Q-LEARNING WITH NORMALIZING FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Q-learning algorithms are the dominant approach for reinforcement learning in discrete action spaces, but adapting them for continuous action spaces requires an intractable continuous max to find optimal actions. Maximum entropy reinforcement learning replaces the max with a softmax that allocates action density proportionally to the exponentiated Q-function, but without an efficient way to maximize the Q-function this alone does not suffice for continuous Q-learning. This paper introduces Q Normalizing Flows (QNF), a class of methods that directly represent the softmax density via normalizing flows, allowing effective semi-gradient Q-learning in continuous action spaces. This makes a rich body of representations immediately available, including those most commonly used in off-policy actor critic methods, without coupling heavily to specialized architectures for action selection. Q Normalizing Flows perform similarly to widely used actor-critic algorithms for continuous RL.

1 INTRODUCTION

Q-learning algorithms are often favored in discrete reinforcement learning for their simplicity, interpretability, performance, and theoretical ability to learn from arbitrarily distributed data. In continuous action spaces however this is not the case because the action selection step goes from a simple discrete max to the maximization of an arbitrary function. This makes it unclear how to apply Q-learning algorithms in continuous action spaces, but suggests highly performant algorithms might exist given the ability to sample from these learned distributions.

Action value learning actor critic algorithms are the most sample efficient model-free continuous reinforcement learning algorithms today. While these methods perform well empirically, the policy improvement step introduces additional complexity and potential for failure.

Entropy regularization in reinforcement learning can speed up learning by encouraging exploration, and improve gradient conditioning by smoothing action distributions. The entropy regularizer also turns the problematic max operation used for action selection into a softmax sampling operation. Entropy regularized actor critic algorithms like SAC (Haarnoja et al., 2018b), CrossQ (Bhatt et al., 2024), and MPO learn through an additional policy extraction procedure which learns the softmax distribution of an action value function.

This paper explores a different approach. Instead of explicitly parametrizing the Q function and then learning its softmax distribution, we fit a distribution as a Q-function directly. For this purpose we use normalizing flows, a large family of generative models which can be efficiently sampled and queried for exact densities. This removes the additional step actor critic methods use to extract policies from Q-functions.

2 BACKGROUND

2.1 MARKOV DECISION PROCESSES

We define a discounted MDP as the following tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$$

Where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and ρ_0 is the initial state

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

distribution. In particular we study the case where $\mathcal{A}$ is continuous and thus cannot be covered by a discrete distribution.

We define stochastic policies as state conditioned distributions over actions $\pi : \mathcal{S} \to \Delta\mathcal{A}$. The joint state action density $\mu_t^\pi \in \Delta(\mathcal{S} \times \mathcal{A})$ represents the tth timestep of a trajectory sampled using π starting from the initial state distribution.

2.2 Reinforcement Learning

Reinforcement learning (RL) algorithms optimize the objective

$$J_{\text{RL}}(\pi) = \sum_t^\infty \gamma^t \mathbb{E}_{\substack{s,a\sim\mu_t^\pi \\ s'\sim p(\cdot|s,a)}} \left[r(s,a)\right]. \tag{1}$$

Value based methods like Q learning (Watkins & Dayan, 1992) which learn to estimate action values use a non-parametric max operator to directly extract optimal actions. These work well in environments with discrete action spaces but are hard to apply to continuous actions. Actor-critic algorithms (Barto et al., 1983) operate on a different principle, learning a critic to estimate values, and then using those to improve a parametric actor. Actor-critic algorithms can thus receive bootstrap actions without an online search.

2.3 Maximum Entropy Reinforcement Learning

Maximum entropy RL (Ziebart et al., 2008) adds entropy regularization to the objective which can speed up exploration, accelerating convergence speeds in many environments (Haarnoja et al., 2018b).

$$J_{\text{ME}}(\pi) = \sum_t^\infty \gamma^t \mathbb{E}_{\substack{s,a\sim\mu_t^\pi \\ s'\sim p(s,a)}} \left[r(s,a) + \alpha H^\pi(s)\right] \tag{2}$$

The optimum of entropy regularized maximization is the softmax distribution of the objective.

Lemma 1. *Let $f : \mathcal{A} \to \mathbb{R}$ and let $u > 0$ be a reference density with $z = \int_{\mathcal{A}} e^{f(a)} u(a)\, da < \infty$. Then*

$$\max_{q\in\Delta\mathcal{A}} \mathbb{E}_{a\sim q}[f(a)] - D_{KL}(q \,\|\, u) = \log z,$$

attained uniquely at $q^\star(a) = e^{f(a)}u(a)/z$

This is a standard result derived by Donsker & Varadhan (1975) and used in Haarnoja et al. (2017). The proof is summarized in the appendix.

Following this result, the density of the optimal entropy regularized policy for a given Q-function is proportional to the exponential of the action Q-values

$$\pi^\star = \operatorname*{argmax}_{\pi\in\Delta\mathcal{A}} \mathbb{E}_{a\sim\pi}[Q(a)] + \alpha H[\pi] \tag{3}$$

$$\pi^\star(a) = \frac{e^{\frac{Q(a)}{\alpha}}}{\int_{\mathcal{A}} e^{\frac{Q(a')}{\alpha}} da'} \tag{4}$$

Thus the optimal entropy regularized policy is available as the soft max over the Q function, instead of the usual max in Q learning (Ziebart et al., 2008) i.e.

$$\pi^\star(a|s) \propto e^{Q^\star(s,a)/\alpha} \tag{5}$$

2.4 Normalizing Flows

Normalizing flows represent complex densities by applying flexible invertible transformations to a simply base density. The transformed simple base results in a highly expressive set of queriable densities. The density P_f of a flow with map f and base density P_{x_0} can be computed at any point x through the change-of-variables formula (Rezende & Mohamed, 2015).

$$\log P(x) = \log P_{x_0}\left(f^{-1}(x)\right) + \log\det\left|\frac{\partial f^{-1}(x)}{\partial x}\right| \tag{6}$$

When training flow models, it is numerically convenient for the Jacobian determinant computation to be cheap and precise. The structure of most flows is heavily constrained by this requirement. Discrete flow models are usually designed to have block triangular Jacobians reducing the determinant computation to just a diagonal product. Several flow models exist in this class trading off expressivity and simple density computations (Dinh et al., 2017; Kingma & Dhariwal, 2018; Chen et al., 2018). All of them are are invertible and thus their density is available through the change of variables formula 6.

3 Related Work

On-Policy Methods On-Policy methods for tuning flow policies are simple and stable, and several works have investigated their use for tuning expressive policies defined via normalizing flows. FPO (McAllister et al., 2025) works similarly to PPO, but uses a CNF as its policy class, and uses the flow matching loss as a density surrogate for clipping and differentiation. DPPO (Ren et al., 2025) uses PPO directly, unrolling the base MDP into a much longer one where each integration step becomes a state transition. Reinflow (Zhang et al., 2025) similarly augments PPO but uses ground truth density estimates with noise injected into the policy during integration for better gradients. The stability of on-policy RL comes at the cost of sample-efficiency. Since the data assumes no mismatch on old behavior data, new data must constantly be collected to ensure gradient accuracy.

Off-Policy Methods RL has also been used to update flow models off-policy through per-step adjustment using the Q function (Gao et al., 2026; Li & Levine, 2026; Li et al., 2026). These methods learn a Q function in each state and train a policy to match the learned Q function using stationary flow-tuning techniques. The challenge of these approaches is Q function instability. As Q function maximization is applied to the policy network, the Q-function is biased upward by the policy which is trained to maximize it.

Continuous Action Q Learning One general approach to Q learning in continuous action spaces uses simple Q representations generated by deep neural nets with a trivial max operator. This approach was introduced by NAF (Gu et al., 2016) which makes Q learning feasible in continuous action spaces for Q functions defined by a single convex quadratic function. Another in this family is wire-fitting(Korkmaz et al., 2025) which regresses a special spline parametrization with an available max operation as the Q function. These methods however must continually balance expressivity with the availability of a max operator. Another approach is QT-Opt (Kalashnikov et al., 2018) which learns an unconstrained Q-function but is constrained by the speed of the CEM in the continuous action space finding a satisfactory action. Another learns densities in continuous action spaces to represent action values in entropy regularized RL. Trust-PCL (Nachum et al., 2018) and MEow (Chao et al., 2024) are such methods and are the closest to our method. Trust-PCL focuses on the N-step return case which gives a sampling bias in stochastic environments, and focuses on forming trust regions, which we did not find essential when using more expressive policy classes. MEow ties its functionality to a specialized flow architecture that has layers separating into action-independent scale layers defining the value, and constant scale action-dependent layers differentiating the actions. QNF is an extension of Trust-PCL and MEow, covering a general recipe that uses arbitrary flows.

4 Q Normalizing Flows

We introduce Q Normalizing Flows (QNF), a family of soft Q learning algorithms for continuous action spaces which parametrize Q as a normalizing flow density combined with a per-state scalar value.

$$Q_{\theta,\phi}(s,a) = V_\phi(s) + \alpha \log \pi_\theta(a;s)$$

Members of this class include MEow (Chao et al., 2024) and Trust-PCL (Nachum et al., 2018) where MEow uses a V estimated from the fixed layers of its flow, and Trust-PCL uses a simple gaussian distribution with gradient TD. This method extends soft Q learning (Haarnoja et al., 2017) with flow models which remove the need for policy extraction alongside value learning.

Observe that the value function of a soft optimal policy with entropy bonus is the normalizing constant of the softmax distribution of the action value.

$$V^\star(s) = \mathop{E}_{a\sim\pi^\star(\cdot;s)}[Q^\star(s,a)] + \alpha H^{\pi^\star}(s) \tag{7}$$

$$= \mathop{E}_{a\sim\pi^\star(\cdot;s)}[Q^\star(s,a) - \alpha \log \pi^\star(a;s)] \tag{8}$$

$$= \mathop{E}_{a\sim\pi^\star(\cdot;s)}\left[Q^\star(s,a) - \alpha \log \frac{e^{\frac{Q^\star(s,a)}{\alpha}}}{Z(s)}\right] \tag{9}$$

$$= \alpha \log Z(s) = \alpha \log \int_{\mathcal{A}} e^{\frac{Q^\star(s,a)}{\alpha}}\, da \tag{10}$$

A normalizing flow provides a simple non-parametric softmax of its density via its forward map.

By construction the QNF Q estimator has the following two properties.

$$\pi_\theta(a;s) = \frac{e^{Q_{\theta,\phi}(s,a)/\alpha}}{\int e^{Q_{\phi,\theta}(s,a')/\alpha}\, da'} \tag{11}$$

$$V_\phi(s) = \mathbb{E}_{a\sim\pi_\theta(\cdot;s)}\left[Q_{\phi,\theta}(s,a)\right] + \alpha H^{\pi_\theta}(s) \tag{12}$$

Equation 11 is a consequence of the normalization of the generative model, and equation 12 follows from π_θ being the softmax distribution of $Q_{\phi,\theta}$.

4.1 Soft TD Loss

Having constructed a Q estimator with a non-parametric softmax operator, we fit it using a similar procedure as the one outlined in Mnih et al. (2015).

$$\bar{Q}(s,s',r) = r + \gamma\left(\alpha H^{\pi_\theta}(s') + \mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot;s')}\left[\min\left(Q_{\theta,\phi_0}(s',a'), Q_{\theta,\phi_1}(s',a')\right)\right]\right) \tag{13}$$

$$= r + \gamma \min(V_{\bar{\phi}_0}(s'), V_{\bar{\phi}_1}(s')) \tag{14}$$

Where $\bar{Q}$ denotes the one step TD target. Similarly to SAC(Haarnoja et al., 2018a) we find it necessary to use a min of two critics to avoid value overestimation. We also use the same critic EMA delay rate of $\tau = 0.005$.

V_ϕ and π_θ can be updated using the same TD loss as in soft Q learning (Haarnoja et al., 2017)

$$\mathcal{L}_{\text{TD}}(\theta,\phi) = \mathop{\mathbb{E}}_{(s,a,r,s')\sim D}\left[\left(Q_{\phi,\theta}(s,a) - |\bar{Q}(s,a)|_{\text{sg}}\right)^2\right] \tag{15}$$

Where D is an experience buffer. Through this loss both π_θ and V_ϕ reflect the reward structure of the MDP. This algorithm differs from Trust-PCL (Nachum et al., 2018) which uses the full gradient-TD update on the same residual. While this provides convergence, it also biases the result in stochastic environments.

4.2 α Tuning

The technique outlined in the previous section works and is used in Figure 4, however tuning α is difficult, and means it cannot track policy improvement during training. In existing entropy regularized RL algorithms like SAC (Haarnoja et al., 2018a), MPO (Abdolmaleki et al., 2018), and Trust-PCL (Nachum et al., 2018) the entropy regularization weight can be either defined as a constant in this way, or tuned automatically according a rule. SAC proposes to use α's monotonic relationship to policy entropy to adjust α during training maintaining a predefined entropy constraint. In MPO and Trust-PCL the policy is trained to advance at a constant pace in terms of KL from its previous iteration. Instead QNF measures the existing relationship between advantage and policy density, using it to recover the α that best explains the policy's behavior. The TD loss at a given α is defined as follows:

$$\mathcal{L}_{\text{TD}}(\alpha) = \mathop{\mathbb{E}}_{(s,a,r,s')\sim D}\Big[\big(\underbrace{r + \gamma V_\phi(s') - V_\phi(s)}_{\text{advantage}} - \alpha \log \pi_\theta\big)^2\Big] \tag{16}$$

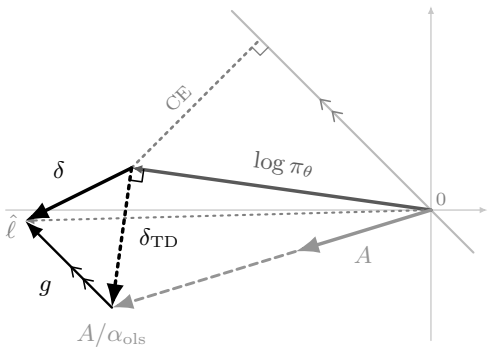

Figure 1: This figure visualizes the gradient update of QNF. The axes correspond to the log density on different (s, a, r, s') samples in the buffer. Raw TD error is shown as δ_{TD}, and the advantage is shown as A. The greedification term is shown as g. The total policy target is shown as $\hat{\ell}$ and the current policy density is $\log \pi_\theta$.

This naturally takes the form of a regression problem where the x-axis is determined by $\log \pi_\theta(a; s)$ and the y-axis is determined by $r + V_\phi(s') - V_\phi(s)$. The minimizer can be computed in closed form and is simply:

$$\alpha_{\text{OLS}} = \frac{\mathop{\mathbb{E}}_{(s,a,r,s')\sim D}\left[(r + \gamma V_\phi(s') - V_\phi(s)) \log \pi_\theta(a; s)\right]}{\mathop{\mathbb{E}}_{(s,a,r,s')\sim D}[\log \pi_\theta^2(a; s)]} \tag{17}$$

4.3 Policy Improvement

Given a converged $Q_{\theta,\phi}$ QNF gives zero residual in expectation and the loss described thus far would cease to further improve the policy. In algorithms with additional constraints like the entropy constraint in SAC or just a fixed α this can only happen when the policy has reached the desired point. For QNF there is no such terminal point (shown in Figure 4), and so to encourage improvement we introduce a greedification term which pushes the policy toward progressively colder softmaxes of the value. To avoid moving too far off the replay buffer, QNF applies a push in a CE-neutral direction. We denote the policy target $A_{V_\phi}(s, r, s', \alpha) = \alpha^{-1}\left(r + \gamma V_\phi(s') - V_\phi(s)\right)$. The cross-entropy between the target and the buffer distribution is $\mathbb{E}_{(s,r,s')\sim D}\left[A_{V_\phi}(s, r, s', \alpha)\right]$. To define the CE neutral greedification direction g, we remove the CE component from the target and RMS normalize:

$$\tilde{A}_{V_\phi}(s, r, s', \alpha) = \alpha^{-1}(r + \gamma V_\phi(s') - V_\phi(s)) - \underbrace{\mathbb{E}_{(s,r,s')\sim D}\left[A_{V_\phi}(s, r, s', \alpha)\right]}_{\text{cross entropy}} \tag{18}$$

$$g_{V_\phi}(s, r, s', \alpha) = \frac{\tilde{A}_{V_\phi}(s, r, s', \alpha)}{\sqrt{\mathbb{E}_{(s,a,s'r)\sim D}[\tilde{A}^2_{V_\phi}(s, r, s', \alpha)]}} \tag{19}$$

To produce the final QNF target we add the advantage target together with the greed term

$$\hat{\ell}_{V_\phi}(s, r, s', \alpha) = A_{v_\phi}(s, r, s', \alpha) + \lambda g_{V_\phi}(s, r, s', \alpha) \tag{20}$$

This greed term allows the policy to constantly push toward improvement without completely departing from the buffer, without slowing down improvement, or setting a ceiling. This, with α_{ols} is the update visualized in Figure 1

5 Experiments

5.1 Setup

Figure 2 compares the QMANSF realization of QNF against several relevant reinforcement learning algorithms. We compare against two MLP based off-policy entropy regularized actor critic baselines

Algorithm 1 Q-Normalizing Flow Training

Require: $\gamma, \lambda, \tau, \eta, N$

Initialize flow policy π_θ, value functions V_{ϕ_0}, V_{ϕ_1}
Replay Buffer $D \leftarrow \emptyset$
Target Networks $\bar{\phi}_0 \leftarrow \phi_0$, $\bar{\phi}_1 \leftarrow \phi_1$
Sample Initial State $s \sim S_0$
while not converged **do**
  Sample action $a \sim \pi(\cdot; s)$
  Sample next state $s' \sim T(s, a)$
  Insert transition to replay buffer $D \leftarrow D \cup (s, a, s')$
  Iterate state $s \leftarrow s'$
  Sample batch $B \leftarrow \{(s_i, a_i, s'_i, r_i)\}_{i=1}^N \sim D$
  $A_i \leftarrow r_i + \gamma \min_{\phi_0,\phi_1} V_\phi(s'_i) - \min_{\phi_0,\phi_1} V_\phi(s_i)$
  $\alpha \leftarrow \dfrac{\frac{1}{N}\sum_{i=1}^N A_i \log \pi_\theta(a_i; s_i)}{\frac{1}{N}\sum_{i=1}^N \log \pi_\theta^2(a_i; s_i)}$
  $\bar{A} \leftarrow \dfrac{1}{N}\sum_{i=1}^N A_i$ ▷ Compute target CE
  $\tilde{A}_i \leftarrow \dfrac{A_i - \bar{A}}{\alpha}$ ▷ Project advantage
  $g_i \leftarrow \dfrac{\tilde{A}_i}{\sqrt{\frac{1}{N}\sum_{i=1}^N \tilde{A}_i^2}}$ ▷ Normalize greed
  $\delta_i \leftarrow \min(V_{\phi_0}(s_i), V_{\phi_1}(s_i)) + \alpha \log \pi_\theta(a_i; s_i) - r_i - \gamma \min(V_{\bar{\phi}_0}(s'_i), V_{\bar{\phi}_1}(s'_i))$
  $\phi_0 \leftarrow \phi_0 - \eta_V \sum_{i=1}^N \delta_i \partial_\phi V_{\phi_0}(s_i)$, $\phi_1 \leftarrow \phi_1 - \eta_V \sum_{i=1}^N \delta_i \partial_\phi V_{\phi_1}(s_i)$ ▷ Critic update
  $\theta \leftarrow \theta - \eta_\pi \sum_{i=1}^N (\delta_i - \lambda g_i) \partial_\theta \log \pi_\theta(a_i; s_i)$ ▷ Actor update
  $\bar{\phi}_0 \leftarrow \tau * \phi_0 + (1-\tau) * \bar{\phi}_0$, $\bar{\phi}_1 \leftarrow \tau * \phi_1 + (1-\tau)\bar{\phi}_1$ ▷ Target update
end while

SAC (Haarnoja et al., 2018a) and CrossQ (Bhatt et al., 2024), one off-policy flow-matching actor critic baseline FMER (Gao et al., 2026), one on-policy flow matching baseline FPO (McAllister et al., 2025), and one normalizing flow baseline MEow (Chao et al., 2024).

Figure 3 compares our choice of masked autoregressive neural spline flows (Durkan et al., 2019) (QMANSF) against several notable flow architectures. Namely we compare against masked autoregressive flows with affine coupling (Papamakarios et al., 2017) (QMAF), affine coupling flows (Dinh et al., 2017) (QRNVP), spline coupling flows (Durkan et al., 2019) (QCNSF), energy based flows to compare against MEow (Chao et al., 2023) (QEBFlow), an MLP with bijective activations and full rank linear layers (Papamakarios et al., 2021) (QMLP), and just a simple parametric gaussian distribution (QGauss).

In Figure 4 we ablate the effect of our alpha autotuning scheme and our greedification scheme by watching our α tuning against fixed settings of α, and against various settings of λ and show the trajectory of α experienced during training.

Across all figures we evaluate on several deepmind control tasks (Tassa et al., 2018). We selected tasks of medium difficulty and dimension prioritizing dense rewards to show the raw policy improvement performance of QNF. We omit tasks like hopper-hop, acrobot-swingup, and point_mass-hard because no algorithms perform sufficiently well within the 500k-step budget tested.

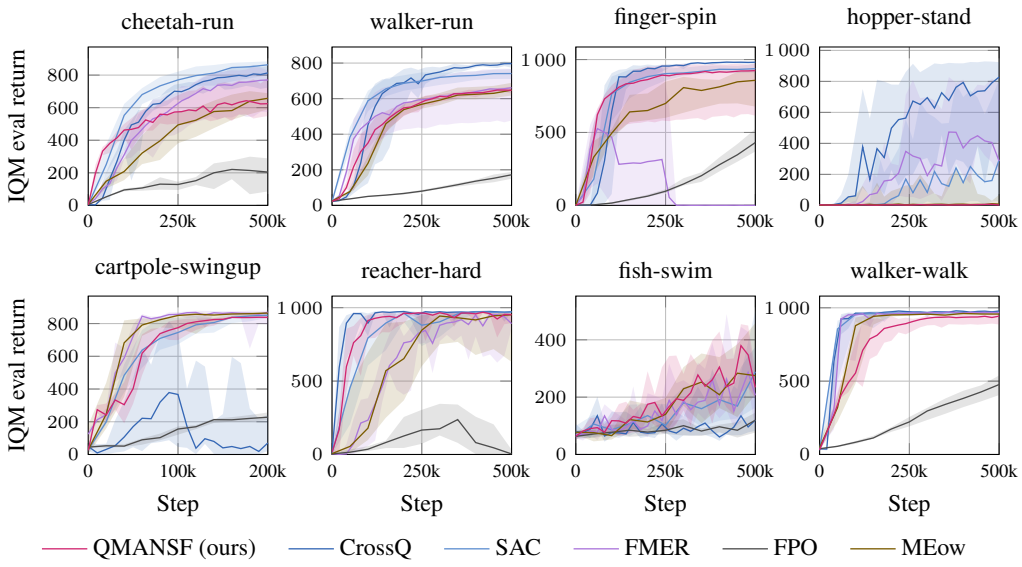

Figure 2: IQM eval return (±95% bootstrap CI) on DeepMind Control environments.

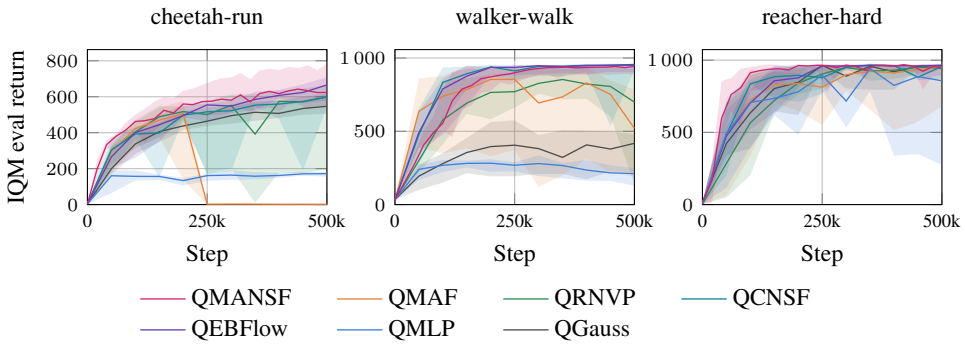

Figure 3: IQM eval return (±95% bootstrap CI) of QNF with different flow architectures.

5.2 Implementation Details

We use the same set of hyperparameters across all envs tested for all environments tested. For MEow we built this from the default for gym's Hopper (Brockman et al., 2016) with an entropy regularization constant tuned down for the DeepMind Control tasks. Also, in the official SAC implementation α is initialized to 1.0, while we follow other implementations and initialize to 0.2 which gives a comparatively faster start in cold environments like those in DeepMind Control For evaluation, 5 seeds were trained and the IQM performance across the 5 seeds is reported with 95 percent bootstrap CIs following Agarwal et al. (2021). Each seed is evaluated at each evaluation step on 10 rollouts, so each point in Figure 2 is the result of 50 evaluations. To be fair to the normalizing flows all evaluations were done in stochastic mode since most flows do not have an option for deterministic evaluation.

5.3 Results

We find in our algorithm comparison experiments Figure 2 that Q Normalizing Flows performs competitively against strong off-policy actor-critic baselines and methods designed for flows. In the running envs, which reward high precision and exploitation, QMANSF underperforms the actor critic baselines despite early performance. In the exploration environments such as hopper-stand it also underperforms while CrossQ performs well. While CrossQ sharpens to the entropy setpoint

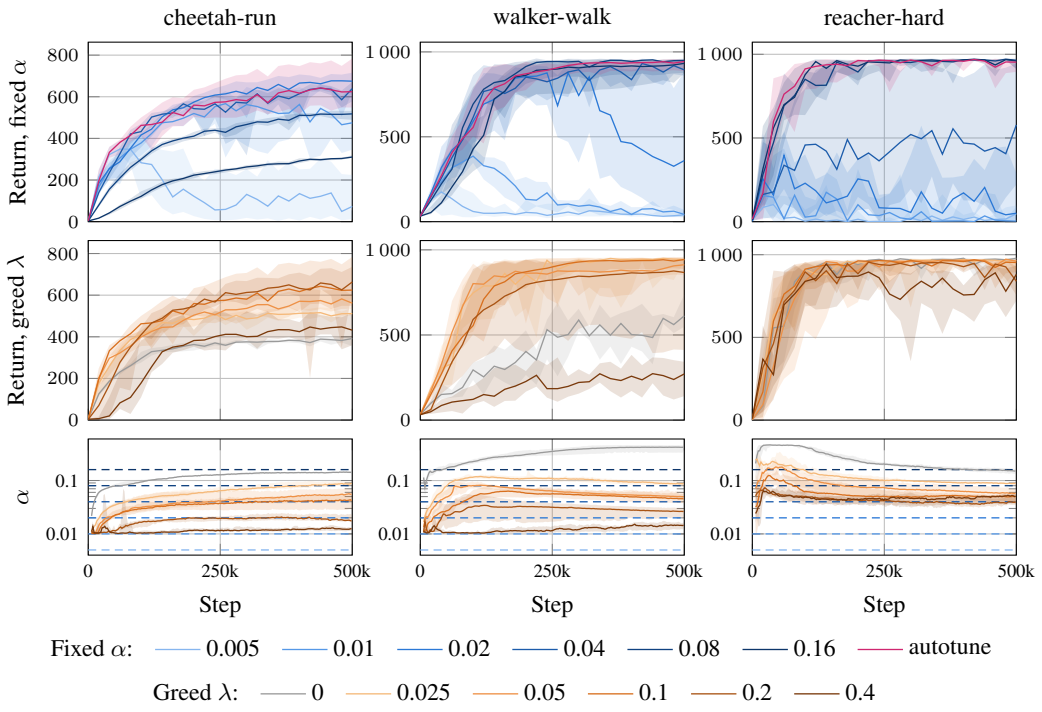

Figure 4: IQM eval return and autotuned α ($\pm$95% bootstrap CI) of QMANSF. Top: fixed α (blue, light to dark with increasing α) and autotuned α. Middle: autotuned α at different greedification scales λ (gray: $\lambda = 0$; orange, light to dark with increasing λ); $\lambda = 0.1$ is the autotuned run of the top row. Bottom: the autotuned α at each step for each λ; dashed lines mark the fixed values of the top row.

even without sufficient return data, QNF algorithms wait for reward signal without sharpening, keeping a constant greedification push that could be equilibrated. This approach fails to find the standup behavior in hopper-stand, and generally fails to explore with a weak reward signal.

In the flow architecture comparison Figure 3 we find that QMANSF is consistently among the best performers, with comparatively few layers that each have a high degree of expressivity.

The grid of α settings in Figure 4 reveals a high sensitivity to α which varies between environments. The stateless greedy α tuning shows performance on-par with the best per-env tuning of α without a preset scale. The environments show α adjusting dynamically to account for policy improvement, and freezing once the policy has converged as is visible in the latter half of the reacher-hard results.

The greed results show that higher greed values drive faster improvement but can collapse, validating that $\lambda = 0.1$ seems like the optimal setting on the environments under test.

Conclusion

Discussion In this work we establish Q Normalizing Flows as an effective family of Q-learning methods that do not require separate action-critic and policy models. By basing the Q-function on flow models combined with a simple scalar value network, a growing set of flexible and expressive models is inherited and immediately made available. This architecture simplifies existing actor-critic algorithms by removing the separate actor improvement regression, leaving only evaluation steps.

Limitations This study is limited to simulated experiments and relatively dense domains. QNF could be extended to more challenging high dimensional and sparse domains. The least computationally intensive flows were selected for this method to align with computational demands of existing RL techniques, and because they were the simplest, but this limits the generality of the

results. Flows are generally more computationally expensive than a gaussian actor critic which is generally one forward pass for sampling and density estimation. Flows also often have instability issues, requiring tricks like the split quadratic solve to remain stable, while actor-critic algorithms usually operate in their high density regime, and so are less susceptible to this problem. A more thorough study would include ablations on higher dimensional problems and would include the wall time and probe the space of actor critic combinations possible with the flow models tested.

Future Work Extending these results to continuous normalizing flows remains a promising avenue toward further improving the fit of the advantage function which could raise the performance beyond what actor-critic algorithms can achieve on precise environments like cheetah-run and walker-run.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods, designing or providing feedback on research methodology or experiments. We have not used generative AI tools for generating synthetic datasets, assisting in the writing of proofs, proposing or refining hypotheses, supporting qualitative and thematic data analysis, and interpreting results. Helping develop novel theoretical models or conceptual frameworks, helping formulate mathematical claims, providing critical ingredients for proving them, assisting with translation, cleaning and reformatting datasets are not applicable to this work. Additionally, we used generative AI tools for typesetting, creating figures, suggesting hyperparameters, and proofreading of the manuscript. We have reviewed all AI-assisted work. We have checked the output of the AI generated work by reading its code and checking the outputs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

There are no ethical concerns with this work.

REPRODUCIBILITY STATEMENT

For reproducibility the supplementary materials will be sufficient to reproduce the results in the paper. The code will be made available via an anonymous link and will reproduce the results exactly.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.*, 13(5):834–846, 1983. ISSN 0018-9472. doi: 10.1109/TSMC.1983.6313077. URL https://doi.org/10.1109/TSMC.1983.6313077.

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Represen-*

tations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=PczQtTsTIX.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016. URL https://arxiv.org/abs/1606.01540.

Chen-Hao Chao, Wei-Fang Sun, Yen-Chang Hsu, Zsolt Kira, and Chun-Yi Lee. Training energy-based normalizing flow with score-matching objectives. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/8882d370cdafec9885b918a8cfac642e-Abstract-Conference.html.

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. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/661c37f3b098bdee53fd7d9c4ef6964a-Abstract-Conference.html.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6572–6583, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Monroe D. Donsker and S. R. S. Varadhan. On a variational formula for the principal eigenvalue for operators with maximum principle. *Proceedings of the National Academy of Sciences*, 72(3): 780–783, 1975. doi: 10.1073/pnas.72.3.780. URL https://www.pnas.org/doi/abs/10.1073/pnas.72.3.780.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7509–7520, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/7ac71d433f282034e088473244df8c02-Abstract.html.

Ting Gao, Stavros Orfanoudakis, Nan Lin, Elvin Isufi, Winnie Daamen, and Serge Hoogendoorn. Flow Matching Policy with Entropy Regularization, 2026. URL https://arxiv.org/abs/2603.17685.

Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2829–2838. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/gu16.html.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th*

International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2017. URL http://proceedings.mlr.press/v70/haarnoja17a.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018a. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications, 2018b. URL https://arxiv.org/abs/1812.05905.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 651–673. PMLR, 2018. URL https://proceedings.mlr.press/v87/kalashnikov18a.html.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 10236–10245, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/d139db6a236200b21cc7f752979132d0-Abstract.html.

Yigit Korkmaz, Urvi Bhuwania, Ayush Jain, and Erdem Biyik. Actor-free continuous control via structurally maximizable q-functions. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/2f8928efe957139e9c0efc98f173f4be-Abstract-Conference.html.

Qiyang Li and Sergey Levine. Q-learning with Adjoint Matching, 2026. URL https://arxiv.org/abs/2601.14234.

Zeqiao Li, Yijing Wang, Haoyu Wang, Zheng Li, and Zhiqiang Zuo. Boosting Maximum Entropy Reinforcement Learning via One-Step Flow Matching, 2026. URL https://arxiv.org/abs/2602.01606.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow Matching Policy Gradients, 2025. URL https://arxiv.org/abs/2507.21053.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015. ISSN 0028-0836. doi: 10.1038/NATURE14236. URL https://doi.org/10.1038/nature14236.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-pcl: An off-policy trust region method for continuous control. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HyrCWeWCb.

George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2338–2347, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6c1da886822c67822bcf3679d04369fa-Abstract.html.

George Papamakarios, Eric T. Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22:57:1–57:64, 2021. URL https://jmlr.org/papers/v22/19-1028.html.

William H Press, Saul A Teukolsky, William T Vetterling, Brian P Flannery, et al. *Numerical recipes in C*, volume 2. Cambridge university press Cambridge, 1992.

Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1530–1538. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/rezende15.html.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. DeepMind Control Suite, 2018. URL https://arxiv.org/abs/1801.00690.

Christopher J. C. H. Watkins and Peter Dayan. Technical Note Q-Learning. *Mach. Learn.*, 8(3): 279–292, 1992. ISSN 0885-6125. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/98d6f928497d9eaad9f320fb2db3040d-Abstract-Conference.html.

Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. 2008.

A Appendix

A.1 Proofs

Lemma 1. *Let $f : \mathcal{A} \to \mathbb{R}$ and let $u > 0$ be a reference density with $z = \int_{\mathcal{A}} e^{f(a)} u(a)\, da < \infty$. Then*

$$\max_{q \in \Delta\mathcal{A}} \mathbb{E}_{a\sim q}[f(a)] - D_{KL}(q \,\|\, u) = \log z,$$

attained uniquely at $q^\star(a) = e^{f(a)} u(a)/z$

Proof.

$$\max_{q \in \Delta A} \mathbb{E}_{a \sim q}[f(a)] + H[q] \tag{21}$$

$$= \max_{q \in \Delta A} \int_A q(a) \left(f(a) - \log \frac{q(a)}{u(a)} \right) da \tag{22}$$

$$= \max_{q \in \Delta A} \int_A q(a) \left(-\log \frac{q(a)}{e^{f(a)} u(a)} \right) da \tag{23}$$

$$= \max_{q \in \Delta A} \int_A q(a) \left(-\log \frac{q(a) z}{e^{f(a)} u(a)} + \log z \right) da \tag{24}$$

$$= \max_{q \in \Delta A} -D_{\mathrm{KL}} \left(q(a) \| \frac{e^{f(a)} u(a)}{z} \right) + \log z \tag{25}$$

Where H is the negative relative entropy, u is the reference density (usually uniform) which must have $u(a) > 0$ for all a in A, z is the normalizer $z = \int_A e^{f(a)} u(a)\, da$. Since the q minimizing $D_{\mathrm{KL}}(q||p)$ is p this means that the maximizer $q^*(a)$ is proportional to $e^{f(a)} u(a)$ □

A.2 Details

A.2.1 Spline Stability

When training neural spline flows with QNF we find that the reverse solves can run into severe numerical instabilities. We found because of this that it was essential to use numerically stable quadratic solving technique described in Press et al. (1992).

A.2.2 Hyperparameters

Table 1: Shared hyperparameters and those of QMANSF. $|\mathcal{A}|$ is the action dimension. All networks are trained with Adam.

Shared	
Discount γ	0.99
Seeds	5 per environment
Evaluation episodes	10 per evaluation
Training steps	500k (200k on cartpole-swingup)
Batch size	256
Off-policy methods (all but FPO)	
Warm-up steps	5000 (uniform random actions)
Replay buffer	every transition of the run
Updates per environment step	1
Observation normalization	running mean and variance (QMANSF, SAC, CrossQ, FPO)
QMANSF (ours)	
Policy flow	1 masked autoregressive rational-quadratic spline layer on $[0, 1]^{\lvert\mathcal{A}\rvert}$, 8 bins, learned endpoint slopes
Spline bounds	minimum bin width 0.01, minimum knot slope 0.01
Base transform	state-conditional Kumaraswamy inverse CDF before the spline, shapes in $[0.8, 5]$
Conditioner	causal MLP, 3 hidden layers, width 128 per action dimension and 128 for the state, dropout 0.2
State encoder	1 hidden layer, width 1024
Value network	2 hidden layers, width 256; twin value heads; EMA target, $\tau = 0.005$
α	closed-form least squares, clipped to $[0.01, 0.5]$
Greed	0.1 (batch-centered advantage push, RMS-normalized)
Learning rate	3×10^{-4}

Table 2: Hyperparameters of the baselines, on top of the shared settings in Table 1.

SAC	
Actor / critic learning rate	3×10^{-4} / 10^{-3}
Networks	2 hidden layers, width 256; twin Q
Target update	$\tau = 0.005$; actor update every 2 critic steps
α	learned, initial 0.2, target entropy $-\lvert\mathcal{A}\rvert$
CrossQ	
Actor / critic learning rate	10^{-3} / 10^{-3}
Networks	2 hidden layers; actor width 256, critic width 2048 with batch renormalization; twin Q, no target network
Actor update	every 3 critic steps
α	learned, initial 1.0, target entropy $-\lvert\mathcal{A}\rvert$
FMER	
Policy	flow matching, 10 integration steps, 8 candidate actions ranked by the twin-minimum Q
Networks	3 hidden layers, width 256; time embedding 16
Learning rates	actor 3×10^{-4} decayed linearly to 5×10^{-5} over steps 25k–75k; critic and duals 3×10^{-4}
Gradient clipping	critic elementwise 50 and global norm 10; actor global norm 2
Target update	$\tau = 0.005$; actor update every 2 critic steps
α	learned, initial e, target entropy ramped from 0 to $-\lvert\mathcal{A}\rvert$ over steps 10k–200k
Advantage temperature β	learned, initial 0.5, effective-sample-size target 3
Reward normalization	running standard deviation, clipped to $[-10, 10]$
FPO	
Rollout / epochs / minibatch	10000 steps / 4 / 256
Learning rate	3×10^{-4}
Clip ϵ, GAE λ	0.05, 0.95
Value coefficient, reward scale	0.25, 10
Flow	8 Monte Carlo samples, 8 solver steps, time embedding 16
Networks	policy 3 hidden layers, width 256; value 3 hidden layers, width 256
MEow	
Learning rate, gradient clipping	10^{-3}, global norm 30
Flow	2 layers, width 64, scale network width 256, 2 hidden layers, dropout 0.1
Log-scale bounds	$[-5, -0.3]$
α	fixed, 0.02
Target	twin, EMA, $\tau = 0.005$
Evaluation	stochastic actions