Maximum Entropy Reinforcement Learning via Energy-Based Normalizing Flow

Project Presentation - Team 9

Shayan Meshkat Alsadat, Soham Karandikar,
Aman Chandak, and Abhinav V Pillai

Arizona State University

November 23, 2025

Problem Statement

- Existing MaxEnt RL methods for continuous action spaces rely on actor-critic frameworks, leading to complex dual optimization and potential misalignment between actor and critic networks.
- Sampling actions directly from the underlying energy-based model of the soft Q-function is computationally expensive, often necessitating a separate actor network for efficient sampling.
- Calculating the soft value function involves an intractable integral, forcing reliance on Monte Carlo approximations that introduce estimation errors and variance.

Introduction

- **Maximum Entropy RL**: Integrates policy entropy as intrinsic reward
  - Improves exploration-exploitation balance
  - Enhanced robustness in continuous control
- **Key Challenges**:
  - Learning soft Q-function requires costly MCMC sampling
  - Soft value calculation needs computationally infeasible integration
  - Actor-critic separation introduces optimization errors
- **Authors' Solution**: MEow - A unified framework using Energy-Based Normalizing Flows

Background: Maximum Entropy RL

MaxEnt Objective

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right)\right]$$

- Optimal policy expressed via soft Q-function and soft value function:

$$\pi^*(a|s) = \exp\left(\frac{1}{\alpha}(Q^*(s,a) - V^*(s))\right)$$

$$V^*(s) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q^*(s,a)\right) da$$

- **Problem**: Computing $V^*(s)$ requires intractable integration

Two Estimation Methods:

1. **SQL**: Importance sampling

$$V(s) \approx \alpha \log \frac{1}{N} \sum_{i=1}^{N} \exp\left(\frac{1}{\alpha} Q(s, a_i)\right)$$

2. **SAC**: Policy approximation

$$V(s) \approx \mathbb{E}_a[Q(s,a) - \alpha \log p(a|s)]$$

Can we do unified approach with exact computation?!

Energy-Based Normalizing Flows

- **Normalizing Flows**: Invertible transformations for density modeling

$$p(x;\theta) = p_Z(f_\theta^{-1}(x)) \prod_{i=1}^{L} \left| \det \frac{\partial f_{\theta,i}}{\partial f_{i-1}} \right|^{-1}$$

- **Energy-Based View**: Decompose Jacobian by linear/nonlinear layers

$$p(x;\theta) = \frac{1}{Z(\theta)} \exp(-E(x;\theta))$$

where:

  - Energy: $E(x;\theta) = -\log p_Z(f_\theta^{-1}(x)) - \sum_{i \in \mathcal{I}_N} \log \left| \det \frac{\partial f_i}{\partial f_{i-1}} \right|$
  - Normalizing constant: $Z(\theta) = \prod_{i \in \mathcal{I}_L} \left| \det \frac{\partial f_i}{\partial f_{i-1}} \right|$

- **Key Property**: $Z(\theta)$ is independent of input x!

MEow Framework: Core Idea

Is there a way to calculate optimal value function without approximating it with Monte-Carlo methods?

It turns out we could!!

$$V^*(s) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q^*(s,a)\right) da$$

Unified Policy Representation

$$\pi(a|s;\theta) = \frac{1}{Z(s;\theta)} \exp\left(\frac{1}{\alpha} Q(s,a;\theta)\right)$$

Theorem (Exact Soft Value)

$$V(s;\theta) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q(s,a;\theta)\right) da = -\alpha \log Z(s;\theta)$$

- Single training objective (no actor-critic split)
- Exact value computation (no approximation error)
- Efficient sampling via normalizing flow

Contributions: MEow Framework

1. **Unified Policy Representation**
   - Single model for both Q-function and policy
   - Eliminates actor-critic separation
2. **Exact Soft Value Computation**
   - No Monte Carlo approximation needed
   - $V(s) = -\alpha \log Z(s; \theta)$ computed exactly
3. **Enhanced Training Techniques**
   - Learnable Reward Shifting (LRS) for numerical stability
   - Shifting-Based Clipped Double Q-Learning (SCDQ) for overestimation bias
4. **Superior Empirical Performance**
   - MuJoCo benchmarks
   - Omniverse Isaac Gym robotic tasks

Training and Inference

Training Objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')} \left[\left(Q(s,a;\theta) - r - \gamma V(s';\theta') \right)^2 \right]$$

- Single loss function
- Exact $V(s')$ via $-\alpha \log Z(s';\theta')$
- Target network with Polyak averaging

Inference:

- **Stochastic**: Sample $z \sim p_Z(z)$, compute $a = f_\theta(z|s)$
- **Deterministic**:

$$a^*(s) = f_\theta(\mathbb{E}[z]|s)$$

Multi-modal Support:

- Normalizing flows can represent complex, multi-modal action distributions

Training Algorithm

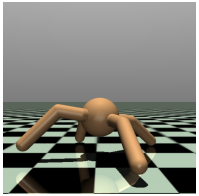
Init: $\theta,\ \theta' \leftarrow \theta,\ \alpha, \gamma, \tau, \beta,\ g_\theta,\ b_\theta^{(1)},\ b_\theta^{(2)},$ buffer $\mathcal{D}$ // Initialize networks, targets, replay buffer

for *each step* **do**

- Sample $z \sim p_z$ // Sample latent noise
- $a_t \leftarrow g_\theta^{-1}(z \mid s_t)$ // Invert flow to get action
- $s_{t+1} \sim p^T(\cdot \mid s_t, a_t)$ // Environment transition
- Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$ // Add to replay
- Sample batch $(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$ // Replay sampling
- $Q_\theta = \alpha \log\left[p_z(g_\theta(a_t \mid s_t)) \prod_{i \in \mathcal{S}_n} |\det(J_{g_\theta^i})|\right]$ // Flow-based Q
- $V_{\theta'} = -\alpha \log \prod_{i \in \mathcal{S}_\ell} |\det(J_{g_{\theta'}^i})|$ // Flow-based V
- $Q_\theta^{(k)} = Q_\theta + b_\theta^{(k)}$ // Twin critics
- $V_{\theta'}^{clip} = V_{\theta'} + \min(b_{\theta'}^{(1)}, b_{\theta'}^{(2)})$ // Clipped target
- $y_t = r_t + \gamma V_{\theta'}^{clip}(s_{t+1})$ // TD target
- $\mathcal{L} = \frac{1}{2}(Q_\theta^{(1)} - y_t)^2 + \frac{1}{2}(Q_\theta^{(2)} - y_t)^2$ // Bellman loss
- $\theta \leftarrow \theta + \beta \nabla_\theta \mathcal{L}$ // Gradient update
- $\theta' \leftarrow (1-\tau)\theta' + \tau\theta$ // Soft update

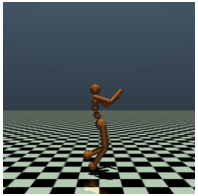
Results: MuJoCo Benchmarks Overview

Ant-v4



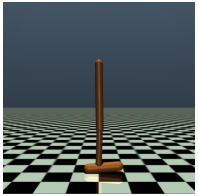
State dim: 27 | Action dim: 8

Humanoid-v4



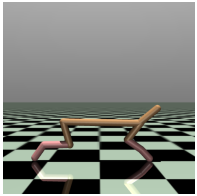
State dim: 376 | Action dim: 17

Hopper-v4



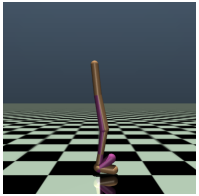
State dim: 11 | Action dim: 3

HalfCheetah-v4



State dim: 17 | Action dim: 6

Walker2d-v4



State dim: 17 | Action dim: 6

Tasks: Ant, Humanoid, Hopper, HalfCheetah, Walker2d

Setup: 1.5M - 5M timesteps depending on task, 5 seeds per method

State/Action Dimensions:

- Hopper: 11/3 (simplest)
- Humanoid: 376/17 (most challenging)

MuJoCo Results: Simpler Tasks

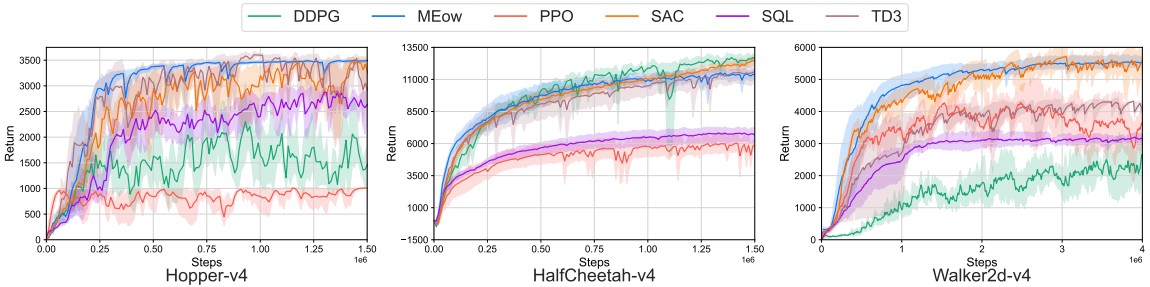


Figure: Results for Hopper-v4, HalfCheetah-v4 and Walker2d-v4.

Key Observations:

- MEow achieves comparable performance to SAC across all three tasks
- Competitive learning speed and stable convergence
- Outperforms DDPG, TD3, and PPO consistently

MuJoCo Results: Simpler Tasks

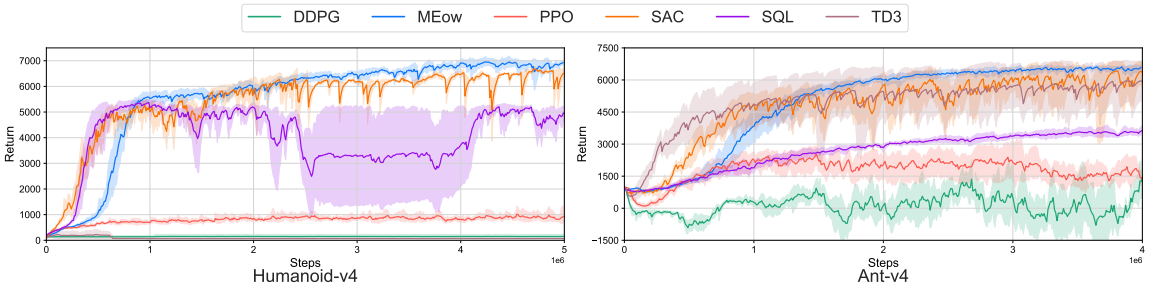


Figure: Results for Humanoid-v4 and Ant-v4.

Key Observations:

- Notably smoother curves with fewer evaluation spikes
- Reduced variance suggests more stable optimization
- Superior sample efficiency and training stability across both high-dimensional environments

Results: Omniverse Isaac Gym - Overview

6 Robotic Tasks: Designed for real-world deployment scenarios

Key Results Summary

- MEow **consistently outperforms SAC** across all tasks
- Substantial improvements in sample efficiency
- Better asymptotic performance
- Smoother learning curves

Isaac Gym: Manipulation Tasks

AllegroHand - Dexterous Manipulation:

- MEow: **550** return vs SAC: 350
- **28% improvement**
- 16 actuators, 88-dimensional state space
- Better handling of **multi-modal action distributions**

FrankaCabinet - Robotic Arm Manipulation:

- MEow: **3000** return vs SAC: 1500
- **29% improvement**
- Contact-rich interactions with cabinet doors
- Exact value estimation particularly beneficial

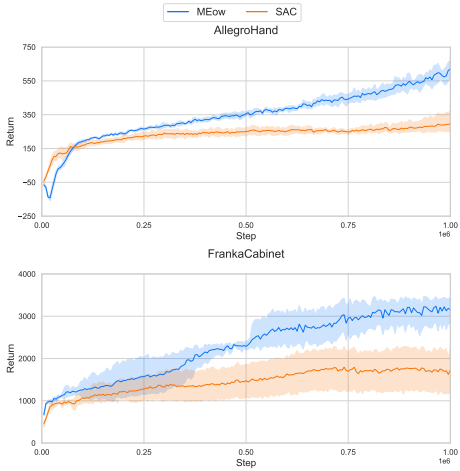


Figure: Learning curves for manipulation tasks

Isaac Gym: Locomotion Tasks

ANYmal - Quadruped Locomotion:

- MEow: **45** return vs SAC: 15
- **17% improvement**
- Notably smoother learning curves
- Stable locomotion emerges **faster**
- Reduced optimization variance

Ingenuity - Helicopter Flight Control:

- MEow: **5000** return vs SAC: 5000
- comparable performance
- Challenging underactuated system
- Performance gap emerges **early in training**
- Better exploration efficiency

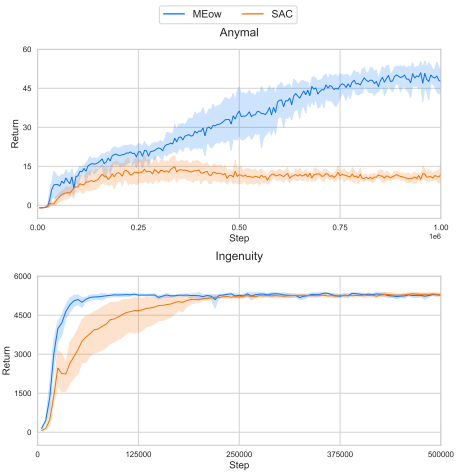


Figure: Learning curves for locomotion tasks

Summary: Bottlenecks and Conclusions

MEow Bottlenecks

- $2\times$ slower than SAC during training
- Slow real-time interaction for high-frequency control
- Sensitive to hyperparameter tuning

Application Challenges:

- Poor generalization to unseen states
- Limited hardware/deployment readiness
- Difficulty scaling to high-dimensional actions

MEow Key Contributions

- Unified objective (no actor–critic split)
- Exact soft value (no MC approximation)
- Improved stability (LRS + SCDQ)
- Strong empirical results

Advantages:

- No actor–critic inconsistency
- Supports multi-modal policies
- Efficient, deterministic sampling

Future Directions:

- Trade-off between expressiveness and speed

Thank You!

Questions?

Original paper: Maximum Entropy Reinforcement Learning via Energy-Based Normalizing Flow, NeurIPS 2024

Contact: {smeshka1, skarand4, achan160, avpilla2}@asu.edu