A KL-Divergence Trust Region Projection Layer

As already mentioned in the main text, TRPLs [9] present a scalable and mathematically sound approach for enforcing trust regions in step-based deep RL. The layer takes the output of a standard Gaussian policy as input in terms of mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ and projects it into the trust region if the given mean and variance violate their respective bounds. This projection is done for each input state individually. Subsequently, the projected Gaussian policy distribution with parameters $\tilde{\boldsymbol{\mu}}$, $\tilde{\boldsymbol{\Sigma}}$ is used for any further steps, e. g. for sampling and/or loss computation. Formally, the layer solves the following two optimization problems for each state $\boldsymbol{s}$

$$\arg\min_{\tilde{\boldsymbol{\mu}}_s} d_{\text{mean}}\left(\tilde{\boldsymbol{\mu}}_s, \boldsymbol{\mu}(s)\right), \quad \text{s.t.} \quad d_{\text{mean}}\left(\tilde{\boldsymbol{\mu}}_s, \boldsymbol{\mu}_{\text{old}}(s)\right) \leq \epsilon_{\boldsymbol{\mu}}, \quad \text{and} \tag{1}$$

$$\arg\min_{\tilde{\boldsymbol{\Sigma}}_s} d_{\text{cov}}\left(\tilde{\boldsymbol{\Sigma}}_s, \boldsymbol{\Sigma}(\boldsymbol{s})\right), \quad \text{s.t.} \quad d_{\text{cov}}\left(\tilde{\boldsymbol{\Sigma}}_s, \boldsymbol{\Sigma}_{\text{old}}(\boldsymbol{s})\right) \leq \epsilon_{\Sigma}, ` \tag{2}$$

where $\tilde{\boldsymbol{\mu}}_s$ and $\tilde{\boldsymbol{\Sigma}}_s$ are the optimization variables for input state $\boldsymbol{s}$ and ϵ_μ and ϵ_Σ are the trust region bounds for mean and covariance, respectively. Finally, μ_{old} and Σ_{old} are the reference mean and covariance for the trust region and d_{mean} as well as d_{cov} are the similarity metrics for the mean and covariance of a decomposable distance or divergence measure. As we only leverage the KL-divergence projection, we will provide only details for this particular projection below. For the other two projections we refer the reader to Otto et al. [9].

Inserting the mean part of the Gaussian KL divergence into Equation 1 yields

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

After differentiating the dual w.r.t. $\tilde{\mu}$, we can solve for the projected mean

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

leveraging the optimal Lagrange multiplier ω. Similarly, we can insert the covariance part of the Gaussian KL divergence into Equation 2, which results in

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

where d is the number of degrees of freedom (DoF). Once again, differentiating and solving the dual $g(\eta)$ for the projected covariance yields

$$\tilde{\Sigma} = \left(\frac{\eta^*\Sigma_{\text{old}}^{-1} + \Sigma^{-1}}{\eta^*+1}\right)^{-1} \quad \text{with} \quad \eta^* = \arg\min_\eta g(\eta), \ \text{s.t.}\ \eta \geq 0.$$

Here, the the optimal Lagrange multiplier η^* cannot be computed in closed form, however, a standard numerical optimizer, such as BFGS, is able to efficiently find it. This can be made differentiable by taking the differentials of the KKT conditions of the dual. For more details, we refer to the original work [9].

B Environment Details

B.1 Box Pushing

The goal of the box pushing task is to move a box to a specified goal location and orientation using the seven DoF Franka Emika Panda. Hence, the context space for this task is the goal position $x \in [0.3, 0.6]$, $y \in [-0.45, 0.45]$ and the goal orientation $\theta \in [0, 2\pi]$. In addition to the contexts, the observation space for the step-based algorithms contains information about joints and end-effector as well as the current box location and orientation. To the original torque from the policy we add

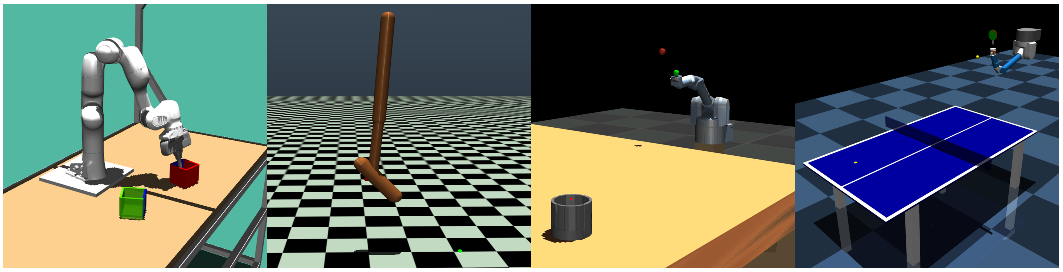

Figure 5: Visualization of the four control tasks box pushing, hopper jumping, beer pong, and table tennis.

gravity compensation in each time step. The task is considered successfully solved if the position distance ≤ 0.05m and the orientation error ≤ 0.5rad. For the total reward we consider different sub-rewards. First, the distance to the goal

$$R_{\text{goal}} = ||\mathbf{p} - \mathbf{p}_{goal}||,$$

where $\mathbf{p}$ is the box position and $\mathbf{p}_{goal}$ the goal position itself. Second, the rotation distance

$$R_{\text{rotation}} = \frac{1}{\pi} \arccos |\mathbf{r} \cdot \mathbf{r}_{goal}|,$$

where $\mathbf{r}$ and $\mathbf{r}_{goal}$ are the box orientation and goal orientation in quaternion, respectively. Third, an incentive to keep the rod within the box

$$R_{\text{rod}} = \text{clip}(||\mathbf{p} - \mathbf{h}_{pos}||, 0.05, 10)$$

where $\mathbf{h}_{pos}$ is the position of the rod tip. Fourth, a similar incentive that encourages to maintain the rod in a desired rotation

$$R_{\text{rod\_rotation}} = \text{clip}(\frac{2}{\pi} \arccos |\mathbf{h}_{rot} \cdot \mathbf{h}_0|, 0.25, 2),$$

where $\mathbf{h}_{rot}$ and $\mathbf{h}_0 = (0.0, 1.0, 0.0, 0.0)$ are the current and desired rod orientation in quaternion, respectively. And lastly, we utilize the following error

$$\text{err}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_{i \in \{i | |q_i| > |q_i^b|\}} (|q_i| - |q_i^b|) + \sum_{j \in \{j | |\dot{q}_j| > |\dot{q}_j^b|\}} (|\dot{q}_j| - |\dot{q}_j^b|).$$

Here, $\mathbf{q}$, $\dot{\mathbf{q}}$, $\mathbf{q}^b$, and $\dot{\mathbf{q}}^b$ are the robot joint's position and velocity as well as their respective bounds. Additionally, we consider an action cost in each time step t

$$\tau_t = 5 \cdot 10^{-4} \sum_i^K (a_t^i)^2,$$

where $K = 7$ is the number of DoF. In total we consider three different rewards.

Dense Reward. The dense reward provides information about the goal and rotation distance in each time step t on top of the utility rewards

$$R_{\text{tot}} = -R_{\text{rod}} - R_{\text{rod\_rotation}} - \tau_t - \text{err}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - 3.5 R_{\text{goal}} - 2 R_{\text{rotation}}.$$

Time-Dependent Sparse Reward. The time-dependent sparse reward is similar to the dense reward, but only returns the goal and rotation distance in the last time step T

$$R_{\text{tot}} = \begin{cases} -R_{\text{rod}} - R_{\text{rod\_rotation}} - \tau_t - \text{err}(\mathbf{q}, \dot{\mathbf{q}}), & t < T, \\ -R_{\text{rod}} - R_{\text{rod\_rotation}} - \tau_t - \text{err}(\mathbf{q}, \dot{\mathbf{q}}) - 350 R_{\text{goal}} - 200 R_{\text{rotation}}, & t = T. \end{cases}$$

Time- and Space-Dependent Sparse Reward. The second sparse reward additionally adds sparsity based on the position and only returns goal and rotation distance in the last time step when the box is near the goal location

$$R_{\text{tot}} = \begin{cases} -R_{\text{rod}} - R_{\text{rod\_rotation}} - \tau_t - \text{err}(\mathbf{q}, \dot{\mathbf{q}}) \cdots & \\ \cdots - \text{clip}(1050 R_{\text{goal}}, 0, 100) - \text{clip}(15 R_{\text{rotation}}, 0, 100) + 300, & t = T \text{ and } R_{\text{goal}} \leq 0.1, \\ -R_{\text{rod}} - R_{\text{rod\_rotation}} - \tau_t - \text{err}(\mathbf{q}, \dot{\mathbf{q}}), & \text{else}. \end{cases}$$

B.2 Hopper Jump

In the hopper jump task the agent has to learn to jump as high as possible and land on a certain goal position at the same time. We consider five basis functions per joint resulting in an 15 dimensional weight space. The context is four-dimensional consisting of the initial joint angles $\theta \in [-0.5, 0]$, $\gamma \in [-0.2, 0]$, $\phi \in [0, 0.785]$ and the goal landing position $x \in [0.3, 1.35]$. We consider a non-Markovian reward function for the episode-based algorithms and a step-based reward for PPO, which we have extensively designed to obtain the highest possible jump.

Non-Markovian Reward. In each time-step t we provide an action cost

$$\tau_t = 10^{-3} \sum_i^K (a_t^i)^2,$$

where $K = 3$ is the number of DoF. In the last time-step T of the episode we provide a reward which contains information about the whole episode as

$$\begin{aligned} R_{height} &= 10 h_{max}, \\ R_{gdist} &= ||p_{foot,T} - p_{goal}||_2, \\ R_{cdist} &= ||p_{foot,contact} - p_{goal}||_2, \\ R_{healthy} &= \begin{cases} 2 & \text{if } z_T \in [0.5, \infty] \text{and } \theta, \gamma, \phi \in [-\infty, \infty] \\ 0 & \text{else} \end{cases} , \end{aligned}$$

where h_{max} is the maximum jump height in z-direction of the center of mass reached during the whole episode, $p_{foot,t}$ is the x-y-z position of the foot's heel at time step t, $p_{foot,contact}$ is the foot's heel position when having a contact with the ground after the first jump, p_{goal} is the goal landing position of the heel. $R_{healthy}$ is a slightly modified reward of the healthy reward defined in the original hopper task. The hopper is considered as 'healthy' if the z position of the center of mass is within the range $[0.5m, \infty]$. This encourages the hopper to stand at the end of the episode. Note that all states need to be within the range $[-100, 100]$ for $R_{healthy}$. Since this is defined in the hopper task from OpenAI already, we haven't mentioned it here. The total reward at the end of an episode is given as

$$R_{tot} = -\sum_{t=0}^{T} \tau_t + R_{height} + R_{gdist} + R_{cdist} + R_{healthy}.$$

Step-Based Reward. We consider a step-based alternative reward such that PPO is also able to learn a meaningful behavior on this task. We have tuned the reward such that we can obtain the best performance. The observation space is the same as in the original hopper task from OpenAI extended with the goal landing position and the current distance of the foot's heel and the goal landing postion. We again consider the action cost in each time-step t

$$\tau_t = 10^{-3} \sum_i^K (a_t^i)^2,$$

and additionally consider the rewards

$$\begin{aligned} R_{height,t} &= 3 h_t \\ R_{gdist,t} &= 3||p_{foot,t} - p_{goal}||_2 R_{healthy,t} \quad = \begin{cases} 1 & \text{if } z_t \in [0.5, \infty] \text{and } \theta, \gamma, \phi \in [-\infty, \infty] \\ 0 & \text{else} \end{cases} , \end{aligned}$$

where these rewards are now returned to the agent in each time-step t, resulting in the reward per time-step

$$r_t(s_t, a_t) = -\tau_t + R_{height,t} + R_{gdist,t} + R_{healthy,t}.$$

B.3 Beer Pong

In the Beer Pong task the $K = 7$ Degrees of Freedom (DoF) robot has to throw a ball into a cup on a big table. The context is defined by the cup's two dimensional position on the table which lies in the range $x \in [-1.42, 1.42]$, $y \in [-4.05, -1.25]$. For the step-based algorithms we consider cosine and sine of the robot's angles, the angle velocities, the ball's distance to the cup bottom, the ball's distance to the cup's top, the cup position and the current time step. The action space for the step-based algorithms is defined as the torques for each joint, the parameter space for the episode-based methods is 15 dimensional which consists of the two weights for the basis functions per joint and the duration of the throwing trajectory, i.e. the ball release time.

We generally consider action penalties

$$\tau_t = \frac{1}{K} \sum_i^K (a_t^i)^2,$$

consisting of the sum of squared torques per joint. For $t < T$ we consider the reward

$$r_t(s_t, a_t) = -\alpha_t \tau_t,$$

with $\alpha_t = 10^{-2}$. For $t = T$ we consider the non-Markovian reward

$$R_{task} = \begin{cases} -4 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 \cdots \\ \cdots - 2||p_{c,bottom} - p_{b,k}||_2^2 - \alpha_T \tau, & \text{if cond. 1} \\ -4 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 - \alpha_T \tau, & \text{if cond. 2} \\ -2 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 - \alpha_T \tau, & \text{if cond. 3} \\ -||p_{c,bottom} - p_{b,T}||_2^2 - \alpha_T \tau, & \text{if cond. 4} \end{cases}$$

$$R_{task} = \left\{ \begin{array}{ll} -4 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 \cdots & \\ \cdots - 2||p_{c,bottom} - p_{b,k}||_2^2 - \alpha_T \tau, & \text{if cond. 1} \\ -4 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 - \alpha_T \tau, & \text{if cond. 2} \\ -2 - min(||p_{c,top} - p_{b,1:T}||_2^2) - 0.5||p_{c,bottom} - p_{b,T}||_2^2 - \alpha_T \tau, & \text{if cond. 3} \\ -||p_{c,bottom} - p_{b,T}||_2^2 - \alpha_T \tau, & \text{if cond. 4} \end{array} \right. ,$$

where $p_{c,top}$ is the position of the top edge of the cup, $p_{c,bottom}$ is the ground position of the cup, $p_{b,t}$ is the position of the ball at time point t, and τ is the squared mean torque over all joints during one rollout and $\alpha_T = 10^{-4}$. The different conditions are:

- cond. 1: The ball had a contact with the ground before having a contact with the table.
- cond. 2: The ball is not in the cup and had no table contact
- cond. 3: The ball is not in the cup and had table contact
- cond. 4: The ball is in the cup.

Note that $p_{b,k}$ is the ball's and the ground's contact position and is only given, if the ball had a contact with the ground first.

At time step $t = T$ we also give information whether the agent's chosen ball release time B was reasonable

$$R_{release} = \left\{ \begin{array}{ll} -30 - 10(B - B_{min})^2, & \text{if } B < B_{min} \\ -30 - 10(B - B_{max})^2, & \text{if } B < B_{max} \end{array} \right. ,$$

where we define $B_{min} = 0.1s$ and $B_{max} = 1s$, such that the agent is encouraged to throw the ball within the time range $[B_{min}, B_{max}]$.

The total return over the whole episode is therefore given as

$$R_{tot} = \sum_{t=1}^{T-1} r_t(s_t, a_t) + R_{task} + R_{release}$$

A throw is considered as successfull if the ball is in the cup at the end of an episode.

B.4 Table Tennis

We consider table tennis for the entire table, i. e. incoming balls are anywhere on the side of the robot and goal locations anywhere on the opponents side. The goal is to use the seven degree of freedom robotic arm to hit the incoming ball based on its landing position and return it as close as possible to the specified goal location. As context space we consider the initial ball position $x \in [-1, -0.2]$, $y \in [-0.65, 0.65]$ and the goal position $x \in [-1.2, -0.2]$, $y \in [-0.6, 0.6]$. The observation space again contains additional information about the joints and the ball. For this experiment, we do not use any gravity compensation and allow in the episode-based setting to learn the start time t_0 and the trajectory duration T. The task is considered successful if the returned ball lands on the opponent's side of table and within ≤ 0.2m to the goal location. The reward is defined as

$$r_{task} = \begin{cases} 0, & \text{if cond. 1} \\ 0.2 - 0.2\tanh\left(\min||\mathbf{p}_r - \mathbf{p}_b||^2\right), & \text{if cond. 2} \\ 3 - 2\tanh\left(\min||\mathbf{p}_r - \mathbf{p}_b||^2\right) - \tanh\left(||\mathbf{p}_l - \mathbf{p}_{goal}||^2\right), & \text{if cond. 3} \\ 6 - 2\tanh\left(\min||\mathbf{p}_r - \mathbf{p}_b||^2\right) - 4\tanh\left(||\mathbf{p}_l - \mathbf{p}_{goal}||^2\right), & \text{if cond. 4} \\ 7 - 2\tanh\left(\min||\mathbf{p}_r - \mathbf{p}_b||^2\right) - 4\tanh\left(||\mathbf{p}_l - \mathbf{p}_{goal}||^2\right), & \text{if cond. 5} \end{cases}$$

where $\mathbf{p}_r$ is the position of racket center, $\mathbf{p}_b$ is the position of the ball, $\mathbf{p}_l$ is the ball landing position, $\mathbf{p}_{goal}$ is the target position. The different conditions are

- cond. 1: the end of episode is not reached
- cond. 2: robot did not hit the ball
- cond. 3: robot did hit the ball but the ball did not land on table or floor
- cond. 4: robot did hit the ball and returned it to the table or floor but it did not cross the net
- cond. 5: robot did hit the ball and returned it to the table or floor and cross the net

The episode ends when any of the following conditions are met

- the maximum horizon length is reached
- ball did land on the floor without hitting
- ball did land on the floor or table after hitting

For BBRL-PPO and BBRL-TRPL, the whole desired trajectory is obtained ahead of environment interaction, making use of this property we can collect some samples without physical simulation. The reward function based on this desired trajectory is defined as

$$r_{traj} = -\sum_{(i,j)} |\tau_{ij}^d| - |q_j^b|, \quad (i,j) \in \{(i,j) \mid |\tau_{ij}^d| > |q_j^b|\}$$

where τ^d is the desired trajectory, i is the time index, j is the joint index, q^b is the joint position upperbound. The desired trajectory is considered as invalid if $r_{traj} < 0$, an invalid trajectory will not be executed by robot. The overall reward for BBRL is defined as:

$$r = \begin{cases} r_{traj}, & r_{traj} < 0 \\ r_{task}, & \text{otherwise} \end{cases}$$

C Additional Evaluations

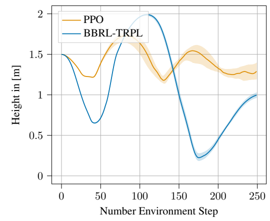

Figure 6: The improved performance on the Hopper Jump task is also demonstrated on the jumping profile for a fixed context. While BBRL-TRPL jumps once as high as possible, PPO constantly tries to maximize the height at each time step which leads to several jumps throughout the episode and consequently to a lower maximum height.

D Hyperparameters

For all methods we optimized the learning rate, sample size, batch size, number of layers, and the number of epochs. For all BBRL methods and NDP, we additionally optimized the number of basis functions. Moreover, we found that NDP requires tuning of the scale of the predicted DMP weights, which was hard-coded to 100 in the original code base. However, this value only worked for the meta-world tasks, but not for the other tasks, hence we adjusted it to allow for a fair comparison.

Table 1: Hyperparameters for the modified reacher experiments.

<table>
<tr><th></th><th>PPO</th><th>NDP</th><th>BBRL-PPO</th><th>BBRL-TRPL</th></tr>
<tr><td>number samples</td><td colspan="2">16000</td><td colspan="2">64</td></tr>
<tr><td>GAE λ</td><td colspan="2">0.95</td><td colspan="2">n.a.</td></tr>
<tr><td>discount factor</td><td colspan="2">0.99</td><td colspan="2">n.a.</td></tr>
<tr><td>ϵ_μ</td><td colspan="3">n.a.</td><td>0.05</td></tr>
<tr><td>ϵ_Σ</td><td colspan="3">n.a.</td><td>0.0005</td></tr>
<tr><td>optimizer</td><td colspan="4">adam</td></tr>
<tr><td>epochs</td><td colspan="2">10</td><td colspan="2">100</td></tr>
<tr><td>learning rate</td><td colspan="4">3e-4</td></tr>
<tr><td>use critic</td><td colspan="2">True</td><td colspan="2">False</td></tr>
<tr><td>epochs critic</td><td colspan="2">10</td><td colspan="2">n.a.</td></tr>
<tr><td>learning rate critic</td><td colspan="2">3e-4</td><td colspan="2">n.a.</td></tr>
<tr><td>number minibatches</td><td colspan="2">32</td><td colspan="2">n.a.</td></tr>
<tr><td>trust region loss weight</td><td colspan="3">n.a</td><td>10.0</td></tr>
<tr><td>entropy loss penalty</td><td colspan="4">0</td></tr>
<tr><td>normalized observations</td><td colspan="2">True</td><td colspan="2">False</td></tr>
<tr><td>normalized rewards</td><td colspan="2">True</td><td colspan="2">False</td></tr>
<tr><td>observation clip</td><td colspan="2">10.0</td><td colspan="2">n.a.</td></tr>
<tr><td>reward clip</td><td colspan="2">10.0</td><td colspan="2">n.a.</td></tr>
<tr><td>critic clip</td><td colspan="3">0.2</td><td>n.a.</td></tr>
<tr><td>importance ratio clip</td><td colspan="3">0.2</td><td>n.a.</td></tr>
<tr><td>hidden layers</td><td colspan="4">[32, 32]</td></tr>
<tr><td>hidden layers critic</td><td colspan="2">[32, 32]</td><td colspan="2">n.a.</td></tr>
<tr><td>hidden activation</td><td colspan="4">tanh</td></tr>
<tr><td>initial std</td><td colspan="4">1.0</td></tr>
<tr><td>number basis functions</td><td>n.a.</td><td colspan="3">5</td></tr>
<tr><td>number zero basis</td><td colspan="2">n.a.</td><td colspan="2">1</td></tr>
<tr><td>weight scale</td><td>n.a.</td><td>20</td><td colspan="2">n.a.</td></tr>
</table>

Table 2: Hyperparameters for the box pushing experiments.

	PPO	NDP	BBRL-PPO	BBRL-TRPL
number samples	16000		160	
GAE λ	0.95		n.a.	
discount factor	0.99		n.a.	
ϵ_μ	n.a.			0.005
ϵ_Σ	n.a.			0.0005
optimizer	adam			
epochs	10		100	
learning rate	3e-4		1e-4	
use critic	True		True	
epochs critic	10		100	
learning rate critic	3e-4		1e-4	
number minibatches	40		n.a.	
trust region loss weight	n.a			25.0
entropy loss penalty	0			
normalized observations	True		False	
normalized rewards	True		False	
observation clip	10.0		n.a.	
reward clip	10.0		n.a.	
critic clip	0.2			n.a.
importance ratio clip	0.2			n.a.
hidden layers	[256, 256]		[128, 128]	
hidden layers critic	[256, 256]		[32, 32]	
hidden activation	tanh			
initial std	1.0		1.0	
number basis functions	n.a.	5		
number zero basis	n.a.		1	
weight scale	n.a.	10	n.a.	

Table 3: Hyperparameters for the Meta-World experiments.

	PPO	NDP	BBRL-PPO	BBRL-TRPL
number samples	16000		16	
GAE λ	0.95		n.a.	
discount factor	0.99		n.a.	
ϵ_μ	n.a.			0.005
ϵ_Σ	n.a.			0.0005
optimizer	adam			
epochs	10		100	
learning rate	3e-4			
use critic	True		False	
epochs critic	10		n.a.	
learning rate critic	3e-4		n.a.	
number minibatches	32		n.a.	
trust region loss weight	n.a			10.0
entropy loss penalty	0			
normalized observations	True		False	
normalized rewards	True		False	
observation clip	10.0		n.a.	
reward clip	10.0		n.a.	
critic clip	0.2			n.a.
importance ratio clip	0.2			n.a.
hidden layers	[128, 128]		[32, 32]	
hidden layers critic	[128, 128]		n.a.	
hidden activation	tanh			relu
initial std	1.0		10.0	
number basis functions	n.a.	5		
number zero basis	n.a.		1	
weight scale	n.a.	100	n.a.	

Table 4: Hyperparameters for the hopper jumping experiments.

	PPO	BBRL-PPO	BBRL-TRPL
number samples	16384	320	
GAE λ	0.95	n.a.	
discount factor	0.99	n.a.	
ϵ_μ	n.a.		0.005
ϵ_Σ	n.a.		0.0005
optimizer	adam		
epochs	10	100	
learning rate	3e-4	1e-4	5e-5
use critic	True	False	
epochs critic	10	n.a.	
learning rate critic	3e-4	n.a.	
number minibatches	32	n.a.	
trust region loss weight	n.a		25.0
entropy loss penalty	0		
normalized observations	True	False	
normalized rewards	True	False	
observation clip	10.0	n.a.	
reward clip	10.0	n.a.	
critic clip	0.2		n.a.
importance ratio clip	0.2		n.a.
hidden layers	[128, 128]	[32, 32]	
hidden layers critic	[128, 128]	n.a.	
hidden activation	tanh		
initial std	1.0	1.0	
number basis functions	n.a.	5	
number zero basis	n.a.	1	

Table 5: Hyperparameters for the Beer Pong experiments.

	PPO	BBRL-PPO	BBRL-TRPL
number samples	16384	160	
GAE λ	0.95	n.a.	
discount factor	0.99	n.a.	
ϵ_μ	n.a.		0.005
ϵ_Σ	n.a.		0.0005
optimizer	adam		
epochs	10	100	
learning rate	1e-4	1e-4	5e-5
use critic	True	False	
epochs critic	10	n.a.	
learning rate critic	3e-4	n.a.	
number minibatches	32	n.a.	
trust region loss weight	n.a		25.0
entropy loss penalty	0		
normalized observations	True	False	
normalized rewards	True	False	
observation clip	10.0	n.a.	
reward clip	10.0	n.a.	
critic clip	0.2		n.a.
importance ratio clip	0.2		n.a.
hidden layers	[128, 128]	[32, 32]	
hidden layers critic	[128, 128]	n.a.	
hidden activation	tanh		
initial std	1.0	1.0	
number basis functions	n.a.	2	
number zero basis	n.a.	2	

Table 6: Hyperparameters for the Table Tennis experiments.

	PPO	BBRL-PPO	BBRL-TRPL
number samples	16000	200	
GAE λ	0.95		n.a.
discount factor	0.99		n.a.
ϵ_μ	n.a.		0.0005
ϵ_Σ	n.a.		0.00005
optimizer	adam		
epochs	10	100	
learning rate	3e-4	1e-4	3e-4
use critic	True		
epochs critic	10	100	
learning rate critic	3e-4	1e-4	3e-4
number minibatches	32	n.a.	
trust region loss weight	n.a		25.0
entropy loss penalty	0		
normalized observations	True	False	
normalized rewards	True	False	
observation clip	10.0	n.a.	
reward clip	10.0	n.a.	
critic clip	0.2		n.a.
importance ratio clip	0.2		n.a.
hidden layers	[256, 256]	[256]	
hidden layers critic	[256, 256]	[256]	
hidden activation	tanh		
initial std	1.0		
number basis functions	n.a.	3	
number zero basis	n.a.	1	