APPENDIX

A AWR DERIVATION

In this section, we derive the AWR algorithm as an approximate optimization of a constrained policy search problem. Our goal is to find a policy that maximize the expected *improvement* $\eta(\pi) = J(\pi) - J(\mu)$ over a sampling policy $\mu(\mathbf{a}|\mathbf{s})$. We start with a lemma from Kakade & Langford (2002), which shows that the expected improvement can be expressed in terms of the advantage $A^{\mu}(\mathbf{s}, \mathbf{a}) = \mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})$ with respect to the sampling policy μ, where $\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}}$ denotes the return obtained by performing action $\mathbf{a}$ in state $\mathbf{s}$ and following μ for the following timesteps, and $V^{\mu}(\mathbf{s}) = \int_{\mathbf{a}} \mu(\mathbf{a}|\mathbf{s}) \mathcal{R}^{\mathbf{a}}_{\mathbf{s}} \, d\mathbf{a}$ corresponds to the value function of μ,

$$\mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\mu}(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{16}$$

$$= \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V^{\mu}(\mathbf{s}_{t+1}) - V^{\mu}(\mathbf{s}_t) \right) \right] \tag{17}$$

$$= \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left[-V^{\mu}(\mathbf{s}_0) + \sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{18}$$

$$= -\mathbb{E}_{\mathbf{s}_0 \sim p(\mathbf{s}_0)} \left[V^{\mu}(\mathbf{s}_0) \right] + \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{19}$$

$$= -J(\mu) + J(\pi) \tag{20}$$

We can rewrite Equation 21 with an expectation over states instead of trajectories:

$$\eta(\pi) = \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\mu}(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{21}$$

$$= \sum_{t=0}^{\infty} \int_{\mathbf{s}} p(\mathbf{s}_t = \mathbf{s}|\pi) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \gamma^t A^{\mu}(\mathbf{s}, \mathbf{a}) \, d\mathbf{a} \, d\mathbf{s} \tag{22}$$

$$= \int_{\mathbf{s}} \sum_{t=0}^{\infty} \gamma^t p(\mathbf{s}_t = \mathbf{s}|\pi) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) A^{\mu}(\mathbf{s}, \mathbf{a}) \, d\mathbf{a} \, d\mathbf{s} \tag{23}$$

$$= \int_{\mathbf{s}} d_{\pi}(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s}) \right] \, d\mathbf{a} \, d\mathbf{s}, \tag{24}$$

where $d_{\pi}(\mathbf{s}) = \sum_{t=0}^{\infty} \gamma^t p(\mathbf{s}_t = \mathbf{s}|\pi)$ represents the unnormalized discounted state distribution induced by the policy π (Sutton & Barto, 1998), and $p(\mathbf{s}_t = \mathbf{s}|\pi)$ is the likelihood of the agent being in state $\mathbf{s}$ after following π for t timesteps.

The objective in Equation 24 can be difficult to optimize due to the dependency between $d_{\pi}(\mathbf{s})$ and π, as well as the need to collect samples from π. Following Schulman et al. (2015), we can optimize an approximation $\hat{\eta}(\pi)$ of $\eta(\pi)$ using the state distribution of μ,

$$\hat{\eta}(\pi) = \int_{\mathbf{s}} d_{\mu}(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s}) \right] \, d\mathbf{a} \, d\mathbf{s}. \tag{25}$$

$\hat{\eta}(\pi)$ matches $\eta(\pi)$ to first order (Kakade & Langford, 2002), and provides a reasonable estimate of η if π and μ are similar. Using this objective, we can formulate the following *constrained* policy search problem:

$$\arg\max_{\pi} \quad \int_{\mathbf{s}} d_{\mu}(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s}) \right] \, d\mathbf{a} \, d\mathbf{s} \tag{26}$$

$$\text{s.t.} \quad \mathrm{D}_{\mathrm{KL}} \left(\pi(\cdot|\mathbf{s}) || \mu(\cdot|\mathbf{s}) \right) \leq \epsilon, \quad \forall \, \mathbf{s} \tag{27}$$

$$\int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \, d\mathbf{a} = 1, \quad \forall \, \mathbf{s}. \tag{28}$$

Since enforcing the pointwise KL constraint in Equation 27 at all states is intractable, we relax the constraint by enforcing it only in expectation $\int_{\mathbf{s}} d_\mu(\mathbf{s}) \mathrm{D}_{\mathrm{KL}}\left(\pi(\cdot|\mathbf{s})||\mu(\cdot|\mathbf{s})\right) d\mathbf{s} \leq \epsilon$. To further simplify the optimization problem, we relax the hard KL constraint by converting it into a soft constraint with coefficient β,

$$\begin{aligned} \arg\max_{\pi} \quad & \left(\int_{\mathbf{s}} d_\mu(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})\right] \, d\mathbf{a} \, d\mathbf{s}\right) + \beta \left(\epsilon - \int_{\mathbf{s}} d_\mu(\mathbf{s}) \mathrm{D}_{\mathrm{KL}}\left(\pi(\cdot|\mathbf{s})||\mu(\cdot|\mathbf{s})\right) d\mathbf{s}\right) \\ \text{s.t.} \quad & \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \, d\mathbf{a} = 1, \quad \forall \, \mathbf{s}. \end{aligned} \tag{29}$$

Next we form the Lagrangian,

$$\begin{aligned} \mathcal{L}(\pi, \beta, \alpha) = & \left(\int_{\mathbf{s}} d_\mu(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})\right] \, d\mathbf{a} \, d\mathbf{s}\right) + \beta \left(\epsilon - \int_{\mathbf{s}} d_\mu(\mathbf{s}) \mathrm{D}_{\mathrm{KL}}\left(\pi(\cdot|\mathbf{s})||\mu(\cdot|\mathbf{s})\right) d\mathbf{s}\right) \\ & + \int_{\mathbf{s}} \alpha_{\mathbf{s}} \left(1 - \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) d\mathbf{a}\right) d\mathbf{s}, \end{aligned} \tag{30}$$

with β and $\alpha = \{\alpha_{\mathbf{s}} \mid \forall \mathrm{s} \in \mathcal{S}\}$ corresponding to the Lagrange multipliers. Differentiating $\mathcal{L}(\pi, \beta, \alpha)$ with respect to $\pi(\mathbf{a}|\mathbf{s})$ results in

$$\frac{\partial \mathcal{L}}{\partial \pi(\mathbf{a}|\mathbf{s})} = d_\mu(\mathbf{s}) \left(\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})\right) - \beta \, d_\mu(\mathbf{s}) \, \mathrm{log}\pi(\mathbf{a}|\mathbf{s}) + \beta d_\mu(\mathbf{s}) \mathrm{log}\mu(\mathbf{a}|\mathbf{s}) - \beta d_\mu(\mathbf{s}) - \alpha_{\mathbf{s}}. \tag{31}$$

Setting to zero and solving for $\pi(\mathbf{a}|\mathbf{s})$ gives

$$\mathrm{log}\pi(\mathbf{a}|\mathbf{s}) = \frac{1}{\beta}\left(\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})\right) + \mathrm{log}\mu(\mathbf{a}|\mathbf{s}) - 1 - \frac{1}{d_\mu(\mathbf{s})} \frac{\alpha_{\mathbf{s}}}{\beta} \tag{32}$$

$$\pi(\mathbf{a}|\mathbf{s}) = \mu(\mathbf{a}|\mathbf{s}) \mathrm{exp}\left(\frac{1}{\beta}\left(\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})\right)\right) \mathrm{exp}\left(-\frac{1}{d_\mu(\mathbf{s})} \frac{\alpha_{\mathbf{s}}}{\beta} - 1\right) \tag{33}$$

Since $\int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \, d\mathbf{a} = 1$, the second exponential term is the partition function $Z(\mathbf{s})$ that normalizes the conditional action distribution,

$$Z(\mathbf{s}) = \exp\left(\frac{1}{d_\mu(\mathbf{s})} \frac{\alpha_{\mathbf{s}}}{\beta} + 1\right) = \int_{\mathbf{a}'} \mu(\mathbf{a}'|\mathbf{s}) \exp\left(\frac{1}{\beta}\left(\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}'} - V^{\mu}(\mathbf{s})\right)\right) d\mathbf{a}'. \tag{34}$$

The optimal policy is therefore given by,

$$\pi^*(\mathbf{a}|\mathbf{s}) = \frac{1}{Z(\mathbf{s})} \, \mu(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta}\left(\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})\right)\right) \tag{35}$$

If π is represented by a function approximator, the optimal policy π^* can be projected onto the manifold of parameterized policies by solving the following supervised regression problem

$$\arg\min_{\pi} \quad \mathbb{E}_{\mathbf{s} \sim d_\mu(\mathbf{s})} \left[\mathrm{D}_{\mathrm{KL}}\left(\pi^*(\cdot|\mathbf{s})||\pi(\cdot|\mathbf{s})\right)\right] \tag{36}$$

$$= \arg\min_{\pi} \quad \mathbb{E}_{\mathbf{s} \sim d_\mu(\mathbf{s})} \left[\mathrm{D}_{\mathrm{KL}}\left(\frac{1}{Z(\mathbf{s})} \, \mu(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta}\left(\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})\right)\right) \middle|\middle| \pi(\cdot|\mathbf{s})\right)\right] \tag{37}$$

$$= \arg\max_{\pi} \quad \mathbb{E}_{\mathbf{s} \sim d_\mu(\mathbf{s})} \mathbb{E}_{\mathbf{a} \sim \mu(\mathbf{a}|\mathbf{s})} \left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta}\left(\mathcal{R}^{\mu}_{\mathbf{s},\mathbf{a}} - V^{\mu}(\mathbf{s})\right)\right)\right], \tag{38}$$

B AWR DERIVATION WITH EXPERIENCE REPLAY

In this section, we extend the derivation presented in Appendix A to incorporate experience replay using a replay buffer containing data from previous policies. To recap, the sampling distribution is a mixture of k past policies $\{\pi_1, \cdots, \pi_k\}$, where the mixture is performed at the trajectory level. First, we define the trajectory distribution $\mu(\tau)$, marginal state-action distribution $\mu(\mathbf{s}, \mathbf{a})$, and marginal state distribution $d_\mu(\mathbf{s})$ of the replay buffer according to:

$$\mu(\tau) = \sum_{i=1}^{k} w_i d_{\pi_i}(\tau), \qquad \mu(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^{k} w_i d_{\pi_i}(\mathbf{s}) \pi_i(\mathbf{a}|\mathbf{s}), \quad d_\mu(\mathbf{s}) = \sum_{i=1}^{k} w_i d_{\pi_i}(\mathbf{s}) \tag{39}$$

where the weights $\sum_i w_i = 1$ specify the probabilities of selecting each policy π_i. The conditional action distribution $\mu(\mathbf{a}|\mathbf{s})$ induced by the replay buffer is given by:

$$\mu(\mathbf{a}|\mathbf{s}) = \frac{\mu(\mathbf{s}, \mathbf{a})}{d_\mu(\mathbf{s})} = \frac{\sum_{i=1}^k w_i d_{\pi_i}(\mathbf{s}) \pi_i(\mathbf{a}|\mathbf{s})}{\sum_{j=1}^k w_j d_{\pi_j}(\mathbf{s})}. \tag{40}$$

Next, using Lemma 6.1 from Kakade & Langford (2002) (also derived in Appendix A), the expected improvement of π over each policy π_i satisfies

$$J(\pi) = J(\pi_i) + \mathbb{E}_{\mathbf{s}\sim d_\pi(\mathbf{s}), a\sim\pi(\mathbf{a}|\mathbf{s})}\left[A^{\pi_i}(\mathbf{s}, \mathbf{a})\right] \tag{41}$$

The expected improvement over the mixture can then be expressed with respect to the individual policies,

$$\begin{aligned}
\eta(\pi) &= J(\pi) - J(\mu) && (42)\\
&= J(\pi) - \sum_{i=1}^k w_i J(\pi_i) && (43)\\
&= \sum_{i=1}^k w_i \left(J(\pi) - J(\pi_i)\right) && (44)\\
&= \sum_{i=1}^k w_i \left(\mathbb{E}_{\mathbf{s}\sim d_\pi(\mathbf{s}), \mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[A^{\pi_i}(\mathbf{s}, \mathbf{a})\right]\right) && (45)
\end{aligned}$$

In order to ensure that the policy π is similar to the past policies, we constrain π against the conditional action distributions of the replay buffer,

$$\mathbb{E}_{\mathbf{s}\sim\mu(\mathbf{s})}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi(\mathbf{a}|\mathbf{s}) \middle|\middle| \mu(\mathbf{a}|\mathbf{s})\right)\right] \leq \varepsilon. \tag{46}$$

Note that constraining π against $\mu(\mathbf{a}|\mathbf{s})$ has a number of desirable properties. First, the constraint prevents the policy π from choosing actions that are vastly different from **all** of the policies $\{\pi_1, \cdots, \pi_k\}$. Second, the mixture weight assigned to each π_i in the definition of μ depends on the marginal state density $d_{\pi_i}(\mathbf{s})$ for the particular policy. This property is desirable as the policy π is now constrained to be similar to π_i only at states that are likely to be visited by π_i. This then yields the following constrained objective:

$$\begin{aligned}
\arg\max_{\pi} \quad & \sum_{i=1}^k w_i \, \mathbb{E}_{\mathbf{s}\sim d_{\pi_i}(\mathbf{s})} \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\mathcal{R}^{\pi_i}_{\mathbf{s},\mathbf{a}} - V^{\pi_i}(\mathbf{s})\right] && (47)\\
\text{s.t.} \quad & \mathbb{E}_{\mathbf{s}\sim d_\mu(\mathbf{s})}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi(\cdot|\mathbf{s}) || \mu(\cdot|\mathbf{s})\right)\right] \leq \epsilon, && (48)\\
& \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \, d\mathbf{a} = 1, \quad \forall\, \mathbf{s}. && (49)
\end{aligned}$$

The Lagrangian of the above objective is given by:

$$\begin{aligned}
\mathcal{L}(\pi, \beta, \alpha) = & \left(\sum_i w_i \mathbb{E}_{\mathbf{s}\sim d_{\pi_i}(\mathbf{s})} \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\mathcal{R}^{\pi_i}_{\mathbf{s},\mathbf{a}} - V^{\pi_i}(\mathbf{s})\right]\right)\\
& + \beta\left(\epsilon - \mathbb{E}_{\mathbf{s}\sim d_\mu(\mathbf{s})} \mathrm{D}_{\mathrm{KL}}\left(\pi(\cdot|\mathbf{s}) \middle|\middle| \frac{\sum_{i=1}^k w_i d_{\pi_i}(\mathbf{s}) \pi_i(\cdot|\mathbf{s})}{\sum_{j=1}^k w_j d_{\pi_j}(\mathbf{s})}\right)\right) && (50)\\
& + \int_{\mathbf{s}} \alpha_{\mathbf{s}} \left(1 - \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) d\mathbf{a}\right) d\mathbf{s},
\end{aligned}$$

Solving the Lagrangian following the same procedure as Appendix A leads to an optimal policy of the following form:

$$\pi^*(\mathbf{a}|\mathbf{s}) = \frac{1}{Z(\mathbf{s})} \, \mu(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta} \frac{\sum_i w_i d_{\pi_i}(\mathbf{s}) \left(\mathcal{R}^{\pi_i}_{\mathbf{s},\mathbf{a}} - V^{\pi_i}(\mathbf{s})\right)}{\sum_j w_j d_{\pi_j}(\mathbf{s})}\right) \tag{51}$$

Finally, if π is represented by a function approximator, the optimal policy π^* can be projected onto the manifold of parameterized policies by solving the following supervised regression problem

$$\arg\min_{\pi} \quad \mathbb{E}_{\mathbf{s},\sim d_\mu(\mathbf{s})}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi^*(\cdot|\mathbf{s})||\pi(\cdot|\mathbf{s})\right)\right] \tag{52}$$

$$= \arg\min_{\pi} \quad \mathbb{E}_{\mathbf{s}\sim d_\mu(\mathbf{s})}\left[\mathrm{D}_{\mathrm{KL}}\left(\frac{1}{Z(\mathbf{s})}\,\mu(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\beta}\frac{\sum_i w_i d_{\pi_i}(\mathbf{s})\left(\mathcal{R}^{\pi_i}_{\mathbf{s},\mathbf{a}} - V^{\pi_i}(\mathbf{s})\right)}{\sum_j w_j d_{\pi_j}(\mathbf{s})}\right)\middle|\middle|\pi(\cdot|\mathbf{s})\right)\right] \tag{53}$$

One of the challenges of optimizing the objective in Equation 53 is that computing the expected return in the exponent requires rolling out multiple policies starting from the same state, which would require the environment to be resettable to any given state. Therefore, to obtain a more practical objective, we approximate the expected return across policies using a single rollout from the replay buffer,

$$\frac{\sum_i w_i d_{\pi_i}(\mathbf{s})\mathcal{R}^{\pi_i}_{\mathbf{s},\mathbf{a}}}{\sum_j w_j d_{\pi_j}(\mathbf{s})} \approx \mathcal{R}^{\mathcal{D}}_{\mathbf{s},\mathbf{a}} \;\text{ such that }\; (\mathbf{s},\mathbf{a}) \in \mathcal{D} \tag{54}$$

This single-sample estimator results in a biased estimate of the exponentiated advantage, because the expectation with respect to the mixture weights appears in the exponent. But in practice, we find this biased estimator to be effective for our experiments. Therefore, the objective used in practice is given by:

$$\arg\max_{\pi} \quad \sum_{i=1}^{k} w_i\, \mathbb{E}_{\mathbf{s}\sim d_{\pi_i}(\mathbf{s})}\mathbb{E}_{\mathbf{a}\sim\pi_i(\mathbf{a}|\mathbf{s})}\left[\log\pi(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\beta}\left(\mathcal{R}^{\pi_i}_{\mathbf{s},\mathbf{a}} - \frac{\sum_j w_j d_{\pi_j}(\mathbf{s})V^{\pi_j}(\mathbf{s})}{\sum_j w_j d_{\pi_j}(\mathbf{s})}\right)\right)\right], \tag{55}$$

where the expectations can be approximated by simply sampling from $\mathcal{D}$ following Line 6 of Algorithm 1. Note, the baseline in the exponent now consists of an average of the value functions of the different policies. One approach for estimating this quantity would be to fit separate value functions V^{π_i} for each policy. However, if only a small amount of data is available from each policy, then V^{π_i} could be highly inaccurate. Therefore, instead of learning separate value functions, we fit a single *mean* value function $\bar{V}(\mathbf{s})$ that directly estimates the weighted average of V^{π_i}'s,

$$\bar{V} = \arg\min_{V} \sum_i w_i\, \mathbb{E}_{\mathbf{s},\sim d_{\pi_i}(\mathbf{s})}\mathbb{E}_{\mathbf{a}\sim\pi_i(\mathbf{a}|\mathbf{s})}\left[\,||\mathcal{R}^{\pi_i}_{\mathbf{s},\mathbf{a}} - V(\mathbf{s})||^2\right] \tag{56}$$

This loss can also be approximated by simply sampling from the replay buffer following Line 5 of Algorithm 1. The optimal solution $\bar{V}(\mathbf{s}) = \frac{\sum_i w_i d_{\pi_i}(\mathbf{s})V^{\pi_i}(\mathbf{s})}{\sum_j w_j d_{\pi_j}(\mathbf{s})}$ is exactly the baseline in Equation 55.

C EXPERIMENTAL SETUP

In our experiments, the policy is represented by a fully-connected network with 2 hidden layers consisting of 128 and 64 ReLU units respectively (Nair & Hinton, 2010), followed by a linear output layer. The value function is modeled by a separate network with a similar architecture, but consists of a single linear output unit for the value. Stochastic gradient descent with momentum is used to update both the policy and value function. The stepsize of the policy and value function are 5×10^{-5} and 1×10^{-4} respectively, and a momentum of 0.9 is used for both. The weight clipping threshold $\omega_{\max}$ is set to 20. At each iteration, the agent collects a batch of approximately 2000 samples, which are stored in the replay buffer $\mathcal{D}$ along with samples from previous iterations. The replay buffer stores 50k of the most recent samples. Updates to the value function and policy are performed by uniformly sampling minibatches of 256 samples from $\mathcal{D}$. The value function is updated with 200 gradient steps per iteration, and the policy is updated with 1000 gradient steps.

To set the value of the Lagrange multiplier β, we found that a simple heuristic of setting β adaptively through advantage normalization works well in practice, and alleviates the need for extensive task-specific tuning. For a given advantage value $A(\mathrm{s},\mathbf{a})$, the normalized advantage $\bar{A}(\mathrm{s},\mathbf{a})$ is given by,

$$\bar{A}(\mathrm{s},\mathbf{a}) = \frac{A(\mathrm{s},\mathbf{a}) - \mu_A}{\sigma_A}, \tag{57}$$

where μ_A and σ_A represents the mean and standard deviation of all advantage values in the replay buffer. This method is akin to setting $\beta = \sigma_A$. The normalized advantages $\bar{A}(\mathrm{s}, \mathbf{a})$ are then used to compute the weights $\omega_{\mathbf{s},\mathbf{a}} = \exp\left(\bar{A}(\mathbf{s}, \mathbf{a})\right)$ for the policy update. This advantage normalization technique is commonly used in standard implementations of algorithms such as PPO (Dhariwal et al., 2017).

D SIMILARITIES TO POLICY GRADIENTS

On the surface, the AWR policy update bears striking similarities to a conventional policy gradient (PG) update (Sutton et al., 2000):

$$\mathbb{E}_{\mathbf{s}\sim d_\pi(\mathbf{s})}\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\nabla_\pi \log \pi(\mathbf{a}|\mathbf{s}) \ \left(\mathcal{R}^\pi_{\mathbf{s},\mathbf{a}} - V^\pi(\mathbf{s})\right)\right] \tag{Policy Gradient}$$

$$\mathbb{E}_{\mathbf{s}\sim d_\mu(\mathbf{s})}\mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[\nabla_\pi \log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta}\left(\mathcal{R}^\mu_{\mathbf{s},\mathbf{a}} - V^\mu(\mathbf{s})\right)\right)\right]. \tag{AWR}$$

However, there are a number of subtle but important differences between the two. First, basic policy gradient algorithms are on-policy methods, which requires the data to be sampled from the same policy π that is being optimized $\mathbf{s} \sim d_\pi(\mathbf{s})$ and $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$, whereas AWR can in principle learn using data from any sampling distribution μ. This requirement for policy gradient methods is because PG directly differentiates through the sampling distribution to compute the gradient of the expected return with respect to the policy parameters. But with AWR and other EM algorithms, they first construct an estimate of the optimal action distribution at each state, and then projects that action distribution onto the space of parameterized policies. Therefore AWR does not need to differentiate through the sampling distribution, which is a critical feature for settings such as batch RL, where the sampling distribution (e.g. demo policy) may not be available to the agent. In AWR, the log probability of an action $\log\pi(\mathbf{a}|\mathbf{s})$ is weighted by the exponentiated advantage $\exp\left(\frac{1}{\beta}\left(\mathcal{R}_{\mathbf{s},\mathbf{a}} - V(\mathbf{s})\right)\right)$, while in PG the log probability is weighted just by the advantage $(\mathcal{R}_{\mathbf{s},\mathbf{a}} - V(\mathbf{s}))$ without the exponential. Since the exponentiated advantage is non-negative, the objective used in the AWR update is a maximum likelihood objective that tries to maximize the likelihood of all actions, but to varying amounts depending on the exponentiated advantage. In the case of PG, the advantage can be both positive and negative, therefore PG updates decrease the likelihood of actions with negative advantages, and thus it is not a conventional maximum likelihood objective. In practice, negative TD updates are often a source of instability when applying PG to off-policy data.

E ADDITIONAL EXPERIMENTS

A comprehensive comparison of AWR with prior methods on all of the tasks considered are available in Figure 6 and 7. Due to the slow wall-clock times of TD3 and SAC, some training runs did not have sufficient time to collect as many samples as other algorithms.

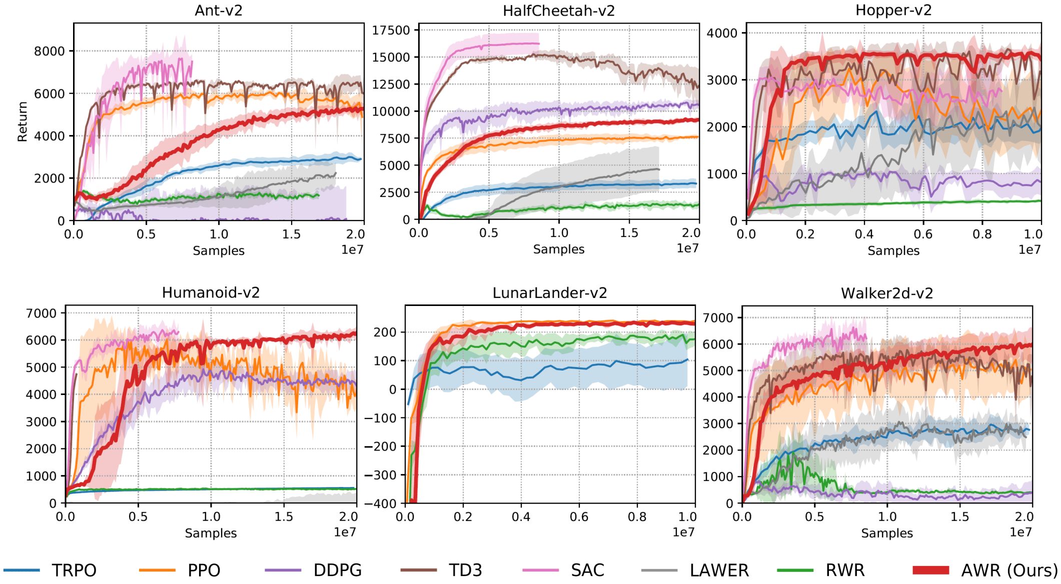

Figure 6: Learning curves of the various algorithms when applied to OpenAI Gym tasks. Results are averaged over 10 random seeds. AWR is generally competitive with the best current methods.

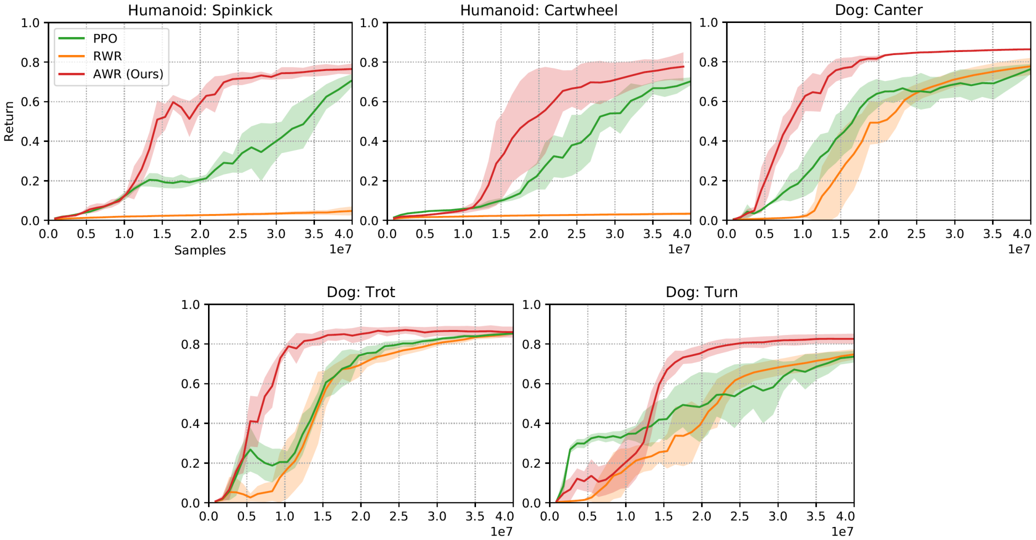

Figure 7: Learning curves on motion imitation tasks. On these challenging tasks, AWR generally learns faster than PPO and RWR.

Figure 8 illustrates more focused comparisons between AWR and closely related algorithms, including RWR (Peters & Schaal, 2007), LAWER (Neumann & Peters, 2009), and REPS (Abdolmaleki et al., 2018a). AWR consistently outperforms these prior methods on standard continuous control benchmarks. The performance figures of the prior algorithms are consistent with prior work (Duan et al., 2016), which have also evaluated these methods on this suite of tasks. While AWR share some resemblances with these prior methods, our design decisions appear vital for effective performance with neural network function approximators. In the case of REPS, we found that the procedure of fitting a value function by optimizing a dual function tends to result in instability during training, which prevents the policy from learning effective behaviors. While in AWR, the value function is fitted with simple least squares regression, which yields a much more stable update procedure.

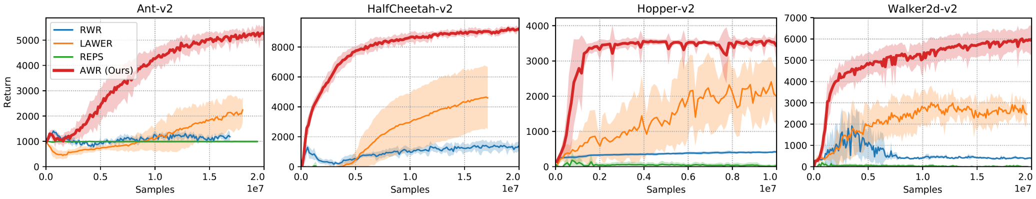

Figure 8: Learning curves comparing AWR to closely related algorithms on OpenAI Gym tasks. AWR consistently outperforms these prior methods.

E.1 WEIGHT CLIPPING

To analyze the effects of weight clipping on the stability of AWR, we compare learning curves of policies trained with weight clipping using a threshold of $\omega_{\max} = 20$, and policies trained without weight clipping. Figure 9 compares the learning curves with and without clipping. 10 separate AWR runs with different random seeds are visualized separately. With weight clipping, performance remains stable throughout training. Policies trained without weight clipping are substantially more unstable, exhibiting drastic fluctuations in performance as a result of exploding gradients from excessively large weights. Some training runs without clipping are terminated early due to exploding gradients causing the networks to output NaNs. These experiments suggest that weight clipping is vital for ensuring stable training with AWR.

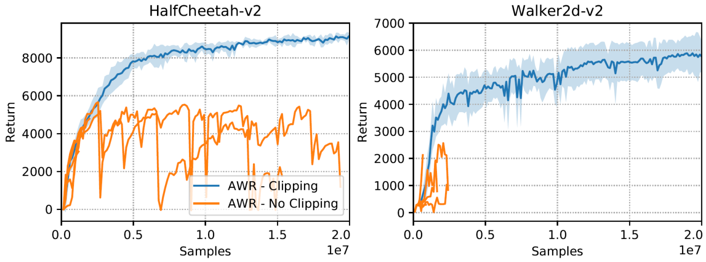

Figure 9: Learning curves comparing AWR policies trained with and without weight clipping. Weight clipping is vital for ensuring stable training with AWR. Policies trained without weight clipping are susceptible to exploding gradients due to excessively large weights.

E.2 NORMALIZATION TECHNIQUES

Our implementation of AWR includes additional normalization techniques such as state normalization and reward scaling, which are also commonly incorporated into widely used RL frameworks, such as OpenAI Baselines (Dhariwal et al., 2017), RLKit (Pong, 2019), Softlearning, TFAgents (Hafner et al., 2017). These techniques are also used by other algorithms, such as PPO, in our experiments. Data whitening is also standard practice in most machine learning applications. Here, we evaluate the effects of these design decisions on the perform of AWR.

The state features are normalized using the running mean $\mu_{\mathbf{s}}$ and standard deviation $\sigma_{\mathbf{s}}$ computed from data collected by the agent

$$\bar{\mathbf{s}} = \frac{\mathbf{s} - \mu_{\mathbf{s}}}{\sigma_{\mathbf{s}}}, \tag{58}$$

where $\mathbf{s}$ denotes the original unnormalized state, $\bar{\mathbf{s}}$ represents the normalized state, and all operations are applied element-wise to each feature. The normalized state is then used as input to the policy and value function. The rewards are also scaled according to

$$\bar{r} = (1 - \gamma)r, \tag{59}$$

where γ is the discount factor. This scaling reduces the magnitude of the returns, which can improve stability when training the value function. While we have found these techniques to be useful for reducing the amount of hyperparameter tuning required for AWR, they are by no means critical components of the algorithm. To evaluate the effects of these normalization techniques, we evaluate the performance of AWR when these normalization techniques are disabled. Figure 10 compares learning curves for AWR with and without these additional normalization techniques. Performance is similar on most tasks even when these techniques are removed, indicating that these standard techniques from prior work are not critical to the performance of AWR.

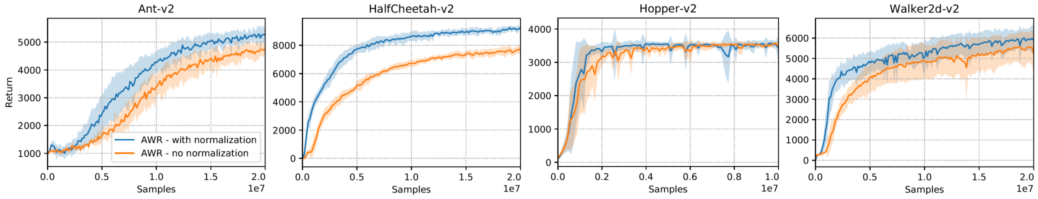

Figure 10: Learning curves comparing AWR with and without normalization techniques. Performance is similar on most tasks even when these techniques are disabled.