Maximum Entropy Reinforcement Learning via Energy-Based Normalizing Flow

Chen-Hao Chao*[1,2], **Chien Feng***[1], **Wei-Fang Sun**[2], **Cheng-Kuang Lee**[2], **Simon See**[2], and **Chun-Yi Lee**[1]

(* equal contribution)

[1] *Elsa Lab, National Tsing Hua University, Hsinchu City, Taiwan*

[2] *NVIDIA AI Technology Center, NVIDIA Corporation, Santa Clara, CA, USA*

TL'DR

We introduce a novel Maximum Entropy (MaxEnt) Reinforcement Learning (RL) framework that supports:

- Single training loss optimization process
- Exact soft value calculation
- Multi-modal action distribution modeling

Our method achieves superior performance on:

- the MuJoCo benchmark
- the Omniverse Isaac Gym Environments

compared to widely-adopted baselines (e.g., SAC [1]).

Github Paper Website

Background

Maximum Entropy Reinforcement Learning

MaxEnt RL augments the standard RL objective with the entropy of a policy as follows:

$$\pi^*_{\mathrm{MaxEnt}} = \underset{\pi}{\mathrm{argmax}} \sum_t \mathbb{E}_{\rho_\pi}[r_t + \alpha\mathcal{H}(\pi(\cdot\,|s_t))], \quad (1)$$

where $\mathcal{H}$ is the entropy and α is a temperature parameter. The solution π^*_{MaxEnt} can be expressed as:

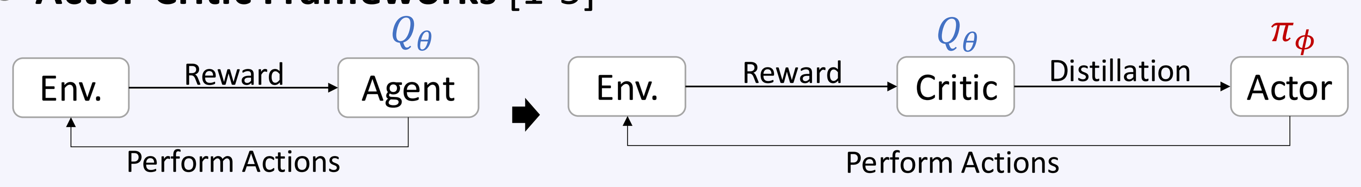
$$\pi^*_{\mathrm{MaxEnt}}(\boldsymbol{a}_t|\boldsymbol{s}_t) = \exp\left(\frac{1}{\alpha}\left(Q^*_{\mathrm{soft}}(\boldsymbol{s}_t,\boldsymbol{a}_t) - V^*_{\mathrm{soft}}(\boldsymbol{s}_t)\right)\right), \quad (2)$$

where $Q^*_{\mathrm{soft}}(\boldsymbol{s}_t,\boldsymbol{a}_t)$ is the optimal soft Q-function and $V^*_{\mathrm{soft}}(\boldsymbol{s}_t) = \alpha \log \int \exp\left(\frac{1}{\alpha} Q^*_{\mathrm{soft}}(\boldsymbol{s}_t,\boldsymbol{a})\right) d\boldsymbol{a}$. Given an experience reply buffer $\mathcal{D}$, the objective of Q_θ is written as:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t) - (r_t + \gamma V_\theta(\boldsymbol{s}_{t+1}))\right)^2\right]. \quad (3)$$

Actor-Critic Designs

- **Two Key Challenges of MaxEnt RL with Conti. $\boldsymbol{a}_t$**
  - Inefficient sampling using $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) \propto \exp(Q_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t))$
  - Intractable integration operation in $V_\theta(\boldsymbol{s}_t)$
- **Actor-Critic Frameworks** [1-5]



- Actor Loss (Reverse KL Divergence $\mathbb{D}_{KL}[\pi_\phi||\pi_\theta]$):

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathcal{D},\pi_\phi}\left[-(Q_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t) - \alpha \log \pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t))\right] \quad (4)$$

- Soft Value Estimation (Monte Carlo Estimation):
- Soft Q-Learning (SQL) [1]: $V_{\theta,\phi}(\boldsymbol{s}_t) \approx \alpha \log \frac{1}{M}\sum_{i=1}^{M}\left(\frac{\exp(Q_\theta(\boldsymbol{s}_t,\boldsymbol{a}^i)/\alpha)}{\pi_\phi(\boldsymbol{a}^i|\boldsymbol{s}_t)}\right)$ (5)
- Soft Actor-Critic (SAC) [2]: $V_{\theta,\phi}(\boldsymbol{s}_t) \approx \frac{1}{M}\sum_{i=1}^{M}(Q_\theta(\boldsymbol{s}_t,\boldsymbol{a}^i) - \alpha \log \pi_\phi(\boldsymbol{a}^i|\boldsymbol{s}_t))$ (6)

Training Process (Vanilla)

For each training iter.:
1. **Interact with env.**
   $\boldsymbol{a}_t \sim \pi_\theta(\cdot\,|\boldsymbol{s}_t)$
   $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{s}_t,\boldsymbol{a}_t,r_t,\boldsymbol{s}_{t+1})$
2. **Update** θ
   $(\boldsymbol{s}_t,\boldsymbol{a}_t,r_t,\boldsymbol{s}_{t+1}) \sim \mathcal{D}$
   $\theta \leftarrow \theta + \nabla_\theta \mathcal{L}(\theta)$

Training Process (Actor-Critic)

For each training iter.:
1. **Interact with env.**
   $\boldsymbol{a}_t \sim \pi_\phi(\cdot\,|\boldsymbol{s}_t)$ ⬅ (a)
   $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{s}_t,\boldsymbol{a}_t,r_t,\boldsymbol{s}_{t+1})$
2. **Update** θ
   $(\boldsymbol{s}_t,\boldsymbol{a}_t,r_t,\boldsymbol{s}_{t+1}) \sim \mathcal{D}$
   Estimate $V_\theta(\boldsymbol{s}_t)$ ⬅ (b)
   $\theta \leftarrow \theta + \nabla_\theta \mathcal{L}(\theta)$
3. **Update** ϕ
   For each distillation iter.: ⬅ (c)
   $\phi \leftarrow \phi + \nabla_\phi \mathcal{L}(\phi)$

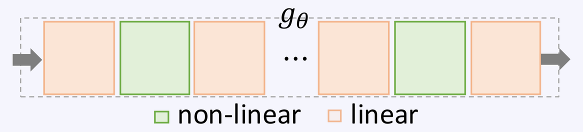
Energy-based Normalizing Flows

- **Normalizing Flows**

The probability density functions (pdf) p_θ are parameterized using a prior distribution $p_{\mathbf{u}}$ of a variable $\mathbf{u}$ and an invertible mapping $g = g_L \circ \cdots \circ g_1$, where $g^i_\theta: \mathbb{R}^D \to \mathbb{R}^D, i \in \{1,\ldots,L\}$. Based on the change of variable theorem, p_θ can be expressed as:

$$p_\theta(\mathbf{x}) = p_{\mathbf{u}}(g_\theta(\mathbf{x})) \prod_{i=1}^{L} \left|\det(\boldsymbol{J}_{g^i_\theta}(\mathbf{x}^{i-1}))\right|, \quad (7)$$

where and $\boldsymbol{J}_{g_i}$ represents the Jacobian of g_i.



- **Energy-based Normalizing Flows (EBFlow)** [6]

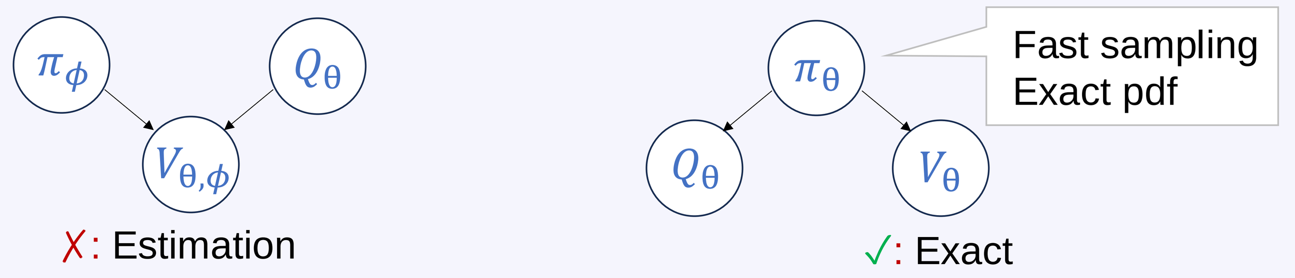
Let $\mathcal{S}_n$ and $\mathcal{S}_l$ be the index sets of non-linear and linear layers in g_θ. EBFlow reinterprets Eq. (7) as a Boltzmann distribution and factorizes it as follows:

$$p_\theta(\mathbf{x}) = \underbrace{p_{\mathbf{u}}(g_\theta(\mathbf{x})) \prod_{i\in\mathcal{S}_n} \left|\det(\boldsymbol{J}_{g^i_\theta}(\mathbf{x}^{i-1}))\right|}_{\triangleq\, \exp(-E_\theta(\mathbf{x}))} \underbrace{\prod_{i\in\mathcal{S}_l} \left|\det(\boldsymbol{J}_{g^i_\theta})\right|}_{\triangleq\, Z_\theta^{-1}} \quad (8)$$

Methodology

MaxEnt RL via EBFlow (MEow) Framework

- **Previous:** Model Q_θ and π_ϕ and derive $V_{\theta,\phi}$
- **Ours:** Model π_θ and extract Q_θ and V_θ



The MEow framework parameterizes the policy as a state-conditioned EBFlow:

$$\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) = \underbrace{p_{\mathbf{u}}(g_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)) \prod_{i\in\mathcal{S}_n} \left|\det(\boldsymbol{J}_{g^i_\theta}(\boldsymbol{a}^{i-1}_t|\boldsymbol{s}_t))\right|}_{\triangleq\, \exp\left(\frac{1}{\alpha}Q_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t)\right)} \underbrace{\prod_{i\in\mathcal{S}_l} \left|\det(\boldsymbol{J}_{g^i_\theta}(\boldsymbol{s}_t))\right|}_{\triangleq\, \exp\left(-\frac{1}{\alpha}V_\theta(\boldsymbol{s}_t)\right)} \quad (9)$$

Proposition 1. *Eq. (9) satisfies the following statements:*

(i) Given that the Jacobian of g_θ is non-singular, $V_\theta(\boldsymbol{s}) \in \mathbb{R}$ and $Q_\theta(\boldsymbol{s},\boldsymbol{a}) \in \mathbb{R}$.

(ii) $V_\theta(\boldsymbol{s}) = \alpha \log \int \exp\left(\frac{1}{\alpha} Q_\theta(\boldsymbol{s},\boldsymbol{a})\right) d\boldsymbol{a}$.

- **Expressivity**
  - (✓) Normalizing flows are universal approximators [7]
  - (✓) Q_θ and V_θ can represent any real number (i.e., Proposition 1)
- **Training**
  - (✗ → ✓) Single training loss optimization process (i.e., Eq. (3))
  - (✗ → ✓) Exact soft value calculation (i.e., Eq. (9))
- **Inference**
  - (✗ → ✓) Consistent policy for training and inference (i.e., π_θ)

Deterministic Inference Technique of MEow

Proposition 2. Given that the Jacobians are constants w.r.t. the input, then:

$$\mathrm{argmax}_{\boldsymbol{a}_t}\ Q_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t) = g_\theta^{-1}(\mathrm{argmax}_{\boldsymbol{u}}\, p_{\mathbf{u}}(\boldsymbol{u})\,|\boldsymbol{s}_t). \quad (10)$$

- **Practical Implementation**
  - Adopt NICE-like [10] architecture for π_θ
  - Using the mean $\boldsymbol{\mu}$ of Gaussian (i.e., $g_\theta^{-1}(\boldsymbol{\mu}|\boldsymbol{s}_t)$) for inference.

Experiments

- **Setups**
  - **Environments:** Multi-goals [1], MuJoCo [8], Isaac [9]
  - **Architecture:** Additive coupling layers [10]; Element-wise linear layer
- **Baselines**
  - SAC
  - SQL
  - PPO
  - DDPG
  - TD3

MuJoCo Benchmark

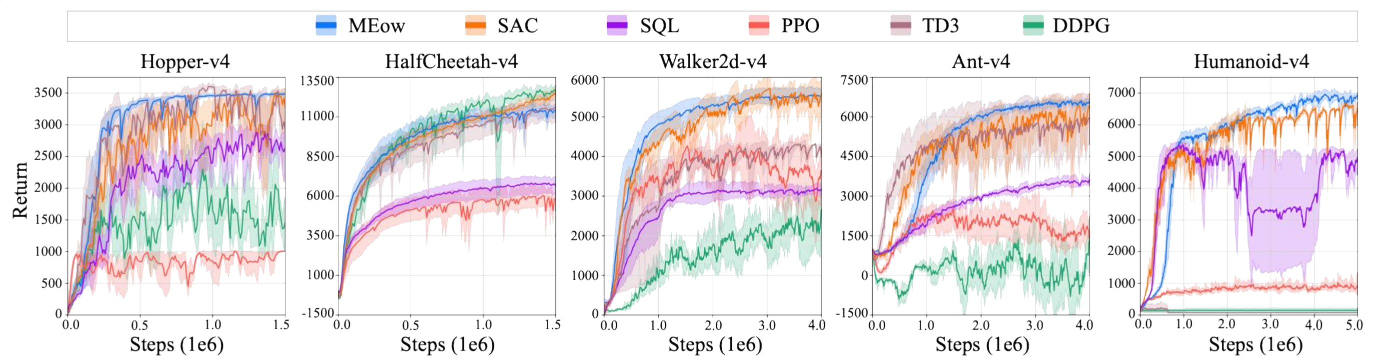


Figure 2. The results in terms of total returns versus the number of training steps evaluated on five MuJoCo environments. Each curve represents the mean performance, with shaded areas indicating the 95% confidence intervals, derived from five independent runs with different seeds.

Multi-Goal Environment

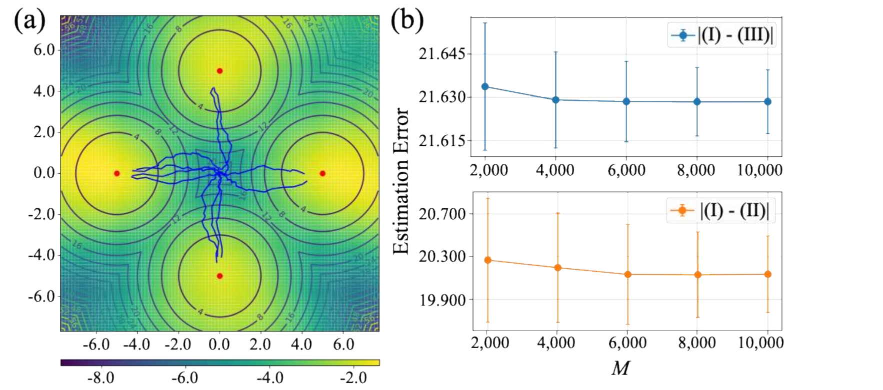


Figure 1. (a) The soft value function and the trajectories generated using our method on the multi-goal environment. (b) The estimation error evaluated at the initial state under different choices of M. In this plot, (I), (II), and (III) represent V_θ in Eq. (9), Eq. (5), and Eq. (6), respectively.

Omniverse Isaac Gym Environments

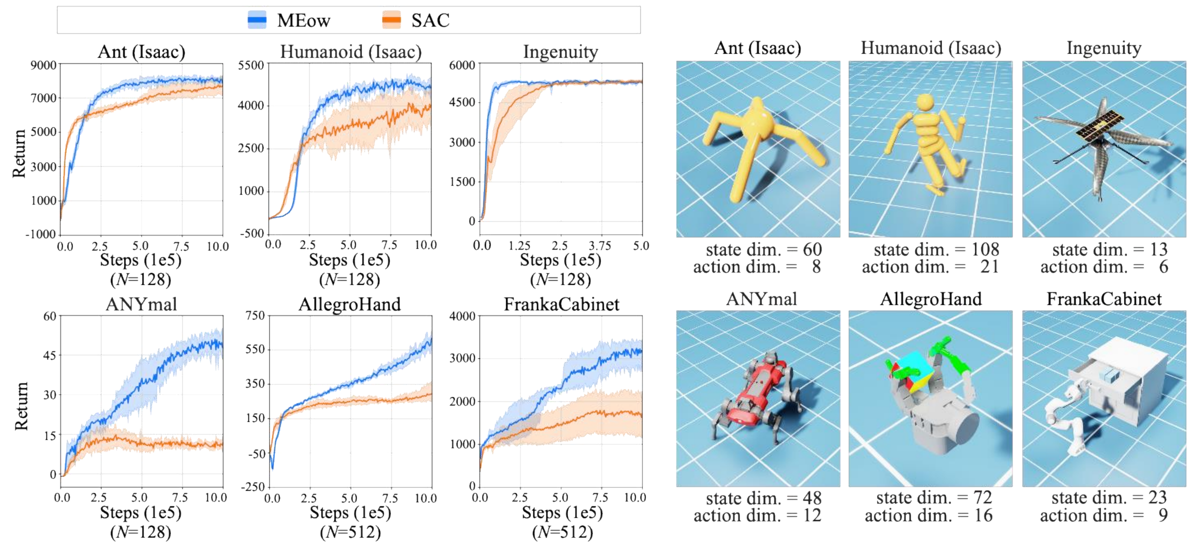


Figure 3 (Left). A comparison of SAC and MEow on six Isaac Gym environments. Each curve represents the mean performance of five runs, with shaded areas indicating the 95% confidence intervals. 'Steps' in the x-axis represents the number of training steps, each of which consists of N parallelizable interactions with the environments.

Figure 3 (Right). A demonstration of six Isaac Gym environments. The dimensionalities of the state and action for each environment are denoted below each subfigure.

References

[1] Haarnoja *et al*. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, *ICML* 2018.
[2] Haarnoja *et al*. Reinforcement Learning with Deep Energy-Based Policies, *ICML* 2017.
[3] Haarnoja *et al*. Latent Space Policies for Hierarchical Reinforcement Learning, *ICML* 2018.
[4] Zhang et al. Latent State Marginalization as a Low-cost Approach for Improving Exploration, ICLR 2023.
[5] Messaoud et al. S2AC: Energy-Based Reinforcement Learning with Stein Soft Actor Critic, ICLR 2024.
[6] Chao et al. Training Energy-Based Normalizing Flow with Score-Matching Objectives, NeurIPS 2023.
[7] Papamakarios et al. Normalizing Flows for Probabilistic Modeling and Inference. JMLR 2019.
[8] Todorov et al. MuJoCo: A physics engine for model-based control. IROS 2012.
[9] Makoviychuk et al. Isaac Gym: High-Performance GPU-Based Physics Simulation For Robot Learning. 2021.
[10] Dinh et al. NICE: Non-linear Independent Components Estimation. ICLR Workshop 2015.

Acknowledgements