Extended Abstract

Motivation Diffusion models exhibit superior expressivity and multimodality, offering significant advantages for tackling complex Reinforcement Learning (RL) tasks. Existing work has predominantly applied diffusion models to offline RL due to challenges in online implementation. MaxEntDP provides a solution for these challenges but lacks automated exploration and temperature-tuning mechanisms, limiting its broader practical use. Since diffusion models have proven so successful in offline RL, developing a method to apply them in the online setting would be a very significant finding.

Method This project extends the MaxEntDP framework by incorporating efficient exploration schemes and a temperature parameter autotuning method to make eliminate a brittle hyperparameter. We integrate Random Network Distillation (RND) for intrinsic rewards, incentivizing exploration based on state novelty. Additionally, we use uncertainty bonuses from an ensemble of Q-networks to encourage exploration in uncertain areas of the action-value space. To reduce hyperparameter tuning complexity, we implement automatic temperature tuning inspired by the Soft Actor-Critic (SAC) algorithm.

Implementation Our methods replicate MaxEntDP based upon the QSM implementation (on which the authors of MaxEntDP also base their official implementation), using Python and JAX. Because MaxEntDP uses MuJoCo for evaluation, we adopt the same for consistency, on the continuous control benchmarks HalfCheetah, Swimmer, Hopper, Walker2d, Ant, and Humanoid.

Results We successfully replicated the original MaxEntDP baseline across six standard MuJoCo benchmarks. The automatic temperature tuning achieved competitive performance in four out of six environments and notably improved performance in the Swimmer environment, surpassing the baseline by approximately 72%. However, it experienced performance degradation in the Ant environment due to suboptimal automatic temperature settings. The Random Network Distillation (RND) method significantly enhanced exploration and performance in Swimmer, outperforming the baseline by about 53%. In other environments, RND did not notably alter final performance, indicating sufficient exploration via entropy maximization alone. Combining RND with automatic temperature tuning in HalfCheetah partially mitigated performance loss. Conversely, uncertainty bonuses from ensemble Q-networks exhibited minimal benefit and often negatively impacted performance due to strong hyperparameter sensitivity and rapid decay of exploration bonuses.

Discussion The effectiveness of exploration methods, particularly uncertainty bonuses, heavily depends on careful hyperparameter tuning, and limited hyperparameter exploration may have contributed to suboptimal results. There is a clear trade-off between performance and usability, with automatic temperature tuning offering significant usability benefits despite potential performance degradation in more complex environments. Random Network Distillation (RND) provided substantial exploratory benefits beyond simple entropy maximization, especially evident in cases where entropy-based exploration alone was inadequate. Specifically, RND effectively addressed exploration shortcomings observed with improper hyperparameter settings, notably in the Swimmer and HalfCheetah environments, highlighting its complementary role in enhancing exploration and overall performance.

Conclusion We extended MaxEntDP by incorporating automatic temperature tuning, Random Network Distillation (RND), and ensemble-based uncertainty bonuses. Temperature autotuning improved usability with minimal performance degradation in most environments. Future research should explore diffusion-specific temperature tuning strategies and theoretically grounded entropy targets. Additionally, more comprehensive hyperparameter studies are needed to fully assess the value of RND and uncertainty bonuses.

Temperature Autotuning and Efficient Exploration in Online MaxEnt Diffusion RL

Javier Nieto
Department of Computer Science
Stanford University
jgnieto@stanford.edu

Abstract

Diffusion models offer powerful capabilities for Reinforcement Learning (RL) tasks due to their expressivity and ability to model multimodal distributions. However, their application has primarily been restricted to offline RL due to practical challenges in online settings. This paper extends the Maximum Entropy Diffusion Policy (MaxEntDP) framework by integrating efficient exploration methods, specifically Random Network Distillation (RND) and uncertainty bonuses derived from ensemble Q-networks, along with automatic temperature tuning inspired by Soft Actor-Critic (SAC). Our experiments on standard MuJoCo benchmarks show that automatic temperature tuning significantly enhances practical usability, maintaining competitive results compared to manual tuning. RND notably improves performance in environments with limited baseline exploration and mitigates the performance hit from using temperature autotuning. However, uncertainty bonuses proved sensitive to hyperparameters, requiring further refinement. We discuss in detail possible reasons why each method worked or not.

1 Introduction

The project's objective is to attempt to augment the MaxEntDP method by experimenting with different efficient exploration schemes. Diffusion models are the state of the art in image generation, and they have also found significant uses in Reinforcement Learning. Particularly, a diffusion model's greater expressivity and multimodality when compared to, for example, Gaussians, make them more useful in complex RL tasks. However, work on applying diffusion models to RL has historically focused on offline methods due to the inherent difficulty of applying diffusion to online RL. A state-of-the-art method to overcome this problem is MaxEntDP (Dong et al., 2025). However, the authors do not explore using further efficient exploration schemes and suggest using an automatic temperature tuning as future work. Therefore, we integrate the efficient exploration schemes of Random Network Distillation and Uncertainty Bonus based on ensemble Q-networks (Burda et al., 2018; Chen et al., 2017). We also use the temperature autotuning method from Soft Actor-Critic (Haarnoja et al., 2019).

The motivation is that diffusion models have already been shown to be greatly successful in offline Reinforcement Learning. Consequently, making them effective in online settings, which can have better performance thanks to the ability to gather new data with the current policy at a given time, could displace state-of-the-art methods.

2 Related Work

Diffusion models have recently been widely explored in RL. For instance, Ying et al. (2025) propose the Exploratory Diffusion Policy (EDP) which introduces a score-based intrinsic reward derived from

a diffusion model trained on replay buffer data, promoting exploration of unseen states. Similarly, Diffusion Augmented Agents (DAAG) utilize diffusion models to relabel past experiences, enhancing sample efficiency in instruction-following tasks (Palo et al., 2024). In online RL, integrating diffusion models poses challenges due to the intractability of computing exact likelihoods and the need for sampling from unknown target distributions, which are unknown in online RL.

To address this, Q-weighted Variational Policy Optimization (QVPO) introduces a variational loss weighted by Q-values, enabling policy optimization with diffusion models in an online setting (Ding et al., 2024). Furthermore, QVPO does entropy regularization by training the model both on sampled data and also uniformly distributed data, which has maximum entropy. However, these approaches often require careful tuning and may not fully leverage the potential of diffusion models for exploration.

Ma et al. (2025) propose SDAC, a novel approach termed Reverse Sampling Score Matching (RSSM). RSSM leverages an insightful interpretation of diffusion models as noise-perturbed energy-based models (EBMs), specifically relying only on the state-action value function (Q-function) as an energy function. This approach eliminates the necessity for sampling directly from the optimal policy distribution. By framing the diffusion policy training problem within the RSSM framework, the authors effectively address the two primary challenges of diffusion models in online RL. Namely, the sampling bottleneck and the computational complexity.

Unlike prior approaches that rely on behavior cloning or computationally expensive full-diffusion backpropagation, QSM updates the denoising network efficiently by minimizing the distance between the score and $\nabla_a Q(s, a)$, enabling stable and sample-efficient off-policy learning (Psenka et al., 2025). MaxEntDP's method and implementation is based on QSM.

Celik et al. (2025) propose DIME derives a tractable lower bound on the maximum entropy objective, enabling efficient training of expressive diffusion-based policies without the need for external exploration noise. The authors propose a provably convergent policy iteration scheme and a practical algorithm that leverages reparameterized stochastic gradients, automatic tuning of key hyperparameters, and CrossQ-style Q-function updates.

3 Background: MaxEntDP

The method which we attempt to improve is Maximum Entropy Diffusion Policy by Dong et al. (2025). It makes two main contributions: a way to estimate the diffusion policy with the Q-function instead of the target actions and a method to approximate the log probability of a given action (necessary to estimate the entropy).

3.1 Q-weighted noise estimation

In SAC, the policy is optimized to approximate the exponential of the Q-function:

$$\pi(a|s) = \frac{\exp(\frac{1}{\beta}Q(s,a))}{Z(s)}, \tag{1}$$

However, since $\pi(a|s)$ in our case is modeled as a denoising diffusion process, it is not trivial to optimize the function. After a series of transitions following the noising equation

$$p(a_t|a_0) = \mathcal{N}\left(a_t|\sqrt{\sigma(\alpha_t)}a_0, \sigma(-\alpha_t)\mathbf{I}\right), \tag{2}$$

the original a_0 must be estimated by using the score function $\nabla a_t \log p(a_t)$ for each intermediate step t. In a traditional diffusion setting, such as image generation, examples $a_0^i \sim p(a_0)$ are known during training and can directly be used as the target. Then, the diffusion policy can be optimized with the loss:

$$\mathcal{L}(\phi) = \mathbb{E}_{p(a_t)}\left[\|\epsilon_\phi(a_t, \alpha_t) - \epsilon^*(a_t, \alpha_t)\|_2^2\right]. \tag{3}$$

However, since the target for our policy is to approximate the exponential of the Q-value, a clever strategy must be used. The authors of MaxEntDP prove that the following expression can be used as the diffusion target:

$$\epsilon^*(a_t, \alpha_t) = -\sum_{i=1}^{K} \text{softmax}\left(\frac{1}{\beta}Q\left(a_0^{1:K}\right)\right)_i \epsilon^i. \tag{4}$$

3.2 Log probability approximation

SAC's policy is arrived at by stochastically maximizing the following objective:

$$J(\pi) = \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right], \tag{5}$$

where $\mathcal{H}(\pi(a_t|s_t)) = -\mathbb{E}_{a_t\sim\pi}[\log \pi(a_t|s_t)]$ is the entropy of π over expected actions. Notably, the entropy requires computing the log probability of an action being the result of a given state $\log \pi(a_t|s_t)$. Although there exists an exact formula (Kong et al., 2023; Wu et al., 2024):

$$\log \pi(a_t|s_t) = -\frac{d}{2}\log(2\pi e) + \frac{d}{2}\int_{-\infty}^{\infty} \sigma(\alpha_t)\mathrm{d}\alpha_t - \frac{1}{2}\int_{-\infty}^{\infty} \mathbb{E}_\epsilon \left[\|\epsilon - \epsilon^*(a_t, \alpha_t)\|_2^2 \right] \mathrm{d}\alpha_t, \tag{6}$$

where $d = \dim(a_0)$, it is intractable to exactly compute this probability, and this limitation has long curtailed the use of diffusion policies. The authors of MaxEntDP propose the following tractable approximation:

$$\log \pi(a_t|s_t) \approx -\frac{d}{2}\log(2\pi e) + \frac{1}{2}\sum_{i=1}^{T} w_{t_i}(d \cdot \sigma(\alpha_{t_i}) - \tilde{\epsilon}_\phi(a_{t_i}, \alpha_{t_i})), \tag{7}$$

where $w_{t_i} = \frac{\sigma(\alpha_{t_{i-1}}) - \sigma(\alpha_{t_i})}{\sigma(\alpha_{t_i})\sigma(-\alpha_{t_i})}$ and $\tilde{\epsilon}_\phi(a_{t_i}, \alpha_{t_i}) = \frac{1}{N}\sum_{j=1}^{N} \|\epsilon^j - \epsilon_\phi(a_{t_i}^j, \alpha_{t_i})\|_2^2$.

4 Methods

Since multiple strategies were attempted, we describe the methodology for each one individually.

4.1 Temperature autotuning

A significant limitation to the adoption of MaxEntDP in real-world applications is the need to tune the temperature hyperparameter α because different environments require different levels of exploration. Similarly to the original SAC, this hyperparameter is brittle and needs to be tuned specifically to each environment. In their conclusion, the authors acknowledge the limitation and briefly suggest the possibility of integrating a temperature autotuning scheme.

As a straightforward attempt, we have implemented the scheme described in the second revision of Soft Actor-Critic Haarnoja et al. (2019), which consists of treating the temperature as a parameter which can change during training to target a specific policy entropy. Thus, the following objective function is iterated on every gradient step:

$$J_\alpha(\alpha) \;=\; \mathbb{E}_{(s,a)\sim\pi}\Big[-\alpha \left(\log \pi(a \mid s) + \mathcal{H}_{\text{target}} \right) \Big]. \tag{8}$$

We have adapted this method to use the approximate log probability defined in Equation 7. Although it makes it necessary to tune the learning rate of α itself and a target entropy $\mathcal{H}_{\text{target}}$, these two hyperparameters are less brittle across tasks. In fact, SAC uses $\mathcal{H}_{\text{target}} = -\dim(a_0)$ successfully across all MuJoCo environments. This is because if the action space were bounded, and we used a uniform policy across the full action space, the entropy would also grow with dimensionality. We decided not to tune $\mathcal{H}_{\text{target}}$ and leave it at $-\dim(a_0)$ since the main advantage of this technique is not the performance gain but the ease of use.

4.2 Random Network Distillation

Random Network Distillation (RND) is an exploration method for reinforcement learning that measures state novelty by quantifying the prediction error of a neural network trained to replicate the output of a fixed, randomly initialized target network (Burda et al., 2018). The target network defines an embedding function $f : \mathcal{O} \to \mathbb{R}^k$, where $\mathcal{O}$ is the space of observations, and f is kept fixed after random initialization. A separate predictor network $\hat{f}(x;\theta)$ is trained via gradient descent to minimize the mean squared error (MSE) between its output and that of the target network. The intrinsic reward used to guide exploration is defined as the prediction error:

$$i_t = \left\| \hat{f}(x_t) - f(x_t) \right\|^2,$$

where x_t is the observation at time t. Since the predictor improves over time on familiar observations, this error serves as a proxy for how novel a state is to the agent.

The rationale behind RND is based on the observation that neural networks trained with gradient descent tend to generalize imperfectly and exhibit low prediction error on inputs similar to their training data, while maintaining higher error on novel inputs. Unlike exploration bonuses based on forward dynamics, which can be corrupted by stochastic transitions (e.g., the "noisy TV" problem), RND uses a deterministic target function, thus eliminating aleatoric uncertainty from the intrinsic reward signal. This makes the prediction error primarily reflect epistemic uncertainty due to lack of prior exposure, which is precisely what is needed for incentivizing exploration. Therefore, RND offers a computationally simple yet effective way to guide exploration in high-dimensional and complex environments.

4.3 UCB Uncertainty bonus with ensemble Q-networks

As explained above, the original MaxEntDP algorithm uses two Q-networks and uses the minimum in each case to mitigate overestimation. Chen et al. (2017) propose a Q-ensemble approach to approximate Bayesian Q-learning, integrating upper-confidence bound (UCB) exploration strategies originally developed for bandit problems. The authors construct an ensemble of independently initialized Q-functions, each estimating optimal action-values. The action selection leverages uncertainty derived from the ensemble, choosing actions based on an upper-confidence bound defined by:

$$a_t = \arg\max_a \left\{ \tilde{\mu}(s_t, a) + \lambda \cdot \tilde{\sigma}(s_t, a) \right\},$$

where $\tilde{\mu}(s_t, a)$ and $\tilde{\sigma}(s_t, a)$ represent the empirical mean and standard deviation across the ensemble's Q-values at a given state-action pair. This makes intuitive sense because it encourages exploration in states and actions about which the ensemble disagrees, thus systematically pushing the agent to explore uncertain yet potentially rewarding areas of the environment. Optimism in the face of uncertainty is a well-established principle: it ensures exploration of uncertain state-action pairs sufficiently often, allowing agents to efficiently gather information about potentially optimal strategies.

5 Experimental Setup

In order to test the effectiveness of the above described methods, we conducted a series of experiments in simulated environments. We chose the MuJoCo v3 environments HalfCheetah, Swimmer, Hopper, Walker, Ant, and Humanoid because they are the industry standard for continuous control tasks and our baseline, MaxEntDP, uses it. That also means that many hyperparameters were already tuned for these environments (including the important and difficult to tune temperature).

First, we ran a baseline experiment on each environment to ensure that the implemented baseline MaxEntRL worked correctly. Next, we ran MaxEntRL with RND and with automatic temperature on all environments to see if they worked well across them. Finally, since upper confidence bound required more hyperparameter tuning we focused our efforts on only two environments: HalfCheetah and Swimmer. We ran five experiments on each one with a variety of hyperparameters. For a complete list, see Table 1

Training occurred over 10^6 steps on an NVIDIA RTX 4090, taking consistently close to 210 minutes across all environments with 80% GPU utilization. The batch size was 256 with 10 episodes of evaluation every $10,000$ steps.

6 Results

6.1 Quantitative results

Figures 1, 2, and 3 are various visualizations of the results. Table 1 contains the numerical results of all plotted experiments. Please see the Figures' captions and the analysis in Subsection 6.2 for more details.

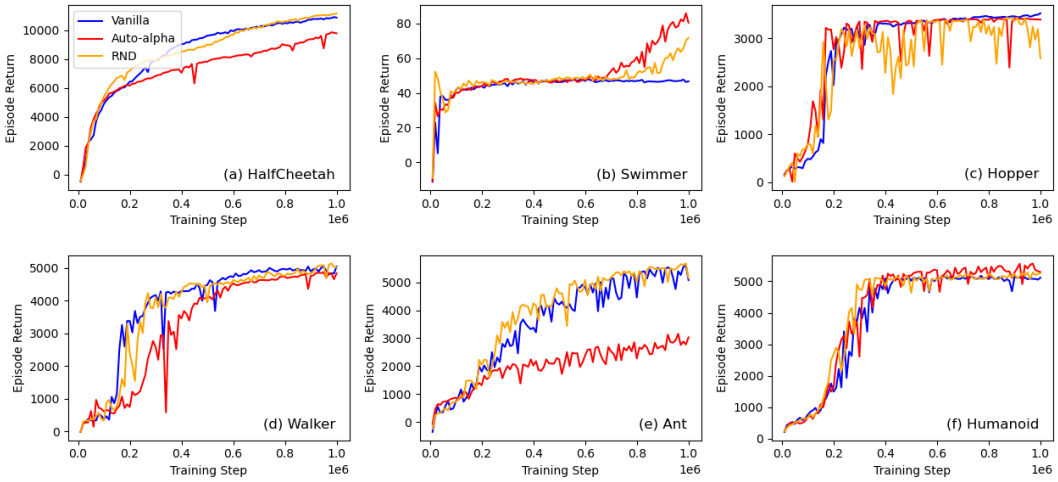

Figure 1: Experiments conducted in six MuJoCo v3 environments with Vanilla MaxEntDP (blue), MaxEntDP with automatic temperature α (red) and MaxEntDP with RND (yellow).

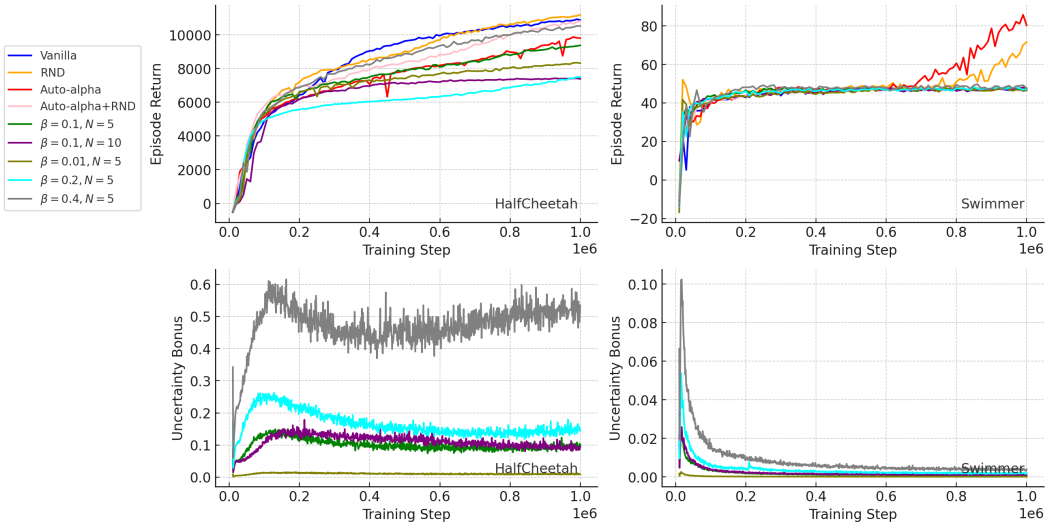

Figure 2: All HalfCheetah (left) and Swimmer (right) experiments, including Vanilla MaxEntDP (blue), MaxEntDP with RND (orange), MaxEntDP with Auto-alpha (red), MaxEntDP with Auto-alpha and RND (pink), and multiple Uncertainty Bonus experiments with different hyperparameters. For Uncertainty Bonus experiments, we also show the uncertainty bonus itself over time.

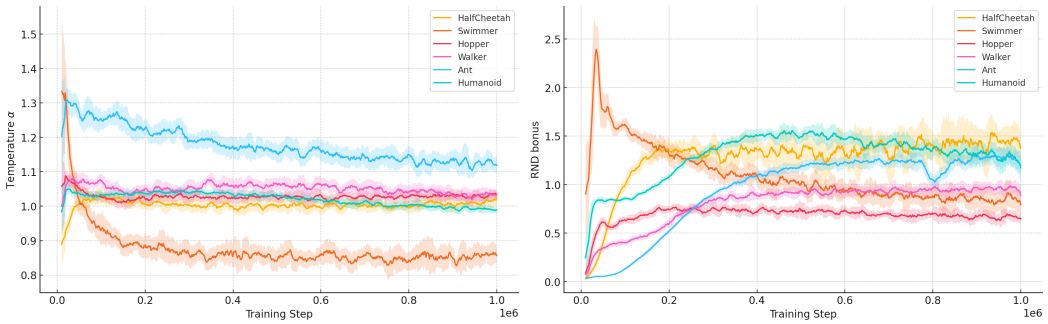

Figure 3: Left shows temperature α over time, normalized such that 1 is the optimal temperature found by the authors of MaxEntDP for each environment. Right shows RND bonus over time for all six tested environments.

Environment	**Name**	**Uncertainty bonus β**	**Num Critics**	**Return (Std)**
HalfCheetah-v3	Vanilla	N/A	N/A	10881.7 (56.5)
	RND	N/A	N/A	**11184.9 (75.1)**
	Auto-alpha	N/A	N/A	9799.4 (127.8)
	Auto-alpha + RND	N/A	N/A	10820.7 (91.8)
	Uncertainty Bonus	0.01	5	8320.2 (83.5)
	Uncertainty Bonus	0.10	5	9373.5 (62.7)
	Uncertainty Bonus	0.10	10	7389.6 (47.2)
	Uncertainty Bonus	0.20	5	7509.4 (84.7)
	Uncertainty Bonus	0.40	5	10523.5 (98.4)
Swimmer-v3	Vanilla	N/A	N/A	46.7 (1.1)
	RND	N/A	N/A	71.6 (1.5)
	Auto-alpha	N/A	N/A	**80.4 (18.3)**
	Auto-alpha + RND	N/A	N/A	47.3 (1.6)
	Uncertainty Bonus	0.01	5	46.6 (1.2)
	Uncertainty Bonus	0.10	5	47.6 (1.0)
	Uncertainty Bonus	0.10	10	47.6 (1.4)
	Uncertainty Bonus	0.20	5	47.3 (1.4)
	Uncertainty Bonus	0.40	5	47.7 (1.8)
Hopper-v3	Vanilla	N/A	N/A	**3524.3 (4.9)**
	RND	N/A	N/A	2585.6 (1097.7)
	Auto-alpha	N/A	N/A	3393.2 (7.8)
Walker2d-v3	Vanilla	N/A	N/A	**5036.1 (48.3)**
	RND	N/A	N/A	5014.3 (70.9)
	Auto-alpha	N/A	N/A	4828.6 (23.4)
Ant-v3	Vanilla	N/A	N/A	5080.8 (1315.8)
	RND	N/A	N/A	**5149.5 (1404.6)**
	Auto-alpha	N/A	N/A	3029.3 (720.4)
Humanoid-v3	Vanilla	N/A	N/A	5113.5 (16.5)
	RND	N/A	N/A	5260.2 (17.1)
	Auto-alpha	N/A	N/A	**5291.7 (54.4)**

Table 1: Experiment results grouped by environment

6.2 Qualitative analysis

Temperature Autotuning Figure 1 shows the performance of temperature autotuning compared to the Vanilla baseline across all environments and Table 1 shows numerical results. In all except Swimmer and HalfCheetah, using temperature autotuning makes no difference to the final results. This result is significant since it allows users of MaxEntDP to integrate the system into their workflow much more easily and without having to tune the brittle temperature hyperparameter. In fact, we see in the Swimmer environment that it can even be better than manual tuning. As we will see, the increased performance of Swimmer is likely that the hyperparameter was tuned to the incorrect number, which goes to show the pitfalls of manual adjustment. However, using temperature autotuning is still a trade-off, as seen in the Ant environment, which took a significant performance hit.

To understand why temperature autotuning's effectiveness varies based on the environment, we turn to Figure 3, which shows the temperature parameter α over time. The figure has been normalized to make 1 be the baseline hyperparameter chosen by the MaxEntDP authors. Two observations provide an explanation for the results.

First, all environments except for Swimmer and HalfCheetah remained close to 1 throughout training. Therefore, running with automatic temperature is almost like running with the manual hyperparameter, and that is why we observe such small deviation in the final results. Second, both Swimmer and Ant deviate significantly (and stay deviated) throughout the run. We can conclude that Swimmer's baseline temperature was incorrectly tuned and that using autotuning helped performance. However, the opposite effect was likely true with Ant, in which α was far from the optimal value throughout the run. It is possible that the relatively high number of dimensions (8, second only to Humanoid), and the fact that it is a challenging environment led to these poor results.

Random Network Distillation Figure 1 shows the performance of RND compared to the Vanilla baseline across all environments and Table 1 shows numerical results. In all cases except for one, RND has no discernible effect on performance. The exception is the Swimmer environment, in which RND significantly outperforms vanilla MaxEntDP. As discussed above, it is likely that the temperature hyperparameter for the Swimmer environment had been misconfigured. Assuming that the effect of this mistuning is that effectively no efficient exploration technique was being used, the addition of RND efficient exploration should boost performance.

All other environments likely already enjoyed sufficiently good efficient exploration thanks to entropy maximization at the core of MaxEntDP and so no performance improvement was observed. This effect is seen in Figure 3, which shows the RND bonus over time. The bonus which looks most like what we would expect is that of Swimmer, with more exploration at the beginning and a steady decrease throughout training. In contrast, the other environments maintain or even increase the RND bonus.

However, Figure 2 shows that RND does provide a significant benefit when used in combination with temperature autotuning in HalfCheetah. Although it is less performant than the baseline MaxEntDP, it significantly mitigates the performance loss from using temperature autotuning.

Uncertainty Bonus Figure 2 shows the performance of multiple experiments with Uncertainty Bonus compared to the Vanilla baseline in the HalfCheetah and Swimmer environments, as well as the value of the bonus over time. Table 1 shows numerical results. Using uncertainty bonus did not show any significant improvement in Swimmer, and it hurt performance in HalfCheetah. The corresponding bonus-over-time plot reveals that early in training, the bonus was high, indicating large epistemic uncertainty, but it decayed rapidly, often too quickly to sustain meaningful exploratory behavior. In Swimmer, no configuration led to performance improvements, consistent with a flat learning curve and persistently low bonus values throughout training. These results suggest that the added exploration incentive was either redundant in simpler environments or poorly aligned with the training dynamics in more complex ones.

7 Discussion

A notable limitation of our study is the relatively shallow hyperparameter search conducted for each method, particularly for the Uncertainty Bonus experiments. While we explored a range of

values for β and the number of ensemble critics, the chosen configurations were coarse and may not represent the optimal balance between exploration and exploitation. Diffusion-based policies, given their sensitivity to noise and stochasticity during training, may require more targeted hyperparameter tuning to reveal their full potential under new exploration schemes. As such, the underperformance of some methods, particularly in HalfCheetah, should not be interpreted as definitive failure, but rather as an indication that further tuning could be necessary.

Random Network Distillation (RND) appears to show meaningful promise even when used alongside MaxEnt-based methods, which already contain a form of intrinsic exploration through entropy regularization. In Swimmer, RND significantly outperformed the baseline, and in HalfCheetah it helped recover much of the performance lost when paired with automatic temperature tuning. This suggests that RND captures a complementary aspect of exploration not fully addressed by maximum entropy objectives. Given its simplicity, computational efficiency, and empirical benefit in select environments, we believe RND warrants further investigation and inclusion in future work on diffusion-based reinforcement learning.

While temperature autotuning did not improve performance in all environments, its inclusion simplifies the use of MaxEntDP considerably. The trade-off between performance and usability is central here: in environments where manual tuning is brittle or infeasible, autotuning provides a viable alternative. The results in Swimmer show clear benefits when the baseline temperature was poorly chosen, while other environments maintained near-baseline performance with autotuning. Thus, despite some degradation (e.g., in Ant), this method could be key to making diffusion policies more practical in broader applications where manual tuning is prohibitively expensive or unreliable.

Finally, while the Uncertainty Bonus based on ensemble Q-networks performed poorly in our experiments, this result should be interpreted with caution. Our exploration of hyperparameters was limited, and the technique may be particularly sensitive to the ensemble architecture and the choice of β. Therefore, we cannot conclusively dismiss its utility based on the present results alone.

8 Conclusion

In this study, we augmented the MaxEntDP algorithm with exploration methods including temperature autotuning, Random Network Distillation (RND), and uncertainty bonuses using ensemble Q-networks. Our results demonstrate that integrating automatic temperature tuning enhances usability significantly, albeit with minor trade-offs in performance for certain environments. Notably, RND provided meaningful complementary exploration, particularly evident when temperature hyperparameters were suboptimally configured.

Future work should attempt to develop an auto-α method that is specific to diffusion policies. Perhaps theoretical work can show a better optimal entropy target than $\dim(a_0)$. To confirm whether using RND and Uncertainty Bonuses significantly improves performance, more exhaustive hyperparameter searches should be conducted.

Changes from Proposal The original proposal contemplated iterating similar improvements on the SDAC paper. I pivoted away from it because that paper was too complex and left several unclear points which made replication impossible. When I got in contact with the authors, they stated that they would release a new version of the paper by June, so any work on the old one would have become moot almost immediately.

References

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018. Exploration by Random Network Distillation. arXiv:1810.12894 [cs.LG] `https://arxiv.org/abs/1810.12894`

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palanicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. 2025. DIME:Diffusion-Based Maximum Entropy Reinforcement Learning. arXiv:2502.02316 [cs.LG] `https://arxiv.org/abs/2502.02316`

Richard Y. Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. 2017. UCB Exploration via Q-Ensembles. arXiv:1706.01502 [cs.LG] `https://arxiv.org/abs/1706.01502`

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. 2024. Diffusion-based Reinforcement Learning via Q-weighted Variational Policy Optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=UWUUVKtKeu

Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. 2025. Maximum Entropy Reinforcement Learning with Diffusion Policy. arXiv:2502.11612 [cs.LG] https://arxiv.org/abs/2502.11612

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. 2019. Soft Actor-Critic Algorithms and Applications. arXiv:1812.05905 [cs.LG] https://arxiv.org/abs/1812.05905

Xianghao Kong, Rob Brekelmans, and Greg Ver Steeg. 2023. Information-Theoretic Diffusion. arXiv:2302.03792 [cs.LG] https://arxiv.org/abs/2302.03792

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. 2025. Efficient Online Reinforcement Learning for Diffusion Policy. arXiv:2502.00361 [cs.LG] https://arxiv.org/abs/2502.00361

Norman Di Palo, Leonard Hasenclever, Jan Humplik, and Arunkumar Byravan. 2024. Diffusion Augmented Agents: A Framework for Efficient Exploration and Transfer Learning. arXiv:2407.20798 [cs.LG] https://arxiv.org/abs/2407.20798

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. 2025. Learning a Diffusion Model Policy from Rewards via Q-Score Matching. arXiv:2312.11752 [cs.LG] https://arxiv.org/abs/2312.11752

Yunshu Wu, Yingtao Luo, Xianghao Kong, Evangelos E. Papalexakis, and Greg Ver Steeg. 2024. Your Diffusion Model is Secretly a Noise Classifier and Benefits from Contrastive Training. arXiv:2407.08946 [cs.LG] https://arxiv.org/abs/2407.08946

Chengyang Ying, Huayu Chen, Xinning Zhou, Zhongkai Hao, Hang Su, and Jun Zhu. 2025. Exploratory Diffusion Policy for Unsupervised Reinforcement Learning. arXiv:2502.07279 [cs.LG] https://arxiv.org/abs/2502.07279