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 seemingly unrelated implementation choices. 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. We 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 for Gaussian policies, realized as differentiable neural network layers. These layers constrain the change in successive policies by projecting the updated policy onto trust regions. First, this approach is more stable with respect to what Engstrom et al. (2020) refer to as *code-level optimizations* than other approaches. Second, it comes with the benefit of imposing constraints for individual states, allowing for the possibility of state-dependent trust regions. This allows us to constrain the state-wise maximum change of successive policies. In this we differ from previous works, that constrain only the expected change and thus cannot rely on exact guarantees of monotonic improvement. 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. The last layer of the projected policy is now the the trust region layer which relies on the old policy as input. This would result in a ever-growing stack of policies which is clearly infeasible. To circumvent this issue we add an additional loss to the reinforcement learning objective which ensures the input and output of the projection stay close together. 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 implementation choices are essential for achieving state-of-the-art results with PPO. *Code-level optimizations*, such as reward scaling as well as observation, reward, gradient and value function clipping, can even compensate for removing core parts of the algorithm, e. g, 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, our projections enforce the constraints per state. This allows for better control of the change between subsequent policies and for state-dependent trust regions.

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 $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{P}_0, \gamma)$. We assume the state space $\mathcal{S}$ and action space $\mathcal{A}$ are continuous and the transition probabilities $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describe the probability transitioning to state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$. We denote the initial state distributions as $\mathcal{P}_0 : \mathcal{S} \to [0, 1]$. 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 $R^\gamma = \mathbb{E}_{\mathcal{T},\mathcal{P}_0,\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)\right]$. 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

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

where $A^\pi(s, a) = \mathbb{E}\left[R^\gamma | s_0 = s, a_0 = a; \pi\right] - \mathbb{E}\left[R^\gamma | s_0 = s; \pi\right]$ describes the advantage function, and the expectation is w.r.t $\pi_{\theta_{\text{old}}}$, i.e. $s' \sim \mathcal{T}(\cdot|s, a), a \sim \pi_{\theta_{\text{old}}}(\cdot|s), s_0 \sim \mathcal{P}_0(s_0), s \sim \rho_{\pi_{\theta_{\text{old}}}}$ where $\rho_{\pi_{\theta_{\text{old}}}}$ is a stationary distribution of policy $\pi_{\theta_{\text{old}}}$. 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 policies $\pi_{\theta_{\text{old}}}(a_t|s_t) = \mathcal{N}(a_t|\mu_{\text{old}}(s_t), \Sigma_{\text{old}}(s_t))$ and $\pi_\theta(a_t|s_t) = \mathcal{N}(a_t|\mu(s_t), \Sigma(s_t))$ represent the old as well as the current policy, respectively. We explore different trust regions on top of Equation 1 that employ different similarity measures between old and new distributions, i. e. the frequently 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

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

where dim is the dimensionality of μ_1, μ_2. The KL uses the Mahalanobis distance to measure the similarity between the two mean vectors. The difference of the covariances is measured by the difference in shape, i.e., the difference in scale, given by the log ratio of the determinants, plus the difference in rotation, given by the trace term. Given the KL is non-symmetric, it is clearly not a distance, yet still a frequently used divergence between distributions. We will use the more common reverse KL for our trust region, where the first argument is the new policy and the second is 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

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

A key difference to the KL divergence is that the 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

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

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 measures, 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 a distance or divergence $d(\pi(\cdot|s), \pi_{\text{old}}(\cdot|s))$ between probability distributions. Complementing Equation 1 with the trust region constraint leads to

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

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. 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 new policy is then given by the output of the trust region layer. Since the old policy distribution is fixed, 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 μ and Σ while satisfying the trust region constraints. The projection is based on the same distance or divergence which was used to define the respective trust region. Formally, this corresponds to the following optimization problems for each s

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

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

where $\tilde{\mu}_s$ and $\tilde{\Sigma}_s$ are the optimization variables for state s. Here, d_{mean} is the mean dependent part and d_{cov} is the covariance dependent part of the employed distance or divergence. For brevity of notation we will neglect all dependencies on the state in the following. We denote the projected policy as $\tilde{\pi}(a|s) = \mathcal{N}(a|\tilde{\mu}, \tilde{\Sigma})$.

4.1 PROJECTION OF THE MEAN

For all three trust region objectives we 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

$$\arg\min_{\tilde{\mu}} \left(\mu - \tilde{\mu}\right)^{\mathrm{T}} \Sigma_{\text{old}}^{-1} \left(\mu - \tilde{\mu}\right) \quad \text{s.t.} \quad \left(\mu_{\text{old}} - \tilde{\mu}\right)^{\mathrm{T}} \Sigma_{\text{old}}^{-1} \left(\mu_{\text{old}} - \tilde{\mu}\right) \leq \epsilon_{\mu}. \tag{5}$$

By making use of the method of Lagrangian multipliers (see Appendix B.2), we can formulate the dual and solve it for the projected mean $\tilde{\mu}$ as

$$\tilde{\mu} = \frac{\mu + \omega \mu_{\text{old}}}{1 + \omega} \quad \text{with} \quad \omega = \sqrt{\frac{\left(\mu_{\text{old}} - \mu\right)^{\mathrm{T}} \Sigma_{\text{old}}^{-1} \left(\mu_{\text{old}} - \mu\right)}{\epsilon_{\mu}}} - 1. \tag{6}$$

This equation can directly be used as mean for the Gaussian policy, while it easily allows to compute gradients. Note, that for the mean part of the KL we would need to use the Σ^{-1} instead of Σ_{old}^{-1} in the objective of Equation 5. Yet, this objective still results in a valid trust region problem which is much easier to optimize.

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 again use the method of Lagrangian multipliers (see Appendix B.3) and obtain the covariance $\tilde{\Sigma}$ as

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

where η is the corresponding Lagrangian multiplier.

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

$$\begin{aligned} &\arg\min_{\tilde{\Sigma}} \operatorname{tr}\left(\Sigma_{\text{old}}^{-1}\Sigma + \Sigma_{\text{old}}^{-1}\tilde{\Sigma} - 2\Sigma_{\text{old}}^{-1}\left(\Sigma^{1/2}\tilde{\Sigma}\Sigma^{1/2}\right)^{1/2}\right), \\ &\text{s.t.} \quad \operatorname{tr}\left(\mathbb{I} + \Sigma_{\text{old}}^{-1}\tilde{\Sigma} - 2\Sigma_{\text{old}}^{-1}\left(\Sigma_{\text{old}}^{1/2}\tilde{\Sigma}\Sigma_{\text{old}}^{1/2}\right)^{1/2}\right) \leq \epsilon_{\Sigma}, \end{aligned} \tag{8}$$

where $\mathbb{I}$ is the identity matrix. A closed form solution to this optimization problem can be found by using the methods outlined in Takatsu (2011). However, we found the resulting solution for the projected covariance matrices to be numerically unstable. Therefore, we made the simplifying assumption that both the current Σ and the old covariance Σ_{old} commute with $\tilde{\Sigma}$. Further, under the common assumption of diagonal covariances, 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 8. Again, we introduce Lagrange multipliers and solve the dual problem to obtain the optimal primal and dual variables (see Appendix B.4). 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

$$\tilde{\Sigma}^{1/2} = \frac{\Sigma^{1/2} + \eta \Sigma_{\text{old}}^{1/2}}{1 + \eta} \quad \text{with} \quad \eta = \sqrt{\frac{\operatorname{tr}\left(\mathbb{I} + \Sigma_{\text{old}}^{-1}\Sigma - 2\Sigma_{\text{old}}^{-1/2}\Sigma^{1/2}\right)}{\epsilon_{\Sigma}}} - 1, \tag{9}$$

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.

[1]Here, we assume the true matrix square root $\Sigma = \Sigma^{1/2}\Sigma^{1/2}$ and not a Cholesky factor $\Sigma = LL^{\mathrm{T}}$ as it naturally appears in the expressions for the projected covariance.

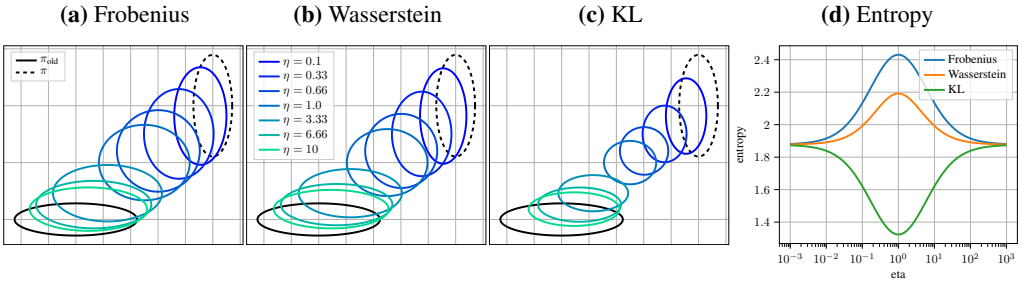

Figure 1: **(a)**, **(b)**, and **(c)**: Interpolated covariances for the different projections for various values of η. For Frobenius and Wasserstein the intermediate distributions clearly have a larger entropy, while for the KL projection the intermediate entropy is smaller. **(d)**: Entropy of the interpolated distributions. 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 transitioning between the distributions, while it decreases for the KL. A more general statement regarding this can be found in Theorem 1.

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

$$\arg\min_{\tilde{\Sigma}} \operatorname{tr}\left(\Sigma^{-1}\tilde{\Sigma}\right) + \log\frac{|\Sigma|}{|\tilde{\Sigma}|}, \quad \text{s.t.} \quad \operatorname{tr}\left(\Sigma_{\text{old}}^{-1}\tilde{\Sigma}\right) - \dim(a) + \log\frac{|\Sigma_{\text{old}}|}{|\tilde{\Sigma}|} \leq \epsilon_{\Sigma}, \tag{10}$$

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

$$\tilde{\Lambda} = \frac{\eta^*\Lambda_{\text{old}} + \Lambda}{\eta^* + 1}, \quad \eta^* = \arg\min_{\eta} g(\eta), \text{ s.t. } \eta \geq 0, \tag{11}$$

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 B.5 for more details and derivations.

Entropy Control. Previous works Akrour et al. (2019) and Abdolmaleki et al. (2015) have shown the benefits of introducing an entropy constraint $H(\tilde{\pi}) = \beta$ in addition to the trust region constraints. Such a constraint allows for more control over the exploration behavior of the policy. In order to endow our algorithm with this improved exploration behavior, 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 (Equation 7), for the W2 projection between their respective matrix square roots (Equation 9), and for the KL projection between their inverses (Equation 11). 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 prove 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 7) and the Wasserstein projection (Equation 9), 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 11).

The proof is based on the multiplicative version of the Brunn-Minkowski inequality and can be found in Appendix B.1. Intuitively, this implies that the Frobenius and Wasserstein projections

Table 1: Mean return and 95% confidence interval of 20 epochs after completing 20% of the total training and for the last 20 epochs. We trained each experiment with 40 different seeds and computed five evaluation rollouts per epoch. For each environment we highlight the best method. However, the projections with (-E) and without entropy control are considered separately, therefore, each column may have up to two best runs.

	Hopper-v2		Walker2d-v2		Halfcheetah-v2		Ant-v2		Humanoid-v2	
	20%	final	20%	final	20%	final	20%	final	20%	final
FROB	1646 ± 19	$\mathbf{2578 \pm 17}$	2142 ± 23	3443 ± 19	2525 ± 11	3552 ± 14	1265 ± 11	3035 ± 26	2176 ± 65	5202 ± 23
W2	1586 ± 31	2490 ± 19	2284 ± 20	3390 ± 17	$\mathbf{2586 \pm 9}$	3692 ± 15	1362 ± 12	3086 ± 28	2502 ± 87	5057 ± 25
KL	1584 ± 20	2476 ± 10	2071 ± 36	$\mathbf{3583 \pm 14}$	2369 ± 9	$\mathbf{4255 \pm 17}$	1460 ± 26	$\mathbf{3335 \pm 21}$	$\mathbf{2923 \pm 53}$	$\mathbf{5510 \pm 27}$
PAPI	1378 ± 20	$\mathbf{2549 \pm 12}$	1663 ± 21	3232 ± 20	1875 ± 5	2380 ± 6	645 ± 5	3198 ± 17	1824 ± 74	5367 ± 22
PPO-M	1030 ± 23	2321 ± 19	1994 ± 18	2771 ± 40	1922 ± 15	3272 ± 18	1494 ± 9	2783 ± 32	604 ± 10	5172 ± 23
PPO	$\mathbf{1881 \pm 30}$	2515 ± 15	$\mathbf{2490 \pm 33}$	$\mathbf{3447 \pm 17}$	2048 ± 8	2880 ± 8	$\mathbf{1657 \pm 10}$	2852 ± 25	1723 ± 67	4969 ± 18
FROB-E	1587 ± 30	2478 ± 11	2037 ± 21	3370 ± 27	$\mathbf{2762 \pm 10}$	4568 ± 14	730 ± 9	$\mathbf{3475 \pm 26}$	3662 ± 44	5807 ± 18
W2-E	1518 ± 36	2437 ± 13	2174 ± 19	3303 ± 25	2266 ± 8	4213 ± 13	855 ± 27	3361 ± 30	3658 ± 56	$\mathbf{5844 \pm 8}$
KL-E	1502 ± 18	2497 ± 16	2215 ± 31	3171 ± 28	2611 ± 12	$\mathbf{4584 \pm 18}$	955 ± 16	3437 ± 21	$\mathbf{3801 \pm 42}$	5430 ± 16

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. Note, however, that at each epoch i the policy π_i predicted by the network before the projection layer does not respect the constraints and thus has to rely on calling this layer. Yet, the policy of the projection layer $\tilde{\pi}_i$ depends on the parameters of π_i and the old policy network $\pi_{i,\text{old}} = \tilde{\pi}_{i-1}$. This would result in an ever-growing stack of policy networks which would become increasingly costly to evaluate. In other words, $\tilde{\pi}_i$ is computed using all stored networks of $\pi_i, \pi_{i-1}, \ldots, \pi_0$. We now discuss the parametrization of $\tilde{\pi}$ via amortized optimization.

We need to encode the information of the projection layer into the parameters θ of the next policy, i.e. $\tilde{\pi}(a|s;\theta) = \mathrm{p} \circ \pi_\theta(a|s)$ is a composition function in which p denotes the projection layer. Basically, the output of π_θ is (μ, Σ), and p computes $(\tilde{\mu}, \tilde{\Sigma})$ according Equations 6, 7, 9, or 11. Formally, we aim to find a set of parameters $\theta^* = \arg\min_\theta \mathbb{E}_{\boldsymbol{s}\sim\rho_{\pi_\text{old}}} \left[d\left(\tilde{\pi}(\cdot|\boldsymbol{s}), \pi_\theta(\cdot|\boldsymbol{s})\right)\right]$, where ρ_{π_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.,

$$\arg\min_\theta \mathbb{E}_{(s,a)\sim\pi_{\theta_\text{old}}} \left[\frac{\tilde{\pi}(a|s;\theta)}{\pi_{\theta_\text{old}}(a|s)} A^{\pi_\text{old}}(s,a) \right] - \alpha \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_\text{old}}} \left[d\left(\tilde{\pi}(\cdot|\boldsymbol{s};\theta), \pi_\theta(\cdot|\boldsymbol{s})\right)\right]. \tag{12}$$

Note that the importance sampling ratio is computed based on Gaussian distribution generated by the trust region layer and not directly from the network output. Furthermore, the gradient for the regression penalty does not flow through the projection, it is solely acting as supervised learning signal. As appropriate similarity measures d for the penalty, we resort to the measures used in each projection. For a detailed algorithmic view see Appendix A.

5 Experiments

Mujoco Benchmarks 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 that it can can achieve similar performance to PPO. For our experiments, the PAPI projection and its conservative PPO version are executed in the setting sent to us by the author. The hyperparameters for all three projections and PPO have been selected with Optuna (Akiba et al., 2019). See Appendix D for a full listing of all hyperparameters. We use a common set of hyperparameters for all environments but the Humanoid, for which we optimized separately. Next to the standard PPO implementation with all code-level optimizations we further evaluate PPO-M, which only leverages the core PPO algorithm. Our projections and PPO-M solely use the observation normalization, network architecture, and initialization from the original PPO implementation. All

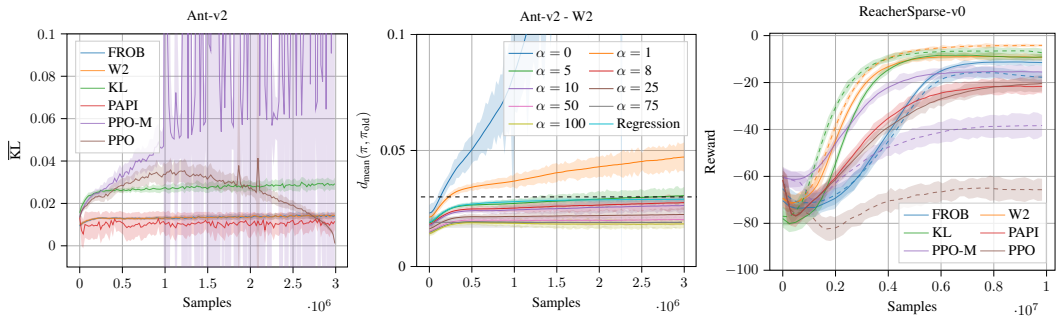

Figure 2: Left: Mean KL divergence for Ant-v2 as a unifying measure to make the policy changes comparable between all methods. Center: Mahalanobis distance between the mean values of the unprojected and old policy when using different weights α for the regression penalty in comparison to a full regression. The mean bound for the W2 projection is set to 0.03 (dotted black line). Right: Mean cumulative reward and a 95% confidence interval based on 40 seeds for a 5-link Reacher task with semi-sparse rewards. For each method, besides PAPI, we train policies with (dashed) and without (solid) contextual covariance.

algorithms parametrize the covariance as a non-contextual diagonal matrix. We refer to the Frobenius projection as *FROB*, the Wasserstein projection as *W2*, and the KL-projection as *KL*.

Table 1 gives an overview of the final performance and convergence speed on the Mujoco benchmarks, Figure 4 in the appendix displays the full learning curves. After every epoch, we evaluate five episodes without applying exploration noise to obtain the return values. However, note that we did not include the entropy projection for PAPI to provide a fair comparison to PPO. The results show that our trust region layers are able to perform similarly or better than PPO and PAPI across all tasks. While the performance on the Hopper-v2 is comparable, the projections significantly outperform all baselines on the HalfCheetah-v2. The KL projection even demonstrates the best performance on the remaining three environments. Besides that, the experiments present a relatively balanced performance between projections, PPO, and PAPI. The differences are more apparent when comparing the projections to PPO-M, which uses the same implementation details as our projections. The asymptotic performance of PPO-M is on par for the Humanoid-v2, but it convergences much slower and is, moreover, noticeably weaker on the remaining tasks. Consequently, the approximate trust region of PPO alone is not sufficient for good performance, only paired with certain implementation choices. Still, the original PPO cannot fully replace a mathematically sound trust region as ours, although it does not exhibit a strong performance difference. For this, Figure 2 visualizes the mean KL divergence at the end of each epoch for all methods. Despite the fact that neither W2 nor Frobenius projection use the KL, we leverage it here as a unifying measure to compare the change in the policy distribution. All projections are characterized by an almost constant change between each epoch, whereas for PPO-M the changes are highly inconsistent. The code-level optimizations of PPO can mitigate this to some extend but cannot properly enforce the desired constant change in the policy distribution. We found especially the learning rate decay as major factor that contributes well to the rather good behavior of PPO. Albeit, PAPI provides a similar principled trust region projection as we do, it still has some inconsistency by approaching the bound iteratively.

Entropy Control To demonstrate the effect of combining our projections with entropy control, as described in Section 4.2, we evaluate all Mujoco tasks again for this extended setting. The target entropy in each iteration i is computed by exponentially decaying the initial entropy $\mathcal{H}_0$ to κ with temperature τ as $\kappa + (\mathcal{H}_0 - \kappa)\tau^{\frac{10i}{N}}$, where N is the total number of training steps. The bottom of Table 1 shows the results for our projections with entropy control. Especially on the more complex tasks, all three projections significantly benefit from the entropy control. Their asymptotic performance for the HalfCheetah-v2, Ant-v2, and Humanoid-v2 increases and yields a much faster convergence in the latter. For the other Mujoco tasks the performance remains largely constant as the complexity of these tasks is insufficient to benefit from the explicit entropy control, as also noted by Pajarinen et al. (2019); Abdolmaleki et al. (2015). Entropy control is particularly helpful in environments which require more exploration.

Contextual Covariances. To emphasize the advantage of *state-wise* trust regions we consider the case of policies with state-dependent covariances. Existing methods, such as PPO and TRPO, are rarely used in this setting. In addition, PAPI cannot project the covariance in the contextual case.

Further, Andrychowicz et al. (2020) demonstrated that for the standard Mujoco benchmarks, contextual policies are not beneficial in an on-policy setting. Therefore, we evaluate on a task motivated from optimal control which benefits from a contextual covariance. We extend the Mujoco *Reacher-v2* to a 5-link planar robot, the distance penalty to the goal is now only provided in the last time step, $t = 200$, and the observation space now contains the current time step t. This semi-sparse reward specification imposes a significantly harder exploration problem as the agent is only provided with feedback at the last time step. We again tuned hyperparameters using Optuna Akiba et al. (2019) and did not use entropy control. The results are presented in Figure 2 (right), comparing the approaches with and without contextual covariances, except for PAPI which is not applicable in the contextual case. All three projections significantly outperform the baseline methods with the non-contextual covariance. Additionally, both the W2 and KL projection improve their results in the contextual case. In contrast, all baseline methods decrease in performance and are not able to leverage the advantage of contextual information. The poor performance for the baselines mainly originates from incorrect exploitation. PPO reduces the covariance too quickly, whereas PAPI reduces it too slowly, which leads for both to a suboptimal performance. However, the Frobenius projection also does not benefit from the contextual policy because of numerical instabilities, arising from small covariance values towards convergence. Those issues can be mitigated using a smaller covariance bound, but they cannot be fully avoided. The KL projection, while yielding the best results throughout all experiments, is the computationally most expensive. Hence, if contextual covariances are not required, the KL is the best choice for most problems. As a bonus, it has all properties of existing KL-based trust region methods that have monotonic improvement guarantees. Nevertheless, for quick benchmarks or contextual covariances, the W2 is preferred, given it is slightly less prone to hyperparameter choices and does not add any computational overhead as the KL does.

Trust Region Regression Loss. Lastly, we investigate the main approximation of our approach, the trust region regression loss (Equation 12). In the following ablation, we evaluate how different choices of the regression weight α affect the constraint satisfaction. Figure 2 (center) shows the Mahalanobis distance between the current policy mean at the end of each epoch without the projection and the old policy mean for different α values. In addition, for one run we remove the regression penalty from the loss but instead run several iterations of regressing the current policy towards the projection after each policy update. One key observation is that decreasing the penalty up to a certain threshold 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. As the penalty is only added to the loss, when the bound is violated, large changes in the policy are punished while making small steps might slightly lower the achieved reward, but does not incur any other negative effect. By selecting a larger α, this behavior is amplified. Furthermore, we can see that small values of α yield a behavior which is similar to the full regression setting. Consequently, it is justified to use a computationally simpler penalty, 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. Unlike previous approaches that only constrain the expected change between successive policies and for which monotonic improvement guarantees thus only hold approximately, we can constrain the maximum change. 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 leverage the Wasserstein distance and 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 continue our work with complex environments, in particular with contextual covariances. 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 ALGORITHM

Algorithm 1 Differentiable Trust Region Layer. The trust region layer acts as final layer after predicting a Gaussian distribution. It projects this predicted Gaussian onto the trust region in case it is violating the specified bounds. As output it generates a projected mean and covariance that satisfy the respective trust region bound. The entropy control in the last step can be disabled.

Initialize bounds $\epsilon_\mu, \epsilon_\Sigma$, temperature τ as well as target κ and initial entropy $\mathcal{H}_0$.

procedure TRUSTREGIONLAYER($\mu, \Sigma, \mu_{\text{old}}, \Sigma_{\text{old}}$)
  if $d_{\text{mean}}(\mu, \mu_{\text{old}}) > \epsilon_\mu$ **then**
    Compute $\tilde{\mu}$ with Equation 6
  else
    $\tilde{\mu} = \mu$
  if $d_{\text{cov}}(\Sigma, \Sigma_{\text{old}}) > \epsilon_\Sigma$ **then**
    Compute $\tilde{\Sigma}$ with Equations 7, 9, or 11
  else
    $\tilde{\Sigma} = \Sigma$
  $\beta = \kappa + (\mathcal{H}_0 - \kappa)\tau^{\frac{10i}{N}}$ ▷ (Optional) entropy control as described in Section 4.2
  if $\mathcal{H}(\Sigma) < \beta$ **then**
    $c = \exp\left\{\left(\beta - \mathcal{H}(\Sigma)\right)/\dim(a)\right\}$
    $\tilde{\Sigma} = c\tilde{\Sigma}$
  return $\tilde{\mu}, \tilde{\Sigma}$

Algorithm 2 Algorithmic view of the proposed Trust Region Projections. The trust region projections itself do not require approximations, the old policy update in the last step is the only point where we introduce an approximation. This update would normally require additional supervised regression steps that minimize the distance between the network output and the projection. However, by leveraging the regression penalty during policy optimization this optimization step can be omitted. Both approaches yield a policy, which is independent of the old policy distribution, i. e. it can act without the projection while maintaining the trust region. However, the penalty does not require additional computation and the policy can directly generate new trajectories, equivalently to other trust region methods, such as PPO.

Initialize policy $\theta_{0,0}$ and value function ϕ_0
for $i = 0, 1, \ldots, N$ **do** ▷ epoch
  Collect set of trajectories $\mathcal{D}_i = \{\tau_k\}$ with $\pi(\theta_{i,0})$
  Compute advantage estimates $\hat{A}_t$ with GAE
  for $j = 0, 1, \ldots, M$ **do**
    Use $\pi(\theta_{i,j})$ to predict Gaussian action distributions $\mathcal{N}(\mu_{i,j}, \Sigma_{i,j})$ for $\mathcal{D}_i$
    $\tilde{\pi} = \text{TRUSTREGIONLAYER}(\mu_{i,j}, \Sigma_{i,j}, \mu_{i,0}, \Sigma_{i,0})$
    Update policy with Adam using the following policy gradient:

$$\theta_{i,j+1} \leftarrow \text{Adam}\left(\nabla_\theta \left[\mathbb{E}_{\pi(\theta_{i,0})}\left[\frac{\tilde{\pi}(a|s;\theta)}{\pi(a|s;\theta_{i,0})}\hat{A}_t\right] - \alpha\mathbb{E}_{\boldsymbol{s}\sim p_{\pi(\theta_{i,0})}}\left[d\left(\tilde{\pi}(\cdot|\boldsymbol{s};\theta), \pi(\cdot|\boldsymbol{s};\theta)\right)\right]\right]\Big|_{\theta=\theta_{i,j}}\right)$$

  Successive policy update: $\theta_{i+1,0} \leftarrow \theta_{i,M}$

B DERIVATIONS

B.1 PROOF OF THEOREM 1

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

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

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 dimensionality 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

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

B.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

$$\mathcal{L}(\tilde{\mu}, \omega) = (\mu - \tilde{\mu})^{\mathrm{T}} \Sigma_{\text{old}}^{-1} (\mu - \tilde{\mu}) + \omega \left((\mu_{\text{old}} - \tilde{\mu})^{\mathrm{T}} \Sigma_{\text{old}}^{-1} (\mu_{\text{old}} - \tilde{\mu}) - \epsilon_\mu \right). \tag{13}$$

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 $\tilde{\mu}$ gives

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

Inserting the optimal mean $\tilde{\mu}^*$ in Equation 13 results in

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

Thus, differentiating w.r.t ω yields

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

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

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

B.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

$$\mathcal{L}(\tilde{\Sigma}, \eta) = \left\|\tilde{\Sigma} - \Sigma\right\|_F^2 + \eta \left(\left\|\tilde{\Sigma} - \Sigma_{\text{old}}\right\|_F^2 - \epsilon_\Sigma\right). \tag{14}$$

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 14 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

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

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

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

B.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

$$\begin{aligned} \mathcal{L}(\tilde{\Sigma}, \eta) = & \operatorname{tr}\left(\tilde{\Sigma} + \Sigma - 2\left(\Sigma^{1/2}\tilde{\Sigma}\Sigma^{1/2}\right)^{1/2}\right) \\ & + \eta\left(\operatorname{tr}\left(\tilde{\Sigma} + \Sigma_{\text{old}} - 2\left(\Sigma_{\text{old}}^{1/2}\tilde{\Sigma}\Sigma_{\text{old}}^{1/2}\right)^{1/2}\right) - \epsilon_\Sigma\right). \end{aligned} \tag{15}$$

Assuming that $\tilde{\Sigma}$ commutes with Σ as well as Σ_{old}, Equation 15 simplifies to

$$\begin{aligned} \mathcal{L}(\tilde{\Sigma}, \eta) &= \operatorname{tr}\left(\tilde{\Sigma} + \Sigma - 2\tilde{\Sigma}^{1/2}\Sigma^{1/2}\right) + \eta\left(\operatorname{tr}\left(\tilde{\Sigma} + \Sigma_{\text{old}} - 2\tilde{\Sigma}^{1/2}\Sigma_{\text{old}}^{1/2}\right) - \epsilon\right) \\ &= \operatorname{tr}\left(S^2 + \Sigma - 2S\Sigma^{1/2}\right) + \eta\left(\operatorname{tr}\left(S^2 + \Sigma_{\text{old}} - 2S\Sigma_{\text{old}}^{1/2}\right) - \epsilon\right), \end{aligned} \tag{16}$$

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

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

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

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

B.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

$$\tilde{\pi}^* = \arg\max_{\tilde{\pi}} -\mathrm{KL}\left(\tilde{\pi}||\pi_\theta\right) \quad \text{s.t.} \quad \mathrm{KL}\left(\tilde{\pi}||\pi_{\theta_{\text{old}}}\right) \leq \epsilon, \quad \mathrm{H}\left(\tilde{\pi}\right) \geq \beta, \tag{17}$$

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 17

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

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

$$\tilde{\Lambda} = \frac{\eta\Lambda_{\text{old}} + \Lambda}{\eta + 1 + \omega} \quad \text{and} \quad \tilde{q} = \frac{\eta q_{\text{old}} + q}{\eta + 1 + \omega}. \tag{18}$$

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

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

Given the optimal Lagrangian multipliers, η^* and ω^* we obtain the parameters of the optimal distribution $\tilde{\pi}^*$ using Equation 18.

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

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

$$m_1(-\eta^*) = 0 \quad \text{and} \quad m_2(-\omega^*) = 0 \quad \text{(Complementary Slackness)}$$

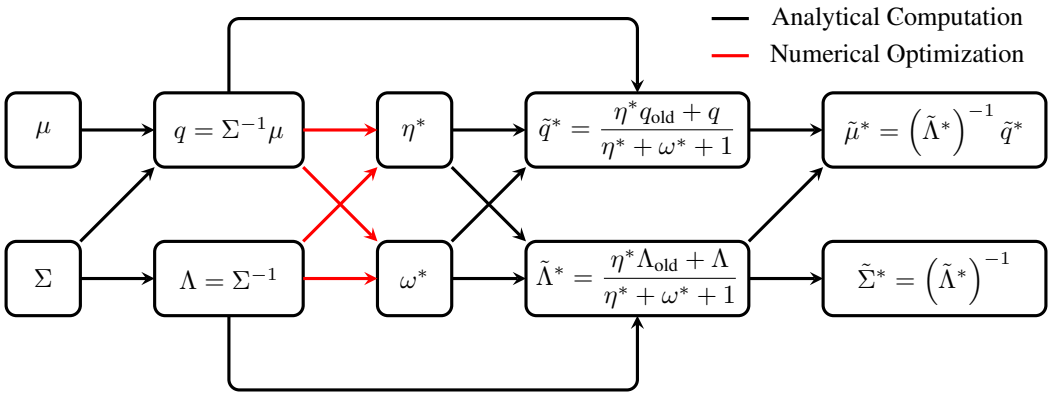

Figure 3: Compute graph of the KL projection layer. The layer first computes the natural parameters of π from the mean and covariance. Then it numerically optimizes the dual to obtain the optimal Lagrangian multipliers which are used to get the optimal natural parameters. Ultimately, the optimal mean 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.

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.

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

C 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 and PPO. In Figure 5 the projections also leverage the Entropy control based on the results from from Akrour et al. (2019).

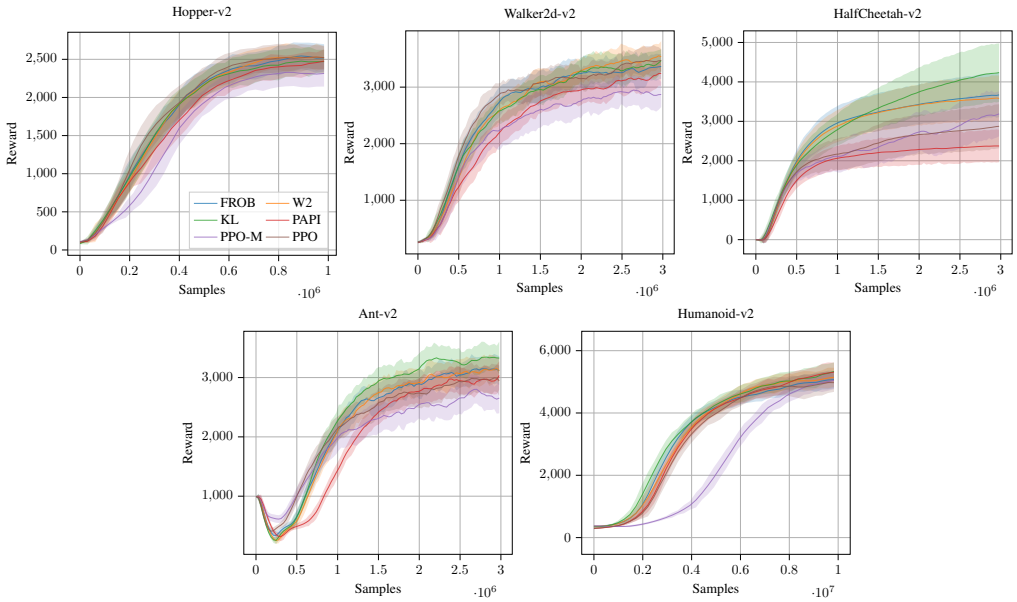

Figure 4: Training curves for the projection layer as well as PPO and PAPI on the test environment. We trained 40 agents with different seeds for each environment using five evaluation episodes for every data point. The plot shows the total mean reward with 95% confidence interval.

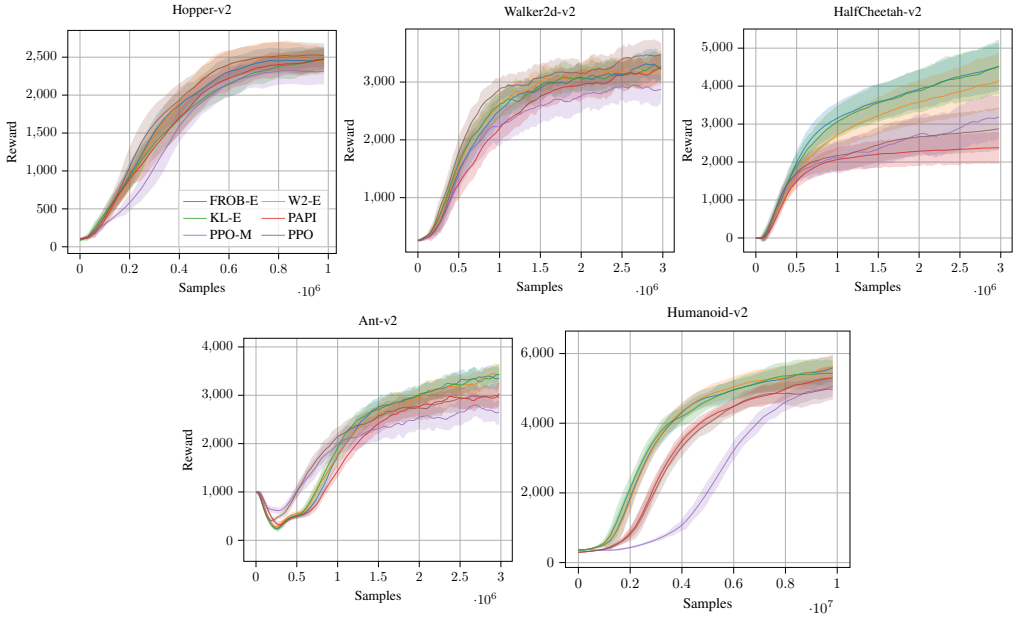

Figure 5: Training curves for the projection layer with entropy control (-E) as well as PPO and PAPI on the test environment. We trained 40 agents with different seeds for each environment using five evaluation episodes for every data point. The plot shows the total mean reward with 95% confidence interval.

D 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, PPO. and PPO-M on the Mujoco benchmarks from Table 1

	Frobenius	Wasserstein	KL-divergence	PAPI	PPO	PPO-M
rollouts			2048			
GAE λ			0.95			
discount factor			0.99			
ϵ_μ/ϵ		0.03		0.015		n.a.
ϵ_Σ		0.001			n.a.	
target entropy		0			n.a.	
temperature		0.5			n.a.	
entropy equality		False		False		n.a.
optimizer			adam			
epochs vf			10			
epochs		20			10	
lr		5e-5			3e-4	
lr vf		4.5e-4			2.5e-4	
minibatch size		32			64	
regression weight α		8.0			n.a.	
entropy loss penalty			0			
normalized observations			True			
normalized rewards		False		True		False
observation clip		n.a.		10		n.a.
reward clip		n.a.		10		n.a.
vf clip		n.a.		0.2		n.a.
importance ratio clip		n.a.			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.

	Frobenius	Wasserstein	KL-divergence	PAPI	PPO	PPO-M
rollouts			16384			
GAE λ			0.95			
discount factor			0.99			
ϵ_μ/ϵ		0.05		0.015	n.a.	
ϵ_Σ	1e-4	0.01	1e-4		n.a.	
target entropy		0			n.a.	
temperature		0.2			n.a.	
entropy equality		True		False	n.a.	
optimizer			adam			
epochs vf			10			
epochs			10			
lr		1e-4			1e-4	
lr vf		4.5e-4			1e-4	
minibatch size		256			512	
entropy loss penalty			0			
max entropy weight			0			
regression weight		8.0			n.a.	
normalized observations			True			
normalized rewards		False		True		False
observation clip		n.a.		10		n.a.
reward clip		n.a.		10		n.a.
vf clip		n.a.		0.2		n.a.
importance ratio clip		n.a.			0.2	
contextual std			False			
hidden layers			[64, 64]			
hidden layers vf			[64, 64]			
hidden activation			tanh			

Table 4: Hyperparameters for all three projection as well as PAPI and PPO on our ReacherSparse-v0 task from Figure 2.

	Frobenius	Wasserstein	KL-divergence	PAPI	PPO	PPO-M
rollouts			16384			
GAE λ			0.95			
discount factor			0.99			
ϵ_μ/ϵ		0.03		0.03	n.a.	
ϵ_Σ	5e-5	1e-3	5e-5/1e-3 (context)		n.a.	
target entropy		n.a.			n.a.	
temperature		n.a.			n.a.	
entropy equality		n.a.		False	n.a.	
optimizer			adam			
epochs vf			10			
epochs			20			
lr		3e-4			1e-4	
lr vf		4.5e-4			1e-4	
minibatch size		256			512	
entropy loss penalty			0			
max entropy weight			0			
regression weight		8.0			n.a.	
normalized observations			True			
normalized rewards		False		True		False
observation clip		n.a.		10		n.a.
reward clip		n.a.		10		n.a.
vf clip		n.a.		0.2		n.a.
importance ratio clip		n.a.			0.2	
contextual std			False			
hidden layers			[64, 64]			
hidden layers vf			[64, 64]			
hidden activation			tanh			