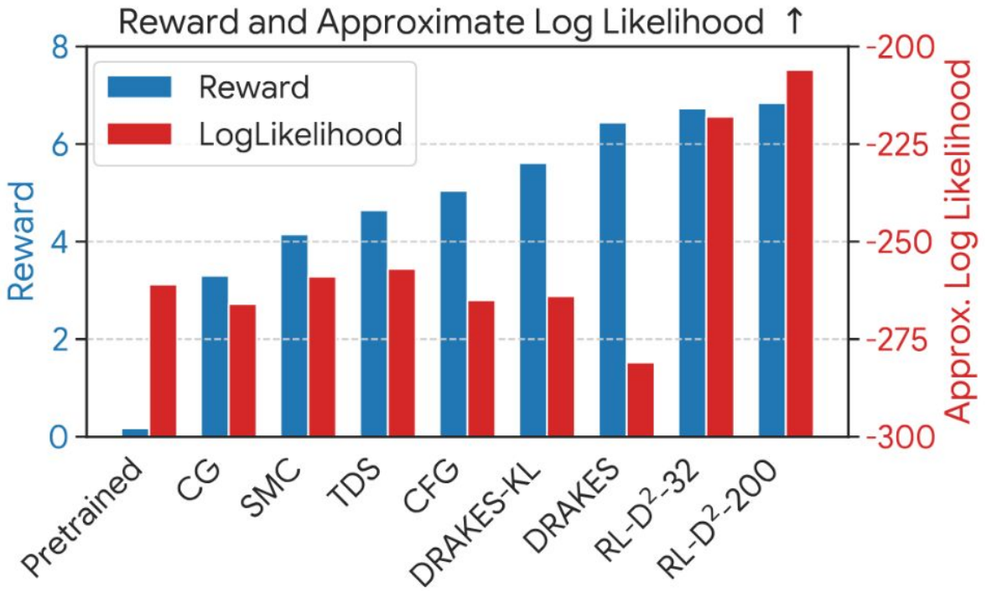
Experiments

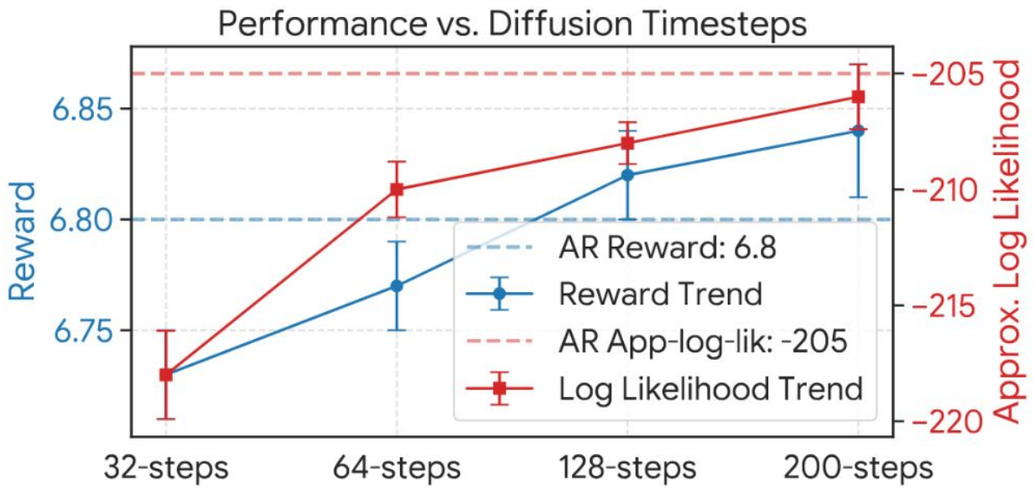


DNA sequence generation (single step)

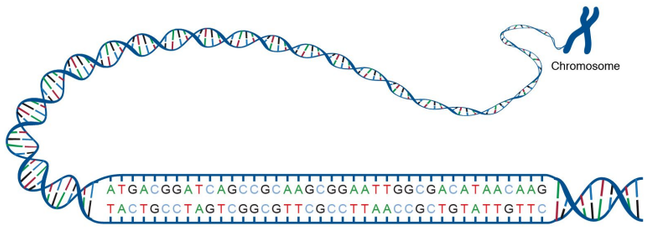
- RL fine tuning w.r.t a pretrained model
- Reward measure **Predicted Activity** in liver cancer cells
- Sequence length: 200
- Action space size: 4^{200}
- Objective: Self Imitation (FKL)

Experiments





- Outperform AR with half the number of steps
- SOTA compared to DRAKES



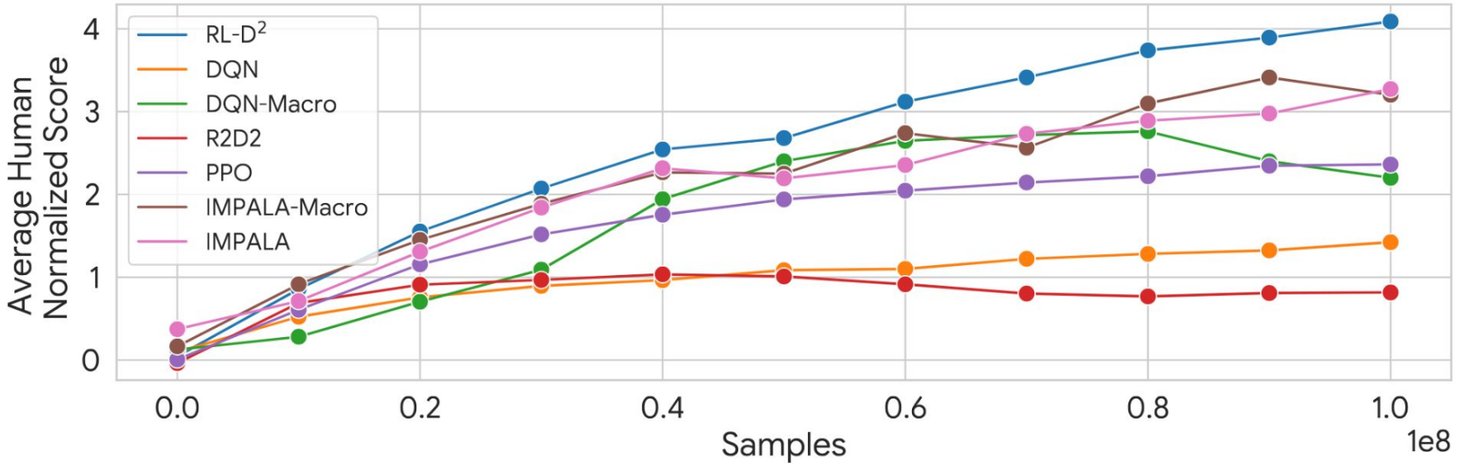
Experiments

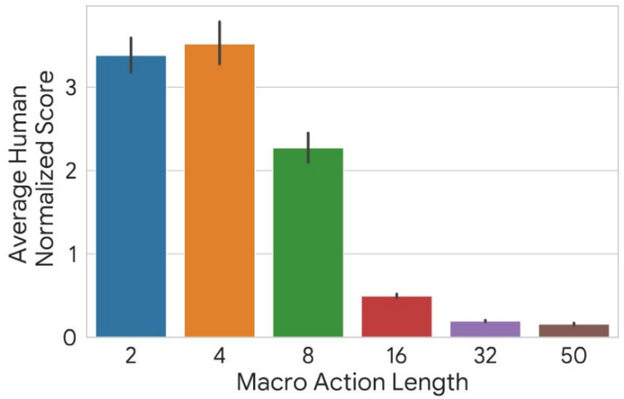
Action sequence generation (Atari)

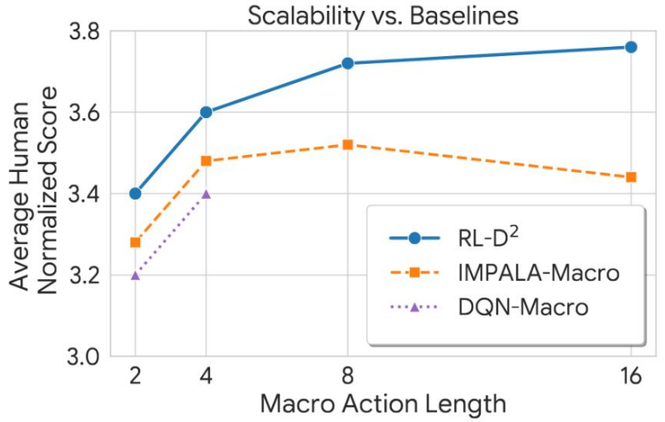
- From scratch
- Macro-action length: 4
- Action space size: 20^4
- Objective: Self Imitation (FKL)

Experiments

Action sequence generation (Atari)







Experiments

Google Research Football

- From scratch
- Different scenarios: 11vs11, 5vs5, corner
- Sequence length: players number
- Action space size: $19^{\#players}$
- Objective: RKL (PPO) and FKL
- Reward: +1 for goal, -1 for the opponent goal
- Stochastic environment

Experiments

small budget

	PASS & SHOOT	RUN PASS & SHOT	3 VS 1	CT-EASY	CT-HARD	CORNER	5 VS 5	11 VS 11	11 VS 11 SM
AUTOREGRESSIVE	78.2 ± 40.1	70.2 ± 35.2	75.8 ± 35.1	69.3 ± 30.6	33.6 ± 40.1	0.0 ± 0.0	71.1 ± 30.5	34.4 ± 14.4	0.0 ± 0.0
RL-D^2 RKL (OURS)	$\mathbf{100 \pm 0.0}$	$\mathbf{98.5 \pm 1.6}$	$\mathbf{98.6 \pm 1.0}$	$\mathbf{99.1 \pm 0.6}$	$\mathbf{97.7 \pm 2.2}$	$\mathbf{96.3 \pm 3.4}$	$\mathbf{99.6 \pm 0.4}$	$\mathbf{99.8 \pm 0.2}$	0.0 ± 0.0
RL-D^2 FKL (OURS)	$\mathbf{99.0 \pm 1.0}$	$\mathbf{98.6 \pm 0.5}$	$\mathbf{97.3 \pm 2.0}$	98.0 ± 1.1	$\mathbf{97.1 \pm 2.9}$	93.2 ± 2.9	98.4 ± 1.0	97.4 ± 0.9	$\mathbf{67.2 \pm 8.1}$
MAT	$\mathbf{97.9 \pm 2.1}$	$\mathbf{98.3 \pm 1.2}$	92.9 ± 1.1	87.9 ± 2.1	88.2 ± 3.8	95.3 ± 2.5	93.7 ± 0.9	92.0 ± 2.4	9.0 ± 3.7
MAPPO	99.5 ± 0.2	73.2 ± 3.6	93.2 ± 1.5	70.1 ± 3.8	63 ± 2.1	53.1 ± 5.3	95.4 ± 1.6	52.6 ± 3.1	5.0 ± 0.2

- Practically solve Gfootball ~100% success for all scenarios
- Outperform SOTA AR method (MAT)
- FKL much more efficient but asymptotically underperform
- Needs a lot of samples..