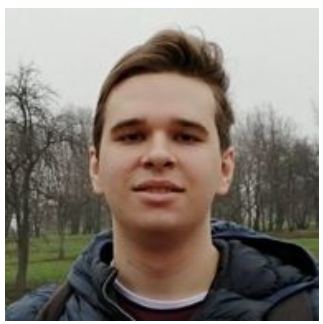
# Generative Flow Networks as Entropy-Regularized RL

Daniil Tiapkin[1,3], Nikita Morozov[1], Alexey Naumov[1], Dmitry Vertrov [1]

[1]HSE University [2]AIRI [3]École Polytechnique

Based on https://arxiv.org/abs/2310.12934




# Problem

Suppose we have a finite discrete space of objects $\mathcal{X}$ with a black-box non-negative reward function $\mathcal{R}(x)$ given for every $x \in \mathcal{X}$.

**Goal:** Construct and train a model that will sample objects from $\mathcal{X}$ from the probability distribution $\mathcal{R}(x)/\mathrm{Z}$ proportional to the reward, where $\mathrm{Z}$ is an unknown normalizing constant.

**Standard approach**: MCMC methods ...

# MCMC

- What can be done if drawing i.i.d. samples from $\mathcal{R}(\cdot)/\mathrm{Z}$ is not an option?
- If we run the (ergodic) Markov chain $(Z_k)_{k\geq 0}$ for a long time (started from anywhere), then for large $N$ the distribution of $Z_N$ will be approximately invariant: $\mathrm{Law}(Z_N) \approx \mathcal{R}(\cdot)/\mathrm{Z}$. We can then set $X_1 = Z_N$, and then restart and rerun the Markov chain to obtain $X_2, X_3, \ldots$,

## Important question

How to construct $\mathrm{P}(x, A)$ such that the distribution of $X_n$ converges to invariant distribution $\mathcal{R}(\cdot)/\mathrm{Z}$ as quickly as possible for arbitrary initial distribution $\xi$?

# Example: Metropolis-Hastings algorithm

Let $Q(x, A) = \int_A q(x, y)\mathrm{d}y$ be some MK (e.g. Gaussian)

1. Choose $X_0$.
2. Given $X_k$, a candidate move $Y_{k+1}$ is sampled from $Q(X_k, \cdot)$
3. $X_{k+1} = Y_{k+1}$ with probability $\alpha(X_k, Y_{k+1})$, otherwise $X_{k+1} = X_k$, where acceptance ratio

$$\alpha(x, y) = \min\left\{1, \frac{\mathcal{R}(y)q(y, x)}{\mathcal{R}(x)q(x, y)}\right\}$$

Example: Random walk MH
Take $q(x, y) = \overline{q}(y - x)$, where $\overline{q}(x) = \overline{q}(-x)$. Then

$$Y_{k+1} = X_k + Z_{k+1}, \quad Z_{k+1} \sim \overline{q}$$

In this case

$$\alpha(x, y) = \min\left\{1, \frac{\mathcal{R}(y)}{\mathcal{R}(x)}\right\}$$

# Example: Langevin Dynamics

Langevin Dynamics Itô SDE:

$$\mathrm{d}X_t = -\nabla U(X_t)\,\mathrm{d}t + \sqrt{2}\mathrm{d}W_t,$$

Invariant measure: $\mathcal{R}(x) = e^{-U(x)}$ and $\mathrm{Law}(X_t) \to \mathcal{R}(\cdot)/\mathrm{Z}$ as $t \to \infty$.

1. Take $\mathcal{R}(x)/\mathrm{Z} = (2\pi)^{-1/2}\mathrm{e}^{-x^2/2}$.
2. SDE: $\mathrm{d}X_t = X_t\,\mathrm{d}t + \sqrt{2}\mathrm{d}W_t$, $X_0$ is independent of $W$. This is Ornstein–Uhlenbeck process
3. Apply Ito's formula to obtain

$$X_t = X_0\mathrm{e}^{-t} + \sqrt{2}\int_0^t e^{-(t-s)}\mathrm{d}W_s$$

4. Since the Itô integral of deterministic integrand is normally distributed, we readily have

$$\mathrm{Law}(X_t) = \mathcal{N}(\theta_0\mathrm{e}^{-t}, 1-\mathrm{e}^{-2t}) \to \mathcal{N}(0,1)$$

# Example: Langevin Dynamics

Itô SDE:

$$\mathrm{d}X_t = -\nabla U(X_t)\,\mathrm{d}t + \sqrt{2}\mathrm{d}W_t,$$

Invariant measure: $\mathcal{R}(x) = e^{-U(x)}$

1. First-order discretization (Unadjusted Langevin Algorithm, ULA):

$$Y_{k+1} = Y_k - \gamma\nabla U(Y_k) + \sqrt{2\gamma}Z_{k+1}, \quad i.i.d.\ Z_k \sim \mathrm{N}(0, \mathrm{I}_d)$$

   Equivalently, $Y_{k+1} \sim \mathcal{N}(Y_k - \gamma\nabla U(Y_k), 2\gamma\,\mathrm{I}_d)$
2. Metropolis-adjusted Langevin Algorithm (MALA):
   ULA + Metropolis-Hastings correction;
3. Demo: https://chi-feng.github.io/mcmc-demo
4. If we can't calculate $\nabla U$ replace it by its estimate over batch (SGLD, SGLD-FP, SAGA etc)
5. Mixture of global (importance sampling) and local steps with trainable proposals (Ex$^2$MCMC), see [Samsonov et al., 2022]

# Problem

Suppose we have a finite discrete space of objects $\mathcal{X}$ with a black-box non-negative reward function $\mathcal{R}(x)$ given for every $x \in \mathcal{X}$.

**Goal:** Construct and train a model that will sample objects from $\mathcal{X}$ from the probability distribution $\mathcal{R}(x)/\mathrm{Z}$ proportional to the reward, where $\mathrm{Z}$ is an unknown normalizing constant.

**Standard approach**: MCMC methods usually suffer from the mixing problem.

**GFlowNet as an alternative:** it learns a stochastic policy that starts with "empty" objects and modifies them until completion. GFlowNet generates each sample independently and, like RL methods, relies on exploration and the generalization power of ML to guess and discover new modes of the reward function (or regions of low energy).

**References:** [Bengio et al., 2021], [Bengio et al., 2023] etc.

# Flows Over Directed Acyclic Graphs

**DAG**: $\mathcal{G} = (\mathcal{S}, \mathcal{E})$: $\mathcal{S}$ is a state space, $\mathcal{E} \subseteq \mathcal{S} \times \mathcal{S}$ is a set of edges. For $s \in \mathcal{S}$, $\mathcal{A}_s$ is the set of actions, i.e. possible next states: $s' \in \mathcal{A}_s \Leftrightarrow s \to s' \in \mathcal{E}$; $s_0 \in \mathcal{S}$ is an initial state ("empty" object) and the set of terminal states (states with no outgoing edges) directly corresponds to $\mathcal{X}$

Denote $\mathcal{T}$ as a set of all complete trajectories in the graph $\tau = (s_0 \to s_1 \to \ldots \to s_{n_\tau})$, where $n_\tau$ is the trajectory length, $s_i \to s_{i+1} \in \mathcal{E}$, $s_0$ is always the initial state, and $s_{n_\tau}$ is always a terminal state, thus $s_{n_\tau} \in \mathcal{X}$

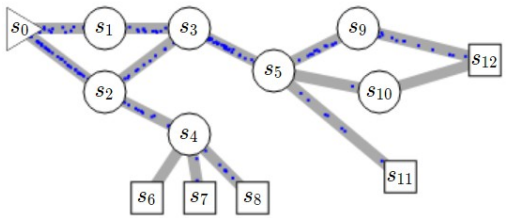



Figure: Directed Acyclic Graphs

A **trajectory flow** is any nonnegative function $\mathcal{F}: \mathcal{T} \to \mathbb{R}_{\geq 0}$.

$$\mathcal{F}(s) \triangleq \sum_{\tau \ni s} \mathcal{F}(\tau),$$

$$\mathcal{F}(s \to s') \triangleq \sum_{\tau \ni (s \to s')} \mathcal{F}(\tau).$$


Figure: Molecule synthesis experiment

# Reward Matching Condition

If $\mathcal{F}$ is not identically zero, it can be used to introduce a probability distribution over complete trajectories:

$$\mathcal{P}(\tau) \triangleq \frac{1}{\mathrm{Z}_{\mathcal{F}}}\mathcal{F}(\tau), \quad \mathrm{Z}_{\mathcal{F}} \triangleq \sum_{\tau \in \mathcal{T}} \mathcal{F}(\tau).$$

Then, $\mathcal{F}(s)/\mathrm{Z}_{\mathcal{F}}$ corresponds to the probability that a random trajectory contains the state $s$, and $\mathcal{F}(s \to s')/\mathrm{Z}_{\mathcal{F}}$ corresponds to the probability that a random trajectory contains the edge $s \to s'$. These definitions also imply $\mathrm{Z}_{\mathcal{F}} = \mathcal{F}(s_0) = \sum_{x \in \mathcal{X}} \mathcal{F}(x)$.

Denote $\mathcal{P}(x)$ as the probability that $x$ is the terminal state of a trajectory sampled from $\mathcal{P}(\tau)$. If *the reward matching constraint* is satisfied for all terminal states $x \in \mathcal{X}$:

$$\mathcal{F}(x) = \mathcal{R}(x), \tag{1}$$

$\mathcal{P}(x)$ will be equal to $\mathcal{R}(x)/\mathrm{Z}$. **Thus by sampling trajectories we can sample the objects from the distribution of interest**

# Markovian flow

The trajectory flow $\mathcal{F}$ is said to be Markovian if $\forall s \in \mathcal{S} \setminus \mathcal{X}$ there exist distributions $\mathcal{P}_{\mathrm{F}}(-|s)$ over the children of $s$ such that $\mathcal{P}$ can be factorized as

$$\mathcal{P}(\tau = (s_0 \to \ldots \to s_{n_\tau})) = \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1}).$$

Equivalently, $\mathcal{F}$ is Markovian if $\forall s \in \mathcal{S} \setminus \{s_0\}$ there exist distributions $\mathcal{P}_{\mathrm{B}}(-|s)$ over the parents of $s$ such that $\forall x \in \mathcal{X}$

$$\mathcal{P}(\tau = (s_0 \to \ldots \to s_{n_\tau})|s_{n_\tau} = x) = \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t),$$

If $\mathcal{F}$ is Markovian, then $\mathcal{P}_{\mathrm{F}}$ and $\mathcal{P}_{\mathrm{B}}$ can be uniquely defined as

$$\mathcal{P}_{\mathrm{F}}(s'|s) = \frac{\mathcal{F}(s \to s')}{\mathcal{F}(s)}, \quad \mathcal{P}_{\mathrm{B}}(s|s') = \frac{\mathcal{F}(s \to s')}{\mathcal{F}(s')}. \tag{2}$$

We call $\mathcal{P}_{\mathrm{F}}$ and $\mathcal{P}_{\mathrm{B}}$ the **forward policy** and the **backward policy**

# Trajectory and detailed balance

*The detailed balance constraint* is a relation between them:

$$\mathcal{F}(s)\mathcal{P}_{\mathrm{F}}(s'|s) = \mathcal{F}(s')\mathcal{P}_{\mathrm{B}}(s|s'). \tag{3}$$

*The trajectory balance constraint* states that for any complete trajectory

$$\prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1}) = \frac{\mathcal{F}(s_{n_\tau})}{Z_{\mathcal{F}}} \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t). \tag{4}$$

Thus any Markovian flow can be uniquely determined from $Z_{\mathcal{F}}$ and $\mathcal{P}_{\mathrm{F}}(-|s)\ \forall s \in \mathcal{S} \setminus \mathcal{X}$ or from $\mathcal{F}(x)\ \forall x \in \mathcal{X}$ and $\mathcal{P}_{\mathrm{B}}(-|s)\ \forall s \in \mathcal{S} \setminus \{s_0\}$.

# Training GFlowNet

To train GFlowNet we parameterize a Markovian flow by some model and train it to minimize some objective. If the objective is globally minimized, we should have $\mathcal{F}(x) = \mathcal{R}(x)$.

*Trajectory Balance* (TB): for each complete trajectory $\tau = (s_0 \to s_1 \to \ldots \to s_{n_\tau} = x)$:

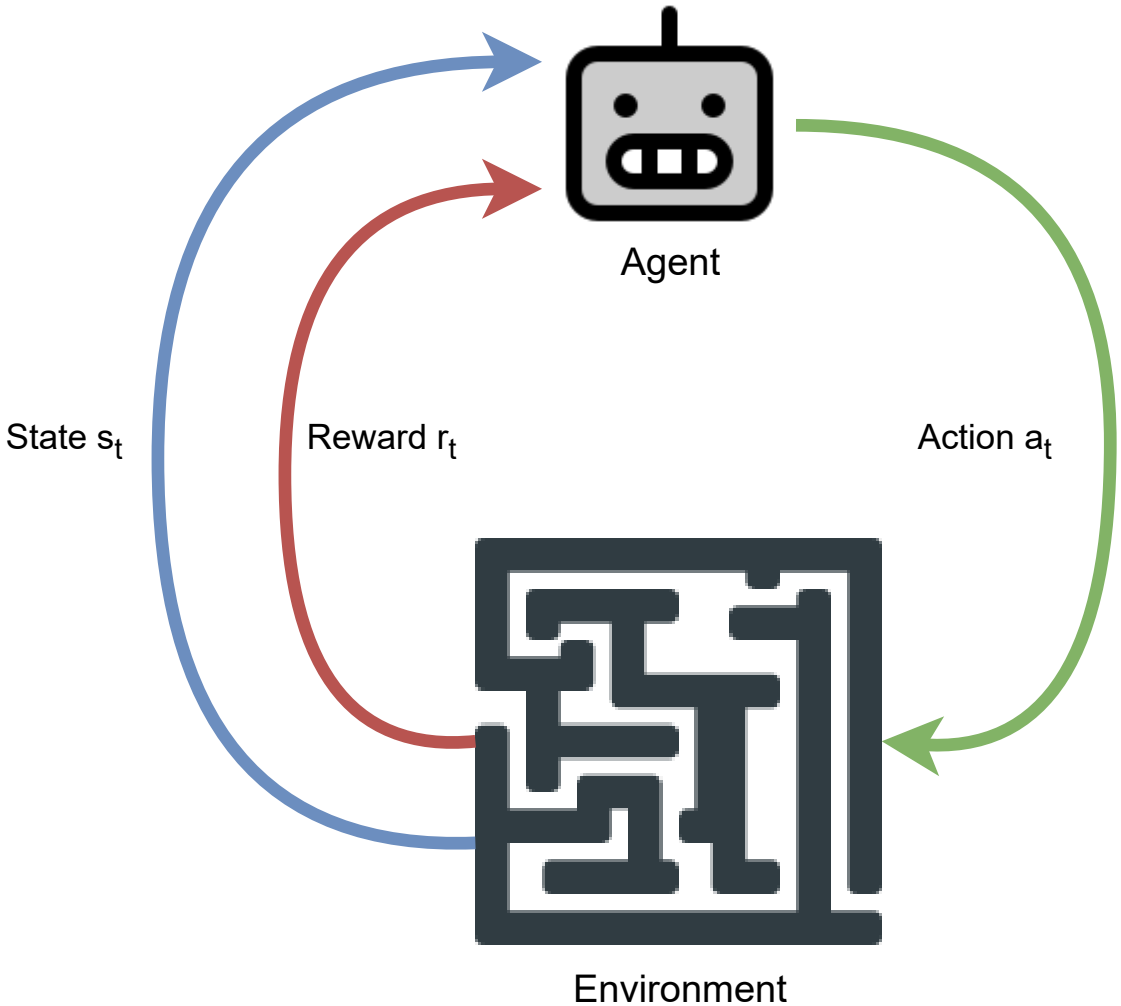
$$\mathcal{L}_{\mathrm{TB}}(\tau) = \left(\log \frac{Z_\theta \prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{F}}(s_t|s_{t-1},\theta)}{\mathcal{R}(x)\prod_{t=1}^{n_\tau} \mathcal{P}_{\mathrm{B}}(s_{t-1}|s_t,\theta)}\right)^2. \tag{5}$$

*Detailed Balance* (DB): for each edge $s \to s' \in \mathcal{E}$:

$$\mathcal{L}_{\mathrm{DB}}(s,s') = \left(\log \frac{\mathcal{F}_\theta(s)\mathcal{P}_{\mathrm{F}}(s'|s,\theta)}{\mathcal{F}_\theta(s')\mathcal{P}_{\mathrm{B}}(s|s',\theta)}\right)^2. \tag{6}$$

For terminal states $\mathcal{F}_\theta(x)$ is substituted by $\mathcal{R}(x)$ in the objective.

# Reinforcement Learning



# Bridging tabular and Deep RL

**Tabular setting:** $S \approx 100$;

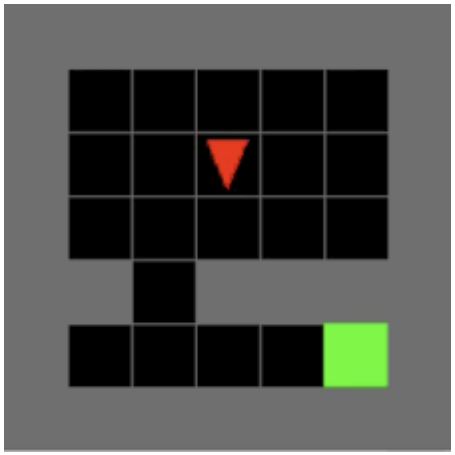


Figure: Minigrid environment

**Deep RL setting:** $S \approx 10^{100}$;


Figure: Atari Breakout

*Is there an algorithm that at the same time*

- provably optimal in tabular setting?
- practically good in Deep RL setting?

# Soft Reinforcement learning

**Markov Decision Process (MDP)**: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathrm{P}, r, \gamma, s_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are finite state and action space, $\mathrm{P}$ is a Markovian transition kernel, $r$ is a bounded deterministic reward function, $\gamma \in [0, 1]$ is a discount factor, and $s_0$ is a fixed initial state. At step $t$

- state $s_t \in \mathcal{S}$;
- action $a_t \in \mathcal{A}$;
- next state $s_{t+1} \sim \mathrm{P}(\cdot|s_t, a_t)$;
- reward $r(s_t, a_t)$ - known.

An RL agent interacts with MDP by a policy $\pi$ that maps each state $s$ to a probability distribution over possible actions.

# Entropy-Regularized RL

**Performance measure:**

$$V_\lambda^\pi(s) \triangleq \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \lambda \mathcal{H}(\pi(s_t))) | s_0 = s \right], \tag{7}$$

where $\lambda$ is a regularization coefficient, $\mathcal{H}$ is a Shannon entropy function, $a_t \sim \pi(s_t), s_{t+1} \sim \mathrm{P}(s_t, a_t)$ for all $t \geq 0$. In a similar manner we can define regularized Q-values $Q_\lambda^\pi(s, a)$ as an expected (discounted) sum of rewards augmented by Shannon entropy given fixed initial state $s_0 = s$ and action $a_0 = a$.

**Goal**: find a **regularized optimal policy** $\pi_\lambda^\star$ that maximizes $V_\lambda^\pi(s)$ for any initial state $s$. In the case $\lambda = 1$ we will drop a subscript $\lambda$ for brevity.

# Soft Bellman Equations

Let $V^\star_\lambda$ and $Q^\star_\lambda$ be a value and Q-value of an optimal policy $\pi^\star_\lambda$ correspondingly. Then for any $s \in \mathcal{S}, a \in \mathcal{A}_s$

$$\begin{aligned} Q^\star_\lambda(s,a) &\triangleq r(s,a) + \gamma \mathbb{E}_{s' \sim \mathrm{P}(s,a)}[V^\star_\lambda(s')] \\ V^\star_\lambda(s) &\triangleq \mathrm{LogSumExp}_\lambda(Q^\star_\lambda(s,\cdot)), \end{aligned} \tag{8}$$

where $\mathrm{LogSumExp}_\lambda \triangleq \lambda \log(\sum_{i=1}^d \mathrm{e}^{x_i/\lambda})$, and an optimal policy is defined as a $\lambda$-softmax policy with respect to optimal Q-values $\pi^\star_\lambda(s) = \mathrm{Softmax}_\lambda(Q^\star_\lambda(s,\cdot))$, where $[\mathrm{Softmax}_\lambda(x)]_i \triangleq \mathrm{e}^{x_i/\lambda}/(\sum_{j=1}^d \mathrm{e}^{x_j/\lambda})$, or, alternatively

$$\pi^\star_\lambda(a|s) = \exp\big(\tfrac{1}{\lambda}(Q^\star_\lambda(s,a) - V^\star_\lambda(s))\big). \tag{9}$$

Notice that as $\lambda \to 0$ we recover the well-known Bellman equations.

# Soft Deep Q-Network

The goal of `SoftDQN`: approximate a solution to soft Bellman equations (8) using a Q-value parameterized by a neural network.

Let $Q_\theta$ be an online Q-network, it represents the current approximation of optimal regularized Q-value, and let $\pi_\theta = \mathrm{Softmax}_\lambda(Q_\theta)$. Then, akin to `DQN`, agent interacts with MDP using policy $\pi_\theta$ (with an additional $\varepsilon$-greedy exploration or even without it), collect transitions $(s_t, a_t, r_t, s_{t+1})$ for $r_t = r(s_t, a_t)$ in a replay buffer $\mathcal{B}$.

Then `SoftDQN` performs stochastic gradient descent on the regression loss $\mathcal{L}_{\texttt{SoftDQN}}(\mathcal{B}) = \mathbb{E}_{\mathcal{B}}[(Q_\theta(s_t, a_t) - y_{\texttt{SoftDQN}}(r_t, s_{t+1}))^2]$ with a target Q-value

$$y_{\texttt{SoftDQN}}(r_t, s_{t+1}) = r_t + \gamma \mathrm{LogSumExp}_\lambda(Q_{\bar{\theta}}(s_{t+1}, \cdot)), \tag{10}$$

where $Q_{\bar{\theta}}$ is a target Q-network with parameters $\bar{\theta}$ that are periodically copied from parameters of online Q-network $\theta$.

# GFlowNet Training as Soft RL Problem

Our main result is the reduction of GFlowNet learning problem to Soft RL:

- Construct a deterministic MDP based on GFlowNet DAG. Add an absorbing state $s_f$ and add edges from all terminal states to it.
- For any $s$ we select $\mathcal{A}_s \subseteq \mathcal{S}$ as a set of actions in state $s$ in $\mathcal{S} \setminus \mathcal{X}$, and define $\mathcal{A}_x \triangleq \{s_f\}$ for any $x \in \mathcal{X} \cup \{s_f\}$.
- Deterministic Markov kernel is simple: $\mathrm{P}(s'|s,a) = \mathbb{1}\{s' = a\}$ since $a$ corresponds to a next state $\mathcal{S}$ given a corresponding action. In the sequel we will write $s'$ instead of $a$ to emphasize this connection.
- Reward function

$$r(s,s') = \begin{cases} \log \mathcal{P}_{\mathrm{B}}(s|s') & s \notin \mathcal{X} \cup \{s_f\}, \\ \log \mathcal{R}(s) & s \in \mathcal{X}, \\ 0 & s = s_f \end{cases} \tag{11}$$

- Set $\gamma = 1$.

# GFlowNet Training as Soft RL Problem

**Theorem**

*The optimal policy* $\pi^\star(s'|s)$ *for a regularized MDP with a coefficient* $\lambda = 1$ *is equal to* $\mathcal{P}_\mathrm{F}(s'|s)$ *for all* $s \in \mathcal{S} \setminus \mathcal{X}, s' \in \mathcal{A}_s$*;*

*Moreover,* $\forall s \neq s_f, s' \neq s_f$ *regularized optimal value* $V^\star(s)$ *and Q-value* $Q^\star(s, s')$ *coincide with* $\log \mathcal{F}(s)$ *and* $\log \mathcal{F}(s \to s')$ *respectively, where* $\mathcal{F}$ *is the Markovian flow defined by* $\mathcal{P}_\mathrm{B}$ *and the GFlowNet reward* $\mathcal{R}$*.*

# Soft DQN in the realm of GFlowNets

During the interaction, the agent collects transitions $(s_t, a_t, r_t, s_{t+1})$ into the replay buffer $\mathcal{B}$ and optimizes a regression loss $\mathcal{L}_{\texttt{SoftDQN}}$ using targets (10) with rewards $r_t = \log \mathcal{P}_{\mathrm{B}}(s_t|s_{t+1})$ for any $s_t \notin \mathcal{X}$. For terminal transitions with $s_t \in \mathcal{X}$ and $s_{t+1} = s_f$ the target is equal to $\log \mathcal{R}(s_t)$.

Using an interpretation $Q_\theta(s, s') = \log \mathcal{F}_\theta(s \to s')$ we obtain the following loss

$$\mathcal{L}_{\texttt{SoftDQN}}(\mathcal{B}) = \mathbb{E}_{\mathcal{B}}\left[\left(\log \frac{\mathcal{F}_\theta(s_t \to s_{t+1})}{\mathcal{P}_{\mathrm{B}}(s_t|s_{t+1})\mathcal{F}_{\bar{\theta}}(s_{t+1})}\right)^2\right], \tag{12}$$

where $\mathcal{F}_{\bar{\theta}}(x) = \mathcal{R}(x)\ \forall x \in \mathcal{X}$, and otherwise $\mathcal{F}_{\bar{\theta}}(s) \triangleq \sum_{s'} \mathcal{F}_{\bar{\theta}}(s \to s')$ and $\bar{\theta}$ are parameters of a target network that is updated with weights $\theta$ from time to time.

# Interpretation of Value

For a policy $\pi$ in a deterministic MDP $\mathcal{M}_{\mathcal{G}}$ with all trajectories of length $n_\tau \leq N$ we can define induced trajectory distribution as follows

$$q^\pi(\tau) = \prod_{i=1}^{N} \pi(s_i|s_{i-1}) = \prod_{i=1}^{n_\tau} \pi(s_i|s_{i-1}),$$

where $\tau = (s_0 \to \ldots \to s_N)$ with $s_N = s_f$ and $s_{n_\tau} \in \mathcal{X}$. The last identity holds since $\pi(s_f|s_f) = \pi(s_f|x) = 1$ for any $x \in \mathcal{X}$. Additionally, we defined a marginal distribution of sampling by policy $\pi$ as $d^\pi(x) = \mathbb{P}_\pi[s_{n_\tau} = x]$.

Additionally, for a fixed GFlowNet reward $\mathcal{R}$ and a backward policy $\mathcal{P}_\mathrm{B}$ we define the corresponding distribution over trajectories with a slight abuse of notation

$$\mathcal{P}_\mathrm{B}(\tau) = \prod_{i=1}^{n_\tau} \mathcal{P}_\mathrm{B}(s_{i-1}|s_i) \cdot \frac{\mathcal{R}(s_{n_\tau})}{\mathrm{Z}}.$$

Notice that by trajectory balance constraint (4) $\mathcal{P}_\mathrm{B}(\tau)$ is equal to a $\mathcal{P}(\tau)$ and, by Theorem 1, to $q^{\pi^\star}(\tau)$.

# Interpretation of Value

Then for any policy $\pi$ its regularized value $V^\pi$ in the initial state $s_0$ is equal to

$$V^\pi(s_0) = \log \mathrm{Z} - \mathrm{KL}(q^\pi \| \mathcal{P}_\mathrm{B}).$$

**Takeaway:** Maximization of the value in the prescribed MDP $\mathcal{M}$ is equivalent to minimization the KL-divergence between trajectory distribution induced by the current policy and by a backward policy.

Moreover, it shows that all near-optimal policies in soft RL-sense introduce a flow with close trajectory distributions, and, moreover, close marginal distribution:

$$V^\star(s_0) - V^\pi(s_0) = \mathrm{KL}(q^\pi \| \mathcal{P}_\mathrm{B}) \geq \mathrm{KL}(d^\pi \| \mathcal{R}(x)/\mathrm{Z}),$$

# Hypergrid Environment

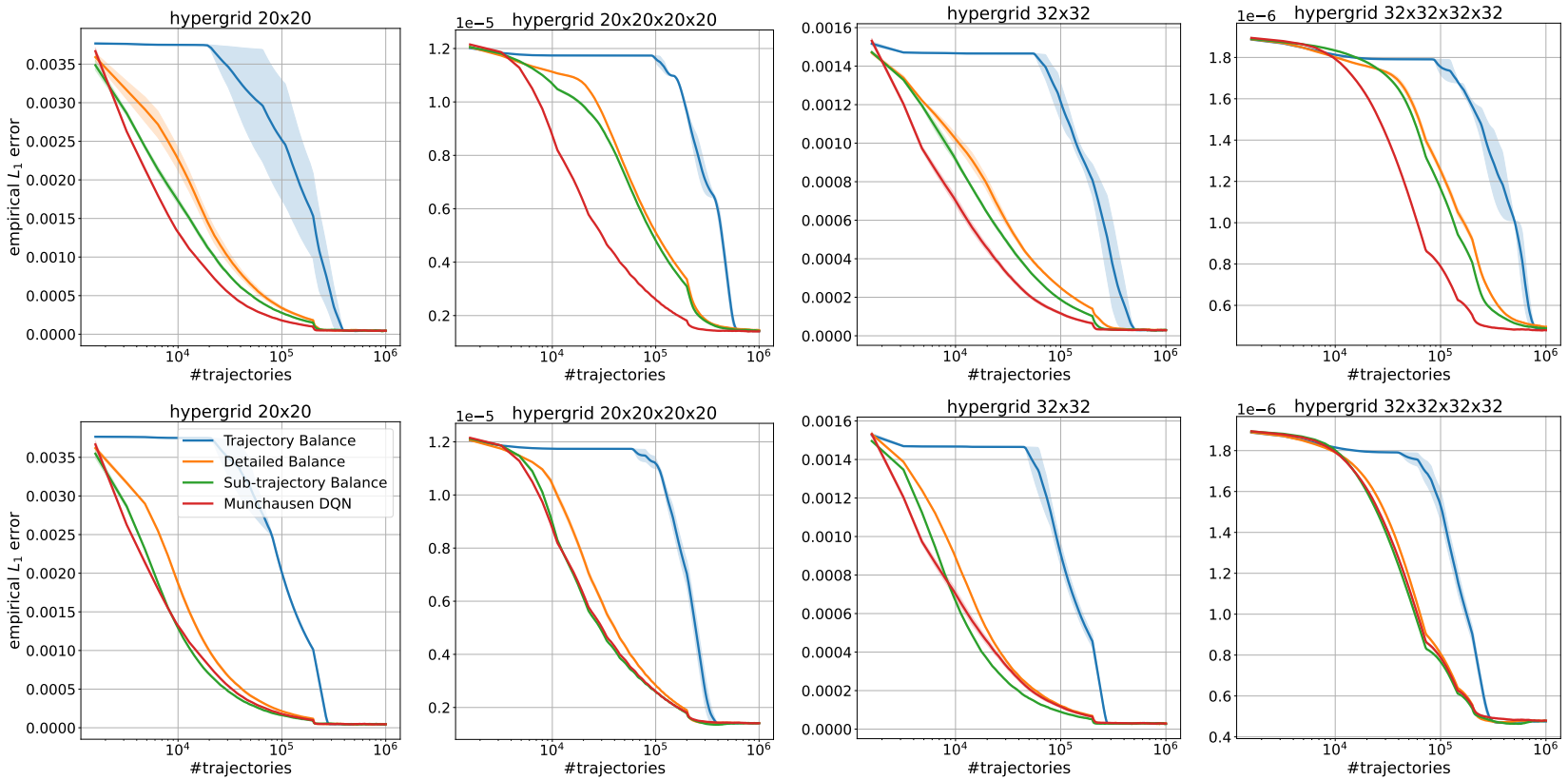



Figure: $L_1$ distance between target and empirical GFlowNet distributions over the course of training on hypergrid environment. *Top row:* $\mathcal{P}_{\mathrm{B}}$ is fixed to be uniform for all methods. *Bottom row:* $\mathcal{P}_{\mathrm{B}}$ is learnt for the baselines and fixed to be uniform for `M-DQN`. Mean and std values are computed over 3 runs.

# Small Molecule Generation

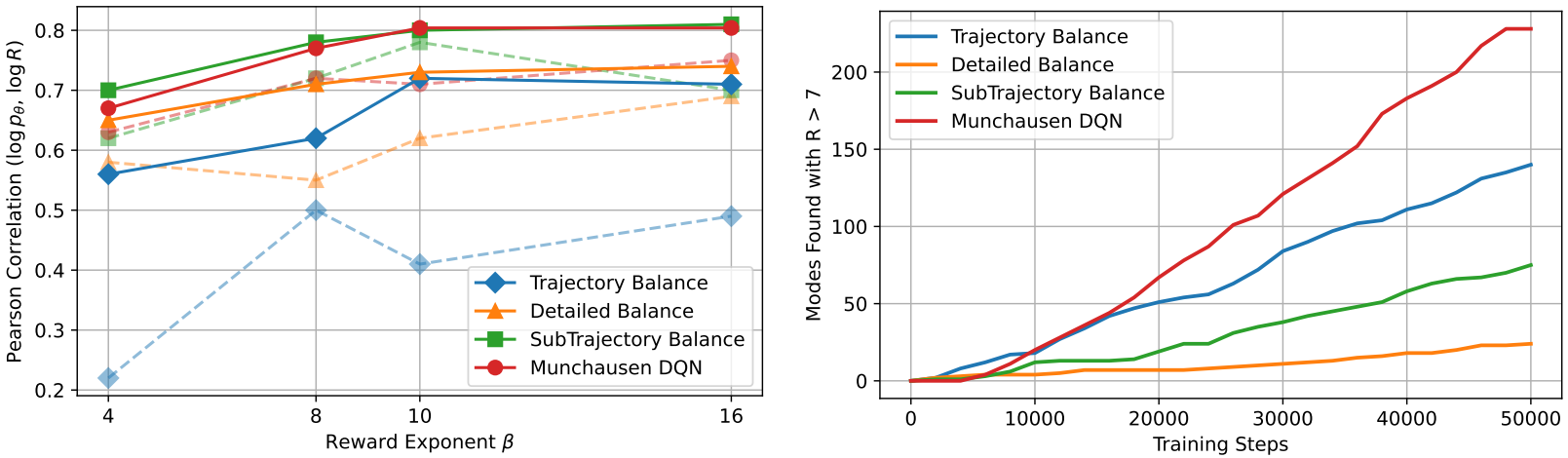
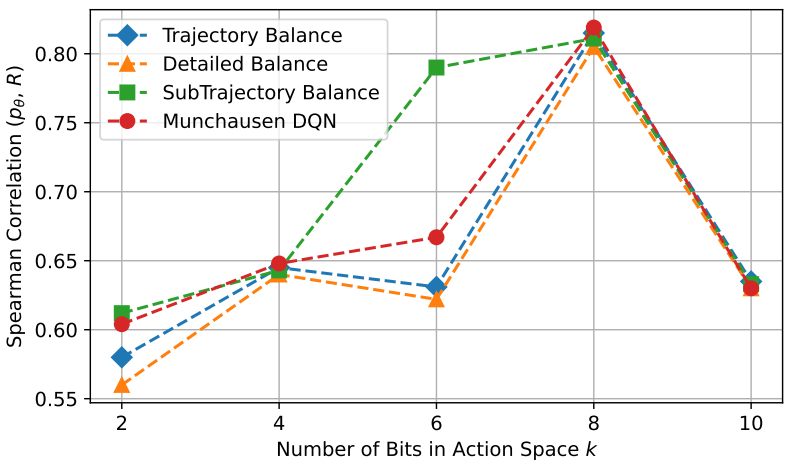



Figure: Small molecule generation results. *Left:* Pearson correlation between $\log \mathcal{R}$ and $\log \mathcal{P}_\theta$ on a test set for each method and varying $\beta \in \{4, 8, 10, 16\}$. Solid lines represent the best results over choices of learning rate, dashed lines — mean results. *Right:* Number of Tanimoto-separated modes with $\tilde{\mathcal{R}} > 7.0$ found over the course of training for $\beta = 10$.

# Bit sequence generation



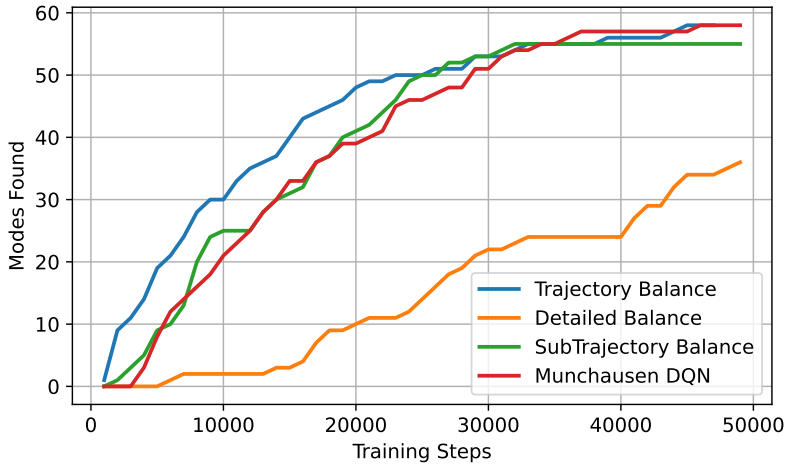




Figure: Bit sequence generation results. *Left:* Spearman correlation between $\mathcal{R}$ and $\mathcal{P}_\theta$ on a test set for each method and varying $k \in \{2, 4, 6, 8, 10\}$. *Right:* The number of modes discovered over the course of training for $k = 8$.

# Problem

We can interpret chain-of-thought reasoning, a paradigm of reasoning in language models, as a problem of intractable posterior inference. Given a question-answer pair $(X, Y)$, we are interested in finding latent chains of thought – token sequences $Z$ that contribute the most to the conditional likelihood

$$p(Y|X) = \sum_Z p_{LM}(ZY|X) = \sum_Z p_{LM}(Y|XZ)p_{LM}(Z|X),$$

where $p_{LM}$ denotes the likelihood assigned to a sequence by a language model and apposition of variables (e.g., $XZY$) denotes the concatenation of the token sequences. We are interested in sampling from

$$p(Z|X, Y) = \frac{p_{LM}(XZY)}{\sum_{Z'} p_{LM}(XZ'Y)}$$

# Amortizing intractable inference in large language models

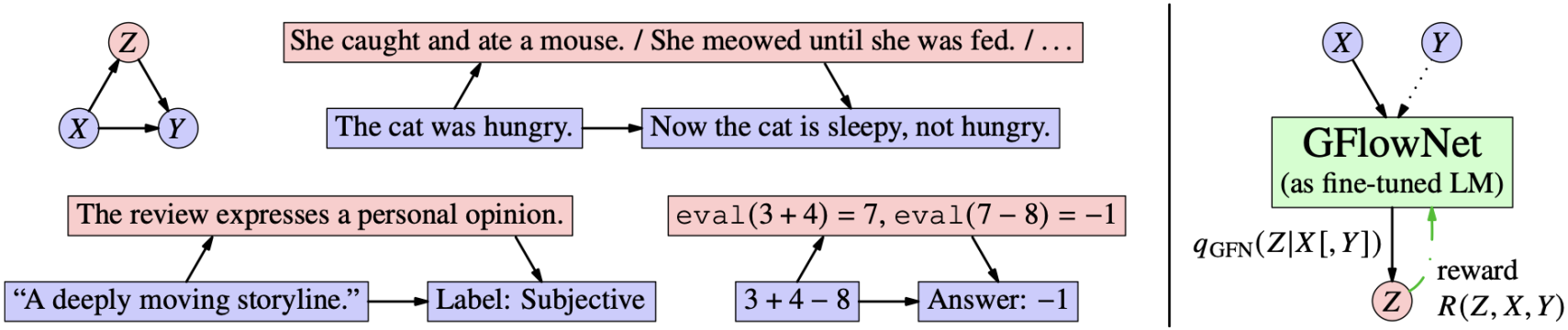



Figure 1: **Left:** Three problems of reasoning in language – sentence infilling, chain-of-thought reasoning, and problem-solving with external tool use – can all be seen as instances of the latent variable model at the top left, where an input ($X$) generates the output ($Y$) via a latent variable ($Z$). **Right:** We fine-tune an LLM to sample from the Bayesian posterior over $Z$, conditioning on $X$ and optionally on $Y$. If conditioned on $Y$, the trained policy can be used to sample diverse latent sequences (*e.g.*, for infilling, §4.2). If not conditioned on $Y$, the policy can sample $Z$, and thus predict $Y$, for inputs $X$ not seen during training (*e.g.*, for classification and multi-step reasoning, §4.3, 4.4). As shown in §4.4, modeling the full diversity of the posterior aids generalization.

Figure: See https://arxiv.org/pdf/2310.04363.pdf

Thank you!

# Bibliography I


Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). Flow network based generative models for non-iterative diverse candidate generation.
Advances in Neural Information Processing Systems, 34:27381–27394.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. (2023).
Gflownet foundations.
Journal of Machine Learning Research, 24(210):1–55.

Samsonov, S., Lagutin, E., Gabrié, M., Durmus, A., Naumov, A., and Moulines, E. (2022).
Local-global mcmc kernels: the best of both worlds.
In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, Advances in Neural Information Processing Systems, volume 35, pages 5178–5193. Curran Associates, Inc.