6.1 POLICY OPTIMIZATION ALGORITH

While generalized advantage estimation can be used along with a variety of different policy gradient methods, for these experiments, we performed the policy updates using trust region policy optimization (TRPO) (Schulman et al., 2015). TRPO updates the policy by approximately solving the following constrained optimization problem each iteration:

$$\begin{aligned}
&\underset{\theta}{\text{minimize}}\ L_{\theta_{old}}(\theta) \\
&\text{subject to}\ \ \overline{D}_{\mathrm{KL}}^{\theta_{old}}(\pi_{\theta_{old}}, \pi_\theta) \le \epsilon \\
&\text{where}\ L_{\theta_{old}}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \frac{\pi_\theta(a_n \mid s_n)}{\pi_{\theta_{old}}(a_n \mid s_n)} \hat{A}_n \\
&\qquad \overline{D}_{\mathrm{KL}}^{\theta_{old}}(\pi_{\theta_{old}}, \pi_\theta) = \frac{1}{N}\sum_{n=1}^{N} D_{KL}(\pi_{\theta_{old}}(\cdot \mid s_n) \,\|\, \pi_\theta(\cdot \mid s_n))
\end{aligned} \tag{31}$$

As described in (Schulman et al., 2015), we approximately solve this problem by linearizing the objective and quadraticizing the constraint, which yields a step in the direction $\theta - \theta_{old} \propto -F^{-1}g$, where F is the average Fisher information matrix, and g is a policy gradient estimate. This policy update yields the same step direction as the natural policy gradient (Kakade, 2001a) and natural actor-critic (Peters & Schaal, 2008), however it uses a different stepsize determination scheme and numerical procedure for computing the step.

Since prior work (Schulman et al., 2015) compared TRPO to a variety of different policy optimization algorithms, we will not repeat these comparisons; rather, we will focus on varying the γ, λ parameters of policy gradient estimator while keeping the underlying algorithm fixed.

For completeness, the whole algorithm for iteratively updating policy and value function is given below:

Initialize policy parameter θ_0 and value function parameter ϕ_0.
for $i = 0, 1, 2, \ldots$ **do**
  Simulate current policy π_{θ_i} until N timesteps are obtained.
  Compute δ_t^V at all timesteps $t \in \{1, 2, \ldots, N\}$, using $V = V_{\phi_i}$.
  Compute $\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V$ at all timesteps.
  Compute θ_{i+1} with TRPO update, Equation (31).
  Compute ϕ_{i+1} with Equation (30).
end for

Note that the policy update $\theta_i \to \theta_{i+1}$ is performed using the value function V_{ϕ_i} for advantage estimation, not $V_{\phi_{i+1}}$. Additional bias would have been introduced if we updated the value function first. To see this, consider the extreme case where we overfit the value function, and the Bellman residual $r_t + \gamma V(s_{t+1}) - V(s_t)$ becomes zero at all timesteps—the policy gradient estimate would be zero.

6.2 EXPERIMENTAL SETU

We evaluated our approach on the classic cart-pole balancing problem, as well as several challenging 3D locomotion tasks: (1) bipedal locomotion; (2) quadrupedal locomotion; (3) dynamically standing up, for the biped, which starts off laying on its back. The models are shown in Figure 1.

6.2.1 ARCHITECTU

We used the same neural network architecture for all of the 3D robot tasks, which was a feedforward network with three hidden layers, with 100, 50 and 25 tanh units respectively. The same architecture was used for the policy and value function. The final output layer had linear activation. The value function estimator used the same architecture, but with only one scalar output. For the simpler cart-pole task, we used a linear policy, and a neural network with one 20-unit hidden layer as the value function.