UNSUPERVISED PRETRAINING FOR NEURAL VALUE APPROXIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks are powerful function approximators and have successfully been employed for the parameterization of value functions in deep reinforcement learning. Neural value approximation is a powerful paradigm for model-free control but it can often result in instability and divergence, especially when combined with off-policy learning and bootstrapping. Recent works have revealed connections between the unstable behavior of neural value approximation and the generalization properties of the value network/critic. Motivated by this, we propose a simple and computationally efficient unsupervised pretraining method to be performed before neural value learning. The method learns initializations of the critic parameters that correspond to Neural Tangent Kernels with desirable generalization structures. We demonstrate the merits of our approach by combining it with the Soft Actor-Critic algorithm and testing its performance on the continuous control environments of the DeepMind Control Suite. Our approach results in considerable improvements in reward accumulation, sample efficiency and stability for the majority of the domain environments. Furthermore, the use of the proposed pretraining enables us to retain the performance gains when changing the in between layers activation function of the critic architecture. Finally, we demonstrate that our proposed pretraining results in performance improvements in pixel-based tasks from the DeepMind Control Suite, where the critic employs a Convolutional neural network as a backbone feature extractor.

1 INTRODUCTION

A crucial question that arises during the designing of Reinforcement Learning (RL) algorithms is how to estimate the sum of rewards that the agent is expected to collect, over the long run, by performing a particular action at a given state. This is the value of the respective state-action pair. Temporal difference methods (Sutton, 1988) comprise a paradigm for such a goal. They connect the value of a state-action pair with immediate rewards and value estimates of subsequent pairs, sidestepping the need for extensive agent rollouts. Estimating the value function becomes extremely challenging in control problems where the state-action space is large or continuous and, therefore, storing the value estimates on a lookup table is not a viable option. A popular approach is to use a deep neural network to parameterize the value function (Tesauro et al., 1995). The combination of deep neural networks and temporal difference learning gives birth to neural value approximation.

In neural value approximation, a deep neural network is trained to map each state-action pair of a Markov Decision Process (MDP) to its expected value under the optimal policy. The quality of the value estimates significantly affects the performance of the algorithm. This can be easily inferred by the fact that the control policy is designed so as to select the action (at each state) that corresponds to the highest value according to the value approximator (Mnih et al., 2015; Lillicrap et al., 2015).

Despite the empirical success of deep neural networks, they often cause instability and divergence. This problematic behavior is more prominent when combined with off-policy learning and bootstrapping, a mingling also known as the deadly triad (Sutton & Barto, 2018). A plethora of algorithmic modifications have been proposed to mitigate divergence. Some of them include the use of delayed copies of the value network for the bootstrapping target (Mnih et al., 2015), ensembling (Van Hasselt et al., 2016; Fujimoto et al., 2018) and n-step returns (Hessel et al., 2018).

Recent results in theoretical deep learning model the evolution of deep neural networks in regression tasks, under gradient descent dynamics, as kernel regression using a gram matrix. The gram matrix is

called the Neural Tangent Kernel (NTK) (Jacot et al., 2018). Every element of the NTK corresponds to the inner product of two gradient vectors of the network. The NTK motivated efforts towards connecting the divergence in value approximation with the generalization properties of the critic.

In Achiam et al. (2019), the authors make the assumption that if the first order approximation of the deep value update is a contraction in the sup norm then learning should be stable. Operating under this presupposition, they prove sufficient conditions of the value NTK for convergence in off-policy value-based RL. The conditions imply that value networks with aggressive generalization (NTK with large off-diagonal elements) are prone to cause divergence. Subsequently, the authors in Yang et al. (2022); Brellmann et al. (2021), inspired by recent improvements in graphics (Tancik et al., 2020; Sitzmann et al., 2020), proposed to preprocess the state-action vector with a Fourier kernel before passing it through the critic. For many benchmark environments and when the critic is feedforward with rectified linear activations, the Fourier preprocessing transforms the value NTK into a stationary kernel with a strong diagonal and smaller off-diagonal elements. The resulting critic provides increase in stability and speed of convergence. The authors in Kumar et al. (2020) report a phenomenon where the rank of the learned representations of the critic drops steeply during training in value-based deep RL with bootstrapping. This rank collapse corresponds to an increase in aliasing of the value estimation across states. The critic generalizes more and performance deteriorates.

Consistent stability in neural value approximation is predicated upon the ability to make explicit trade-offs between generalization and bias for the corresponding NTK. The NTK depends on the architecture class, the state-action vector space and the weights vector at initialization (Narkhede et al., 2022). Therefore, there is the need for designing principled methods that properly "condition" the value NTK for a wide range of MDPs and network architectures. Indeed, in this paper, we propose an architecture-agnostic unsupervised pretraining method that makes use of no reward signals and imposes a controllable generalization structure on the value NTK at initialization. We pretrain the critic on apriori collected transitions, by maximizing an objective that acts as a surrogate to the difference between the diagonal and off-diagonal elements of the value NTK. The objective uses an approximation of the network's gradient, based on function evaluations, to circumvent the computation of Hessian vector products. Subsequently, we use the pretrained value network as the critic initialization of the Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018). We compare the training performance of the pretrained version against a standard implementation of the SAC, the critic of which corresponds to the version of the critic before the pretraining.

The pretraining provides consistent improvements in stability and convergence speed for all the environments of the DeepMind Control Suite (DMC) (Tassa et al., 2018). It achieves about 50% training speedup on the "Quadruped-walk" environment and about 25% increase in average reward accumulation on the "Quadruped-run". Both environments have the largest state-action dimension among the DMC tasks. Furthermore, by using the proposed pretraining, we are able to solve all the DMC environments with proprioceptive states using a feedforward critic with hyperbolic Tangent activations. This is not the case when the the critic is randomly initialized. Finally, our pretraining scheme provides improvements in stability and reward performance for pixel-based tasks of the DMC, where the critic contains Convolutional layers.

2 Background

RL studies the problem where an agent interacts with the environment in a sequence of states, actions and rewards giving birth to a Markov Decision Process (MDP) defined as a tuple $\langle S, A, R, P, p, \gamma \rangle$. S is the state space, A is the action space, $R : S \times A \to \mathbb{R}$ is the reward function and $P : S \times A \to S$ is the transition function (depends on the environment dynamics). $p(s)$ is the distribution of the initial state and $\gamma \in (0, 1)$ is the discount factor that quantifies how "far-sighted" the agent is. The goal of RL is to learn a mapping from states to actions, namely a policy, $\pi(a|s)$ so as to maximize the expected discounted sum of rewards $J = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\Big]$.

The state-action value function Q^π is defined as the expected discounted sum of rewards starting from a state-action pair and following the policy π thereafter.

$$Q^\pi(s, a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | (s_0, a_0) = (s, a)\Big] \tag{1}$$

The optimal value function $Q^*(s, a)$ is the fixed point of the Bellman backup operator:

$$B^*Q(s, a) = \mathbb{E}_{s' \sim P}\Big[R(s, a) + \gamma \max_{a'} Q(s', a')\Big] \tag{2}$$

In off-policy RL, one assumes the existence of an Experience Replay Memory D (Mnih et al., 2015) that contains experiences sampled from the MDP ($D = \{s, a, s', r\}_{i=1}^{N}$). The optimal value function is parameterized as a neural network with parameter vector θ (Q_θ). The estimation of the optimal value function is conducted by minibatch gradient descent with the following update rule.

$$\theta' = \theta + 2\eta \mathbb{E}_{(s,a,s',r)\sim D}\Big[\Big(B^*Q_\theta(s,a) - Q_\theta(s,a)\Big)\nabla_\theta Q_\theta(s,a)\Big] \tag{3}$$

The parameter η is the learning rate. It is common practice to use an additional network, typically called as the target network, with parameters ψ, for calculating the bootstrapping target. The parameter vector ψ slowly tracks the parameter vector θ. We employ a slight abuse of formalism and we use the terms "value network" and "critic" interchangeably, for the rest of the paper. In addition, we use the term "value function" to refer to the function that depends both on the action and the state (Equation 1) and not the function where the actions are marginalized out (Schaul et al., 2015).

2.1 Divergence and the value Neural Tangent Kernel

Let us assume that we sample one experience (s, a, s', r) from the Experience Replay D and update the parameters θ of the value network using the update rule in Equation 3. Subsequently, we get a new parameter vector θ'. Assuming a small η, we examine the updated $Q_{\theta'}$ on state-action pair $(\bar{s}, \bar{a})$ by the first order Taylor expansion around θ, as proposed by Achiam et al. (2019).

$$Q_{\theta'}(\bar{s}, \bar{a}) \approx Q_\theta(\bar{s}, \bar{a}) + \eta \mathbb{E}_{(s,a)\sim P}\Big[K_\theta(\bar{s}, \bar{a}; s, a)\Big(B^*Q_\theta(s,a) - Q_\theta(s,a)\Big)\Big] \tag{4}$$

In the above expression, $K_\theta(\bar{s}, \bar{a}; s, a) = \nabla_\theta Q_\theta(\bar{s}, \bar{a})^T Q_\theta(s, a)$ is the corresponding element of the *value NTK* (the Neural Tangent Kernel of the value network). The quantity $K_\theta(\bar{s}, \bar{a}; s, a)$ corresponds to the inner product between the gradient vectors of the respective state-action pairs and constitutes a measure of the generalization of the value approximator. In essence, **the higher the magnitude of $K_\theta(\bar{s}, \bar{a}; s, a)$, the more the update using (s, a) affects the value estimate of $(\bar{s}, \bar{a})$.**

We briefly follow the analysis by Achiam et al. (2019). Assuming an MDP with N state-action pairs $\{(s_i, a_i)\}_{i=1}^{N}$, then the value NTK, $K_\theta = K$, is an $N \times N$ symmetric matrix where :

- $K_\theta(i, j) = K_{ij} = K_\theta(j, i) = K_{ji} = \nabla_\theta Q_\theta(s_i, a_i)^T \nabla_\theta Q_\theta(s_j, a_j)$
- $K_\theta(i, i) = K_{ii} = \|\nabla_\theta Q_\theta(s_i, a_i)\|^2$

After updating the parameters θ with all N experiences, the state-action pair values before and after the update are related by:

$$Q_{\theta'} \approx Q_\theta + \eta K_\theta D_\rho\Big(B^*Q_\theta - Q_\theta\Big), \tag{5}$$

where D_ρ is the diagonal matrix with entries $\rho_i = \rho_i(s_i, a_i)$ given by the distribution of state-action pairs induced by the content of the Experience Replay.

Assuming that the first order approximation of the deep value update being a contraction in the supremum norm suffices for convergence in neural value learning, the following theorem can be proven.

Theorem 1 *(Achiam et al., 2019) Let indices i, j refer to state-action pairs. Suppose that K, ρ, $\gamma < 1$, η satisfy the following conditions:*

$$\forall i, 2\eta K_{ii}\rho_i \leq 1, \tag{6}$$

$$\forall i, (1+\gamma)\sum_{j\neq i}\|K_{ij}\|\rho_j \leq (1-\gamma)K_{ii}\rho_i. \tag{7}$$

Then the neural value update operator is a contraction in the supremum norm, with its fixed point being the optimal value function Q^*.

The off-diagonal elements of the value NTK determine the generalization of the value approximator. For a given MDP, a value network that corresponds to a value NTK with large off-diagonal elements in comparison to the elements of the main diagonal generalizes relatively aggressively.

Let us examine Theorem 1. Assuming that $\rho_i > 0$ everywhere and the discount factor γ is large (typically it is chosen to be 0.99), the theorem implies the following. To achieve stability and convergence when learning the value function with a neural network (off-policy), the underlying value NTK needs to have off-diagonal elements with small magnitude in comparison to the magnitude of the elements of the main diagonal. Therefore, for every state-action pair i, it is required that:

$$\forall j \neq i, \quad \|\nabla_\theta Q_\theta(s_i, a_i)^T \nabla_\theta Q_\theta(s_j, a_j)\| \ll \|\nabla_\theta Q_\theta(s_i, a_i)\|^2 \tag{8}$$

3 UNSUPERVISED PRETRAINING OF THE VALUE NETWORK

Inequality 8 is the main motivation for the unsupervised pretraining method proposed in the current work. The key idea is to pretrain the value network in order to force inequality 8 for the state-action pairs of the MDP of interest. The goal is to ensure that condition 8 holds strong at the beginning of training and, by doing that, to enable stability thereafter (Chizat & Bach).

We assume the availability of an Experience Replay D which is filled with transitions ($D = \{s_i, a_i, s'_i, a'_i, r_i\}_{i=1}^{N}$). The (s_i, a_i) is the current state-action of the transition and (s'_i, a'_i) is the subsequent pair. The natural pretraining regimen to propose is the following.

We set a number of pretraining epochs. At each update step, we sample a minibatch of experiences from the Experience Replay and maximize the following objective with gradient ascent on θ:

$$J_p(\theta) = \mathbb{E}_{(s,a,s',a')\sim D}\Big[\mathbf{a}\|\nabla_\theta Q_\theta(s,a)\|^2 - \mathbf{b}\|\nabla_\theta Q_\theta(s,a)^T \nabla_\theta Q_\theta(s',a')\|\Big] \tag{9}$$

The parameters **a**, **b** are positive scalars. By maximizing $J_p(\theta)$ we incentivize the search over the network's parameter space in order to find areas where the gradient vectors of different state-action pairs are almost orthogonal to each other and the gradient norms are relatively large. The reward is absent from the objective, hence the unsupervised aspect of the method (Hastie et al., 2009).

There is an intrinsic trade-off between generalization and stability in neural value approximation. On one hand, the off-diagonal elements of the NTK need to have small magnitude, relative to the diagonal elements, for stable convergence. On the other hand, the off-diagonal elements control the amount of generalization during value approximation. Therefore, even though a value NTK with a strong diagonal is desirable for convergence, we would like to avoid forcing the off-diagonal elements to be (almost) zero. In the case of (almost) zero off-diagonal elements, the critic is in memorization mode and acts like a lookup table. This degenerate case defeats the purpose of function approximation and the critic is unable to reason (interpolate) for the value of unseen pairs.

The phenomenon where critics go into undergeneralizing modes was empirically studied in Bengio et al. (2020). The authors compute a quantity, throughout temporal difference learning, that they coin as interference. Interference is an estimate to the value of the off-diagonal NTK elements. They report cases where this amount gets very small and the critic simply memorizes.

Choosing the values of the parameters **a**, **b** gives us implicit control over the trade-off between generalization and stability. The "optimal" coefficient values depend on the dynamics, the state-action vector space and the critic architecture. We investigate the effect of **a** and **b** in Section 4.6.

3.1 A SURROGATE TO THE PRETRAINING OBJECTIVE BASED ON FUNCTION EVALUATIONS

The maximization of the pretraining objective (Equation 9) requires computing Hessian vector products at every update step. For environments with high-dimensional state-action spaces, the critic typically has a large number of parameters. Therefore, the Hessian computation induces significant overhead. We propose a different formulation to overcome the computational bottleneck.

The computationally efficient proposition invokes a zeroth order approximation of the gradient of the critic's output with respect to the parameters. The approximation pertains to the use of function evaluations. In particular, let us assume the value network $Q_\theta(s,a)$ and $\mu > 0$ a smoothing parameter. Then the μ-smooth value network (critic) can be defined as follows:

$$Q^\mu_\theta(s,a) \coloneqq \mathbb{E}_{U\sim\mathcal{N}(0,I)}\left[Q_{\theta+\mu U}(s,a)\right] \tag{10}$$

The use of the μ parameter enables us to obtain an unbiased estimate of the gradient of the μ-smooth critic, with respect to the parameters, using only two evaluations. We present the following lemma (modification of similar lemmas in Nesterov & Spokoiny (2017); Kalogerias & Powell (2022))

Lemma 2 *(Nesterov & Spokoiny, 2017; Kalogerias & Powell, 2022) For every $\mu > 0$, the μ-smooth critic surrogate Q^μ_θ is differentiable with respect to the parameter vector and the gradient can be represented as:*

$$\nabla_\theta Q^\mu_\theta(s,a) = \mathbb{E}_{U\sim\mathcal{N}(0,I)}\Big[\frac{Q_{\theta+\mu U}(s,a) - Q_\theta(s,a)}{\mu} U\Big]$$

We can safely assume that, for small enough μ, the true gradient of the critic with respect to the parameters can be approximated by the zeroth order gradient of the μ-smooth critic:

$$\nabla_\theta Q_\theta(s,a) \approx \mathbb{E}_{U\sim\mathcal{N}(0,I)}\Big[\frac{Q_{\theta+\mu U}(s,a)-Q_\theta(s,a)}{\mu}U\Big] \tag{11}$$

Motivated by the above zeroth order approximation of the gradient we formulate a surrogate to the original unsupervised pretraining objective (Equation 9) as follows.

$$J_{SUR}(\theta;a,b,\mu,U) = \mathbb{E}_{(s,a,s',a')\sim D}\Big[\mathbf{a}\|\overbrace{\frac{Q_{\theta+\mu U}(s,a)-Q_\theta(s,a)}{\mu}U}^{\approx\nabla_\theta Q_\theta(s,a)}\|^2 - \mathbf{b}\|\underbrace{\frac{Q_{\theta+\mu U}(s',a')-Q_\theta(s',a')}{\mu}U^TU\frac{Q_{\theta+\mu U}(s,a)-Q_\theta(s,a)}{\mu}}_{\approx\nabla_\theta Q_\theta(s',a')^T\nabla_\theta Q_\theta(s,a)}\|\Big] \tag{12}$$

The parameters μ, $\mathbf{a}$ and $\mathbf{b}$ are fixed. The pretraining is depicted in Algorithm 1.

For the experiments of the paper we stick to the sampling of one U per batch. There can be implementations that involve the sampling of multiple i.i.d U_i and averaging for every batch update. Such implementations trade pretraining time for gradient approximation accuracy.

An interesting alternative is to fill the Experience Replay with transitions from expert demonstrations and apply the proposed pretraining scheme either in the context of deep RL or in the context of Imitation learning (Li et al., 2017; Torabi et al., 2019). This direction is left for future work.

Algorithm 1 Unsupervised Pretraining

Fill Experience Replay D with N transitions from a random policy $D = \{s_i, a_i, s'_i, a'_i\}_{i=1}^N$
Initialize θ
Set μ (smoothing), λ (learning rate), $\mathbf{a}$ and $\mathbf{b}$ (coefficients of the objective terms)
for each epoch **do**
    for each update step **do**
        Sample a batch of transitions from D
        $U \sim \mathcal{N}(0, I)$
        $\theta \leftarrow \theta + \lambda\nabla_\theta J_{SUR}(\theta; \mathbf{a}, \mathbf{b}, \mu, U)$
    end for
end for

4 Experiments

4.1 DeepMind Control Suite

We test our pretraining scheme on the continuous control tasks from the DeepMind Control Suite (DMC) (Tassa et al., 2018). For every experiment, we first collect experiences by running a random policy for 100 episodes. Subsequently, we independently train two critic networks for 100 epochs with our proposed pretraining (Algorithm 1). An important nuance is that we pretrain with vanilla stochastic gradient descent (torch.optim.SGD). In our early experiments, we found it to be more stable than other popular optimizers such as Adam (Kingma & Ba, 2014). Finally, we use the pretrained networks as the double critic initialization of the SAC. We benchmark the training performance of the above described scheme against the direct training of the SAC algorithm with a non-pretrained double critic. We use the SAC implementation by Yarats & Kostrikov (2020), which, to the best of our knowledge, constitutes the current *state-of-the-art* for the DMC.

The first set of experiments is depicted in Figure 1. Each critic is a Multilayer Perceptron (MLP) with Rectified Linear (ReLU) activations between layers, abbreviated as ReLU MLP. The architecture follows the one used by Yarats & Kostrikov (2020) (3 latent layers, 1024 neurons each). The unsupervised pretraining of the critic improves the overall stability, convergence and reward accumulation of the algorithm for all the environments. The impact of the pretraining is more profound when it comes to the environments with the largest, in terms of dimension, state-action space. In particular, for the "Quadruped-walk" environment, the pretraining scheme provides about 50% speedup

in terms of sample efficiency. When it comes to the "Quadruped-run" environment, the pretrained version of the SAC performs about 25% better than the non-pretrained version in terms of average reward. The pretraining of the critic improves the stability between seeds for all the environments.

We provide the pretraining hyperparameters (**a**, **b** and μ) for every environment in the Appendix A.1. The learning curves for each environment (Figure 1) correspond to the average performance over 15 different seeds. We investigate the effect of different hyperparameter values in Section 4.6. In general, we found that choosing $\mu = 0.0001$, $\mathbf{a} = 1$ and $\mathbf{b} = 1$ generalizes well across the DMC environments (see Figure 6)

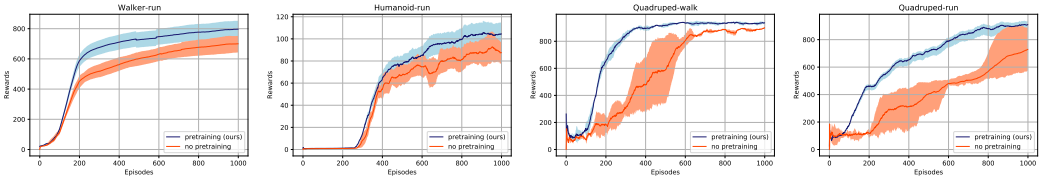

Figure 1: Learning curves for DMC environments. For every environment, the subfigure depicts the learning curve for the direct training of the SAC and the curve corresponding to the SAC with the critic that has been pretrained with Algorithm 1. The environments are ordered by increasing input dimension (left to right). The critic is a ReLU MLP. The pretraining part is not included in the plots.

4.2 Changing the activations of the critic from ReLU to Tanh

Figure 2 presents a set of experiments similar to those of the previous subsection. The only difference is that the activations between the layers of the value network are replaced with the hyperbolic Tangent function (Tanh). This might seem as an insignificant change, but its effect is substantial. Based on the work by Tancik et al. (2020), MLPs with Tanh activations, abbreviated as Tanh MLPs, generally correspond to NTKs with larger off-diagonal elements in comparison to ReLU MLPs. Therefore, it is expected that using Tanh MLP for the critic will cause performance deterioration for some environments. Indeed, if we consider the direct training of the SAC algorithm, without the unsupervised pretraining of the critic, the Tanh MLP performs noticeably worse than the ReLU MLP on the majority of the domain environments. On the other hand, the Tanh MLP critic, after being pretrained, performs on par with the pretrained ReLU MLP critic. For the case of the "Quadruped-run", the version of the SAC with the pretrained Tanh MLP critic performs better than the rest of the variations (Figure 12 in the Appendix for the direct comparison).

A notable case is that of the "Humanoid-run" environment. The version that employs a pretrained critic with Tanh activations performs reasonably well (Note that the current state-of-the-art performance for the "Humanoid-run" is approximately 130 per episode (Yang et al., 2022)). On the other hand, the SAC with a non-pretrained Tanh MLP critic cannot practically solve the environment.

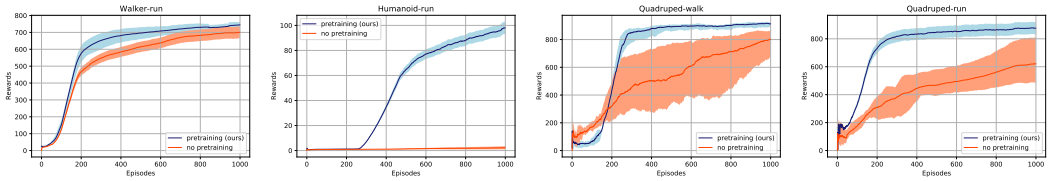

Figure 2: The figure depicts the same set of experiments as Figure 1, but with Tanh MLP critic.

4.3 Pixel-based DMC tasks

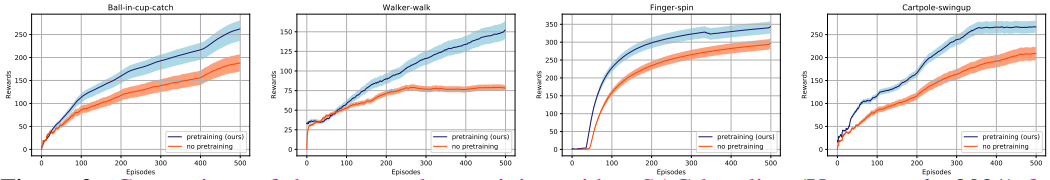

Figure 3: Comparison of the proposed pretraining with a SAC baseline (Yarats et al., 2021) for pixel-based tasks of the DMC. Each curve corresponds to an average over 5 seeds. The pretraining part is not included in the plots.

We test the performance of the proposed pretraining approach on a sample of the DMC tasks where the state is comprised by raw image frames. The results are depicted in Figure 3. We combine our pretraining with the pixel SAC implementation by Yarats et al. (2021), where the critic combines two different architecture classes: first a Convolutional Neural Network (CNN) and then a ReLU MLP, both connected sequentially. As can be extracted by Figure 3, our proposed pretraining provides considerable increase in performance for all depicted tasks. We employ the same pretraining hyperparameter values for all pixel-based environments ($\mu = 0.0001$, $\mathbf{a}, \mathbf{b} = 1$).

4.4 Visualizing the value NTK before and after the pretraining

The intuition behind the proposed unsupervised pretraining method is, given the architecture of the critic and transitions from the MDP, to "condition" the NTK to adhere to inequality 8 that derives, under mild assumptions, from the theoretical result in Achiam et al. (2019). The pretraining process searches through the parameter space to find an initial parameter vector for the critic that corresponds to a NTK that has a strong diagonal and relatively smaller off-diagonal elements.

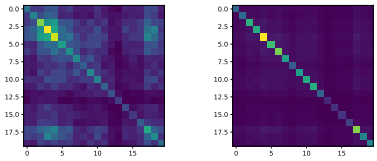

Figure 4: NTK of ReLU MLP critic for the same batch of state-action pairs before (left) and after (right) the pretraining

In order to validate that the pretraining does find suitable initializations, we visualize the NTK of the value network before and after the pretraining for a ReLU MLP (Figure 4) and a Tanh MLP critic (Figure 5). All NTKs of Figures 4 and 5 are computed on the same batch of state-action pairs.

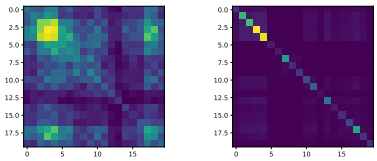

Figure 5: NTK of Tanh MLP critic for the same batch of state-action pairs before (left) and after (right) the pretraining

Two things to be stressed regarding Figures 4 and 5. First of all, the NTK after the pretraining adheres to the condition outlined in 8 for both classes of activation functions of the feedforward critic. It corresponds to a kernel with strong diagonal and small off-diagonal elements. Second, the NTK of the Tanh MLP generalizes more aggressively than the NTK of the ReLU MLP (both of them without being pretrained with the proposed unsupervised scheme), which explains the overall inferior performance of the Tanh MLP on the majority of environments. This comes to a complete agreement with the theoretical result by Achiam et al. (2019) described in Theorem 1, and the differences between ReLU and Tanh MLPs, in terms of their corresponding NTKs, described in Tancik et al. (2020); Yang et al. (2022).

The NTKs of our visualizations were computed with the repository by Engel et al. (2022).

4.5 Comparing with the Fourier Features

The proposed pretraining approach strives to impose a particular structure on the NTK of the critic. The same incentive underlies the proposed works in Yang et al. (2022); Brellmann et al. (2021), where the authors employ a learnable Fourier feature kernel on the state-action vector before passing it through the value network. It is natural to compare our proposed pretraining with the Fourier features proposition for the environments of the DMC. We provide the comparison of our proposed approach with the method by Yang et al. (2022) in Figure 6.

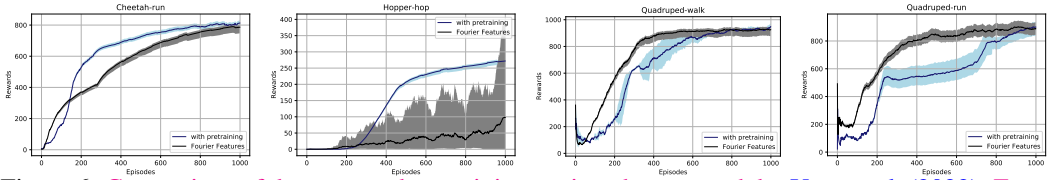

Figure 6: Comparison of the proposed pretraining against the approach by Yang et al. (2022). Every curve corresponds to the average over 6 seeds. The pretraining part is not included in the plots.

The proposed pretraining behaves comparably to the Fourier preprocessing for the depicted environments. The notable cases are the "Quadruped-run", where the Fourier approach performs noticeably better than the pretraining, and the "Hopper-hop" where the proposed pretraining performs significantly better than the Fourier preprocessing. The approach by Yang et al. (2022) introduces one hyperparameter which is the variance of the Normal distribution that the elements of the Fourier kernel are sampled from at initialization. For the experiments of Figure 6, we employ the variance

that was proposed by Yang et al. (2022) for the Fourier preprocessing approach at each environment. For our proposed pretraining, we employ the same hyperparameters for each environment ($\mu = 0.0001$, $\mathbf{a} = 1$ and $\mathbf{b} = 1$).

The methods by Yang et al. (2022); Brellmann et al. (2021) are admittedly a powerful paradigm for improving value learning when the critic is a ReLU MLP. Even though the Fourier preprocessing introduces only one additional hyperparameter, in contrast to the 3 hyperparameters introduced by the proposed pretraining, we observe that it is generally safer to tune the pretraining method. This conjecture is grounded in the following observation. The performance of the pretraining exhibits some variability in terms of the hyperparameter values (see Subsection 4.6) but it is very rare that it will cause catastrophic divergence. This is not always the case for the Fourier preprocessing, where small deviations from the optimal variance can result in reward collapse due to noisy value estimates.

An interesting finding pertains to the joint employment of the Fourier features preprocessing and the proposed unsupervised pretraining. In particular, we noticed that the employment of the proposed pretraining can stabilize the training performance of the SAC for certain DMC environments where the Fourier features exhibit instability. We provide the training performance of the SAC with Fourier preprocessing on the "Hopper-hop" environment, with and without the proposed pretraining, in Figure 7. As can be extracted by the aforementioned Figure, the SAC with the Fourier preprocessing behaves unstably. Even though there are cases where it achieves very high reward, there are seeds where the SAC performance collapses during training. The employment of the proposed pretraining, on top of the Fourier features preprocessing, stabilizes training.

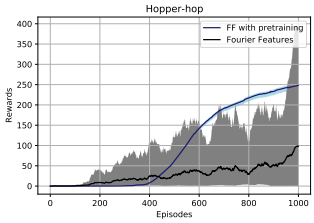

Figure 7: SAC with Fourier preprocessing with and without the pretraining. The pretraining part is not included in the plots.

4.6 Choosing the coefficients for the unsupervised pretraining objective

The values of the parameters $\mathbf{a}$ and $\mathbf{b}$ in the pretraining objective (12) correlate with performance. We provide the performance of SAC for the two DMC environments with the largest input dimension, when the critic is pretrained with 3 different values of the $\frac{\mathbf{a}}{\mathbf{b}}$ ratio, in Figure 8. For each environment, we use the same set of Experiences, the same μ, the same number of pretraining epochs and the same initial (before the pretraining) parameter vector for the critic for all the pretraining experiments. The only parameter that changes is the ratio of the coefficients.

The discrepancy between the performances of the SAC for the different coefficient ratios in the pretraining objective clearly indicates the overall importance of the critic initialization.

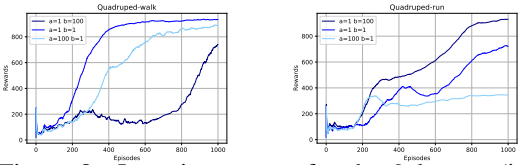

Figure 8: Learning curves for the 2 largest (in terms of state-action dimension) environments of the DMC. For each environment, the subfigure demonstrates the performance of the SAC for 3 different sets of values for the coefficients $\mathbf{a}$ and $\mathbf{b}$. Each curve corresponds to the average over 6 seeds. Variance is omitted to avoid clutter.

If we fix all the pretraining parameters and hyperparameters (Experience Replay content, parameters μ and λ, number of pretraining epochs and initial critic parameter vector), then the $\frac{\mathbf{a}}{\mathbf{b}}$ ratio influences the properties of the resulting NTK. In general, the smaller the ratio $\frac{\mathbf{a}}{\mathbf{b}}$ the stronger the NTK diagonal in comparison to the off-diagonal elements after the pretraining. This can be extrapolated by Figure 9 where the NTK for $\mathbf{a} = 1$ and $\mathbf{b} = 100$ is clearly sparser than the one for $\mathbf{a} = 100$ and $\mathbf{b} = 1$. We should note that the performance of the proposed method does not vary as much for different coefficient ratios on the rest of the environments of the DMC (except for the Quadruped environments). The trade-off between generalization and bias is governed by the relation between diagonal and off-diagonal elements of the NTK and is particular to each environment (dynamics and underlying value function). That is why, for the "Quadruped-walk", best performance is exhibited for $\mathbf{a} = 1$ and $\mathbf{b} = 1$ in contrast to the "Quadruped-run" where best performance corresponds to pretraining with $\mathbf{a} = 1$ and $\mathbf{b} = 100$ (Figure 8). This hints that the "Quadruped-walk" demands "less" critic generalization than the "Quadruped-run".

A reasonable proposition would be to collapse the contribution of the 2 coefficients (**a** and **b**) into 1 ($\frac{\mathbf{a}}{\mathbf{b}}$ or $\frac{\mathbf{b}}{\mathbf{a}}$). We refer the reader to Appendix A.3 for the relevant discussion.

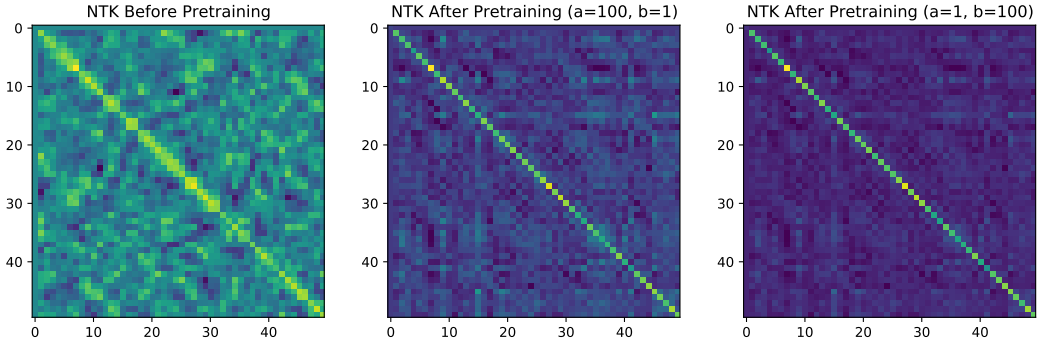

Figure 9: The value NTK for the same batch of state-action pairs (from the "Quadruped-run" environment) before the unsupervised pretraining (left), after the unsupervised pretraining with $\mathbf{a} = 100$ and $\mathbf{b} = 1$ (mid) and after the unsupervised pretraining with $\mathbf{a} = 1$ and $\mathbf{b} = 100$ (right).

5 Related Work

Recent works have shed light on the connections between the instability in neural value learning and the generalization properties of the value network. Our work builds on the theoretical analysis by Achiam et al. (2019). We propose an unsupervised pretraining method which finds critic initializations that correspond to NTKs with strong diagonals and small off-diagonal elements. There is a connection between our proposed approach and the works in Yang et al. (2022); Brellmann et al. (2021) where the authors propose the preprocessing of the input of the critic with a Fourier transformation that makes the resulting NTK stationary. The distinction is that our proposed method supersedes, to a certain degree, the architecture class of the critic and the vector space of the MDP, but requires more computing budget because of the pretraining aspect.

In Kumar et al. (2020), the authors reveal that the rank of the learned representations of the value network reduces significantly during temporal difference learning with bootstrapping. This results in feature aliasing (aggressive generalization) and divergence. To avoid the rank drop, they propose a regularization term that pertains to the difference between the largest and smallest singular value of the feature transformation. Our proposed work targets the issue of aggressive generalization explicitly by conditioning the NTK of the value network. Nonetheless, the effects of our proposed pretraining on the representation aspect of neural value learning are to be examined.

6 Discussion

The purpose of the paper is to introduce a pretraining method that conditions the NTK of the critic to adhere to sufficient conditions for convergence of value approximation. The pretraining scheme is unsupervised (no use of rewards or labels of any kind) and is computationally efficient because it employs a zeroth order approximation of the gradient vector of the value network to circumvent the compution of Hessian vector products. The pretraining finds initializations of the critic network that correspond to NTKs with a strong diagonal and relatively small off-diagonal elements. It constitutes a principled way of network initialization in contrast to classical views that consider the initial parameters as samples from a probabilistic distribution (He et al., 2015; Kumar, 2017).

There has been an emergence of unsupervised representation learning for RL (Seo et al., 2022; Finn et al., 2016; Stooke et al., 2021; Schwarzer et al., 2021; Gupta et al., 2018; Jaderberg et al., 2016). Typically, the underlying motivation behind the merging of RL and unsupervised learning is to extract rich feature representations that capture the factors of variation in data and can be reused for a plethora of downstream tasks. Instead, the current work is a testament towards a different direction that is not necessarily orthogonal to the mainstream one. The objective is to employ unsupervised learning methods so as to imprint desirable generalization and bias properties on the value network in terms of its evolution under gradient descent dynamics. That is the reason why the proposed pretraining scheme operates directly on the properties of the resulting NTK of the value network. We believe that both directions (extracting powerful representations and facilitating the evolution under gradient descent) can be simultaneously progressed towards and connections between the two can be uncovered in the future.

REFERENCES

Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. In *International Conference on Machine Learning*, pp. 767–777. PMLR, 2020.

David Brellmann, Goran Frehse, and David Filliat. Fourier features in reinforcement learning with neural networks. 2021.

L Chizat and F Bach. A note on lazy training in supervised differentiable programming.(2018). *arXiv preprint arXiv:1812.07956*.

Andrew Engel, Zhichao Wang, Anand D Sarwate, Sutanay Choudhury, and Tony Chiang. Torch-ntk: A library for calculation of neural tangent kernels of pytorch models. *arXiv preprint arXiv:2205.12372*, 2022.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519. IEEE, 2016.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pp. 485–585. Springer, 2009.

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*, pp. 1026–1034, 2015.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Dionysios S Kalogerias and Warren B Powell. Zeroth-order stochastic compositional algorithms for risk-aware learning. *SIAM Journal on Optimization*, 32(2):386–416, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems*, 30, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Meenal V Narkhede, Prashant P Bartakke, and Mukul S Sutaone. A review on weight initialization strategies for neural networks. *Artificial intelligence review*, 55(1):291–322, 2022.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pp. 19561–19579. PMLR, 2022.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

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

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, and Yazhe Li. Diego de las casas, david budden, abbas abdolmaleki, josh merel, andrew lefrancq, et al. deepmind control suite. *arXiv preprint arXiv:1801.00690*, 1, 2018.

Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. *arXiv preprint arXiv:2206.04672*, 2022.

Denis Yarats and Ilya Kostrikov. Soft actor-critic (sac) implementation in pytorch. `https://github.com/denisyarats/pytorch_sac`, 2020.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10674–10681, 2021.

A APPENDIX

A.1 HYPERPARAMETERS

In the current subsection, we list the hyperparameters for the unsupervised pretraining for all the environments of the DMC. One thing to be noted is that the pretraining provides improvements for a range of values close to the ones that are reported in this subsection. The provided values correspond to the experiments of Figures 1 and 2. They do not apply for Figures 3 and 6.

Pretraining hyperparameters

Environment	Observation space	Action space	μ	**a**	**b**	λ
Acrobot-swingup	Box(6,)	Box(1,)	1e-6	1	1	1e-6
Finger-turn-hard	Box(12,)	Box(2,)	1e-6	1	1	1e-6
Hopper-hop	Box(15,)	Box(4,)	1e-6	1	1	1e-6
Cheetah-run	Box(17,)	Box(6,)	1e-5	1	10	1e-6
Walker-run	Box(24,)	Box(6,)	1e-5	1	10	1e-6
Humanoid-run	Box(67,)	Box(21,)	1e-5	1	1	1e-6
Quadruped-walk	Box(78,)	Box(12,)	1e-4	1	1	1e-6
Quadruped-run	Box(78,)	Box(78,)	1e-4	1	10	1e-6

A.2 EXTRA EXPERIMENTS

In the current subsection, we provide some additional plots for the performance of SAC with and without the proposed pretraining both for the ReLU MLP and the Tanh MLP case (Figures 10 and 11).

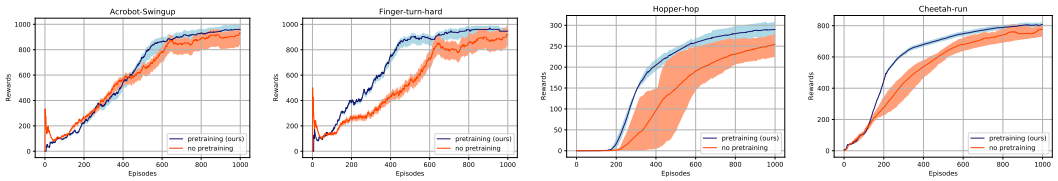

Figure 10: Learning curves for the some of the DMC environments. For every environment, the subfigure depicts the learning curve (15 seeds) that corresponds to the direct training of the SAC and the curve (15 seeds) corresponding to the SAC with the critic that has been pretrained with Algorithm 1. The environments are ordered by increasing input dimension (left to right). The critic is a ReLU MLP. The pretraining part is not included in the plots.

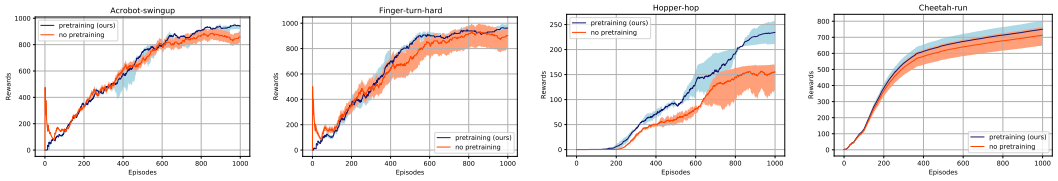

Figure 11: Learning curves for the continuous control environments of the DMC. For every environment, the subfigure depicts the learning curve (15 seeds) that corresponds to the direct training of the SAC and the curve corresponding to the SAC with the critic that has been pretrained with Algorithm 1. The environments are ordered by increasing input dimension (left to right). The critic is a Tanh MLP. The pretraining part is not included in the plots.

We also provide direct comparison for all feedforward variations (ReLU MLP and Tanh MLP, with and without pretraining) on the 4 largest, in terms of input dimension, environments on the DMC (Figure 12)

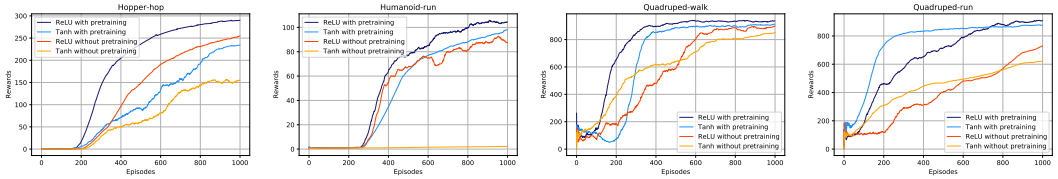

Figure 12: Learning curves for 4 DMC environments. There are 4 curves for each environment. One that corresponds to the performance of the ReLU MLP critic with pretraining (midnight blue), one to the performance of the ReLU MLP critic without the pretraining (orange), one to the performance of the Tanh MLP critic with pretraining (blue) and one to the Tanh MLP critic without pretraining (yellow). Each curve is the average over 15 seeds and the variance is omitted to avoid clutter. The pretraining part is not included in the plots.

A.3 Discussion for the 2 coefficients of the pretraining objective

A reasonable proposition would be to collapse the contribution of the 2 coefficients (**a** and **b**) of Equation 12 into 1 ($\frac{\mathbf{a}}{\mathbf{b}}$ or $\frac{\mathbf{b}}{\mathbf{a}}$). The reason that we stick with the 2 distinct coefficients is that, when μ, λ and number of pretraining epochs are fixed, there is a chance that the optimal NTK can correspond to early stopping of the pretraining. In cases like that, it is the magnitude of the coefficients that matters as well as the ratio. The ratio contributes to the scaling of the learning rate.