Team 9 Project Report: Maximum Entropy Reinforcement Learning via Energy-Based Normalizing Flow

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

***Abstract*— We are Team 9, and we address the prompt of maximum entropy reinforcement learning (MaxEnt RL) and how it can be advanced for robotic control. Maximum entropy reinforcement learning is a popular approach for training agents to balance exploration and exploitation, especially in continuous action spaces like robotics. However, current MaxEnt RL methods often rely on separate actor-critic models and require complex sampling and approximation techniques, slowing down training and reducing accuracy. Our discussion centers on a new framework, inspired by the paper "Maximum Entropy Reinforcement Learning via Energy-Based Normalizing Flow" which uses energy-based normalizing flows to unify policy evaluation and improvement into a single training objective. This approach allows for efficient and exact calculation of value functions, supports flexible action distributions, and enables fast action sampling. We use application problems from standard control benchmarks and challenging robotic tasks in MuJoCo and Isaac Gym to facilitate the discussion and demonstrate that this method outperforms existing MaxEnt RL algorithms in performance and stability.**

***Index Terms*— Maximum Entropy Reinforcement Learning, Normalizing Flows, Energy-Based Models, Actor-Critic Methods, Continuous Control**

I. Introduction

Maximum entropy reinforcement learning (MaxEnt RL) has emerged as a powerful paradigm for training agents to balance exploration and exploitation, particularly in continuous control domains such as robotics, autonomous vehicles, and complex simulation environments. The central issue with current MaxEnt RL methods lies in their reliance on separate actor-critic architectures, which require alternating steps of policy evaluation and improvement. Since calculating value functions usually requires Monte Carlo approximations, which can slow down training and introduce variance, and sampling actions from energy-based models frequently necessitate expensive methods, this separation generates major inefficiencies. These drawbacks are particularly significant in practical applications where activities like autonomous driving, adaptive path planning, and robotic manipulation require quick, reliable, and steady learning. MaxEnt RL has been widely used in fields [1] where making decisions under uncertainty is crucial, such as robotic control, vehicle trajectory planning, multi-agent coordination, and even healthcare and finance. Cutting-edge algorithms like Soft Actor-Critic (SAC) and Soft Q-Learning have achieved remarkable results on widely used platforms like MuJoCo and Isaac Gym [2], setting new standards for sample stability and efficiency. However, as task dimensions and complexity increases, the requirement for complex sampling and approximation approaches continues to be a bottleneck.

The approach discussed in this work, inspired by the paper "Maximum Entropy Reinforcement Learning via Energy-Based Normalizing Flow," [3] fundamentally differentiates itself by unifying policy evaluation and improvement into a single training objective using energy-based normalizing flows. This framework enables exact and efficient calculation of value functions, supports flexible and multi-modal action distributions, and allows for fast action sampling without the drawbacks of traditional actor-critic separation. Experimental results on challenging robotic benchmarks demonstrate that this method outperforms existing MaxEnt RL algorithms in terms of performance and stability and also offers a scalable and robust solution for advanced continuous control tasks.

Team members are with School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, Arizona, USA. {smeshka1, skarand4, achan160, avpilla2}@asu.edu. The original paper can be found at here

II. Problem Formulation

In reinforcement learning, agents must learn robust and diverse strategies for complex environments. Maximum entropy reinforcement learning (MaxEnt RL) enhances exploration through stochastic policies, making agents more adaptable. However, current MaxEnt RL methods face key inefficiencies: actor-critic frameworks alternate between policy evaluation and improvement, sampling from energy-based models is computationally expensive, and Monte Carlo value function estimation introduces variance and errors. The authors propose a unified framework using energy-based normalizing flows (EBFlow) that integrates policy evaluation and improvement into a single objective, enabling exact value calculation and efficient sampling. To validate this approach, they use the humanoid locomotion problem, where a bipedal robot with 21 action dimensions and 108 state dimensions learns stable walking behaviors. This high-dimensional task requires coordinated movement across many joints while maintaining balance. Experiments on MuJoCo Humanoid-v4 and Isaac Gym Humanoid environments, Fig. 1, demonstrate that the MEow framework achieves superior performance compared to state-of-the-art baselines.

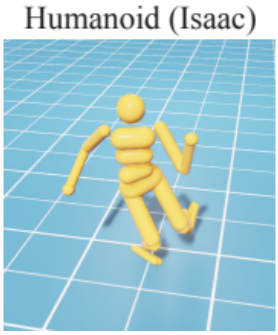

Fig. 1: A demonstration of Humanoid robot learning to walk in Issac Gym enviornment.

III. Background and Preliminaries

A. Maximum Entropy Reinforcement Learning

The manuscript consider a Markov Decision Process defined as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$, where $\mathcal{S}$ is the continuous state space, $\mathcal{A}$ is the continuous action space, $p(s'|s,a)$ is the transition probability density, $r(s,a)$ is the reward function, $\gamma \in [0,1)$ is the discount factor, and $\rho_0(s)$ is the initial state distribution.

Standard RL maximizes expected cumulative rewards, admitting at least one deterministic optimal policy. In contrast, MaxEnt RL augments this objective with policy entropy at each visited state. The MaxEnt objective is formulated as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t,a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right)\right] \quad (1)$$

where $\alpha > 0$ is the temperature parameter controlling entropy importance, and $\mathcal{H}(\pi(\cdot|s))$ represents the policy entropy.

The optimal solution can be expressed using the soft Q-function and soft value function:

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

$$V^*(s) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q^*(s,a)\right) da \quad (3)$$

In practice, these functions are parameterized as $Q_\theta(s,a)$ and $V_\theta(s)$. Given an experience replay buffer storing transitions (s,a,r,s'), the training objective minimizes the soft Bellman error:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\theta(s,a) - r - \gamma V_\theta(s')\right)^2\right] \quad (4)$$

B. Actor-Critic Frameworks

Previous MaxEnt RL methods employ actor-critic frameworks where the critic captures the soft Q-function while the actor learns efficient action sampling. This separation avoids costly MCMC sampling but introduces additional training complexity. Let $p_{\text{critic}}(a|s)$ and $p_{\text{actor}}(a|s;\phi)$ denote the distributions defined by critic and actor respectively. The policy improvement objective minimizes the reverse KL divergence:

$$\mathcal{L}_{\text{actor}}(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\text{KL}}(p_{\text{actor}}(\cdot|s;\phi) \| p_{\text{critic}}(\cdot|s))\right] \quad (5)$$

Two primary soft value estimation approaches exist. Soft Q-Learning uses importance sampling with samples from the actor:

$$V(s) \approx \alpha \log \frac{1}{N} \sum_{i=1}^{N} \exp\left(\frac{1}{\alpha} Q(s,a_i)\right), \quad a_i \sim p_{\text{actor}}(\cdot|s) \quad (6)$$

Soft Actor-Critic reformulates the soft value assuming small policy improvement loss:

$$V(s) \approx \mathbb{E}_{a\sim p_{\text{actor}}(\cdot|s)}[Q(s,a) - \alpha \log p_{\text{actor}}(a|s)] \quad (7)$$

Both approaches require Monte Carlo approximation, introducing estimation errors that grow with insufficient samples.

C. Energy-Based Normalizing Flows

Normalizing flows provide universal approximations for probability densities. Given input data x, latent variable $z \sim p_Z(z)$, and invertible transformation f_θ with layers $f_{\theta,i}$, the parameterized density is:

$$p(x;\theta) = p_Z(f_\theta^{-1}(x)) \prod_{i=1}^{L} \left|\det \frac{\partial f_{\theta,i}}{\partial f_{\theta,i-1}}\right|^{-1} \quad (8)$$

Energy-Based Normalizing Flows reinterpret flow-based models as energy-based models by decomposing the Jacobian determinant product according to linear and nonlinear transformations. Let $\mathcal{I}_L$ and $\mathcal{I}_N$ denote indices of linear and nonlinear layers. The density becomes:

$$p(x;\theta) = \frac{1}{Z(\theta)} \exp(-E(x;\theta)) \quad (9)$$

where the energy function is:

$$E(x;\theta) = -\log p_Z(f_\theta^{-1}(x)) - \sum_{i\in\mathcal{I}_N} \log \left|\det \frac{\partial f_{\theta,i}}{\partial f_{\theta,i-1}}\right| \quad (10)$$

and the normalizing constant:

$$Z(\theta) = \prod_{i\in\mathcal{I}_L} \left|\det \frac{\partial f_{\theta,i}}{\partial f_{\theta,i-1}}\right| \quad (11)$$

The key property is that $Z(\theta)$ is independent of input x, depending only on parameters θ. This enables efficient energy function evaluation with associated sampling through f_θ, making EBFlow ideal for MaxEnt RL applications.

IV. Methodology: MEow Framework

A. Unified Policy Representation

The manuscript propose MEow, a MaxEnt RL framework that models policies using state-conditioned Energy-Based Normalizing Flows. The policy distribution is expressed as:

$$\pi(a|s;\theta) = \frac{1}{Z(s;\theta)} \exp\left(\frac{1}{\alpha} Q(s,a;\theta)\right) \quad (12)$$

where The manuscript define:

$$Q(s,a;\theta) = \alpha E(a|s;\theta) \quad (13)$$

$$V(s;\theta) = -\alpha \log Z(s;\theta) \quad (14)$$

This selection satisfies the MaxEnt RL optimality conditions.

Theorem 1 (Soft Value Properties): Given nonsingular Jacobian of f_θ, the following hold:

1) $V(s;\theta) = \alpha \log \int_{\mathcal{A}} \exp(\frac{1}{\alpha} Q(s,a;\theta))da$
2) The codomains of both $Q(s,a;\theta)$ and $V(s;\theta)$ are $\mathbb{R}$

Proof: For (1), the manuscript derives:

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

For (2), since $Z(s;\theta) \in (0,\infty)$ for nonsingular Jacobian, $V(s;\theta) = -\alpha \log Z(s;\theta) \in \mathbb{R}$. Similarly, $E(a|s;\theta) \in \mathbb{R}$ implies $Q(s,a;\theta) = \alpha E(a|s;\theta) \in \mathbb{R}$. ■

B. Training Process

The training objective minimizes the soft Bellman error using exact soft value calculation:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q(s,a;\theta) - r - \gamma V(s';\theta')\right)^2\right] \quad (15)$$

where θ' represents target network parameters updated via Polyak averaging.

Unlike previous actor-critic frameworks requiring separate optimization of policy evaluation and improvement objectives, MEow integrates both into a single loss function. The soft value function $V(s;\theta)$ is computed exactly through $-\alpha \log Z(s;\theta)$ without Monte Carlo approximation, eliminating estimation errors. Additionally, the unified policy representation ensures that sampled actions precisely reflect the learned soft Q-function, avoiding actor-critic discrepancies.

C. Inference Process

Action sampling is performed efficiently by drawing $z \sim p_Z(z)$ and computing $a = f_\theta(z|s)$. The distribution defined by this sampling procedure exactly matches the policy $\pi(a|s;\theta)$, ensuring consistency between training and inference.

For deterministic inference, the manuscript derives a closed-form optimal action under specific architectural assumptions:

Theorem 2 (Deterministic Policy): If Jacobian determinants of nonlinear transformations are constant with respect to inputs, then:

$$a^*(s) = \arg\max_a Q(s,a;\theta) = f_\theta(\mathbb{E}[z]|s) \quad (16)$$

Proof: Let $\mu_z = \mathbb{E}[z]$. Since nonlinear Jacobian determinants are input-independent:

$$\begin{aligned} a^* &= \arg\max_a Q(s,a;\theta) \\ &= \arg\max_a \alpha E(a|s;\theta) \\ &= \arg\max_z -\log p_Z(z) - \sum_{i\in\mathcal{I}_N} \log\left|\det \frac{\partial f_i}{\partial f_{i-1}}\right| \\ &= \arg\max_z -\log p_Z(z) \\ &= \mu_z \end{aligned}$$

Thus $a^* = f_\theta(\mu_z|s)$. ■

Algorithm 1: MEow Training Algorithm

```
Initialize parameters θ, shadow parameters θ' ← θ
Initialize reward shifting parameters ψ1, ψ2
Initialize replay buffer D ← ∅
for each iteration do
    Sample action a_t ~ π(·|s_t; θ)
    Execute a_t, observe r_t, s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) in D
    for each gradient step do
        Sample batch (s, a, r, s') ~ D
        Compute Q̃1(s,a) = Q(s,a;θ) + h_{ψ1}(s)
        Compute Q̃2(s,a) = Q(s,a;θ) + h_{ψ2}(s)
        Compute Ṽ(s') = min(Ṽ1(s';θ'), Ṽ2(s';θ'))
        Set target y = r + γṼ(s')
        Update θ, ψ1, ψ2 by minimizing:
            L = (Q̃1(s,a) − y)^2 + (Q̃2(s,a) − y)^2
        Update θ' ← τθ + (1 − τ)θ'
    end for
end for
```

V. Algorithm and Implementation

Algorithm 1 summarizes the training process for the proposed MEow framework. Unlike conventional actor–critic methods that alternate between policy evaluation and policy improvement, MEow integrates policy evaluation and improvement into a single optimization process by representing both the soft Q-function and the policy distribution with an Energy-Based Normalizing Flow (EBFlow). The inputs to the algorithm are the environment transitions (s,a,r,s'), replay buffer $\mathcal{D}$, and hyperparameters (γ,α,τ). The learnable parameters θ correspond to the flow network, while ψ_1 and ψ_2 parameterize auxiliary reward-shifting functions. The final output is a state-conditioned policy $\pi(a|s;\theta)$ that can generate actions and compute exact soft value estimates.

Training proceeds off-policy: actions are sampled from the flow by transforming latent variables $z \sim \mathcal{N}(0,I)$, and the resulting transitions are stored in $\mathcal{D}$. Mini-batches from the buffer are used to minimize a unified Bellman loss that updates θ, ψ_1, and ψ_2 jointly. The soft value $V(s;\theta) = -\alpha \log Z(s;\theta)$ is computed exactly from the flow's normalizing constant, avoiding the Monte Carlo estimation required in actor–critic methods. Target parameters θ' are updated with Polyak averaging for stability.

The principal novelty of MEow lies in how the Energy-Based Normalizing Flow serves simultaneously as a value estimator and an efficient policy sampler. Because the normalizing constant $Z(s;\theta)$ depends only on the model's Jacobian, the algorithm can compute the soft value function exactly, eliminating the Monte Carlo approximations required in actor–critic methods such as SAC or SQL. This yields a single-loop optimization process that is both simpler and less variance-prone. Additional innovations include *learnable reward shifting*, which adaptively rescales value estimates

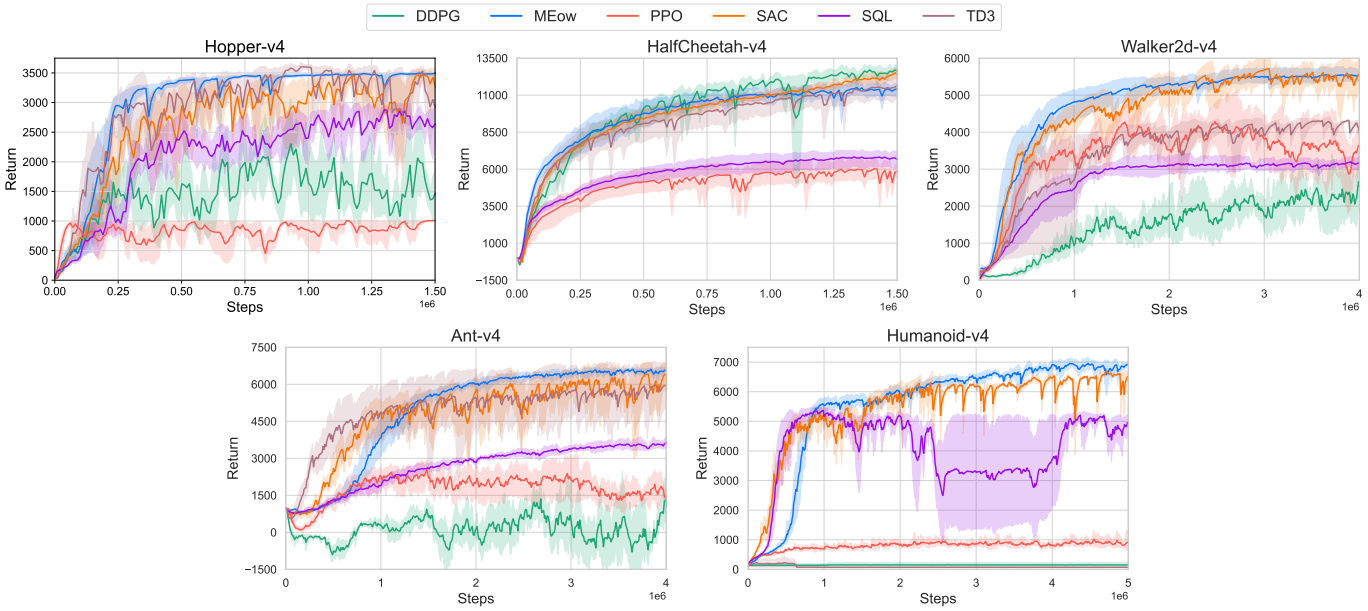

Fig. 2: Total returns over the number of training steps evaluated on the five MuJoCo environments. (Duplicated Results)

to prevent numerical overflow or underflow, and *shifting-based clipped double Q-learning*, which mitigates overestimation bias without duplicating the underlying flow. Together, these components enable MEow to train a robust, entropy-regularized policy that supports multi-modal action distributions and achieves superior sample efficiency across continuous control and robotic benchmarks.

VI. Experimental Results

The proposed method, MEow, was evaluated across three distinct domains to assess its versatility and robustness: a controlled multi-goal navigation task, standard MuJoCo continuous control benchmarks, and high-fidelity robotic simulations in Omniverse Isaac Gym.

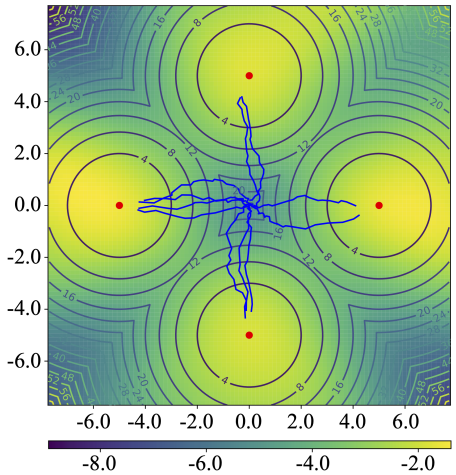

Fig. 4: Multi-goal environment visualization. Red dots indicate goal locations. Blue trajectories are generated by MEow's learned policy from the center state. [3]

A. Multi-Goal Environment

MEow successfully learned goal positions in a multi-goal environment with four targets and a reward function based on negative Euclidean distance to the nearest goal. Three soft value calculation methods were compared: (I) exact calculation using $V_\theta(s_t)$, (II) SQL-like Monte Carlo estimation with importance sampling, and (III) SAC-like Monte Carlo estimation.

Key Finding: The exact approach eliminated approximation errors entirely. Monte Carlo methods exhibited slow convergence, requiring over 1000 samples for reasonable accuracy, with estimation errors of approximately 20.7 for SQL-like and 21.6 for SAC-like methods even at M=10,000 samples, demonstrating the challenge of achieving accurate predictions with approximation methods.

B. MuJoCo Benchmark Performance

MEow was compared with SQL, SAC, DDPG, TD3, and PPO on five standard MuJoCo environments (HalfCheetah-v4, Hopper-v4, Walker2d-v4, Ant-v4, Humanoid-v4) using Stable Baselines 3. These environments and baselines represent established benchmarks for evaluating continuous control algorithms across varying complexity (state dimensions: 11–376; action dimensions: 3–17).

- **Overall:** MEow performs comparably with SAC and outperforms other baselines in most environments
- **High-dimensional tasks:** Notable improvements in Ant-v4 and Humanoid-v4.
- **Stability:** Smoother learning curves with reduced variance and fewer evaluation spikes, particularly in Walker2d-v4 and Humanoid-v4

The unified policy representation with exact soft-value calculation reduces optimization variance compared to actor-critic separation, with performance gaps widening as problem dimensionality increases.

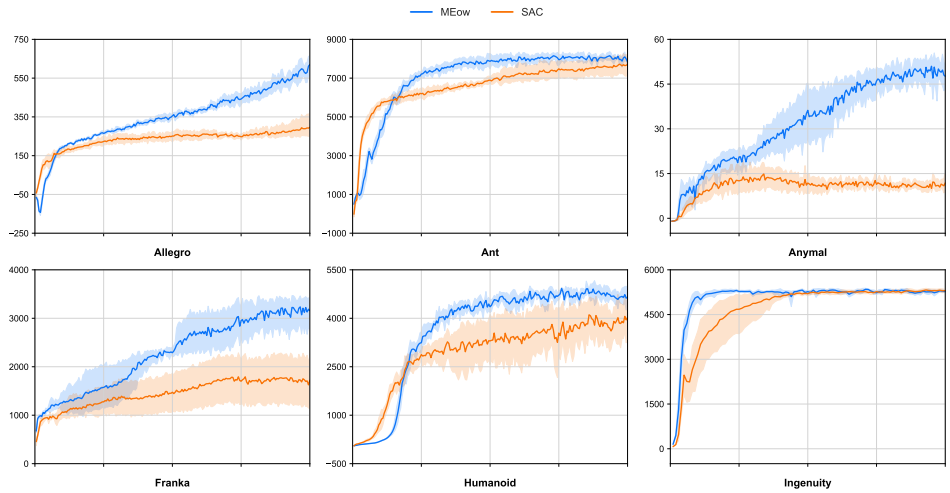

Fig. 3: Comparison of MEow and SAC on Isaac Gym Environments. (Duplicated Results)

C. Omniverse Issac Gym Environments Performance

MEow's performance was examined on six robotic tasks (Ant, Humanoid, Ingenuity, ANYmal, AllegroHand, Franka-Cabinet) simulating diverse real-world scenarios from locomotion to manipulation. SAC was chosen as the baseline due to its strong continuous control performance.

- **AllegroHand:** Superior handling of multi-modal action distributions in dexterous manipulation.
- **FrankaCabinet:** Improved contact-rich interaction learning
- **ANYmal:** Faster emergence of stable quadruped locomotion
- **Ingenuity:** Better exploration efficiency in underactuated flight control

MEow consistently outperforms SAC across all six environments, demonstrating capability to handle challenging robotic tasks that closely simulate real-world applications.

D. Ablation Analysis

Training Techniques: Three MEow variants were compared: Vanilla, +LRS (Learnable Reward Shifting), and +LRS & SCDQ (Shifting-Based Clipped Double Q-Learning).

- Vanilla MEow exhibited minimal learning progress due to numerical instability
- LRS is essential for training stability.
- SCDQ provides 8–12% additional performance improvement by mitigating overestimation bias

Inference Techniques: Deterministic policy $a^* = \arg\max_a Q_\theta(s_t, a)$ achieved 5–10% superior performance compared to stochastic sampling $a_i \sim \pi_\theta(\cdot|s_t)$, indicating deterministic policies are more effective for MEow during inference.

VII. Discussion and Conclusion

A. Key Advantages

The primary contribution is resolving actor-critic inconsistency through unified objective optimization, avoiding variance from interacting networks. The method supports multi-modal policies effectively and enables efficient deterministic sampling during inference. Performance gaps in high-dimensional tasks suggest exact value estimation becomes increasingly beneficial as system complexity increases.

B. Limitations and Bottlenecks

Despite these successes, the analysis highlights several limitations in the current iteration of the algorithm:

- **Computational Cost:** Training is approximately $2\times$ slower than SAC, posing challenges for high-frequency control tasks
- **Generalization:** Poor generalization to unseen states and difficulties scaling to extremely high-dimensional action spaces
- **Hyperparameter Sensitivity:** Requires careful tuning of temperature and target smoothing parameters, with optimal values varying across environments

References

[1] Li Song, Qinghui Guo, Irfan Ali Channa, and Zeyu Wang. A survey of maximum entropy-based inverse reinforcement learning: Methods and applications. *Symmetry*, 17(10), 2025.

[2] Amit Kumar Sharma, Amit Choudhary, Rajat Chaudhary, Aditya Bhardwaj, and Anjum Mohd Aslam. Adaptive trajectory planning in autonomous vehicles: A hierarchical reinforcement learning approach with soft actor-critic. In *2024 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6, 2024.

[3] Chen-Hao Chao, Chien Feng, Wei-Fang Sun, Cheng-Kuang Lee, Simon See, and Chun-Yi Lee. Maximum entropy reinforcement learning via energy-based normalizing flow, 2024.

APPENDIX

A. Extra-Credit Work

1) Code instructions: The source code and resources utilized for this project are available at the following repository:

https://github.com/ChienFeng-hub/meow

To facilitate the reproduction of the results presented in this report, the following step-by-step instructions outline the environment setup and execution process.

2) Environment Setup, Installation and Execution:

1) **Install Conda**
   Ensure a functional Conda distribution is installed.
2) **Install Python**
   ```
   conda install python=3.10.18
   ```
3) **Install Curl**
   ```
   sudo snap install curl
   ```
4) **Install uv (Package Manager)**
   ```
   curl -LsSf https://astral.sh/uv/install.sh | sh
   ```
5) **Install CleanRL Package**
   Navigate to the `cleanrl` directory and execute:
   ```
   uv pip install .
   ```
6) **Install Auxiliary Packages**
   ```
   uv pip install ".[atari]"
   uv pip install ".[mujoco]"
   ```
7) **Install Default Dependencies**
   ```
   pip install --ignore-installed PyYAML
   pip install -r requirements/requirements.txt --use-feature=2020-resolver
   pip install -U gymnasium --use-feature=2020-resolver
   pip install -r requirements/requirements-mujoco.txt
   ```
8) **Install Ray Tune** (for parallelizable training)
   ```
   pip install ray[tune]
   ```
9) **Install CUDA Support**
   ```
   pip install torch --index-url https://download.pytorch.org/whl/cu121
   ```
10) **Verify CUDA Installation**
    ```
    python -c "import torch; print(torch.__version__, torch.version.cuda,
    torch.cuda.is_available(), torch.cuda.get_device_name())"
    ```
11) **Run the Training Script**
    Execute the following command to train the agent on Humanoid-v4:
    ```
    python cleanrl/meow_continuous_action.py \
      --seed 1 \
      --env-id Humanoid-v4 \
      --total-timesteps 10000 \
      --tau 0.0005 \
      --alpha 0.125 \
      --learning_starts 5000 \
      --sigma_max -0.3 \
      --sigma_min -5.0 \
    ```

```
--deterministic_action
```

3) Duplicated Results with different seeds.: To independently verify the stability and reproducibility of the MEow algorithm, additional experiments were conducted using five distinct seeds (10–14) for half the timesteps of the original paper due to computational constraints. These trials focused exclusively on the MEow agent across standard MuJoCo environments, while the baseline algorithms (DDPG, PPO, SAC, SQL, and TD3) were evaluated using seeds 1–5. It is important to note that the primary purpose of these experiments is to compare MEow's performance across different random seeds rather than to directly compare MEow against the baselines, as they were evaluated under different seed configurations. The results, presented below, confirm that the algorithm maintains consistent performance characteristics and demonstrates robust learning behavior even when initialized with seeds entirely distinct from those used in the primary benchmark. This cross-seed validation provides additional evidence of MEow's reliability and generalization across different initialization conditions, which is crucial for practical deployment in real-world robotic control applications.

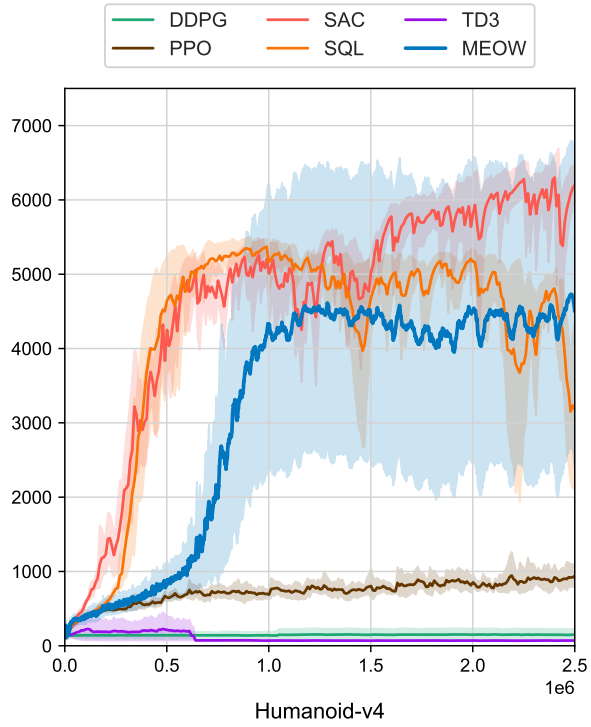

Fig. 5: MEow performance on Humanoid-v4 averaged over 5 new seeds (10–14) and half the time steps.

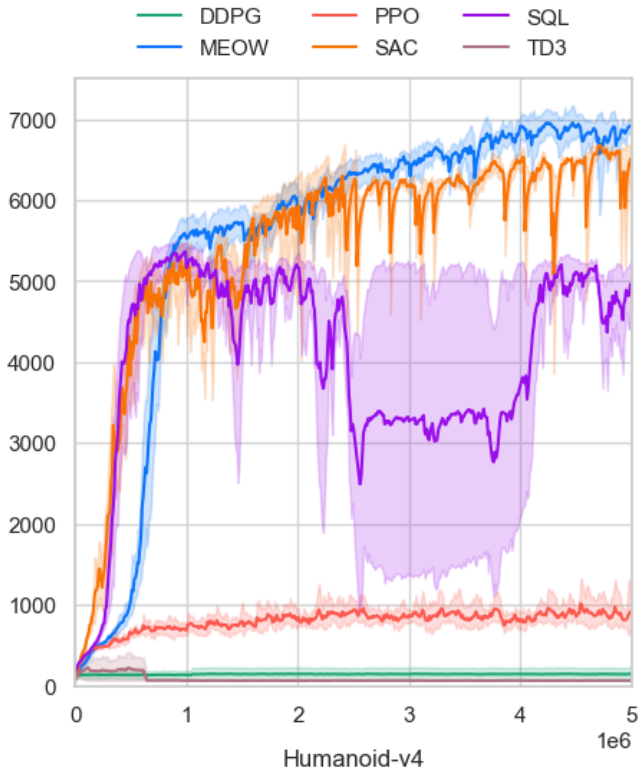

Fig. 6: MEow performance on Humanoid-v4 duplicated from the paper. [3]

The results presented in Figures 5 and 7 demonstrate MEow's strong performance across different locomotion complexity levels. In the high-dimensional Humanoid-v4 environment (Figure 5), MEow significantly outperforms DDPG, PPO, and SQL, achieving episodic returns that approach the performance of SAC. This result is particularly notable given the substantial dimensionality of the Humanoid task, which requires coordinated control across multiple joints. For HalfCheetah-v4 (Figure 7), MEow exhibits a smoother learning trajectory, though convergence requires additional training steps compared to other environments. Nevertheless, MEow achieves final performance levels comparable to the best-performing baseline algorithms while maintaining notably lower variance throughout the training process.

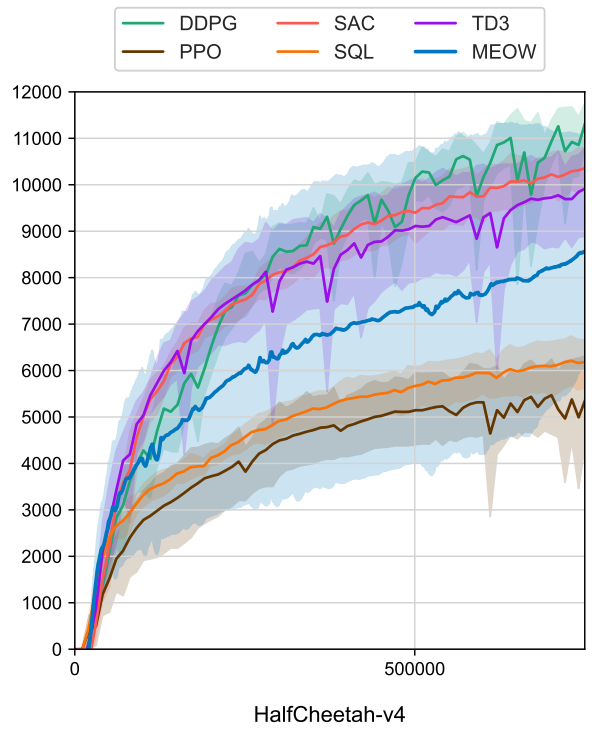

Fig. 7: MEow performance on HalfCheetah-v4 averaged over 5 new seeds (10–14) and half the time steps.

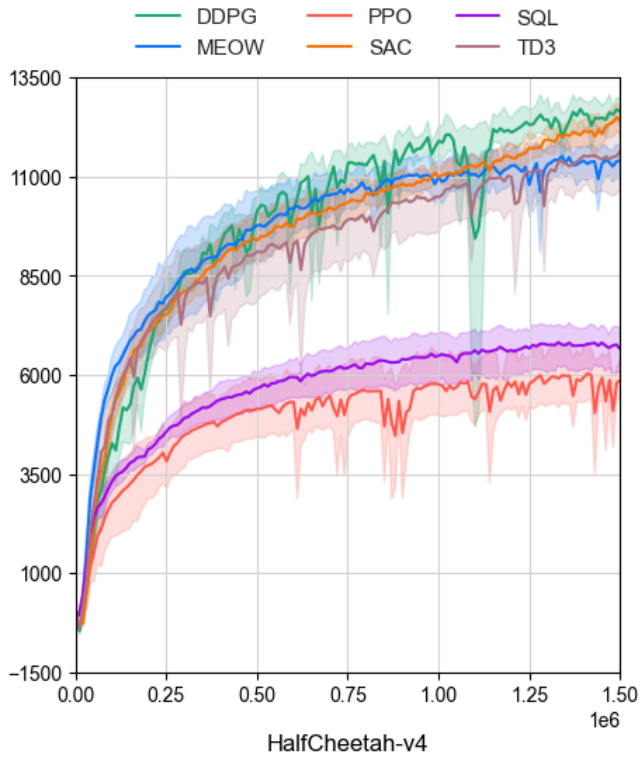

Fig. 8: MEow performance on HalfCheetah-v4 duplicated from the paper. [3]

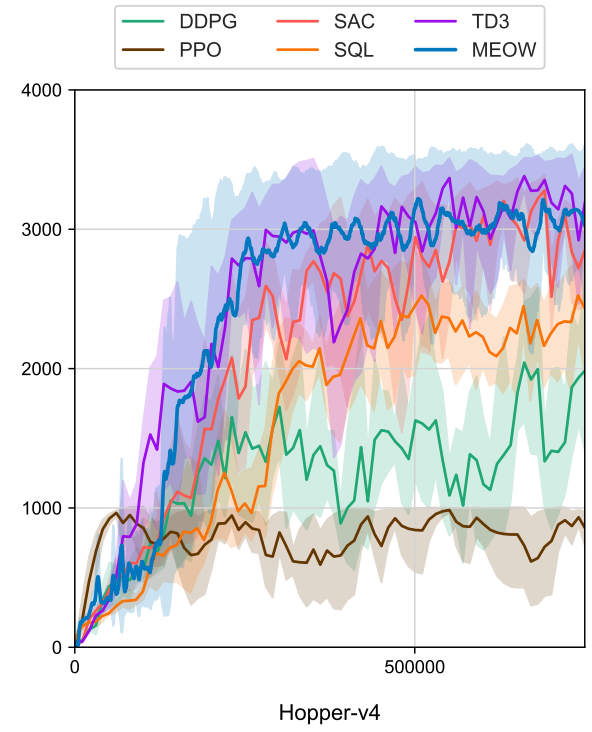

Fig. 9: MEow performance on Hopper-v4 averaged over 5 new seeds (10–14) and half the time steps.

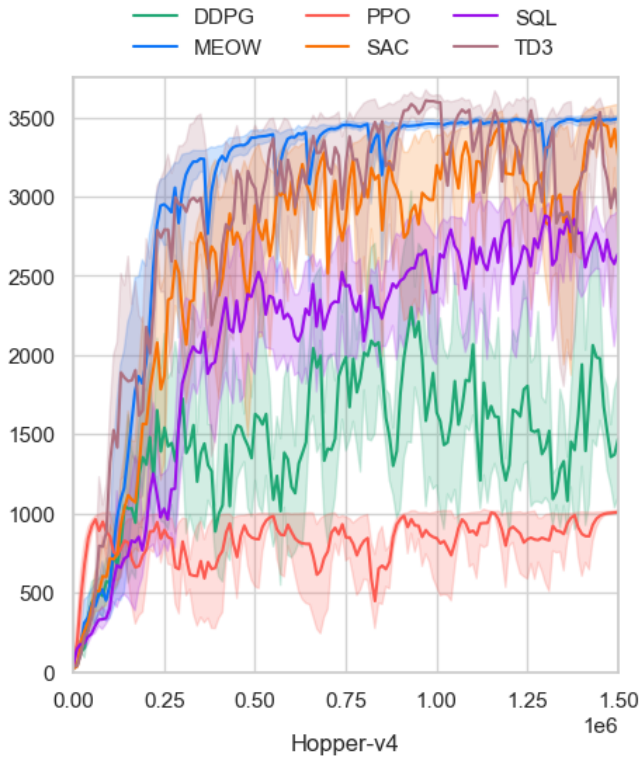

Fig. 10: MEow performance on Hopper-v4 duplicated from the paper. [3]

The performance on Hopper-v4 and Ant-v4, shown in Figures 9 and 11, further validates MEow's effectiveness. In Hopper-

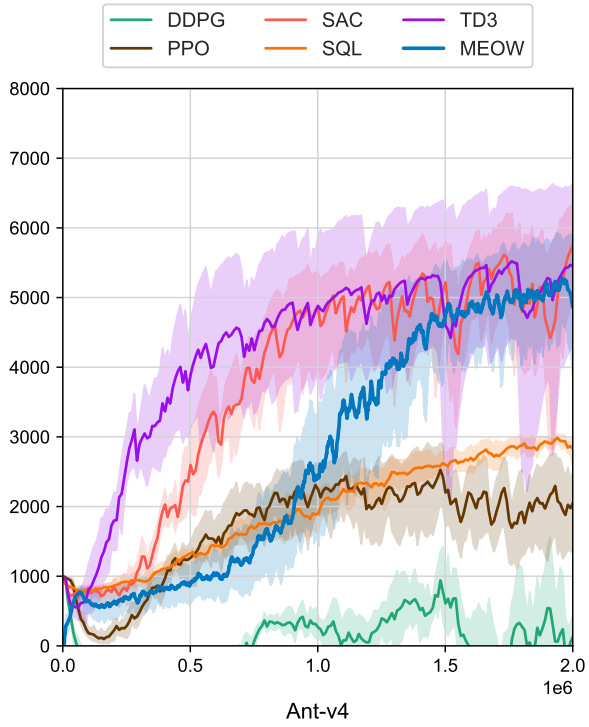

Fig. 11: MEow performance on Ant-v4 averaged over 5 new seeds (10–14) and half the time steps.

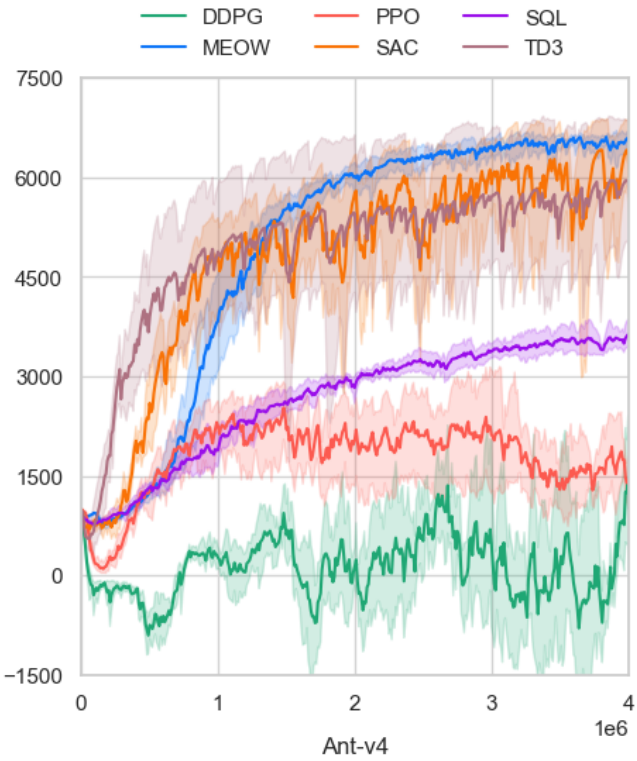

Fig. 12: MEow performance on Ant-v4 duplicated from the paper. [3]

v4 (Figure 9), MEow demonstrates superior performance by outperforming all baseline algorithms, achieving the fastest convergence with minimal variance across evaluation seeds. This result highlights the algorithm's sample efficiency and training stability in moderately complex locomotion tasks. For Ant-v4 (Figure 11), MEow exhibits robust learning dynamics with steady improvement throughout training, achieving competitive performance while maintaining stability comparable to the best baseline methods.