A MISSING BACKGROUND

A.1 VARIATIONAL AUTO-ENCODER

A variational auto-encoder (VAE) (Kingma & Welling, 2014) is a generative model that aims to learn the data distribution $p(X)$ given a set of observations $\{x_i\}_{i=1}^N$. While directly optimizing $p(X)$ is intractable, we can optimize its variational lower-bound:

$$\log p(X) \geq \mathbb{E}_{z\sim q(z)}[\log p(X|z)] - \mathcal{D}_{\text{KL}}(q(z)||p(z)) \tag{12}$$

where $q(Z)$ is the variational distribution and $p(Z)$ is the prior. In VAE, $q(Z)$ is $q(Z|X)$ so that it is an auto-encoder. Optimizing Equation 12 using gradient descent requires back-propagate the gradient through a sample operator. Fortunately, if the latent variable is a multivariate Gaussian distribution, we can use re-parametrization trick. We refer to (Kingma & Welling, 2019) for more mathematical derivations.

B IMPLEMENTATION DETAILS

Policy network Our policy network is a 3-layer feed-forward neural network. The size of each hidden layer is 512. We apply RELU activation (Agarap, 2018) after each hidden layer. Following (Haarnoja et al., 2018a), the output is a Gaussian distribution with diagonal covariance matrix. We apply `tanh` to enforce the action bounds. The log-likelihood after applying the `tanh` function has a simple closed form solution. We refer to (Haarnoja et al., 2018a) Appendix C for more details.

Q network Following (Haarnoja et al., 2018a; Fujimoto et al., 2018a; Wu et al., 2019), we train two independent Q network $\{Q_{\psi_1}, Q_{\psi_2}\}$ to penalize uncertainty over the future states. We maintain a target Q network $\{Q_{\psi'_1}, Q_{\psi'_2}\}$ with the same architecture and update the target weights using a weighted sum of the current Q network and the target Q network. When computing the target Q values, we simply take the minimum value of the two Q networks:

$$Q_{\psi'}(s', a') = \min_{j=1,2} Q_{\psi'_j}(s', a') \tag{13}$$

Each Q network is a 3-layer feed-forward neural network. The size of each hidden layer is 256. We apply RELU activation (Agarap, 2018) after each hidden layer.

Behavior policy network Following the previous work (Fujimoto et al., 2018b; Kumar et al., 2019), we learn a conditional variational auto-encoder (Kingma & Welling, 2014) as our behavior policy network. The encoder takes a pair of states and actions, and outputs a Gaussian latent variable Z. The decoder takes sampled latent code z and states, and outputs a mixture of Gaussian distributions. Both the architecture of the encoder and the decoder is a 3-layer feed-forward neural network. The size of each hidden layer is 512. The activation is relu (Agarap, 2018). To avoid epistemic uncertainty, we train B ensembles of behavior policy networks. At test time, we randomly select one model to perform the calculations. We found $B = 3$ is sufficient for all the experiments. We pre-train the the behavior policy network with 300 epochs. We use cross validation with split ratio 0.1. The batch size is 256.

α network The α network takes in a state and outputs the Lagrange multiplier for the state. The architecture of the α network is a 3-layer feed-forward neural network with relu (Agarap, 2018) activation. The size of each hidden layer is 256. We use `softplus` activation after the output to ensure that all the values are positive.

Table 2: Default hyper-parameters

Hyper-parameter	Value
Optimizer	Adam (Kingma & Ba, 2015)
Policy learning rate	5e−6
Q network learning rate	3e−4
α learning rate	1e−5
Policy update batch size	1000
α batch size	1000
Q network update batch size	10000
Target update rate τ	5e−3
Discount factor γ	0.99
Initial β	10
β learning rate	1e−3
Steps per epoch T	1000
Number of epochs to pre-train	100

Table 3: Task-specific hyper-parameters

Task name	KL divergence threshold ϵ_{KL}	maximum entropy $\mathcal{H}_0$
halfcheetah-rand	9	-3
walker2d-rand	3	0
hopper-rand	6	-6
halfcheetah-med	1.5	-12
walker2d-med	6	-12
hopper-med	3	-6
halfcheetah-med-exp	6	-12
walker2d-med-exp	6	-12
hopper-med-exp	3	-6
halfcheetah-mixed	8	-12
walker2d-mixed	12	-12
hopper-mixed	6	-6