DIFFERENTIABLE TRUST REGION LAYERS FOR DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Trust region methods are a popular tool in reinforcement learning as they yield robust policy updates in continuous and discrete action spaces. However, enforcing such trust regions in deep reinforcement learning is difficult. Hence, many approaches, such as Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO), are based on approximations. Due to those approximations, they violate the constraints or fail to find the optimal solution within the trust region. Moreover, they are difficult to implement, lack sufficient exploration, and have been shown to depend on code-level optimizations. In this work, we propose differentiable neural network layers to enforce trust regions for deep Gaussian policies via closed-form projections. Unlike existing methods, those layers formalize trust regions for each state individually and can complement existing reinforcement learning algorithms. We derive trust region projections based on the Kullback-Leibler divergence, the Wasserstein L2 distance, and the Frobenius norm for Gaussian distributions and empirically demonstrate that those projection layers achieve similar or better results than existing methods while being almost agnostic to specific implementation choices.

1 INTRODUCTION

Deep reinforcement learning has shown considerable advances in recent years with prominent application areas such as games (Mnih et al., 2015; Silver et al., 2017), robotics (Levine et al., 2015), and control (Duan et al., 2016). In policy search, *policy gradient* (PG) methods have been highly successful and have gained, among others, great popularity (Peters & Schaal, 2008). However, often it is difficult to tune learning rates for vanilla PG methods, because they tend to reduce the entropy of the policy too quickly. This results in a lack of exploration and, as a consequence, in premature or slow convergence. A common practice to mitigate these limitations is to impose a constraint on the allowed change between two successive policies. Kakade & Langford (2002) provided a theoretical justification for this in the approximate policy iteration setting. Two of the arguably most favored policy search algorithms, *Trust Region Policy Optimization* (TRPO) (Schulman et al., 2015a) and *Proximal Policy Optimization* (PPO) (Schulman et al., 2017), follow this idea using the *Kullback-Leibler divergence* (KL) between successive policies as a constraint.

We propose closed-form projections, realized as differentiable neural network layers, to enforce change limiting trust regions for Gaussian policies. In addition to being more stable and less dependent on code-level optimizations, our method comes with the benefit of imposing the constraints on the level of individual states, allowing for the possibility of context dependent trust regions. Furthermore, we propose three different similarity measures, the KL divergence, the Wasserstein L2 distance, and the Frobenius norm, to base our trust region approach on. Considering the trust region layers require the old policy as input, which would result in a stack of trust regions for a several sequential policy updates, we formalize an additional trust-region loss in the reinforcement objective such that the new policy without trust layers has to stay close to the output of the projection layer. While this still results in an approximation of the trust region update, we show that the trust regions are properly enforced. We also extend our approach to allow for a controlled evolution of the entropy of the policy, which has been shown to increase performance in difficult exploration problems (Pajarinen et al., 2019; Akrour et al., 2019).

We compare and discuss the effect of the different similarity measures and the entropy control on the optimization process. Additionally, we benchmark our algorithm against existing methods and demonstrate that we achieve similar or better performance.

2 Related Work

Approximate Trust Regions. Bounding the size of the policy update in policy search is a common approach. While, originally, Kakade & Langford (2002) focused on a method based on mixing policies, nowadays most approaches use KL trust regions to bound the updates. Peters et al. (2010) proposed a first approach to such trust regions by formulating the problem as a constraint optimization and provided a solution based on the dual of that optimization problem. Yet, this approach is not straightforwardly extendable to highly non-linear policies, such as neural networks. In an attempt to transfer those ideas to deep learning, TRPO (Schulman et al., 2015a) approximates the KL constraint using the Fisher information matrix and natural policy gradient updates (Peters & Schaal, 2008; Kakade, 2001), along with a backtracking line search to enforce a hard KL constraint. Yet, the resulting algorithm scales poorly. Thus, Schulman et al. (2017) introduced PPO, which does not directly enforce the KL trust region, but clips the the probability ratio in the importance sampling objective. This allows using efficient first-order optimization methods while maintaining robust training. However, Engstrom et al. (2020) and Andrychowicz et al. (2020) recently showed that code-level optimizations are essential for achieving state-of-the-art results with PPO and can even compensate for removing core parts of the algorithm, such as the clipping of the probability ratio. Additionally, PPO heavily relies on its exploration behavior and might get stuck in local optima (Wang et al., 2019). Tangkaratt et al. (2018) use a closed-form solution for the constraint optimization based on the method of Lagrangian multipliers. Yet, they require a quadratic parametrization of the Q-Function, which can limit the performance. Pajarinen et al. (2019) introduced an approach based on compatible value function approximations to realize KL trust regions. Based on the reinforcement learning as inference paradigm (Levine, 2018), Abdolmaleki et al. (2018) introduced an actor-critic approach using an Expectation-Maximization based optimization with KL trust regions in both the E-step and M-step. Song et al. (2020) proposed an on-policy version of this approach using a similar optimization scheme and constraints.

Projections for Trust Regions. Akrour et al. (2019) proposed *Projected Approximate Policy Iteration* (PAPI), a projection-based solution to realize KL trust regions. Their method projects an intermediate policy, which already satisfies the trust region constraint, onto the constraint bounds. This maximizes the size of the update step. However, PAPI relies on other trust region methods to generate this intermediary policy and cannot operate in a stand-alone setting. Additionally, the projection is not directly part of the policy optimization but applied afterwards, which can result in sub-optimal policies. Further, in view of high computational complexity, TRPO and PAPI have to simplify the problem of constraining the change by working with the expected KL-divergence. Opposed to that, we realize the projections as fully differentiable network layers which are directly included in the optimization process. Additionally, the projections enforce the constraints per state and thus also allow for different trust region sizes per state.

For the KL-based projection layer we need to resort to numerical optimization and implicit gradients for convex optimizations (Amos & Kolter, 2017; Agrawal et al., 2019). Thus, we investigate two alternative projections based on the Wasserstein L2 (W2) and Frobenius norm, which allow for closed form solutions. Both, Wasserstein and Frobenius norm have found only limited application in reinforcement learning. Pacchiano et al. (2020) use the Wasserstein distance to score behaviors of agents. Richemond & Maginnis (2017) proposed an alternative algorithm for bandits with Wasserstein based trust regions. Song & Zhao (2020) focus on solving the trust region problem for distributional policies using both KL and Wasserstein based trust regions for discrete action spaces. Our projections are applicable independently of the underlying algorithm and only assume a Gaussian policy, a common assumption for continuous action spaces.

Several authors (Dalal et al., 2018; Chow et al., 2019; Yang et al., 2020) used projections as network layers to enforce limitations in the action or state space given environmental restrictions, such as robotic joint limits.

Entropy Control. Abdolmaleki et al. (2015) introduced the idea of explicitly controlling the decrease in entropy during the optimization process, which later was extended to deep reinforcement learning by Pajarinen et al. (2019) and Akrour et al. (2019). They use either an exponential or linear decay of the entropy during policy optimization to control the exploration process and escape local optima. To leverage those benefits, we embed this entropy control mechanism in our differentiable trust region layers.

3 Preliminaries and Problem Statement

We consider the general problem of a policy search in a *Markov Decision Process* (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. We assume the state space $\mathcal{S}$ and action space $\mathcal{A}$ are continuous and the transition probabilities $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ describe the probability transitioning to state $\boldsymbol{s}_{t+1} \in \mathcal{S}$ given the current state $\boldsymbol{s}_t \in \mathcal{S}$ and action $\boldsymbol{a}_t \in \mathcal{A}$. The reward returned by the environment is given by a function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $\gamma \in [0, 1)$ describes the discount factor. Our goal is to maximize the expected accumulated discounted reward $R_t^\gamma = \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$, where ρ_π represents the state-action marginals of the trajectory distribution with $a_0, s_0, ... \sim \rho_\pi$ based on a policy $\pi(a|s)$. To find the optimal policy, traditional PG methods often make use of the likelihood ratio gradient and an importance sampling estimator. Moreover, instead of directly optimizing the returns, it has been shown to be more effective to optimize the advantage function as this results in an unbiased estimator of the gradient with less variance

$$\max_\theta \hat{J}(\pi_\theta, \pi_{\theta_{\text{old}}}) = \max_\theta \mathbb{E}_{\boldsymbol{s}_t \sim \mathcal{T}(\cdot|\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}), \boldsymbol{a}_t \sim \pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{s}_t)} \left[\frac{\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}_t|\boldsymbol{s}_t)} A^\pi(\boldsymbol{s}_t, \boldsymbol{a}_t) \right], \tag{1}$$

where $A^\pi(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}\left[R_t^\gamma | \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}; \pi\right] - \mathbb{E}\left[R_t^\gamma | \boldsymbol{s}_t = \boldsymbol{s}; \pi\right]$ describes the advantage function. The advantage function can be estimated by *generalized advantage estimation* (GAE) (Schulman et al., 2015b). This objective is used by most TRPO and PPO implementations, even though GAE was not used in the original TRPO paper. Trust region methods use additional constraints for the given objective. Using a constraint on the maximum KL over the states has been shown to guarantee monotonic improvement of the policy (Schulman et al., 2015a). However, as all current approaches do not use a maximum KL constraint but an expected KL constraint, thus the monotonic improvement guarantee also does not hold exactly. We are not aware of such results for the W2 distance or the Frobenius norm.

For our projections we assume Gaussian polices $\pi_{\theta_{\text{old}}}(\boldsymbol{a}_t|\boldsymbol{s}_t) = \mathcal{N}(\boldsymbol{a}_t|\mu_{\text{old}}(\boldsymbol{s}_t), \Sigma_{\text{old}}(\boldsymbol{s}_t))$ as well as $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) = \mathcal{N}(\boldsymbol{a}_t|\mu(\boldsymbol{s}_t), \Sigma(\boldsymbol{s}_t))$ represent the old and the current policy, respectively. We explore different trust regions on top of Equation 1 that employ different similarity metrics between old and new distributions, i. e., the commonly used reverse KL, the Wasserstein L2 distance and the Frobenius norm.

Reverse KL Divergence. The KL-divergence between two Gaussian distributions with means μ_1 and μ_2 and covariances Σ_1 and Σ_2 can generally be written as

$$\text{KL}(\{\mu_1, \Sigma_1\} \parallel \{\mu_2, \Sigma_2\}) = \frac{1}{2}\left[(\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) + \log \frac{|\Sigma_2|}{|\Sigma_1|} - d + \text{tr}\{\Sigma_2^{-1}\Sigma_1\}\right].$$

The KL uses the Mahalanobis distance to measure the similarity between the two mean vectors. The similarity of the covariance is measured by the difference in entropy of both distributions (the log-terms) that measures the difference in the "shape" or rotation of the covariance. The KL is not symmetric but can be used as similarity measure as it is always non-negative. We will use the more common reverse KL for our trust region, where the first argument is the new policy and the second the old policy.

Wasserstein Distance. The Wasserstein distance is a distance measure based on an optimal transport formulation, for more details see Villani (2008). The Wasserstein-2 distance for two Gaussian distributions can generally be written as

$$\mathcal{W}_2\left(\{\mu_1, \Sigma_1\}, \{\mu_2, \Sigma_2\}\right) = |\mu_1 - \mu_2|^2 + \text{tr}\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_2^{1/2}\Sigma_1\Sigma_2^{1/2}\right)^{1/2}\right).$$

A key difference to the KL-divergence is that the the Wasserstein distance is a symmetric distance measure, i. e., $\mathcal{W}_2(q, p) = \mathcal{W}_2(p, q)$. Our experiments also revealed that it is beneficial to measure the W2 distance in a metric space defined by the covariance of the old policy distribution (denoted here as Σ_2) as the distance measure is then more sensitive to the data-generating distribution. The

W2 distance in this metric space reads

$$\mathcal{W}_{2,\Sigma_2}\left(\{\mu_1,\Sigma_1\},\{\mu_2,\Sigma_2\}\right) = (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1) + \operatorname{tr}\left(\mathbb{I}+\Sigma_2^{-1}\Sigma_1 - 2\Sigma_2^{-1}\left(\Sigma_2^{1/2}\Sigma_1\Sigma_2^{1/2}\right)^{1/2}\right).$$

Frobenius Norm. The Frobenius norm is a matrix norm and can be directly applied to the difference of the covariance matrices of the Gaussian distributions. To measure the distance of the mean vectors, we will, similar to the KL-divergence, employ the Mahalanobis distance as this empirically lead to an improved performance in comparison to just taking the squared distance. Hence, we will denote the following metric as Frobenius norm between two Gaussian distributions

$$F(\{\mu_1,\Sigma_1\},\{\mu_2,\Sigma_2\}) = (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1) + \operatorname{tr}\big((\Sigma_2-\Sigma_1)^T(\Sigma_2-\Sigma_1)\big).$$

The Frobenius norm also constitutes a symmetric distance measure.

4 Differentiable Trust-Region Layers for Gaussian Policies

We present projections based on the three similarity metrics, i. e., Frobenius norm, Wasserstein L2 distance, and KL-divergence. These projections realize state-wise trust regions and can directly be integrated in the optimization process as differentiable neural network layers. Additionally, we extend the trust region layers to include an entropy constraint to gain control over the evolution of the policy entropy during optimization. The trust regions are defined by means of a distance or divergence $d(\pi(\cdot|\boldsymbol{s}),\pi_{\text{old}}(\cdot|\boldsymbol{s}))$ between probability distributions. Complementing Equation 1 with the trust region constraint leads to

$$\max_{\theta} \hat{J}(\pi_\theta,\pi_{\theta_{\text{old}}}) \quad \text{s.t.} \quad d(\pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{s})),\pi_\theta(\cdot|\boldsymbol{s})) \leq \epsilon \quad \forall \boldsymbol{s}\in\mathcal{S}. \tag{2}$$

While, in principle, we want to enforce the constraint for every possible state, in practice, we can only enforce them for states sampled from rollouts of the current policy.

To solve the problem in Equation 2, a standard neural network will output the parameters μ,Σ of a Gaussian distribution π_θ, ignoring the trust region bounds. Note that μ,Σ are state dependent, which we will however neglect for ease of notation. These parameters are provided to the trust region layers, together with the mean and covariance of the old policy and a parameter specifying the size of the trust region ϵ. The output of the trust region layer is then considered to be the new policy. As all distances or divergences used in this paper can be decomposed into a mean and a covariance dependent part, we also use separate trust regions and bounds for mean and covariance, as this gives the algorithm more flexibility. The trust region layers aim to project π_θ into the trust region by finding parameters $\tilde{\mu}$ and $\tilde{\Sigma}$ that are closest to the original parameters while not violating the trust region constraints. The projection is based on the same distance that was used to define the respective trust region. Formally, this corresponds to the following optimization problems

$$\arg\min_{\tilde{\mu}} d_\mu\left(\tilde{\mu},\mu\right), \quad \text{s.t.} \quad d_\mu\left(\tilde{\mu},\mu_{\text{old}}\right) \leq \epsilon_\mu, \quad \text{and} \tag{3}$$

$$\arg\min_{\tilde{\Sigma}} d_\Sigma\left(\tilde{\Sigma},\Sigma\right), \quad \text{s.t.} \quad d_\Sigma\left(\tilde{\Sigma},\Sigma_{\text{old}}\right) \leq \epsilon_\Sigma, \tag{4}$$

where d_μ is the mean dependent part and d_Σ is the covariance dependent part of the employed similarity measure. We denote the projected policy as $\tilde{\pi}(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\tilde{\mu},\tilde{\Sigma})$.

4.1 Projection of the Mean

All three trust region projections make use of the same distance measure for the mean, i. e., the Mahalanobis distance. Hence, the optimization problem for the mean is given by

$$\arg\min_{\tilde{\mu}} \left(\mu-\tilde{\mu}\right)^{\mathrm{T}}\Sigma_{\text{old}}^{-1}\left(\mu-\tilde{\mu}\right) \quad \text{s.t.} \quad \left(\mu_{\text{old}}-\tilde{\mu}\right)^{\mathrm{T}}\Sigma_{\text{old}}^{-1}\left(\mu_{\text{old}}-\tilde{\mu}\right) \leq \epsilon_\mu,$$

By making use of the method of Lagrangian multipliers, we can formulate the dual and solve it for the projected mean $\tilde{\mu}$ as

$$\tilde{\mu} = \frac{\mu+\omega\mu_{\text{old}}}{1+\omega} \quad \text{with} \quad \omega = \sqrt{\frac{\left(\mu_{\text{old}}-\mu\right)^{\mathrm{T}}\Sigma_{\text{old}}^{-1}\left(\mu_{\text{old}}-\mu\right)}{\epsilon_\mu}} - 1.$$

This equation can directly be used as mean for the Gaussian policy, while it easily allows to compute gradients.

4.2 PROJECTION OF THE COVARIANCE

Frobenius Projection. The Frobenius projection formalizes the trust region for the covariance with the squared Frobenius norm of the matrix difference, which yields

$$\arg\min_{\tilde{\Sigma}} \operatorname{tr}\left((\Sigma - \tilde{\Sigma})^T(\Sigma - \tilde{\Sigma})\right), \quad \text{s.t.} \quad \operatorname{tr}\left((\Sigma_{\text{old}} - \tilde{\Sigma})^T(\Sigma_{\text{old}} - \tilde{\Sigma})\right) \leq \epsilon_\Sigma.$$

We can again use the method of Lagrangian multipliers and obtain the covariance $\tilde{\Sigma}$ as

$$\tilde{\Sigma} = \frac{\Sigma + \eta\Sigma_{\text{old}}}{1+\eta} \quad \text{with} \quad \eta = \sqrt{\frac{\operatorname{tr}\left((\Sigma_{\text{old}} - \Sigma)^T(\Sigma_{\text{old}} - \Sigma)\right)}{\epsilon_\Sigma}} - 1. \tag{5}$$

where η is the corresponding Lagrangian multiplier.

Wasserstein Projection. Deriving the Wasserstein projection follows the same procedure. We obtain the following optimization problem

$$\begin{aligned} &\arg\min_{\tilde{\Sigma}} \operatorname{tr}\left(\Sigma_{\text{old}}^{-1}\Sigma + \Sigma_{\text{old}}^{-1}\tilde{\Sigma} - 2\Sigma_{\text{old}}^{-1}\left(\Sigma^{1/2}\tilde{\Sigma}\Sigma^{1/2}\right)^{1/2}\right), \\ &\text{s.t.} \quad \operatorname{tr}\left(\mathbb{I} + \Sigma_{\text{old}}^{-1}\tilde{\Sigma} - 2\Sigma_{\text{old}}^{-1}\left(\Sigma_{\text{old}}^{1/2}\tilde{\Sigma}\Sigma_{\text{old}}^{1/2}\right)^{1/2}\right) \leq \epsilon_\Sigma, \end{aligned} \tag{6}$$

where $\mathbb{I}$ is the identity matrix. To find the projected covariance $\tilde{\Sigma}$, we make the simplifying assumption that both the current Σ and the old covariance Σ_{old} commute with $\tilde{\Sigma}$. Finding a closed form solution is also possible without this assumption by using the expression for a geodesic between two Gaussian distributions derived by Takatsu (2011). However, in practice we found this approach to be numerically unstable. Further, under the common assumption of diagonal covarinces, the commutativity assumption holds. For the more general case of arbitrary covariance matrices, we would need to ensure the matrices are sufficiently close together, which is effectively ensured by Equation 6. Again, we introduce Lagrange multipliers and solve the dual problem to obtain the optimal primal and dual variables. Note however, that here we chose the *square root* of the covariance matrix[1] as primal variable. The corresponding projection for the square root covariance $\tilde{\Sigma}^{1/2}$ is then

$$\tilde{\Sigma}^{1/2} = \frac{\Sigma^{1/2} + \eta\Sigma_{\text{old}}^{1/2}}{1+\eta} \quad \text{with} \quad \eta = \sqrt{\frac{\operatorname{tr}\left(\mathbb{I} + \Sigma_{\text{old}}^{-1}\Sigma - 2\Sigma_{\text{old}}^{-1/2}\Sigma^{1/2}\right)}{\epsilon_\Sigma}} - 1. \tag{7}$$

where η is the corresponding Lagrangian multiplier. We see the same pattern emerging as for the Frobenius projection. The chosen similarity measure directly reappears in the expression for the Lagrangian multipliers and the primal variables are weighted averages of the corresponding parameters of the old policy and the predicted one.

KL Projection. Identically to the previous two projections, we reformulate Equation 4 as

$$\arg\min_{\tilde{\Sigma}} \operatorname{tr}\left(\Sigma^{-1}\tilde{\Sigma}\right) + \log\frac{|\Sigma|}{|\tilde{\Sigma}|}, \quad \text{s.t.} \quad \operatorname{tr}\left(\Sigma_{\text{old}}^{-1}\tilde{\Sigma}\right) - \dim(a) + \log\frac{|\Sigma_{\text{old}}|}{|\tilde{\Sigma}|} \leq \epsilon_\Sigma, \tag{8}$$

where $\dim(a)$ is the dimensionality of the action space. It is impossible to obtain a fully closed form solution for this problem. However, following Abdolmaleki et al. (2015), we can obtain the precision $\tilde{\Lambda} = \tilde{\Sigma}^{-1}$ by interpolation between the precision matrices of the old policy π_{old} and the current policy π

$$\tilde{\Lambda} = \frac{\eta^*\Lambda_{\text{old}} + \Lambda}{\eta^* + 1}, \quad \eta^* = \arg\min_{\eta} g(\eta), \text{ s.t. } \eta \geq 0, \tag{9}$$

where η is the corresponding Lagrangian multiplier and $g(\eta)$ the dual function. While this dual cannot be solved in closed form, an efficient solution using a standard numerical optimizer, such as BFGS, is still possible, as it is a 1-D convex optimization. Yet, we want to propagate gradients through our projection and thus also need to propagate them back through the numerical optimization. To this end, we follow Amos & Kolter (2017) and compute those gradients by taking the differentials of the KKT conditions of the dual. We refer to Appendix A.5 for more details and derivations.

[1] We assume the true matrix square root $\Sigma = \Sigma^{1/2}\Sigma^{1/2}$, not a Cholesky factor $\Sigma = LL^{\mathrm{T}}$.

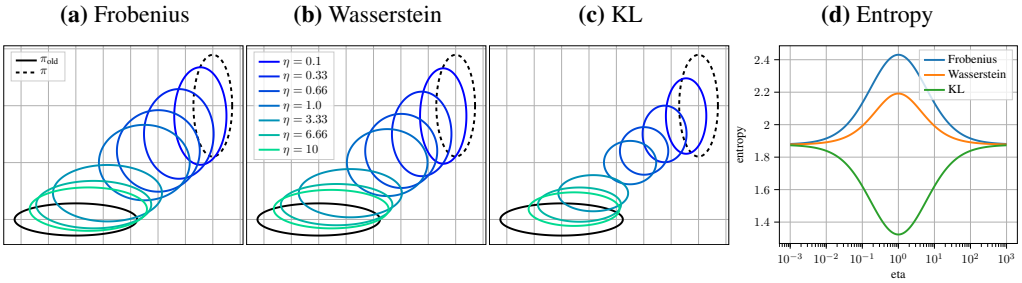

Figure 1: **(a)**, **(b)**, and **(c)**: Interpolated covariances for the different projections for various values of η. For Frobenius and Wasserstein the intermediate distributions clearly have larger entropy, while for the KL projection the intermediate entropy is smaller. **(d)**: Entropy of the interpolated covariances. In this example π and π_{old} have the same entropy. It can be seen that the entropy increases for the Frobenius and Wasserstein projections when moving from one distribution to the other, while it decreases for the KL. A more general statement regarding this can be found in Theorem 1.

Entropy Projection. So far we mainly focused on the trust region constraint and did not consider an entropy constraint $H(\tilde{\pi}) = \beta$. However, for our experiments we propose to complement the trust region layer with an optional entropy projection. Akrour et al. (2019) and Abdolmaleki et al. (2015) already demonstrated the benefit of such a constraint to allow for more control of the exploration. For that, we make use of the results from Akrour et al. (2019) and scale the standard deviation of the Gaussian distribution with a scalar factor $\exp\left\{\left(\beta - \mathcal{H}(\pi_\theta)\right)/\dim(\boldsymbol{a})\right\}$, which can also be individually computed per state.

4.3 Analysis of the Projections

It is instructive to compare the three projections. The covariance update is an interpolation for all three projections, yet the quantities that are interpolated differ. For the Frobenius projection we directly interpolate between the old and current covariances, for the Wasserstein projection between their respective matrix square roots, and for the KL projection between their inverses. In other words, each projection suggests which parametrization to use for the covariance matrix. The different interpolations also have an interesting effect on the entropy of the resulting covariances which can be observed in Figure 1. Further, we can proof the following theorem about the entropy of the projected distributions

Theorem 1 Let π_θ and $\pi_{\theta_{\text{old}}}$ be Gaussian and $\eta \geq 0$, then for the entropy of the projected distribution $\mathrm{H}(\tilde{\pi})$ it holds that $\mathrm{H}(\tilde{\pi}) \geq \text{minimum}(\mathrm{H}(\pi_\theta), \mathrm{H}(\pi_{\theta_{\text{old}}}))$ for the Frobenius (Equation 5) and the Wasserstein projection (Equation 7), as well as, $\mathrm{H}(\tilde{\pi}) \leq \text{maximum}(\mathrm{H}(\pi_\theta), \mathrm{H}(\pi_{\theta_{\text{old}}}))$ for the KL projection (Equation 9).

The proof is based on the multiplicative version of the Brunn-Minkowski inequality and can be found in Appendix A.1. Intuitively, this implies that the Frobenius and Wasserstein projections act more *aggressively*, i. e., they rather yield a higher entropy, while the KL projection acts more *conservatively*, i. e., it rather yields a smaller entropy. This could also explain why many KL based trust region methods lose entropy too quickly and converge prematurely. Yet, by introducing an explicit entropy control, those effects can be mitigated.

4.4 Successive Policy Updates

The above projections can directly be implemented for training the current policy. However, for successive policy updates the projections become rather impractical. Each projection would rely on calling the projection of the preceding policy.

Hence, we need to encode the information of the projection layer into the parameters θ of the next policy. Formally, we aim to find a set of parameters $\theta^* = \arg\min_\theta \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\text{old}}}}\left[d\left(\tilde{\pi}(\cdot|\boldsymbol{s}), \pi_\theta(\cdot|\boldsymbol{s})\right)\right]$, where $p_{\pi_{\text{old}}}$ is the state distribution of the old policy and d is the similarity measure used for the projection, such that we minimize the expected distance or divergence between the projection and the current policy prediction,

The most intuitive way to solve this problem is to use the existing samples for additional regression steps after the policy optimization. Yet, this adds a computational overhead. Therefore, we propose to concurrently optimize both objectives during training by penalizing the main objective, i. e.,

$$\arg\min_{\theta} \mathbb{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\pi_{\theta_{\text{old}}}} \left[\frac{\tilde{\pi}(\boldsymbol{a}_t|\boldsymbol{s}_t,\pi_\theta)}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}_t|\boldsymbol{s}_t)} A^{\pi}(\boldsymbol{a}_t,\boldsymbol{s}_t) \right] - \alpha \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\text{old}}}} \left[d\left(\tilde{\pi}(\cdot|\boldsymbol{s}), \pi_\theta(\cdot|\boldsymbol{s})\right)\right]. \tag{10}$$

Note that the gradient for the penalty is not flowing through the projection, only through the original network outputs for the regression loss. As appropriate similarity measures d for the penalty, we resort to the measures used in each projection.

5 EXPERIMENTS

We evaluate the performance of our trust region layers regarding sample complexity and final reward in comparison to PAPI and PPO on the OpenAI gym benchmark suite (Brockman et al., 2016). We explicitly did not include TRPO in the evaluation, as Engstrom et al. (2020) showed PPO and TRPO can achieve similar performance.

For our experiments, the PPO hyperparameters are based on the original publication (Schulman et al., 2017). The PAPI projection as well as its conservative PPO version are executed in the setting sent to us by the author. For our projections, all parameters are selected with Optuna (Akiba et al., 2019). See Appendix C for a full listing of all hyperparameter. The same hyperparameters are used for all experiments, except for the Humanoid where we optimize all algorithms using new Optuna runs. We reference the Frobenius projection as *FROB*, the Wasserstein projection as *W2*, and the KL-projection as *KL*. Standard PPO is using a lot of code-level optimizations which are not used by our approach. We only reuse the observation normalization as well as the default network architecture and initialization from the PPO implementation. Further, all algorithms leverage the same policy parametrization, a non-contextual diagonal covariance. Engstrom et al. (2020) showed that the performance of PPO degrades without heuristics such as reward normalization.

Algorithm Comparison. Table 1 shows the mean return with a 95% confidence interval for the last 20 epochs as well as for the first 100 epochs over ten seeds to get an overview of the final performance as well as the initial learning speed. We computed five evaluation episodes, i. e., with learning disabled, after every epoch without applying exploration noise to obtain the return values. The learning curves are shown in (Figure 4) in the appendix. The results show that our trust region layers are able to perform similar or better than PPO and PAPI across all tasks. Yet, there is no clear winner between the projections, PPO, and PAPI. PPO and PAPI seem to be slightly better on the Walker-v2 and Hopper-v2, while the projections outperform the other methods on the HalfCheetah-v2. The performance on the Ant-v2 and the Humanoid-v2 is en par. We also note, however, that PAPI is using the entropy projection that we do not include in this experiment to provide a fair comparison to PPO. PAPI has a poor asymptotic performance on the HalfCheetah-v2 task, whereas our projections remain robust and outperform PPO by a larger margin. Overall, we find that our Wasserstein projection is performing especially well in the beginning of the training, while the KL projections tends to achieve a slightly better asymptotic performance. Practically, we found the Frobenius projection to be the least stable variant, which also reflects in its rather low final performance. Nevertheless, by introducing an entropy projection we could alleviate this behavior as discussed in the next section.

Entropy Projection To demonstrate the effect of complementing our projections with an entropy control, as described in Section 4.2, we train additional agents on the more complex Ant-v2 and Humanoid-v2. The target entropy for the entropy projection in each iteration n is computed by exponentially decaying the initial entropy $\mathcal{H}_0$ to κ with temperature τ as $\kappa + (\mathcal{H}_0 - \kappa)\tau^{\frac{10n}{N}}$, where N is the total number of training steps. Figure 2 (left and center) shows the mean total reward and

Table 1: Mean reward of the last 20 epochs (last) and the first 100 epochs (first). We trained ten agents on different seeds for each method and computed five evaluation rollouts per epoch.

	Hopper-v2		Walker2d-v2		Halfcheetah-v2		Ant-v2		Humanoid-v2	
	first	last	first	last	first	last	first	last	first	last
FROB	925 ± 255	2078 ± 68	514 ± 97	3019 ± 103	631 ± 354	3486 ± 77	321 ± 150	2460 ± 123	651 ± 241	4840 ± 34
W2	$\mathbf{1042 \pm 314}$	2273 ± 51	$\mathbf{559 \pm 157}$	2879 ± 123	618 ± 346	4279 ± 68	315 ± 158	2930 ± 72	$\mathbf{770 \pm 384}$	5216 ± 127
KL	1022 ± 298	2186 ± 69	526 ± 126	2692 ± 124	$\mathbf{777 \pm 341}$	$\mathbf{5038 \pm 79}$	235 ± 136	3136 ± 119	736 ± 341	$\mathbf{5254 \pm 113}$
PAPI	829 ± 281	2434 ± 67	382 ± 77	3452 ± 77	197 ± 153	2531 ± 24	$\mathbf{473 \pm 197}$	$\mathbf{3330 \pm 62}$	490 ± 142	5252 ± 248
PPO	971 ± 341	$\mathbf{2477 \pm 64}$	480 ± 110	$\mathbf{3668 \pm 89}$	427 ± 260	3917 ± 37	378 ± 175	2495 ± 71	515 ± 166	5100 ± 257

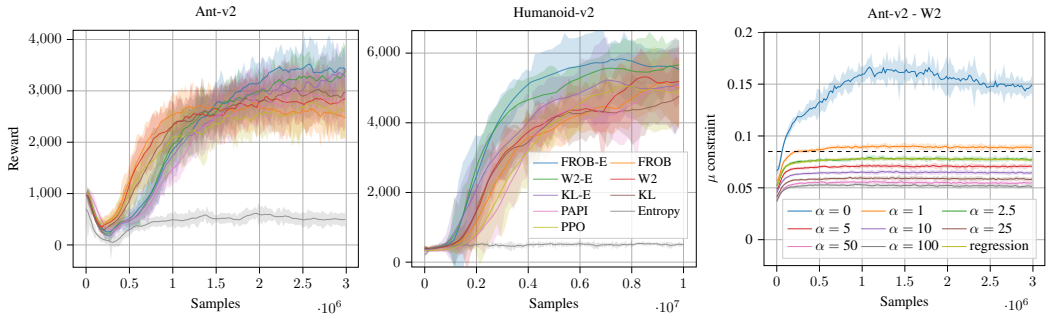

Figure 2: Performance of the trust region projection with (-E) and without entropy control as well as the entropy projection on its own (left and center). We further show the performance for different α values in comparison to a full regression (right). The mean bound is set to 0.085 (dotted black line).

a 95% confidence interval for our projection with (-E) and without entropy, as well as the entropy projection on its own. Exclusively leveraging the entropy projection in the trust region layer yields poor results, for none of the tasks the agents are able to converge. The trust region projections, on the other hand, significantly benefit from the entropy control in both cases. It leads to better asymptotic performance on the Ant-v2, and for the Humanoid-v2 additionally enables faster convergence. For the other Mujoco tasks we did not observe this effect as the complexity of the problems is not high enough to effectively take advantage of explicit entropy control. This aligns with the findings from previous work (Pajarinen et al., 2019; Abdolmaleki et al., 2015) that entropy control is particularly helpful in environments which require more exploration.

Trust Region Regression Loss. Another key element of our algorithm is the trust region regression loss, Equation 10, which helps to enforce the trust region for successive policies. Therefore, we explore how different choices of the regression weight α affect the constraint satisfaction. Figure 2 (right) shows the mean constraint value for the policy predictions without the projection for different α values. In addition we included a comparison to a slightly different version of our algorithm, where we do not use a regression penalty but instead run several iterations of regressing the current policy towards the projection after each policy update.

Without penalty or explicit regressions, i. e., $\alpha = 0$, the projections are not able to enforce the bound. Thus, in the next iteration of the algorithm, the un-projected policy violates the bounds. Another key observation is that decreasing the penalty leads to larger changes in the policy and pushes the mean closer to its maximum bound. Intuitively, this can be explained by the construction of the bound. The penalty is only added to the loss, when the bound is violated. Therefore, large changes in the policy are punished while making small steps might affect the achieved reward, but does not incur any other negative effect. By selecting a larger α, this behavior is reinforced. Furthermore, we can see that small values of α achieve a behavior which is very similar to the full regression setting. Therefore, it is justified to use a penalty, which is computationally more efficient, instead of performing a full regression after each policy update.

6 Discussion and Future Work

In this work we proposed differentiable projection layers to enforce trust region constraints for Gaussian policies in deep reinforcement learning. While being more stable than existing methods, they also offer the benefit of imposing the constraints on a state level. Our results illustrate that trust regions are an effective tool in policy search for a wide range of different similarity measures. Apart from the commonly used reverse KL, we also use the Wasserstein distance und Frobenius norm. We demonstrated the subtle but important differences between those three different types of trust regions and showed our benchmark performance is on par or better than existing methods that use more code-level optimizations. For future work we plan to leverage the property of state level projections also for contextual covariances, which are relevant for more complex environments. Additionally, more sophisticated heuristics or learning methods could be used to adapt the trust region bounds for better performance. Lastly, we are interested in using our trust region layers for other deep reinforcement learning approaches, such as actor-critic methods.

REFERENCES

A. Abdolmaleki, R. Lioutikov, J Peters, N. Lau, L. Reis, and G. Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems 28*, pp. 3537–3545, 2015. URL http://www.ausy.tu-darmstadt.de/uploads/Team/GerhardNeumann/Abdolmaleki_NIPS2015.pdf.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable Convex Optimization Layers. *Advances in Neural Information Processing Systems*, 32, oct 2019. URL http://arxiv.org/abs/1910.12430.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, 2019. URL http://arxiv.org/abs/1907.10902.

Riad Akrour, Joni Pajarinen, Jan Peters, and Gerhard Neumann. Projections for approximate policy iteration algorithms. In *Proceedings of Machine Learning Research*, pp. 181–190, 2019.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *34th International Conference on Machine Learning*, pp. 179–191, 2017. URL http://arxiv.org/abs/1703.00443.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study. 2020. URL http://arxiv.org/abs/2006.05990.

Oleg Arenz, Mingjun Zhong, and Gerhard Neumann. Efficient Gradient-Free Variational Inference using Policy Search. In *Proceedings of Machine Learning Research*, pp. 234–243, 2018. URL http://proceedings.mlr.press/v80/arenz18a.html.

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

Yinlam Chow, Ofir Nachum, Aleksandra Faust, Mohammad Ghavamzadeh, and Edgar Duenez-Guzman. Lyapunov-based Safe Policy Optimization for Continuous Control. In *ICML Workshop RL4RealLife Submission*, 2019.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe Exploration in Continuous Action Spaces. 2018. URL https://arxiv.org/abs/1801.08757.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. *33rd International Conference on Machine Learning*, pp. 2001–2014, 2016. URL http://arxiv.org/abs/1604.06778.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. In *International Conference on Learning Representations*, 2020. URL http://arxiv.org/abs/2005.12729.

Sham Kakade. A natural policy gradient. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pp. 1531–1538, Cambridge, MA, USA, 2001. MIT Press.

Sham M. Kakade and John C. Langford. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 267–274, 2002. URL https://dl.acm.org/doi/10.5555/645531.656005.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018. URL http://arxiv.org/abs/1805.00909.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. In *The Journal of Machine Learning Research*, volume 17, pp. 1334–1373, 2015. URL http://arxiv.org/abs/1504.00702.

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

Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Anna Choromanska, Krzysztof Choromanski, and Michael I Jordan. Learning to Score Behaviors for Guided Policy Optimization. In *Proceedings of the International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/1906.04349.

Joni Pajarinen, Hong Linh Thai, Riad Akrour, Jan Peters, and Gerhard Neumann. Compatible Natural Gradient Policy Search. *Machine Learning*, 108(8-9):1443–1466, 2019. URL http://arxiv.org/abs/1902.02823.

J. Peters, K. Muelling, and Y. Altun. Relative entropy policy search. In *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence (AAAI), Physically Grounded AI Track*, 2010. URL http://www.ias.informatik.tu-darmstadt.de/uploads/Team/JanPeters/Peters2010_REPS.pdf.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008. URL https://www.ias.informatik.tu-darmstadt.de/uploads/Team/JanPeters/Peters2010_REPS.pdf.

K. B. Petersen and M. S. Pedersen. The matrix cookbook, 2012. URL http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html. Version 20121115.

Pierre H. Richemond and Brendan Maginnis. On Wasserstein Reinforcement Learning and the Fokker-Planck equation. *arXiv preprint*, 2017. URL http://arxiv.org/abs/1712.07185.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *Proceedings of Machine Learning Research*, pp. 1889–1897, 2015a. URL http://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations*, 2015b. URL http://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. 2017. URL http://arxiv.org/abs/1707.06347.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. pp. 1–19, 2017. URL http://arxiv.org/abs/1712.01815.

H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin Riedmiller, and Matthew M. Botvinick. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylOlp4FvH.

Jun Song and Chaoyue Zhao. Optimistic Distributionally Robust Policy Optimization. pp. 15872–15882, 2020. URL https://arxiv.org/abs/2006.07815.

Asuka Takatsu. Wasserstein geometry of Gaussian measures. *Osaka Journal of Mathematics*, 48: 1005–1026, 2011.

Voot Tangkaratt, Abbas Abdolmaleki, and Masashi Sugiyama. Guide actor-critic for continuous control. In *Proceedings of the International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1705.07606v2.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Yuhui Wang, Hao He, Xiaoyang Tan, and Yaozhong Gan. Trust region-guided proximal policy optimization. In *Advances in Neural Information Processing Systems 32*, pp. 626–636, 2019. URL https://arxiv.org/abs/1901.10314.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-Based Constrained Policy Optimization. In *International Conference on Learning Representations*, 2020. URL https://sites.google.com/view/iclr2020-pcpo.

A DERIVATIONS

A.1 PROOF OF THEOREM 1

This section provides a proof for Theorem 1. We mainly used the multiplicative version of the Brunn-Minkowski inequality

$$\log(\alpha|\Sigma_1| + \beta|\Sigma_2|) \geq \log(|\Sigma_1|)^{\alpha}(|\Sigma_2|)^{\beta}$$

where Σ_1, Σ_2 are p.s.d, α, β are positive, and $\alpha + \beta = 1$.

Frobenius Projection

$$\begin{aligned}
\mathrm{H}(\tilde{\pi}) &= 0.5 \log |2\pi e \tilde{\Sigma}| \\
&= 0.5 \log \left| 2\pi e \left(\frac{1}{\eta+1}\Sigma + \frac{\eta}{\eta+1}\Sigma_{\text{old}} \right) \right| \\
&\geq 0.5 \log \left| (2\pi e \Sigma)^{\frac{1}{\eta+1}} \det (2\pi e \Sigma_{\text{old}})^{\frac{\eta}{\eta+1}} \right| \\
&= \frac{1}{\eta+1} 0.5 \log |2\pi e \Sigma| + \frac{\eta}{\eta+1} 0.5 \log |2\pi e \Sigma_{\text{old}}| \\
&= \frac{1}{\eta+1}\mathrm{H}(\pi_\theta) + \frac{\eta}{\eta+1}\mathrm{H}(\pi_{\theta_{\text{old}}}) \geq \text{minimum}\left(\mathrm{H}(\pi_\theta), \mathrm{H}(\pi_{\theta_{\text{old}}})\right)
\end{aligned}$$

Wasserstein Projection Let k denote the dimensonality of the distributions under consideration.

$$\begin{aligned}
\mathrm{H}(\tilde{\pi}) &= 0.5 \log (2\pi e)^k |\tilde{\Sigma}| \\
&= 0.5 \log (2\pi e)^k \left| \left(\frac{1}{\eta+1}\Sigma^{0.5} + \frac{\eta}{\eta+1}\Sigma_{\text{old}}^{0.5} \right) \right|^2 \\
&= 0.5 \log (2\pi e)^k + \log \left| \frac{1}{\eta+1}\Sigma^{0.5} + \frac{\eta}{\eta+1}\Sigma_{\text{old}}^{0.5} \right| + \log \left| \frac{1}{\eta+1}\Sigma^{0.5} + \frac{\eta}{\eta+1}\Sigma_{\text{old}}^{0.5} \right| \\
&\geq 0.5 \log (2\pi e)^k + \log \left|\Sigma^{0.5}\right|^{\frac{1}{\eta+1}} \left|\Sigma_{\text{old}}^{0.5}\right|^{\frac{\eta}{\eta+1}} + \log \left|\Sigma^{0.5}\right|^{\frac{1}{\eta+1}} \left|\Sigma_{\text{old}}^{0.5}\right|^{\frac{\eta}{\eta+1}} \\
&= 0.5 \log (2\pi e)^k + \log |\Sigma|^{\frac{1}{\eta+1}} |\Sigma_{\text{old}}|^{\frac{\eta}{\eta+1}} \\
&= 0.5 \log \left(| \left(2\pi e \Sigma\right|^{\frac{1}{\eta+1}} |2\pi e \Sigma_{\text{old}}|^{\frac{\eta}{\eta+1}} \right) \\
&= \frac{1}{\eta+1} 0.5 \log |2\pi e \Sigma| + \frac{\eta}{\eta+1} 0.5 \log |2\pi e \Sigma_{\text{old}}| \\
&= \frac{1}{\eta+1}\mathrm{H}(\pi_\theta) + \frac{\eta}{\eta+1}\mathrm{H}(\pi_{\theta_{\text{old}}}) \geq \text{minimum}\left(\mathrm{H}(\pi), \mathrm{H}(\pi_{\text{old}})\right)
\end{aligned}$$

KL Projection

$$\begin{aligned}
\mathrm{H}(\tilde{\pi}) &= 0.5 \log |2\pi e \tilde{\Sigma}| \\
&= 0.5 \log \left| \frac{1}{\eta+1}(2\pi e \Sigma)^{-1} + \frac{\eta}{\eta+1}(2\pi e \Sigma_{\text{old}})^{-1} \right|^{-1} \\
&= -0.5 \log \left| \frac{1}{\eta+1}(2\pi e \Sigma)^{-1} + \frac{\eta}{\eta+1}(2\pi e \Sigma_{\text{old}})^{-1} \right| \\
&\leq -0.5 \log \left(\left|(2\pi e \Sigma)^{-1}\right|^{\frac{1}{\eta+1}} \left|(2\pi e \Sigma_{\text{old}})^{-1}\right|^{\frac{\eta}{\eta+1}} \right) \\
&= 0.5 \log \left(|2\pi e \Sigma|^{\frac{1}{\eta+1}} |2\pi e \Sigma_{\text{old}}|^{\frac{\eta}{\eta+1}} \right) \quad \text{(use the fact that: } \det(A^{-1}) = 1/\det(A)) \\
&= \frac{1}{\eta+1} 0.5 \log |2\pi e \Sigma| + \frac{\eta}{\eta+1} 0.5 \log |2\pi e \Sigma_{\text{old}}| \\
&= \frac{1}{\eta+1}\mathrm{H}(\pi_\theta) + \frac{\eta}{\eta+1}\mathrm{H}(\pi_{\theta_{\text{old}}}) \leq \text{maximum}\left(\mathrm{H}(\pi_\theta), \mathrm{H}(\pi_{\text{old}})\right)
\end{aligned}$$

A.2 MEAN PROJECTION

First, we consider only the mean objective

$$\begin{aligned} \min_{\tilde{\mu}} \quad & (\mu - \tilde{\mu})^{\mathrm{T}} \Sigma_{\text{old}}^{-1} (\mu - \tilde{\mu}) \\ \text{s.t.} \quad & (\mu_{\text{old}} - \tilde{\mu})^{\mathrm{T}} \Sigma_{\text{old}}^{-1} (\mu_{\text{old}} - \tilde{\mu}) \leq \epsilon_{\mu}, \end{aligned}$$

which give us the following dual

$$\mathcal{L}(\tilde{\mu}, \omega) = (\mu - \tilde{\mu})^{\mathrm{T}} \Sigma_{\text{old}}^{-1} (\mu - \tilde{\mu}) + \omega \left((\mu_{\text{old}} - \tilde{\mu})^{\mathrm{T}} \Sigma_{\text{old}}^{-1} (\mu_{\text{old}} - \tilde{\mu}) - \epsilon_{\mu} \right). \tag{11}$$

Differentiating w.r.t. $\tilde{\mu}$ yields

$$\frac{\partial \mathcal{L}(\tilde{\mu}, \omega)}{\partial \tilde{\mu}} = 2\Sigma_{\text{old}}^{-1} (\tilde{\mu} - \mu) - 2\omega \Sigma_{\text{old}}^{-1} (\tilde{\mu} - \mu_{\text{old}}).$$

Setting the derivative to 0 and solving for $\tilde{\mu}$ yields,

$$\tilde{\mu}^* = \frac{\mu + \omega \mu_{old}}{1 + \omega}.$$

Inserting the optimal mean $\tilde{\mu}^*$ in Equation 11 results in

$$\begin{aligned} L(\omega) = & \left(\frac{\mu + \omega \mu_{old}}{1 + \omega} - \mu \right)^T \Sigma_{\text{old}}^{-1} \left(\frac{\mu + \omega \mu_{old}}{1 + \omega} - \mu \right) + \\ & + \omega \left(\left(\frac{\mu + \omega \mu_{old}}{1 + \omega} - \mu_{old} \right)^T \Sigma_{\text{old}}^{-1} \left(\frac{\mu + \omega \mu_{old}}{1 + \omega} - \mu_{old} \right) - \epsilon_{\mu} \right) \\ = & \frac{\omega^2 (\mu - \mu_{old})^T \Sigma_{\text{old}}^{-1} (\mu - \mu_{old})}{(1 + \omega)^2} + \frac{\omega (\mu - \mu_{old})^T \Sigma_{\text{old}}^{-1} (\mu - \mu_{old})}{(1 + \omega)^2} - \omega \epsilon_{\mu}. \end{aligned}$$

Thus, differentiating w.r.t ω yields

$$\frac{\partial \mathcal{L}(\omega)}{\partial \omega} = \frac{(\mu - \mu_{old})^T \Sigma_{\text{old}}^{-1} (\mu - \mu_{old})}{(1 + \omega)^2} - \epsilon_{\mu}.$$

Now solving $\frac{\partial \mathcal{L}(\omega)}{\partial \omega} \stackrel{!}{=} 0$ for ω, we arrive at

$$\omega^* = \sqrt{\frac{(\mu - \mu_{old})^T \Sigma_{\text{old}}^{-1} (\mu - \mu_{old})}{\epsilon_{\mu}}} - 1.$$

A.3 FROBENIUS COVARIANCE PROJECTION

We consider the following objective for the covariance part

$$\begin{aligned} \min_{\tilde{\Sigma}} \quad & \left\| \tilde{\Sigma} - \Sigma \right\|_F^2 \\ \text{s.t.} \quad & \left\| \tilde{\Sigma} - \Sigma_{\text{old}} \right\|_F^2 \leq \epsilon_{\Sigma} \end{aligned}$$

with the corresponding Lagrangian

$$\mathcal{L}(\tilde{\Sigma}, \eta) = \left\| \tilde{\Sigma} - \Sigma \right\|_F^2 + \eta \left(\left\| \tilde{\Sigma} - \Sigma_{\text{old}} \right\|_F^2 - \epsilon_{\Sigma} \right). \tag{12}$$

Differentiating w.r.t. $\tilde{\Sigma}$ yields

$$\frac{\partial \mathcal{L}(\tilde{\Sigma}, \eta)}{\partial \tilde{\Sigma}} = 2 \left(\left(\tilde{\Sigma} - \Sigma \right) + \eta \left(\Sigma_{\text{old}} - \Sigma \right) \right).$$

We can again solve for $\tilde{\Sigma}$ by setting the derivative to 0, i.e.,

$$\tilde{\Sigma}^* = \frac{\Sigma + \eta \Sigma_{\text{old}}}{1 + \eta}.$$

Inserting $\tilde{\Sigma}^*$ into Equation 12 yields the dual function

$$g(\eta) = \left\| \frac{\Sigma + \eta \Sigma_{\text{old}}}{1 + \eta} - \Sigma \right\|_F^2 + \eta \left(\left\| \frac{\Sigma + \eta \Sigma_{\text{old}}}{1 + \eta} - \Sigma_{\text{old}} \right\|_F^2 - \epsilon_{\Sigma} \right).$$

Differentiating w.r.t. η results in

$$\frac{\partial \mathcal{L}(\eta)}{\partial \eta} = \frac{\|\Sigma - \Sigma_{\text{old}}\|_F^2}{(1+\eta)^2} - \epsilon_\Sigma.$$

Hence, $\frac{\partial \mathcal{L}(\eta)}{\partial \eta} \stackrel{!}{=} 0$ yields

$$\eta^* = \frac{\|\Sigma - \Sigma_{\text{old}}\|_F}{\sqrt{\epsilon_\Sigma}} - 1.$$

A.4 Wasserstein Covariance Projection

As described in the main text, the Gaussian distributions to have been rescaled by Σ_{old}^{-1} to measure the distance in the metric space that is defined by the variance of the data. For notational simplicity, we show the derivation of the covariance projection only for the unscaled scenario. The scaled version can be obtained by a simple redefinition of the covariance matrices. For our covariance projection we are interested in solving the following optimization problem

$$\begin{aligned} \min_{\tilde{\Sigma}} \quad & \operatorname{tr}\left(\tilde{\Sigma} + \Sigma - 2\left(\Sigma^{1/2}\tilde{\Sigma}\Sigma^{1/2}\right)^{1/2}\right) \\ \text{s.t.} \quad & \operatorname{tr}\left(\tilde{\Sigma} + \Sigma_{\text{old}} - 2\left(\Sigma_{\text{old}}^{1/2}\tilde{\Sigma}\Sigma_{\text{old}}^{1/2}\right)^{1/2}\right) \leq \epsilon_\Sigma, \end{aligned}$$

which leads to the following Lagrangian function

$$\begin{aligned} \mathcal{L}(\tilde{\Sigma}, \eta) = & \operatorname{tr}\left(\tilde{\Sigma} + \Sigma - 2\left(\Sigma^{1/2}\tilde{\Sigma}\Sigma^{1/2}\right)^{1/2}\right) \\ & + \eta\left(\operatorname{tr}\left(\tilde{\Sigma} + \Sigma_{\text{old}} - 2\left(\Sigma_{\text{old}}^{1/2}\tilde{\Sigma}\Sigma_{\text{old}}^{1/2}\right)^{1/2}\right) - \epsilon_\Sigma\right). \end{aligned} \tag{13}$$

Assuming that $\tilde{\Sigma}$ commutes with Σ as well as Σ_{old}, Equation 13 simplifies to

$$\begin{aligned} \mathcal{L}(\tilde{\Sigma}, \eta) &= \operatorname{tr}\left(\tilde{\Sigma} + \Sigma - 2\tilde{\Sigma}^{1/2}\Sigma^{1/2}\right) + \eta\left(\operatorname{tr}\left(\tilde{\Sigma} + \Sigma_{\text{old}} - 2\tilde{\Sigma}^{1/2}\Sigma_{\text{old}}^{1/2}\right) - \epsilon\right) \\ &= \operatorname{tr}\left(S^2 + \Sigma - 2S\Sigma^{1/2}\right) + \eta\left(\operatorname{tr}\left(S^2 + \Sigma_{\text{old}} - 2S\Sigma_{\text{old}}^{1/2}\right) - \epsilon\right), \end{aligned} \tag{14}$$

where S is the unique positive semi-definite root of the positive semi-definite matrix $\tilde{\Sigma}$, i.e. $S = \tilde{\Sigma}^{1/2}$. Instead of optimizing the objective w.r.t $\tilde{\Sigma}$, we optimize w.r.t S in order, which greatly simplifies the calculation. That is, we solve

$$\frac{\partial \mathcal{L}(S, \eta)}{\partial S} = (1+\eta)2S - 2\left(\Sigma^{1/2} + \eta\Sigma_{\text{old}}^{1/2}\right) \stackrel{!}{=} 0$$

for S, which leads us to

$$S^* = \frac{\Sigma^{1/2} + \eta\Sigma_{\text{old}}^{1/2}}{1+\eta}, \tilde{\Sigma}^* = \frac{\Sigma + \eta^2\Sigma_{\text{old}} + 2\eta\Sigma^{1/2}\Sigma_{\text{old}}^{1/2}}{(1+\eta)^2}.$$

Inserting this into Equation 14 yields the dual function

$$g(\eta) = \frac{\eta\left(\operatorname{tr}\left(\Sigma + \Sigma_{\text{old}} - 2\Sigma^{1/2}\Sigma_{\text{old}}^{1/2}\right)\right)}{1+\eta} - \eta\epsilon_\Sigma$$

The derivative of the dual w.r.t. η is given by

$$\frac{\partial \mathcal{L}(\eta)}{\partial \eta} = \frac{\operatorname{tr}\left(\tilde{\Sigma} + \Sigma_{\text{old}} - 2\tilde{\Sigma}^{1/2}\Sigma_{\text{old}}^{1/2}\right)}{(1+\lambda)^2} - \epsilon_\Sigma.$$

Now solving $\frac{\partial \mathcal{L}(\eta)}{\partial \eta} \stackrel{!}{=} 0$ for η, we arrive at

$$\eta^* = \sqrt{\frac{\operatorname{tr}\left(\tilde{\Sigma} + \Sigma_{\text{old}} - 2\tilde{\Sigma}^{1/2}\Sigma_{\text{old}}^{1/2}\right)}{\epsilon_\Sigma}} - 1$$

A.5 KL-DIVERGENCE PROJECTION

We derive the KL-Divergence projection in its general form, i.e., simultaneous projection of mean and covariance under an additional entropy constraint

$$\tilde{\pi}^* = \arg\min_{\tilde{\pi}} \mathrm{KL}\left(\tilde{\pi}||\pi_\theta\right) \quad \text{s.t.} \quad \mathrm{KL}\left(\tilde{\pi}||\pi_{\theta_{\text{old}}}\right) \leq \epsilon, \quad \mathrm{H}\left(\tilde{\pi}\right) \geq \beta.$$

Instead of working with this minimization problem we consider the equivalent maximization problem

$$\tilde{\pi}^* = \arg\max_{\tilde{\pi}} -\mathrm{KL}\left(\tilde{\pi}||\pi_\theta\right) \quad \text{s.t.} \quad \mathrm{KL}\left(\tilde{\pi}||\pi_{\theta_{\text{old}}}\right) \leq \epsilon, \quad \mathrm{H}\left(\tilde{\pi}\right) \geq \beta, \tag{15}$$

which is similar to the one considered in *Model Based Relative Entropy Strochastic Search* (MORE) (Abdolmaleki et al., 2015), with a few distinctions. To see those distinctions let η and ω denote the Lagrangian multipliers corresponding to the KL and entropy constraint respectively and consider the Lagrangian corresponding to the optimization problem in Equation 15

$$\begin{aligned}\mathcal{L} &= -\mathrm{KL}(\tilde{\pi}||\pi_\theta) + \eta\left(\epsilon - \mathrm{KL}(\tilde{\pi}||\pi_{\theta_{\text{old}}})\right) + \omega\left(\mathrm{H}(\tilde{\pi}) - \beta\right) \\ &= \mathbb{E}_{\tilde{\pi}}\left[\log \pi_\theta\right] + \eta\left(\epsilon - \mathrm{KL}(\tilde{\pi}||\pi_{\theta_{\text{old}}})\right) + (\omega+1)\mathrm{H}(\tilde{\pi}) - \omega\beta.\end{aligned}$$

Opposed to Abdolmaleki et al. (2015) we are not working with an unknown reward but using the log density of the target distribution π instead. Thus we do not need to fit a surrogate and can directly read off the parameters of the squared reward. They are given by the natural parameters of π, i.e, $\Lambda = \Sigma^{-1}$ and $q = \Sigma^{-1}\mu$. Additionally, we need to add a constant 1 to ω to account for the additional entropy term in the original objective, similar to (Arenz et al., 2018).

Following the derivations from Abdolmaleki et al. (2015) and Arenz et al. (2018) we can obtain a closed form solution for the natural parameters of $\tilde{\pi}$, given the Lagrangian multipliers η and ω

$$\tilde{\Lambda} = \frac{\eta\Lambda_{\text{old}} + \Lambda}{\eta + 1 + \omega} \quad \text{and} \quad \tilde{q} = \frac{\eta q_{\text{old}} + q}{\eta + 1 + \omega}. \tag{16}$$

To obtain the optimal Lagrangian multipliers we can solve the following convex dual function using gradient descent

$$\begin{aligned}g(\eta,\omega) =& \eta\epsilon - \omega\beta + \eta\left(-\frac{1}{2}q_{\text{old}}^T\Lambda_{\text{old}}^{-1}q_{\text{old}} + \frac{1}{2}\log\det\left(\Lambda\right) - \frac{k}{2}\log(2\pi)\right) \\ &+ (\eta+1+\omega)\left(\frac{1}{2}\tilde{q}^T\tilde{\Lambda}^{-1}q - \frac{1}{2}\log\det\left(\tilde{\Lambda}\right) + \frac{k}{2}\log(2\pi)\right) + \text{const}, \\ \frac{\partial g(\eta,\omega)}{\partial\eta} =& \epsilon - \mathrm{KL}(\tilde{\pi}||\pi_{\theta_{\text{old}}}) \quad \text{and} \quad \frac{\partial g(\eta,\omega)}{\partial\omega} = H(\tilde{\pi}) - \beta.\end{aligned}$$

Given the optimal Lagrangian multipliers, η^* and ω^* we obtain the parameters of the optimal distribution $\tilde{\pi}^*$ using Equation 16.

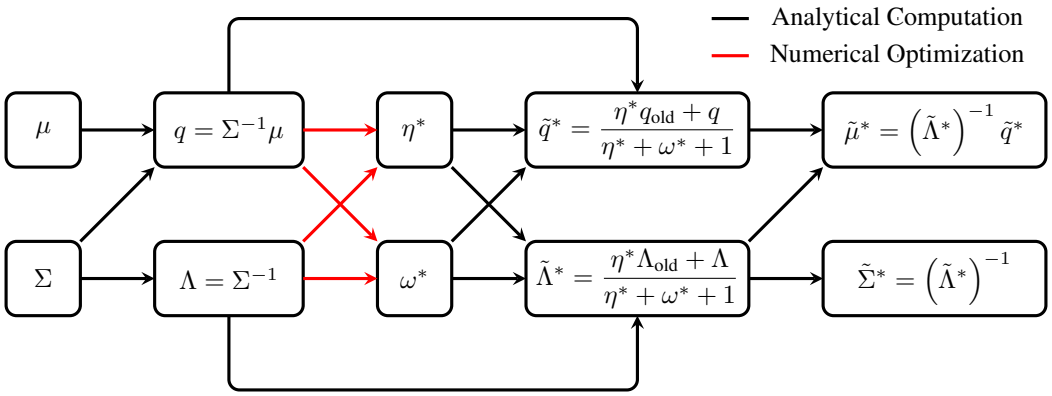

Figure 3: Compute graph of the KL projection layer. The layer first computes the natural parameters of π from the mean and covariance. Then it numerically optimizes the dual to obtain the optimal Lagrangian multipliers which are used to get the optimal natural parameters. Ultimately, the optimal mean and covariance are computed from the optimal natural parameters. We omit the dependency on constants, i.e., the bound ϵ and β as well as the parameters of π_{old} for clarity of the visualization.

Forward Pass For the forward pass we compute the natural parameters of π, solve the optimization problem and compute mean and covariance of $\tilde{\pi}^*$ from the optimal natural parameters. The corresponding compute graph is given in Figure 3.

Backward Pass Given the computational graph in Figure 3 gradients can be propagated back though the layer using standard back-propagation. All gradients for the analytical computations (black arrows in Figure 3) are straight forward and can be found in (Petersen & Pedersen, 2012). For the gradients of the numerical optimization of the dual (red arrows in Figure 3) we follow Amos & Kolter (2017) and differentiate the KKT conditions around the optimal Lagrangian multipliers computed during the forward pass. The KKT Conditions of the dual are given by

$$\nabla g(\eta^*, \omega^*) + m^T \nabla \begin{pmatrix} -\eta^* \\ -\omega^* \end{pmatrix} = \begin{pmatrix} \epsilon - \mathrm{KL}\left(\tilde{\pi}^* || \pi_{\theta_{\text{old}}}\right) - m_1 \\ \mathrm{H}(\tilde{\pi}^*) - \beta - m_2 \end{pmatrix} = 0, \quad \text{(Stationarity)}$$
$$m_1(-\eta^*) = 0 \quad \text{and} \quad m_2(-\omega^*) = 0 \quad \text{(Complementary Slackness)}$$

here $m = (m_1, m_2)^T$ denotes the Lagrangian multipliers for the box constraints of the dual (η and ω need to be non-negative). Taking the differentials of those conditions yields the equation system

$$\begin{pmatrix} -\frac{\partial \mathrm{KL}\left(\tilde{\pi}^* || \pi_{\theta_{\text{old}}}\right)}{\partial \eta^*} & -\frac{\partial \mathrm{KL}\left(\tilde{\pi}^* || \pi_{\theta_{\text{old}}}\right)}{\partial \omega^*} & -1 & 0 \\ \frac{\partial \mathrm{H}(\tilde{\pi}^*)}{\partial \eta^*} & \frac{\partial \mathrm{H}(\tilde{\pi}^*)}{\partial \omega^*} & 0 & -1 \\ -m_1 & 0 & -\eta^* & 0 \\ 0 & -m_2 & 0 & -\omega^* \end{pmatrix} \begin{pmatrix} d\eta \\ d\omega \\ dm_1 \\ dm_2 \end{pmatrix}$$
$$= \begin{pmatrix} \frac{\partial \mathrm{KL}\left(\tilde{\pi}^* || \pi_{\theta_{\text{old}}}\right)}{\partial q} dq + \frac{\partial \mathrm{KL}\left(\tilde{\pi}^* || \pi_{\theta_{\text{old}}}\right)}{\partial \Lambda} d\Lambda \\ -\frac{\partial \mathrm{H}(\tilde{\pi}^*)}{\partial q} dq - \frac{\partial \mathrm{H}(\tilde{\pi}^*)}{\partial \Lambda} d\Lambda \\ 0 \\ 0 \end{pmatrix}$$

which is (analytically) solved to obtain the desired partial derivatives

$$\frac{\partial \eta}{\partial q}, \frac{\partial \eta}{\partial \Lambda}, \frac{\partial \omega}{\partial q} \quad \text{, and} \quad \frac{\partial \omega}{\partial \Lambda}.$$

Implementation We implemented the whole layer using C++, Armadillo, and OpenMP for parallelization. The implementation saves all necessary quantities for the backward pass and thus a numerical optimization is only necessary during the forward pass. Before we perform a numerical optimization we check whether it is actually necessary. If the target distribution $\tilde{\pi}$ is within the trust region we immediately can set $\tilde{\pi}^* = \pi_\theta$, i.e., the forward and backward pass become the identity mapping. This check yield significant speed-ups, especially in early iterations, if the target is still close to the old distribution. If the projection is necessary we use the L-BFGS to optimize the 2D convex dual, which is still fast. For example, for a 17-dimensional action space and a batch size of 512, such as in the Humanoid-v2 experiments, the layer takes roughly 170ms for the forward pass and 3.5ms for the backward pass if the all 512 Gaussians are actually projected[2]. If none of the Gaussians needs to be projected its less than 1ms for forward and backward pass.

Simplifications If only diagonal covariances are considered the implementation simplifies significantly, as computationally heavy operations (matrix inversions and cholesky decompositions) simplify to pointwise operations (divisions and square roots). If only the covariance part of the KL is projected, we set $\mu_{\text{old}} = \mu = \tilde{\mu}^*$ and $d\mu = 0$ which is again a simplification for both the derivations and implementation. If an entropy equality constraint, instead of an inequality constraint, it is sufficient to remove the $\omega > 0$ constraint in the dual optimization.

B Additional Results

Figure 4 shows the training curves for all Mujoco environments with a 95% confidence interval. Besides the projections we also show the performance for PAPI, PPO, and only the Entropy projection from Akrour et al. (2019) without the trust region.

[2]On a 8 Core Intel Core i7-9700K CPU @ 3.60GHz

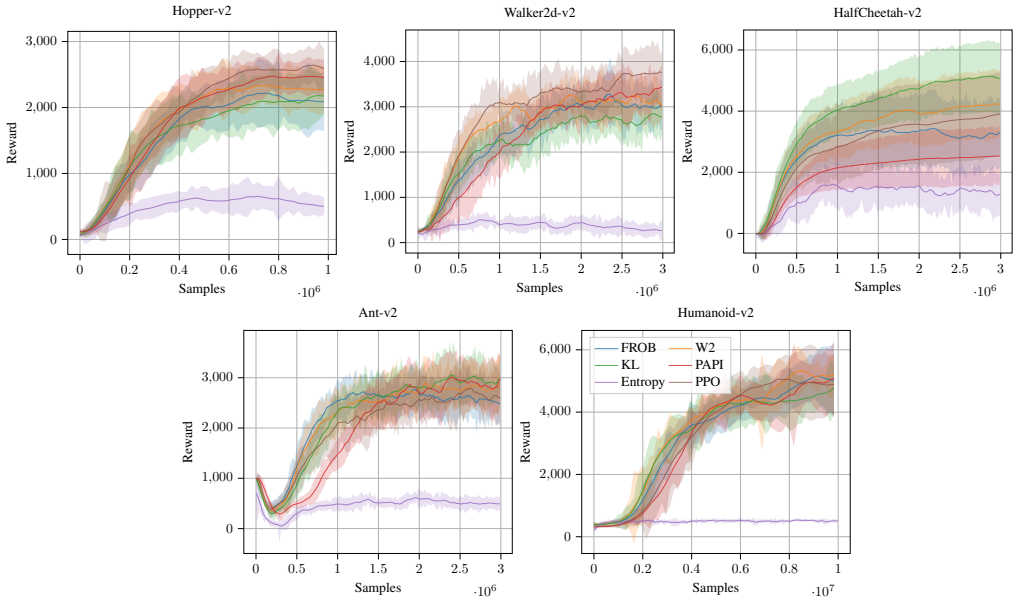

Figure 4: Training curves for the projection layer as well as PPO and PAPI on the test environment. We trained each agent on 10 seeds with 5 evaluation episodes for every data point. The plot shows the total mean reward with 95% confidence interval.

C HYPERPARAMETERS

Tables 2 and 3 show the hyperparameters used for the experiments in Table 1. Target entropy, temperature, and entropy equality are only required when the entropy projection is included in the layer, otherwise those values are ignored.

Table 2: Hyperparameters for all three projections as well as PAPI and PPO on the Mujoco benchmarks from Table 1

	Frobenius	Wasserstein	KL-divergence	PAPI	PPO
rollouts			2048		
GAE λ			0.95		
discount factor			0.99		
ϵ_μ/ϵ		0.085		0.015	/
ϵ_Σ		0.0075		/	/
target entropy		0		/	/
temperature		0.5		/	
entropy equality		True		False	/
optimizer			adam		
epochs vf			10		
epochs		20		10	
lr		5e-5		3e-4	
lr vf		7.5e-4		2.5e-4	
minibatch size		16		64	
entropy loss penalty			0		
max entropy weight			0		
regression weight	75.0	1.0	5.0	/	/
normalized observations			True		
normalized rewards		False		True	
observation clip		/		10	
reward clip		/		10	
importance ratio clip		/		0.2	
vf clip		/		0.2	
contextual std			False		
hidden layers			[64, 64]		
hidden layers vf			[64, 64]		
hidden activation			tanh		

Table 3: Hyperparameters for all three projection as well as PAPI and PPO on the Humanoid-v2 from Table 1.

	Wasserstein	Frobenius	KL-divergence	PAPI	PPO
rollouts			16384		
GAE λ			0.95		
discount factor			0.99		
ϵ_μ/ϵ		0.05		0.015	/
ϵ_Σ		0.0005		/	/
target entropy		0		/	/
temperature		0.2		/	
entropy equality		True		False	/
optimizer			adam		
epochs vf			10		
epochs			10		
lr		5e-5		1e-4	
lr vf		7.5e-4		1e-4	
minibatch size		128		512	
entropy loss penalty			0		
max entropy weight			0		
regression weight	5.0	1.0	5.0	/	/
normalized observations			True		
normalized rewards		False		True	
observation clip		/		10	
reward clip		/		10	
importance ratio clip		/		0.2	
vf clip		/		0.2	
contextual std			False		
hidden layers			[64, 64]		
hidden layers vf			[64, 64]		
hidden activation			tanh		