This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Regularized Parameter Uncertainty for Improving Generalization in Reinforcement Learning

Pehuen Moure Longbiao Cheng Joachim Ott Zuowen Wang Shih-Chii Liu

Institute of Neuroinformatics, ETH Zurich and University of Zurich
{pehuen, longbiao, jott, zuowen, shih}@ini.ethz.ch

Abstract

In order for reinforcement learning (RL) agents to be deployed in real-world environments, they must be able to generalize to unseen environments. However, RL struggles with out-of-distribution generalization, often due to over-fitting the particulars of the training environment. Although regularization techniques from supervised learning can be applied to avoid over-fitting, the differences between supervised learning and RL limit their application. To address this, we propose the Signal-to-Noise Ratio regulated Parameter Uncertainty Network (SNR PUN) for RL. We introduce SNR as a new measure of regularizing the parameter uncertainty of a network and provide a formal analysis explaining why SNR regularization works well for RL. We demonstrate the effectiveness of our proposed method to generalize in several simulated environments; and in a physical system showing the possibility of using SNR PUN for applying RL to real-world applications.

1. Introduction

Reinforcement Learning (RL) has emerged as a powerful framework for teaching agents to make decisions through trial-and-error interactions with dynamic environments. However, a critical challenge arises when agents overfit to their training environments and fail to generalize to new, unseen domains – a problem that is central to the deployment of RL in real-world scenarios [22, 27]. Regularization techniques have been proposed as a solution to this generalization dilemma. These techniques can be categorized into two distinct approaches: noise injection and parameter regularization.

Noise injection strategies, such as entropy-based action regularization [10, 45], state-dependent noise injection [36], and dropout-induced parameter noise [16, 39], introduce stochasticity into the agent's policy or model parameters to promote exploratory behavior and prevent over-reliance on specific features of the training environment . However, the downside of these methods is a potential destabilization of the training process, as the added variability can interfere with the agent's ability to learn an effective policy [11].

On the other hand, parameter regularization techniques aim to directly control the complexity of the model. Weight decay and Lasso regularization are common approaches that penalize large weights, encouraging the network to prefer simpler functions that may generalize better [20, 28, 29]. Despite the effectiveness of parameter regularization techniques in supervised learning, in the context of RL, these approaches can lead to oversimplified networks that lack the capacity to capture the complexities of the environment, limiting the agent's ability to learn the task [44].

Parameter uncertainty represents an intriguing middle ground, potentially combining the explorative benefits of noise injection with the structural discipline of parameter regularization. Although it is well studied in supervised learning [6, 23], and some methods have been applied for exploration in RL [12, 33, 35], they have not been applied to generalization. Additionally, such methods do not properly regulate the parameter uncertainty due to the difficulty in applying standard Kullback–Leibler divergence regularization [12, 35] (see Section 4.1), which can hinder their application for generalization in RL.

In this paper, we propose a regularized parameter uncertainty method for improving generalization in reinforcement learning. The main contributions of this work are summarized as follows:

- We propose a signal-to-noise (SNR)-regulated parameter uncertainty network (SNR PUN). This approach first introduces a parameter-specific ratio of the mean over the standard deviation for each distribution, referred to as the *signal-to-noise ratio (SNR)*, as a measure of parameter uncertainty. Then, instead of using KL divergence to constrain the parameter distribution into a predefined prior, we limit each parameter's uncertainty below a maximum SNR. This regulation allows for both effective training and enhanced network generalization.
- We provide a formal analysis on the relationship be-

tween our proposed max SNR regularization and KL divergence. This analysis indicates that SNR regularization is equivalent to calculating the KL divergence against a broader range of prior distribution sets for each parameter, offering a novel perspective on parameter uncertainty regularization.
- We further integrate the proposed SNR PUN into a Proximal Policy Optimization method for RL. By modifying the parameter sampling strategy, separating the actor and critic networks, and employing a new initialization method, we maximize the network's uncertainty, aims to achieve better generalization, while still ensuring successful training.
- We demonstrate our method's improvement in unseen environments across several RL tasks, highlighting its correlation with better transferability to real-world systems; and underscoring the utility beyond simulated shifts to address the unpredictable variations of a physical system.

2. Related Work

Generalization in RL Network generalization problem is a well-studied area. Readers are referred to Wang et al. [42] for a comprehensive survey. Methods proposed for improving the generalization in RL typically concentrated on the reward function or transition dynamics. In robotics, this issue has been studied extensively, particularly when the exact dynamics of the environment are poorly defined or entirely different from the simulated environment [21]. Randomization using noise has been applied to the input state space [3] or the transition dynamics [34]. Adversarial noise insertion into the domain space has been explored by Mandlekar et al. [26]. Many of the aforementioned methods focus on utilizing noise on the inputs or outputs and not throughout the network. Igl et al. [19] addressed this issue by selectively inserting noise to one latent state to avoid excessive perturbation of the gradient estimation. The authors in Igl et al. [19] and Lu et al. [24] argue that generalization is improved by constraining the available information during training. Still, the noise that can be inserted is applied to only the input, output, or part of the latent space.

Parameter Uncertainty in RL Parameter uncertainty allows for noise to be incorporated into deep neural networks by treating each parameter as a distribution rather than a fixed value [1, 15]. We refer to these methods as Parameter Uncertainty Network (PUN)s. The use of variational inference for parameter uncertainty in RL is a well-studied area [31, 32]. The original variational parameter uncertainty work, Blundell et al. [5], applied PUN to a simple contextual bandits problem. Fortunato et al. [12] applied PUNs to a more complex set of environments and a variety of policy gradient methods to improve exploration. These methods are limited because they do not explicitly regulate the learned distributions. In RL these PUNs are challenging to regulate in standard ways because of the noisy learning. Our work differs from previous approaches because it offers a novel regularization method for PUNs.

3. Preliminaries

3.1. Bayes-by-backprop

The most common method for implementing the parameter uncertainty in a neural network is Bayes-by-Backprop (BBB), where the parameters of a network $\theta = \{\theta_1, \ldots, \theta_N\}$ are learned not as a fixed, deterministic value but as a Gaussian distribution, $\theta_i \sim \mathcal{N}(\mu_i, \sigma_i)$ [25]. The standard deviation σ_i is ensured to be positive by applying the softplus function to auxiliary parameters $\beta_i \in \mathbb{R}$, i.e., $\sigma_i = \log(\exp(\beta_i) + 1)$. During the forward pass of the network, parameter values are sampled from these distributions:

$$\epsilon_i \sim \mathcal{N}(0, 1) \tag{1}$$

$$\theta_i = \mu_i + \log(\exp(\beta_i) + 1) \cdot \epsilon_i \quad \forall i \in N \tag{2}$$

The training process of BBB methods revolves around updating these distributions, represented by μ_i and β_i, to fit the data while ensuring they stay close to the predefined, fixed prior distributions. The loss function is accordingly defined as:

$$\mathcal{L}(\mathcal{D}; \theta) = -\mathbb{E}_{q(\theta)}[\log p(\mathcal{D}|\theta)] + c_\beta * \mathrm{KL}\left[q(\theta)||p(\theta)\right], \tag{3}$$

Here, $\mathcal{D}$ denotes the dataset, $q(\theta)$ is the learned parameter distributions and $p(\theta)$ refers to the predetermined prior distribution. The first negative log-likelihood term is used for training the network to fit the data while the Kullback-Leibler (KL) divergence term is instrumental in maintaining the learned distributions close to their predetermined priors to prevent overfitting. c_β is a scaling factor.

3.2. Proximal Policy Optimization

In RL the most commonly used optimization method is Proximal Policy Optimization (PPO) [38], which has become a benchmark in RL problems due to its empirical robustness and ease of implementation [18]. PPO is an actor-critic algorithm that has two components: the actor decides the actions to be taken, and the critic evaluates the actor's choices. In most approaches, the actor and the critic share a network backbone.

We omit many of the PPO details for brevity and will focus on the loss function. The PPO loss function has three terms: a clipped surrogate objective $\mathcal{L}_t^{\text{CLIP}}$, a value function error term $\mathcal{L}_t^{\text{VF}}$, and an entropy bonus term $S[\pi_\theta](s_t)$ [38]:

$$\mathcal{L}_t^{\text{PPO}}(\theta) = \mathop{\mathbb{E}}_{t}\left[\mathcal{L}_t^{\text{CLIP}}(\theta) - c_1\mathcal{L}_t^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t)\right] \tag{4}$$

here, c_1, c_2 are coefficients. The clipped surrogate objective serves to prevent the new policy from deviating too far from the old policy by limiting the size of the policy update. The value function error term measures the squared difference between the predicted and target state values. The entropy bonus term encourages exploration and prevents the policy from becoming too deterministic.

4. Methods

In this section, we begin by analyzing the challenges of using the BBB framework to improve the generalization ability of trained networks. To address these challenges, we propose infusing uncertainty into network parameters using Signal-to-Noise ratio (SNR)-based regularization. Furthermore, we introduce a training scheme to ensure viable and stable training outcomes.

4.1. Improving Regularization in BBB

In BBB, the Kullback-Leibler (KL) regularization term is a fundamental component for constraining the parameter distributions, as mathematically expressed in Eq. 3. Ideally, $\mathrm{KL}\left[q(\theta)||p(\theta)\right]$ should be minimized to ensure $q(\theta)$ aligns closely with $p(\theta)$. However, the divergence can increase significantly when the posterior mean μ_i for each parameter θ_i diverges from zero, leading to a network behavior that might have limited expressivity if the divergence form the prior is overly regulated.

The RL framework compounds these challenges. RL optimization often has noisy gradient estimates. The inherent noise in these estimates is further complicated by the KL term in BBB. As such, the KL divergence term is often dropped (or set to a value that is effectively 0) when BBB is used for training in RL [12, 35]. As a consequence, the learned distributions are under-regulated, often collapsing each distribution's standard deviation σ_i to zero.

4.2. SNR Constrained Parameter Uncertainty

To mitigate the challenges associated with regularization and to enable effective training in BBB networks, while still maintaining robust learning and network generalization, we propose a SNR-based regularization method. This method is designed to finely control parameter uncertainty, allowing for sufficient model stochasticity without incurring the risk of training collapse.

We define the parameter-specific SNR as:

$$\Omega_{\mathrm{SNR}_i} = \frac{|\mu_i|}{\sigma_i} \tag{5}$$

where μ_i represents the mean and σ_i represents the standard deviation of the i-th parameter in the network. This ratio quantifies the extent of noise relative to the signal in the parameter distribution.

Without regularization, network parameters tend to exhibit high SNRs after training, indicating less uncertainty in parameters and leading to more deterministic behavior. This reduction in parameter uncertainty may yield stable and high-performance results on the training dataset, yet it can lead to overfitting, constraining the network's generalizability to new environment configurations. Conversely, lower SNR values, indicative of increased parameter uncertainty, are associated with wider probability distributions and a flatter loss landscape, which are favorable for generalization.

Max SNR Regularization To control the parameter-specific Ω_{SNR_i} and prevent overfitting within the uncertainty models used for RL, we introduce a regularization term that penalizes deviations of the SNR from a predefined maximum threshold, $\Omega_{\text{Max SNR}}$, which is realized by the following SNR loss term:

$$\mathcal{L}^{\mathrm{SNR}} = \sum_i \left[\max\left(\Omega_{\mathrm{SNR}_i} - \Omega_{\text{Max SNR}}, 0\right)\right]^2 \tag{6}$$

where Ω_{SNR_i} denotes the SNR for the i-th parameter distribution as defined in Eq. 5, and $\Omega_{\text{Max SNR}}$ signifies the upper limit for the SNR. The SNR loss term enforces a boundary on the SNR, ensuring that the network does not select pdfs with an SNR exceeding this limit. This approach imposes a constraint that, while allowing the weights to vary in magnitude, ensures that an appropriate level of noise is incorporated across the parameter spectrum. This trade-off between precision and variability, achieved through the Max SNR regularization, is essential for successfully adapting trained networks to new environments.

Comparing SNR Regularization to KL Divergence Both the proposed SNR Regularization and the conventional KL divergence aim to control the distribution of network parameters, yet they operate on different principles that affect parameter constraints.

On the one hand, the SNR regularization is equivalent to calculating the KL divergence for a parameter $\theta_q = q \sim \mathcal{N}(\mu_q, \sigma_q)$ with the prior $p \sim \mathcal{N}(\mu_p, \sigma_p)$ such that it is the closest value that does not violate the SNR constraint and that $\sigma_p = \sigma_q$. This is mathematically depicted as:

$$\mathrm{KL}[q\|p] = \frac{1}{2}\left[\frac{(\mu_q - \mu_p)^2}{\sigma_p^2} + \frac{\sigma_q^2}{\sigma_p^2} - 1 + \ln\left(\frac{\sigma_p^2}{\sigma_q^2}\right)\right] \tag{7}$$

$$\overset{\sigma_p=\sigma_q}{=} \frac{1}{2}\left[\frac{|\mu_q - \mu_p|}{\sigma_p}\right]^2 \tag{8}$$

$$\overset{\mu_p\mu_q>0}{=} \frac{1}{2}\left[\frac{|\mu_q|}{\sigma_q} - \frac{|\mu_p|}{\sigma_p}\right]^2 \tag{9}$$

$$= \frac{1}{2}\left[\Omega_{\mathrm{SNR}_q} - \Omega_{\text{Max SNR}}\right]^2 = \frac{1}{2}\left[\mathcal{L}^{\mathrm{SNR}}(q)\right] \tag{10}$$

This equivalence, valid as $\sigma_p = \sigma_q$ and μ_p and μ_q always

share the same sign, simplifies the KL divergence to the squared SNR difference between the posterior and prior.

On the other hand, the differences between SNR and KL regularization are evident when examining their impact on parameter distributions, as illustrated in Figure 1. The standard KL term, with its zero-mean Gaussian constraint, results in a more pronounced regularization effect, requiring a smaller constant in the loss function. In contrast, SNR regularization allows a wider range of parameter distributions and can be more heavily penalized during training. This flexibility in SNR regularization is advantageous as it permits a larger selection of the parameter space. Networks trained under SNR are more likely to find solutions that are not only optimal for the current loss but also robust to variations in the parameter space. This robustness contributes to an improved generalization ability of the network, highlighting a key advantage of SNR over the more restrictive KL regularization.

Figure 1 highlights the impact of the KL divergence term $\mathcal{L}_{\text{KL}}(\theta_i)$ and the challenges it presents in the parameter update process. In RL, where gradient estimates are often based on incomplete or imperfect environmental information [43], the sensitivity to the KL term is more accentuated.

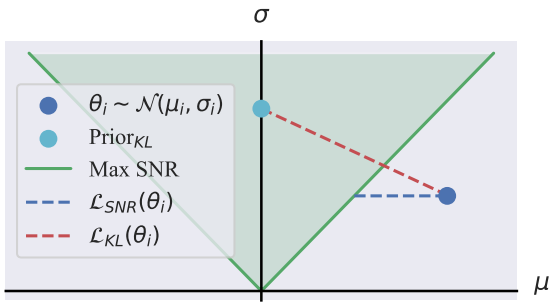

Figure 1. Visualization of parameter-wise KL and SNR regularization term effects on a parameterized distribution θ_i in a PUN (given $\theta_i \sim \mathcal{N}(\mu_i, \sigma_i)$). $\mathcal{L}_{\text{KL}}$ will always be greater than or equal to $\mathcal{L}_{\text{SNR}}$.

4.3. Training Scheme

Integrating the Max SNR loss term, i.e., Eq. 6, into the PPO framework, the modified loss function used in our method is presented as:

$$\mathcal{L}_t^{\text{PPO+SNR}}(\theta) = \mathop{\mathbb{E}}_t \left[\mathcal{L}_t^{\text{CLIP}}(\theta) - c_1\mathcal{L}_t^{\text{VF}}(\theta)\right] + c_2\mathcal{L}^{\text{SNR}} \quad (11)$$

Here, c_1 and c_2 are coefficients; $\mathcal{L}^{\text{CLIP}}$ and $\mathcal{L}^{\text{VF}}$ are the clipping loss and value function loss terms in Eq. 4. To maximize the network's uncertainty while still ensuring successful training, we propose a novel training scheme that incorporates a new sampling strategy, a deterministic critic and a new network initialization method.

Sampling Strategy Our approach differs from traditional methods that sample parameters only once per simulation rollout. We adopt a strategy of continuously resampling the parameters of a stochastic model at every timestep within each simulation. This method introduces a higher level of stochasticity, enhancing the model's adaptability and robustness across various scenarios. The increased stochasticity from this sampling strategy impacts our loss function. In contrast to the standard PPO framework, which uses an entropy bonus to encourage exploration and prevent overly deterministic policies, our continuous resampling inherently makes the policy less deterministic. Therefore, we omit the entropy bonus in our loss function.

Stabilizing Advantage Estimates Our work introduces a novel approach using the SNR constrained PUN, which has a highly probabilistic nature, as the backbone for policy generation, aimed at improving generalization. This benefit comes at the cost of complicating gradient estimates, thereby introducing additional noise into the training process. As a result, this highly stochastic backbone can lead to unstable advantage estimates, posing challenges during training. To address this, we propose a strategy to enhance training stability by using separate actor and critic. Specifically, the SNR constrained PUN is used as the actor, while another deterministic network is utilized for the critic. This approach not only stabilizes the loss function during training but also capitalizes on the benefits of stochastic elements.

Network Initialization The μ_i and σ_i for each parameter are initialized as follows: Each μ_i is sampled from $\mathcal{U}(-\sqrt{\frac{2}{p}}, \sqrt{\frac{2}{p}})$ where p is the number of inputs to each linear layer following He et al. [17]. Each σ_i is sampled from $\mathcal{U}(\sigma_{min}, \sigma_{max})$ where σ_{min} and σ_{max} are tuned hyperparameters.

We did a hyper-parameter search to select the values of σ_{min} and σ_{max} for our environments. These values are an order of magnitude larger than the σ_{min} and σ_{max} values in previous work [12]. We found that using higher sigma values during initialisation lead to better generalisation results from the network.

5. Experimental Setup

Evaluation of the efficacy of our approach is measured by mutating the base environments after training and with no further adjustment; these environments are defined in Section 5.1. All models are trained using PPO (with our changes for the stochastic models), and each combination of parameters is evaluated with several seeds for each possible environment configuration.

Model	Experiment	Cartpole	Acrobot	Quadruped	Walker
MLP	–	529.64 ± 10.19	160.58 ± 19.07	670.57 ± 28.82	212.96 ± 14.89
	+Dropout	597.61 ± 9.67	148.52 ± 23.28	713.87 ± 2.81	180.84 ± 15.84
	+gSDE	571.46 ± 19.51	–	688.08 ± 8.21	166.03 ± 10.38
	+Entropy Maximization	662.03 ± 17.77	145.47 ± 27.96	540.25 ± 45.91	301.19 ± 23.48
	+Weight Decay	642.83 ± 21.31	213.71 ± 30.20	702.04 ± 3.53	210.87 ± 15.27
PUN	–	534.43 ± 48.40	246.57 ± 22.55	738.95 ± 3.91	250.11 ± 16.28
	+Dropout	500.09 ± 64.89	414.34 ± 31.91	686.79 ± 1.71	179.18 ± 13.86
VIB	–	622.72 ± 23.06	302.95 ± 34.92	716.11 ± 3.77	255.27 ± 18.41
	Use Mean	623.06 ± 17.84	303.51 ± 34.93	739.84 ± 16.93	259.18 ± 17.81
SNR PUN	–	670.48 ± 13.42	421.77 ± 32.74	750.47 ± 3.23	**310.96** ± 16.96
	Sample Once	664.88 ± 24.35	**423.43** ± 50.98	**751.21** ± 3.44	309.56 ± 17.33
	Use Mean	**693.86** ± 12.21	417.32 ± 31.97	749.13 ± 3.41	310.70 ± 17.01

Table 1. Model performance measured using Average Task Performance (ATP) as defined in Section 5.2. The standard errors are calculated per-environment configuration and then averaged. The best metrics are highlighted in bold. Our approach is denoted as 'SNR PUN' and the parameter uncertainty model that did not include the $\mathcal{L}^{\text{SNR}}$ is denoted as 'PUN'. In the experiment column, '-' means models without any regularization term, '+Dropout' means dropout is included during training, '+gSDE' means state exploration is included (for Acrobot, gSDE is omitted as it cannot be applied to discrete action spaces), '+Entropy Maximization' means models include entropy maximization, '+Weight Decay' means models include a weight decay regularization term, 'Sample Once' means models that are sampled once per simulation rollout.

5.1. Environments

Experiments are conducted on several control environments, various continuous control tasks and a discrete control task. The tasks are briefly described here, along with the key changes we made to the environment, as well as the domain parameters that were mutated during the evaluation.

CartPoleSwingUp The cart-pole task is a classic control problem where a pole is attached to a cart that slides along a track [4]. The environment's reward function combines the cart's position on the track and the pole's angle from vertical. We use the environment proposed in Gaier and Ha [13] and Gal et al. [14]. The mutable parameters in our domain generalization task include the pole length, the pendulum mass, and the magnitude of the force that can be applied to the cart.

Acrobot-v1 Acrobot is a two-link robot with one end fixed and one end free. The task objective is for the free end of the chain to reach a particular height. The environment is described in Sutton [40] and Sutton and Barto [41]. The mutable parameters here are the link length, link mass, and the magnitude of torque that can be applied to the links. In the base environment, both links have the same values for length and mass. This symmetry is maintained in our study.

Quadruped is a four-legged robot with each leg consisting of two joints; the objective is to move forward with a desired speed and receives a reward proportional to its stability and forward speed, minus energy usage costs [37]. The environment has an observation space of 78 dimensions and an action space of 12 dimensions. The shin length, joint damping, and contact friction are mutable parameters.

Walker is a two-legged extension of the hopper environment defined in Durrant-Whyte et al. [9]. Similar to the quadruped, the agent receives a reward for forward locomotion minus some energy costs. The environment has an observation space and action space of 18 and 6 dimensions respectively. The mutable parameters are thigh length, joint damping, and contact friction are mutable parameters. Both the Walker and Quadruped are implemented in Dulac-Arnold et al. [8].

5.2. Evaluation Parameters and Baselines

We evaluate our approach by performing a combinatorial search over the range of possible values for each environment parameter. These parameter ranges, were proposed in previous work (Akl et al. [2] and Dulac-Arnold et al. [7] as meaningful ranges for our environments. Each combination of environment parameters is evaluated with 10 seeds and the simulations were carried out for 1000 time-steps. The environment configuration space is discretized into sections for each mutable environment parameter, these sections are defined in Tables 2a and 2b of the appendix. This totals 80000 evaluation simulations for CartPoleSwingUp and Acrobot-v1, and 10000 total simulations for Quadruped and Walker. In CartPoleSwingUp, the maximum possible score is 1000, and the minimum possible score is 0. In Acrobot-v1, the maximum possible score is 1000, and the minimum possible score is 0. In Quadruped and Walker the maximum score is approximately 1200. A higher score indicates better performance in all environments.

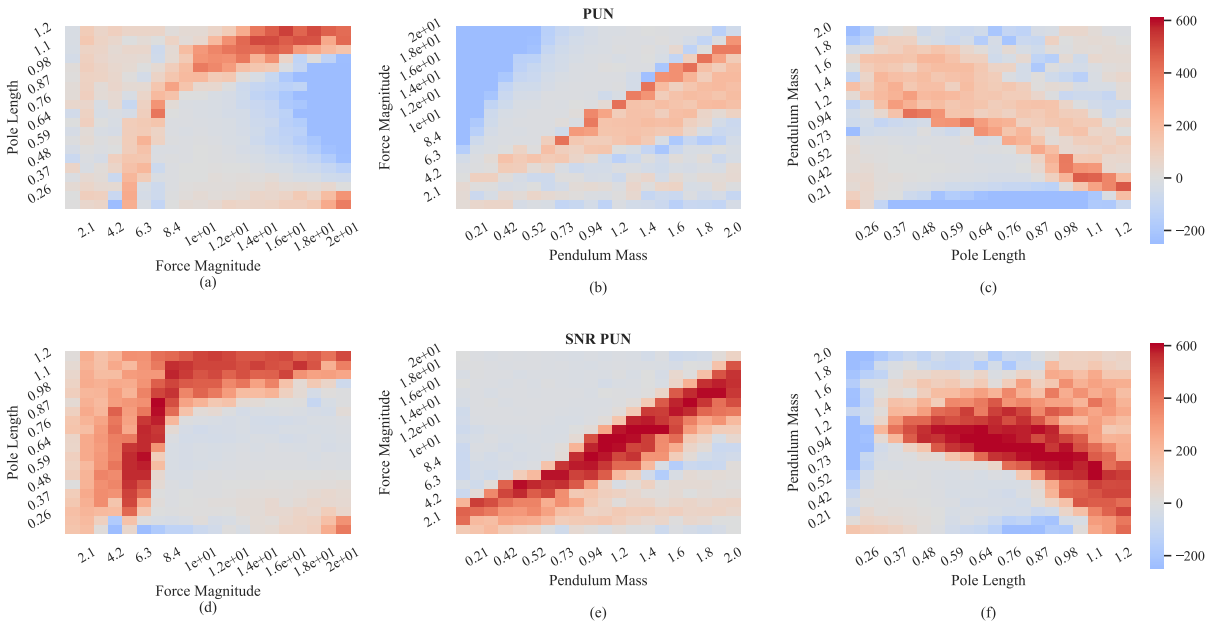

Figure 2. Three heatmaps that show the difference in mean performance between the SNR PUN and PUN models as a function of two of the three CartpoleSwingUp environment parameters while the other parameter is set to a fixed value. The configurations where our model performs better is shown in red, and where the baseline performs better is shown in blue.

Performance Measurement We use the ATP evaluation metric for all models. Formally: $ATP = \frac{1}{M}\sum_m \overline{R}_m$ where $\overline{R}_m$ is the average reward (over all seeds).

The goal of ATP is to capture the average performance over all environment configurations. We intentionally set a large enough search space such that all models fail in some configuration; as such, reporting the worst-/best-case performance is not particularly informative.

Baselines Five additional methods are included for baseline comparison. The first baseline uses dropout regularization. The second baseline is gSDE, a state-of-the-art exploration method proposed by Raffin et al. [36]. gSDE injects state-dependent noise into the latent space of the network during training. Although not explicitly designed for generalization, it provides a meaningful baseline for evaluating the role of injected noise during training and offers a more principled method for noise injection than random noise in the observation or action space. The third baseline method is maximum entropy (ME) RL [45]. This approach introduces an additional regularization term during training to encourage more stochastic policy distributions, that has been shown to improve robustness in RL [10]. We also compare with weight decay [44]. Lastly, we compare with a model that uses a Variational Information Bottleneck (VIB), following the work in Igl et al. [19] and Lu et al. [24] showing VIB's success in generalization for RL.

6. Results and Discussion

6.1. Comparison Between Baselines

Table 1 presents our models' performance compared to the baselines for all environments. Our model, **SNR PUN**, outperforms the baseline models in all environments. We find VIB provides the closest performance to our models. Recent work has shown that layers of stochastic neural networks, like PUNs, perform approximate information compression, as such our method is similar to VIB [30].

Additionally, we find that dropout performs relatively well in Quadruped, while it struggles to show the same performance gain in the other environments. Whereas dropout introduces stark changes in the gradient, PUNs present a much smoother parameter perturbation to the loss function. Interestingly, in Acrobot, we see a gain in combining PUN with Dropout, possibly due to the environment's chaotic nature.

Furthermore, our empirical results show that using the SNR regularization methods consistently improves the performance of the PUN models. We find the model with parameters sampled once per rollout (SNR PUN - Sampled Once) performs slightly worse than the SNR PUN model but still outperforms all baselines, showing the stability of the learned parameter distributions. The overall performance gain in our model indicates that it has better generalization properties to unseen environments.

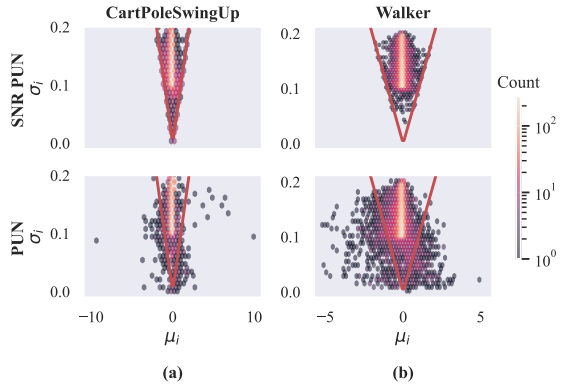

Figure 3. Comparison of the parameter distributions as a function of the mean value for (a) and (b) CartpoleSwingUp and (c) and (d) Quadruped. The maximum possible SNR boundary ($\Omega_{\text{Max SNR}}$) is shown by the red line and the distributions of the stochastic models are shown as heatmaps. The heatmaps represent the frequency with which each combination of μ_i and σ_i is selected by each of the respective models.

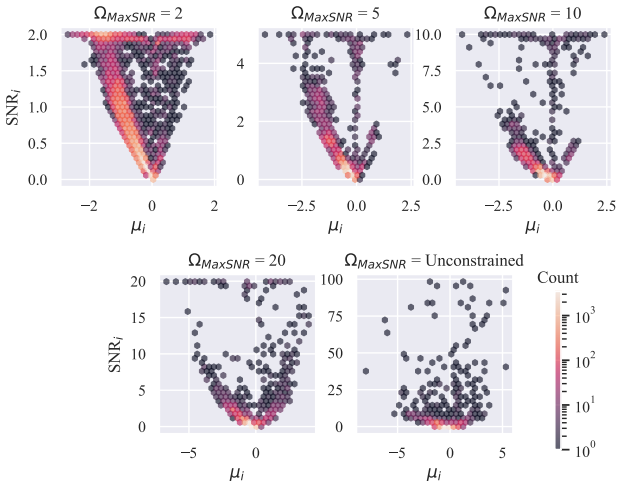

Figure 4. Visualization of selected distributions for different $\Omega_{\text{Max SNR}}$ values and the unconstrained case on the CartpoleSwingUp environment.

6.2. Environment Case Study

Next, we focus on CartpoleSwingUp due to its ease of interpretation and implementation.

Figure 2 shows 2D slices of the evaluation space for each pair of the CartpoleSwingUp environment parameters, while the other parameter is set to a fixed value; this subset can be used to infer configurations where one model outperforms the other. In general, the SNR PUN model outperforms or performs equally as well as the MLP over all environment configurations, shown in Figure 2(d-f). In limited cases, the SNR PUN appears to performs worse, e.g. Figure 2(c) with light poles. In these scenarios, the MLP often spins extremely small and light poles at the center of the track without properly countering the momentum of the pole, whereas the SNR PUN does not build sufficient momentum to lift the pole. In these scenarios, neither model properly balances the pole. In Figure 2(a-c), the same results for a PUN model without the SNR loss term (again compared to the MLP baseline) are presented. We can see a performance increase along the limit of the baseline MLP model. We can also see that the PUN model has more scenarios where it performs slightly worse than MLP. In contrast, the SNR PUN does not, highlighting that the parameter uncertainty regularization term helps generalization.

The experiments presented here, were carried out for all models in Table 1 and each environment; these results can be found in the appendix (Figures 7, 8, 9, and 10). They are omitted here for brevity. Similar results are seen across environments.

6.3. Ablation Study of SNR Term

The following sections discuss the potential benefits of SNR regularization.

SNR Statistics The impact of removing the SNR regularization term ($\mathcal{L}_{\text{SNR}}$) from the loss function during training is demonstrated by the drop in the mean performance of the stochastic ('PUN') models in Table 1. Figure 3 plots the relationship between μ_i and σ_i for the stochastic models with and without the SNR loss term. For the CartpoleSwingUp and Quadruped environments respectively, there is a 2.12% and 10.2% increase in the SNR parameter values that exceed $\Omega_{\text{Max SNR}}$ in the 'PUN' model, whereas no parameter values exceed $\Omega_{\text{Max SNR}}$ in the 'SNR PUN' model. Constraining the SNR of a small subset of the overall parameter distributions could have led to the reduction in information during training and thereby an increase in generalization performance. We leave exploration of this relationship to future studies.

Influence on Max SNR selection The results in the main text were obtained using $\Omega_{\text{Max SNR}} = 10$. Figure 4 presents the visualization of the network parameter distributions for different $\Omega_{\text{Max SNR}}$ values and in the unconstrained case. We provide these results primarily as a sensitivity analysis of the $\Omega_{\text{Max SNR}}$ hyperparameter.

When $\Omega_{\text{Max SNR}}$ is set to a value close to zero it closely resembles training with a KL divergence regularization term, as only a distribution with $\mu_i = 0$ has zero SNR. When the value is set to a large value, it resembles training an unregulated network. In Figure 5 we show the relative performance of varying the $\Omega_{\text{Max SNR}}$ hyperparameter in all environments. More complex and unstable environments require a larger $\Omega_{\text{Max SNR}}$ so that a stable solution is found. Walker is the

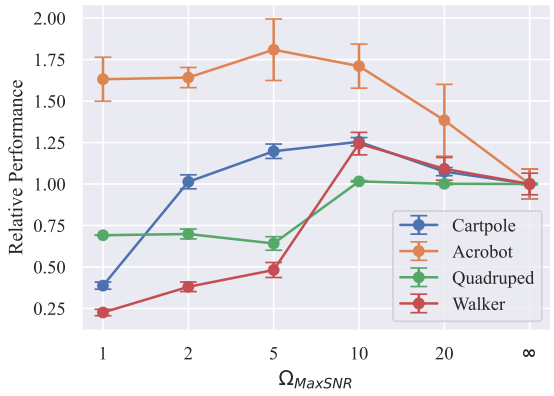

Figure 5. Relative performance (RP) for all environments over $\Omega_{\text{Max SNR}}$. RP is the relative improvement of the ATP at $\Omega_{\text{Max SNR}} = \infty$, which is equivalent to the unconstrained PUN.

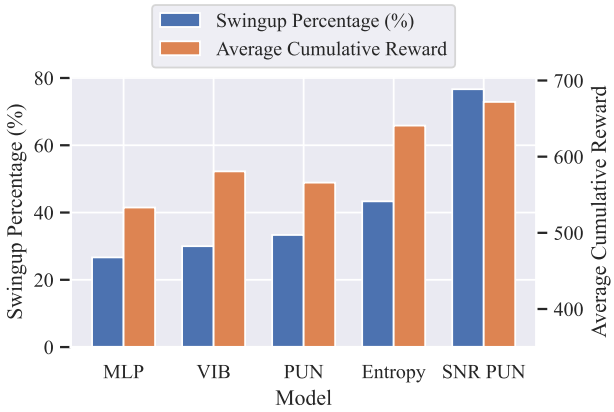

Figure 6. Comparing the percentage of successful balances and reward on physical Cartpole. Although the average reward improvement between SNR PUN and state-of-the-art is small, its effect on successful swingup percentage is 43% higher.

most complex and least stable environment, as such it is most sensitive to changes in $\Omega_{\text{Max SNR}}$ while Acrobot is the simplest and thus the least sensitive.

Furthermore, the sharpness of the curves shows each environment relationship between parameter noise and regularization requirements. In Acrobot heavy regularization is key, in Quadruped minimizing parameter noise to facilitate training is essential, and in Cartpole and Walker finding a balance between the two leads to optimal performance.

6.4. Transferability to Robotic Platform

To evaluate the transferability of our models to a real-world platform, we conducted tests on a physical Cartpole. Detailed information about the equipment used can be found here. We estimated several key parameters for the physical Cartpole: the friction coefficient, pendulum mass and length, maximum force output, sensor noise (including angle, angular velocity, track position, and track velocity), and track length. These parameters were mirrored in the training phase. The physical system operates at 100 Hz, and simulations were conducted at the same frequency. We assessed the MLP, MLP + Entropy, VIB, PUN, and SNR PUN models across 30 trials, each lasting 10 seconds (to match the 1000 timesteps during training). For stochastic models the mean is used. We evaluate all models on two metrics: reward (which is comparable to results in Table 1) and successful swingup percentage (i.e. a measure on if the pole is balanced).

The results are shown in Figure 6. Both PUN and SNR PUN models outperform in the relative reward scores as shown in Table 1, showing the success of parameter uncertainty in facilitating sim-to-real transfer. Our findings indicate that while all models could learn during the training phase, only the SNR PUN model successfully transferred to the physical Cartpole, even for the base task. This suggests that a model demonstrating generalizability in simulated environments can be transferred effectively to a real-world environment.

7. Conclusion and Future Work

In this paper, we introduced **SNR PUN**, a novel SNR-based regularization method for improving the generalization of parameter uncertainty models in RL. Our approach outperforms traditional regularization methods like dropout and entropy maximization by effectively managing the SNR of model parameters. We found that setting the maximum SNR to a low value emulates training with a KL divergence regularization term, while a higher value resembles training in an unregulated network.

The SNR PUN model consistently demonstrated superior or comparable performance in various mutated environments, spanning both discrete and multiple continuous control tasks. Its effectiveness was further underscored when evaluated against all baselines, showing robustness and potential for real-world application.

Our results pave the way for future explorations into the transferability of SNR PUN to real-world scenarios. We aim to extend this research to continual learning in supervised tasks and exploration strategies in RL, leveraging the distinct advantages of SNR regularization. This will help in understanding the full scope of its applicability and effectiveness in broader contexts.

8. Acknowledgements

This work was partially supported by the Swiss National Science Foundation CA-DNNEdge project (208227). We would like to thank João Sacramento, Alexander Nedergaard, and Marcin Paluch for their valuable feedback.

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. 2

[2] Mahmoud Akl, Yulia Sandamirskaya, Deniz Ergene, Florian Walter, and Alois Knoll. Fine-tuning deep reinforcement learning policies with r-STDP for domain adaptation. In *Proceedings of the International Conference on Neuromorphic Systems 2022*, pages 1–8, 2022. 5

[3] Rika Antonova, Silvia Cruciani, Christian Smith, and Danica Kragic. Reinforcement learning for pivoting task. *arXiv preprint arXiv:1703.00472*, 2017. 2

[4] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. 5

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015. 2

[6] Stefan Braun and Shih-Chii Liu. Parameter uncertainty for end-to-end speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5636–5640, 2019. 1

[7] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. *arXiv preprint arXiv:2003.11881*, 2020. 5

[8] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021. 5

[9] Hugh Durrant-Whyte, Nicholas Roy, and Pieter Abbeel. *Infinite-Horizon Model Predictive Control for Periodic Tasks with Contacts*, pages 73–80. 2012. 5

[10] Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. *arXiv preprint arXiv:2103.06257*, 2021. 1, 6

[11] Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018. 1

[12] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017. 1, 2, 3, 4

[13] Adam Gaier and David Ha. Weight agnostic neural networks. *Advances in Neural Information Processing Systems*, 32, 2019. 5

[14] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning Workshop, ICML*, page 25, 2016. 5

[15] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021. 2

[16] Matthew Hausknecht and Nolan Wagener. Consistent dropout for policy gradient reinforcement learning. *arXiv preprint arXiv:2202.11818*, 2022. 1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015. 4

[18] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2

[19] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 6

[20] Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991. 1

[21] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37 (4-5):421–436, 2018. 2

[22] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5542–5550, 2017. 1

[23] Noel Loo, Siddharth Swaroop, and Richard E Turner. Generalized variational continual learning. *arXiv preprint arXiv:2011.12328*, 2020. 1

[24] Xingyu Lu, Kimin Lee, Pieter Abbeel, and Stas Tiomkin. Dynamics generalization via information bottleneck in deep reinforcement learning. *arXiv preprint arXiv:2008.00614*, 2020. 2, 6

[25] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992. 2

[26] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3932–3939. IEEE, 2017. 2

[27] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013. 1

[28] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Pro-*

ceedings of The 28th Conference on Learning Theory, pages 1376–1401, Paris, France, 2015. PMLR. 1

[29] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017. 1

[30] Thanh T Nguyen and Jaesik Choi. Markov information bottleneck to improve information flow in stochastic neural networks. *Entropy*, 21(10):976, 2019. 6

[31] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013. 2

[32] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *International Conference on Machine Learning*, pages 2377–2386. PMLR, 2016. 2

[33] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018. 1

[34] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018. 2

[35] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017. 1, 3

[36] Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In *Conference on Robot Learning*, pages 1634–1644. PMLR, 2022. 1, 6

[37] John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. 5

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2

[39] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 1

[40] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*. MIT Press, 1995. 5

[41] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 5

[42] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 2

[43] Shimon Whiteson, Brian Tanner, Matthew E. Taylor, and Peter Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 120–127, 2011. 4

[44] Zeke Xie, Issei Sato, and Masashi Sugiyama. On the overlooked pitfalls of weight decay and how to mitigate them: A gradient-norm perspective. 2020. 1, 6

[45] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, pages 1433–1438. Chicago, IL, USA, 2008. 1, 6