A Implementation Details

Designer Policy Instead of optimizing Eq. 3 directly, we use a similar objective as in Soft Actor-Critic (SAC) (Haarnoja et al., 2019). Let's assume q is the designer actor and ν_1 and ν_2 are two critics that are initialized independently. We use the following objective to train critics:

$$l(\nu_1, \nu_2) = \mathbb{E}_{(s,a,z)\sim\mathbb{RB}} \left[(\nu_1(z) - r(s,a))^2 + (\nu_2(z) - r(s,a))^2 \right] \tag{4}$$

where $\mathbb{RB}$ is a replay buffer of previous design predictions, z, and corresponding state and action pairs, (s, a), collected via learned behavior policy $\hat{\mu}$ in the simulator. $r(s, a) = -\log(1 - g(s, a))$ is the approximate log ratio estimated using the discriminator as in Section 4.1. We use the following objective to train the designer actor:

$$l(q) = \mathbb{E}_{z\sim q(z)} \left[\alpha \log q(z) - \min(\nu_1(z), \nu_2(z)) + \lambda D_{KL}(q(z)|p(z)) \right] \tag{5}$$

We initialize the prior as $p(z) = q(z)$ but keep its variance fixed and set its mean to be the mean of $q(z)$ in the previous training step. This gives a rudimentary trust-region like optimization algorithm which also discourages the variance of $q(z)$ from collapsing.

To fit the designer policy, we use a Gaussian with tanh (or sigmoid for *geom-friction* parameter) squashing (Haarnoja et al., 2019) where we scale the output to be within a predefined interval for each parameter. We rescale the output to be within $[-5, 5]$ for independent parameters (except *gravity*) and $[-10, 10]$ for all and *gravity* parameters.

In our preliminary experiments, we observed that diversity of design parameters in the initial replay is crucial to get good performance. Even when we run a random designer for a number of steps (200k for D4RL), we saw it might not be sufficient. For example, for HalfCheetah, 200k steps give only 200 different design parameters. Without enough diversity, the discriminator might not able to train to optimality and the designer distribution converges into a sub-optimal solution. We solve this problem by terminating initial episodes early while running a random designer for the same number of steps. This significantly increases diversity of design parameters and enough to train the discriminator successfully.

We also make another observation that, when the actor loss without the KL term in Eq. 5 converges, KL term might take off and force the variance of the designer policy to increase to match the prior. In some cases, this causes the designer to diverge for a number of steps before it can learn the correct parameter distribution again. We mitigate this issue by using an exponential decay function on KL divergence weight λ. This encourages the designer to explore more early on during training and exploit later when discriminator is sufficiently trained.

Discriminator We train the discriminator using samples from the replay buffer $\mathbb{RB}$ and offline dataset D. We use gradient penalty following (Gulrajani et al., 2017).

Online Policy We use SAC (Haarnoja et al., 2019) as the online agent. We use default hyperparameters without tuning.

Offline Policy We use ValueDICE (Kostrikov et al., 2020) and Behavioral Cloning (BC) to learn a behavior policy in ValueDICE and D4RL (+MiniGrid) experiments, respectively. We use default parameters for the ValueDICE without tuning. Different from (Kostrikov et al., 2020), we do not use replay regularization as our preliminary results showed that the performance of ValueDICE can degrade when replay has mixed experience collected from different simulators with possibly different MDPs. We observe this behavior more strongly for the HalfCheetah environment. For MiniGrid experiments, we place both the agent and the goal in randomly selected empty positions in the grid. We use gray scale images and shapes of obstacles in the grid are randomly assigned. We use a 3-layer Convolutional Neural Network (CNN) with an additional MLP architecture as in (Mnih et al., 2015).

We present a list of hyper parameters and corresponding search spaces that we tuned/selected for OTED in Table 3.

B Ground Truth Design Parameters for Mujoco

In Table 4, we give a list of ground truth parameters for Mujoco environments that we experimented with. Note that these parameters are shared across different offline datasets such as ValueDICE or D4RL demonstrations.

Hyper Parameters	**Value**
Designer learning rate	[1e-5, 3e-5]
Designer actor architecture	MLP[256, 256, 256; ReLU]
Designer critic architectures	MLP[256, 256; ReLU]
Designer SAC alpha	0.05
Discriminator architecture	MLP[256, 256; Tanh]
Batch size	256
Buffer size	$5 * 10^6$
Initial episode length	[100, 500, 1000]
Log standard deviation of $p(z)$	[0.05, 0.1]
KL divergence weight λ	[0.05, 0.1, 0.2, 0.5]
KL divergence weight decay rate	[0.95, 1.0]
KL divergence weight decay steps	[3500, 5000, 10000]

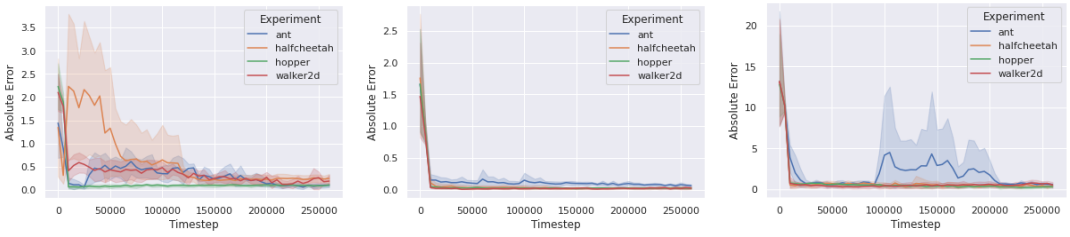
Table 3: Hyper parameter search space for OTED.

	HalfCheetah	Hopper	Walker2D	Ant
geom-friction	0.4	2.0	1.9	1.0
actuator-gainprm	1.0	1.0	1.0	1.0
gravity	-9.81	-9.81	-9.81	-9.81

Table 4: Ground truth design parameters for the Mujoco environments.

C Additional Experiments using ValueDICE Dataset

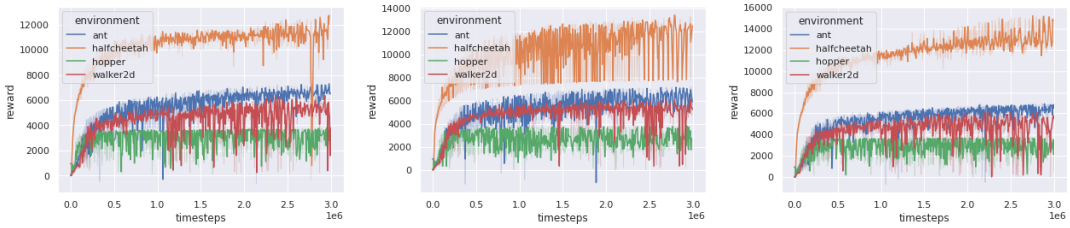
We also perform additional experiments using the same demonstration dataset from ValueDICE (Kostrikov et al., 2019) with an additional Mujoco environment Ant which is missing in the D4RL benchmark datasets. OTED recovers the unknown target environment parameter with low absolute error on all parameterizations (Figure 4). We observe that different tasks show varying sensitivity to different parameters. For example, on *actuator-gainprm*, all environments exhibit low error and relatively low variance while on *geom-friction*, we observe relatively high error and high variance. Ant is one of the most challenging environments where we get higher error on all settings. We hypothesize that the reason is higher number of dimensions in the observation space (111 dimensions which is around 7 times higher than others) which makes it more difficult to learn using a small demonstration dataset.



(a) Friction (b) Gain PRM (c) Gravity

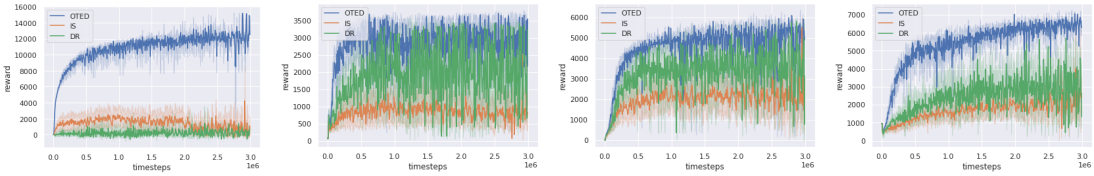
Figure 4: Absolute design error of OTED using different parameterizations on ValueDICE demonstrations. OTED reliably learns to recover the target MDP with a small error on all environments.

In Figure 5, we plot the performance of OTED-SAC trained in tuned simulators using ValueDICE demonstrations. Across all experiments, we observe consistently good results where we achieve on par with an online state-of-the-art SAC agent that is trained on the ground truth target environment. We also compare with baseline domain transfer methods in Figure 6 where OTED-SAC consistently performs better. Figure 6 also demonstrates that a mis-specified simulator would hurt the performance of a SAC agent as shown by lower results of DR and IS compared to our proposed method.



(a) Friction (b) Gain PRM (c) Gravity

Figure 5: Performance of the OTED-SAC using ValueDICE dataset. OTED-SAC outperforms ValueDICE (Kostrikov et al., 2020) on almost all parameter settings.

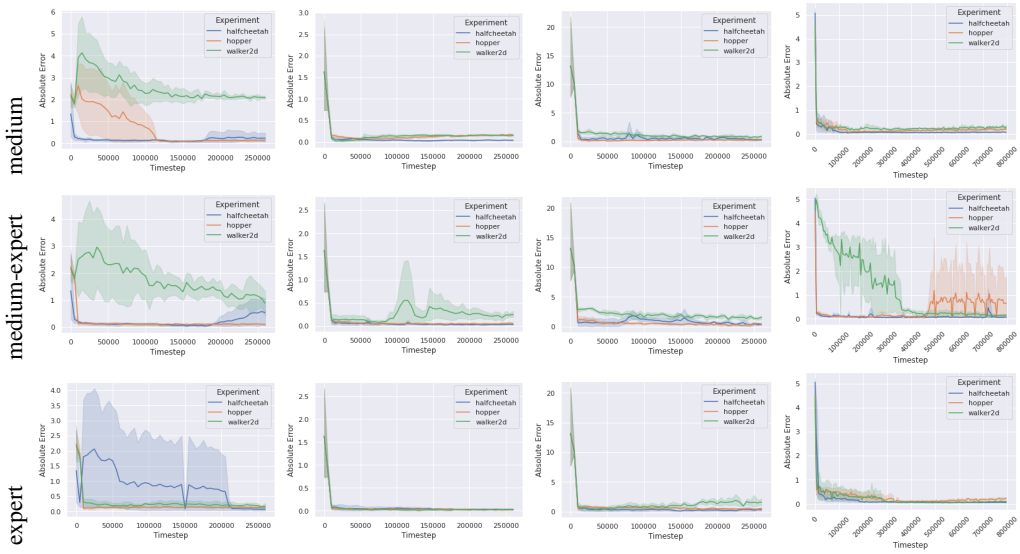


(a) HalfCheetah (b) Hopper (c) Walker2D (d) Ant

Figure 6: Comparison of OTED-SAC to baseline DR and IS methods on ValueDICE dataset. OTED-SAC performs better than both baselines under all settings.

D Detailed Comparison on D4RL Benchmark

In Figure 7, we present detailed design errors of our method on different expert levels on D4RL dataset. As expected, OTED exhibits higher error and variance on the medium-level data than others. This is not only due to the quality of the data but also the performance of the BC as it performs poorly on all medium tasks.

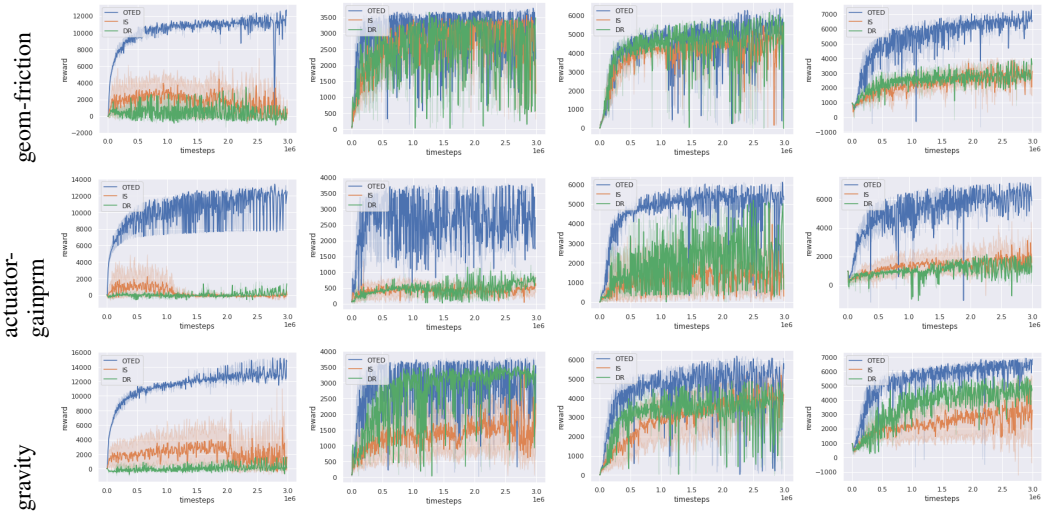


(a) geom-friction (b) actuator-gainprm (c) gravity (d) all parameters

Figure 7: Absolute error of OTED on different simulator parameters using the D4RL benchmark with different expert-levels.

In Figure 8, we present detailed comparison of OTED-SAC to DR and IS on ValueDICE demonstrations. OTED-SAC outperforms both baselines on all parameters and environments, illustrating

that a misspecified simulator yields a poor performance. We observe that on Hopper and Walker2D environments with *geom-friction* parameter, baselines also achieve relatively high performance, showing the sensitivity of the environments to different parameters.

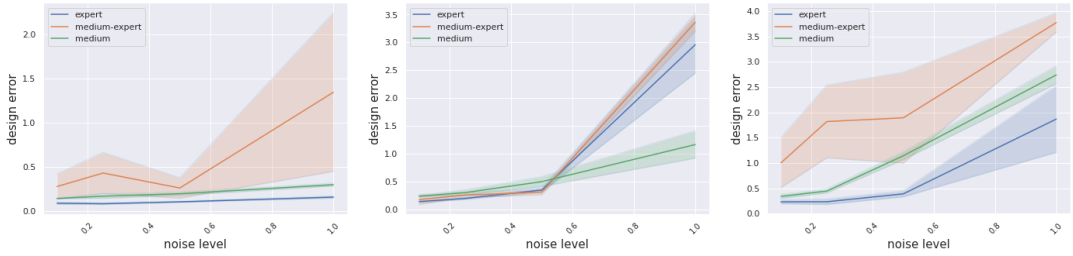


(a) HalfCheetah (b) Hopper (c) Walker2D (d) Ant

Figure 8: Detailed comparison of OTED-SAC to baseline DR and IS methods on ValueDICE dataset.

E CONTROLLED SIM2REAL EXPERIMENTS

We conducted controlled experiments to understand how the performance of OTED changes w.r.t an increasing gap between offline and online data. We used additive zero mean gaussian noise with varying standard deviation to modify the offline data and retrained OTED to learn all parameters in D4RL benchmark. As expected, OTED's performance drops with higher noise but OTED can still be robust up to a certain noise, especially for HalfCheetah and Hopper environments (see Figure 9). We make a similar observation with expert-level data in all environments; no visible increase in design error up to a noise level. The reason for drop in performance is that as offline data becomes less similar to online data, the discriminator overfits easily which leads to poor log ratio estimations and suboptimal designer policies.



(a) HalfCheetah (b) Hopper (c) Walker2d

Figure 9: Performance of OTED with varying gaussian noise levels. OTED's performance drops as noise increases. But, OTED is also robust to certain noise levels. For example, in HalfCheetah, there is no visible drop in performance even with 0.5 standard deviation.

F PERFORMANCE IN A MISSPECIFIED SIMULATOR

In reality, simulators might have more parameters that are not necessarily explicit and thus can't be tuned. To understand OTED's performance in this setting, we fixed the gravity parameter to an incorrect value and let OTED learn the other two. We used different values for gravity to measure the performance w.r.t different error levels. As expected, the performance of OTED drops as the gravity parameter becomes more inaccurate (see Figure 10). The reason is similar to above where the discriminator overfits. But, OTED can be robust up to a certain error level. For example, in HalfCheetah, OTED still gives good performance even when there is a 50% error rate in the parameter (using -4.0 vs the ground truth -9.81). We think that these experiments also shed light on how OTED would perform in more realistic settings.

(a) HalfCheetah (b) Hopper (c) Walker2d

Figure 10: Performance of OTED in a misspecified simulator. OTED's performance drops as the error in the fixed parameter (gravity) increases. But, similar to noisy offline data results, OTED is also robust to certain error levels. For example, in HalfCheetah, OTED still performs well even with 50% error rate on the parameter.

G PERFORMANCE OF OTED IN 2-WAY PARAMETER COMBINATIONS

We ran additional experiments where only 2 of the 3 parameters are learned while the remaining one is assumed to be correct (see Table 5). We observe a similar trend as learning all parameters. We also observe higher errors with gravity involved. The main reason is that gravity has relatively high ground truth value where a relatively small error can still give high absolute error and bias the results.

	All Params	Friction&Prm	Friction&Gravity	Prm&Gravity
halfcheetah-expert	0.064 (0.003)	0.067(0.006)	0.147(0.016)	0.094(0.007)
halfcheetah-medium-expert	0.061 (0.016)	0.044(0.01)	0.668(0.117)	0.368(0.217)
halfcheetah-medium	0.071 (0.007)	0.026(0.004)	0.422(0.025)	0.203(0.022)
hopper-expert	0.196 (0.03)	0.032(0.007)	0.361(0.033)	0.161(0.014)
hopper-medium-expert	0.621 (0.455)	0.289(0.192)	0.185(0.053)	0.206(0.054)
hopper-medium	0.203 (0.025)	0.144(0.034)	0.226(0.015)	0.306(0.038)
walker2d-expert	0.101 (0.02)	0.069(0.025)	0.212(0.043)	0.305(0.038)
walker2d-medium-expert	0.144 (0.03)	0.317(0.076)	0.489(0.114)	0.135(0.032)
walker2d-medium	0.287 (0.038)	0.108(0.025)	0.475(0.063)	0.298(0.026)

Table 5: Design error of OTED using *geom-friction&actuator-gainprm*, *geom-friction&gravity* and *actuator-gainprm&gravity*. We observe similar design errors as learning all parameters.

H PERFORMANCE OF OTED-SAC WITH VARYING NUMBER OF SEEDS

We experimented with using [5, 10, 15, 20, 25] seeds to train an online agent to understand if using more seeds performs better and if our model selection approach gives improvements (see Table 6). We observe that using more than 5 seeds can be beneficial but we don't see a considerable difference when using more than 10 seeds. We also observe that our model selection method (denoted "w\") outperforms just aggregating all available seeds (denoted "w\o").

	5	10		15		20		25	
		w/	w/o	w/	w/o	w/	w/o	w/	w/o
halfcheetah-expert	111.0 (1.7)	109.4 (4.9)	106.1 (3.1)	117.6 (1.4)	112.1 (1.9)	118.4 (3.8)	109.6 (2.0)	**119.3 (2.8)**	108.8 (2.2)
halfcheetah-medium-expert	109.0 (3.2)	113.6 (1.6)	108.4 (2.7)	**114.2 (3.3)**	105.8 (2.1)	111.5 (2.7)	102.6 (2.1)	111.0 (2.2)	104.9 (1.2)
halfcheetah-medium	111.5 (2.2)	112.7 (1.1)	107.9 (1.9)	116.6 (2.4)	111.1 (1.5)	115.2 (2.0)	108.7 (1.8)	**119.0 (2.5)**	108.3 (1.6)
hopper-expert	45.8 (4.7)	**49.0 (4.0)**	48.3 (2.9)	48.2 (4.9)	51.7 (4.7)	42.9 (1.8)	45.4 (1.9)	41.2 (1.7)	45.4 (3.1)
hopper-medium-expert	60.0 (4.4)	54.1 (3.5)	54.9 (4.7)	**81.9 (10.5)**	94.4 (5.6)	66.6 (6.9)	73.7 (5.6)	63.4 (8.7)	73.8 (4.2)
hopper-medium	109.0 (2.7)	**112.3 (0.4)**	107.4 (3.7)	109.8 (2.6)	104.8 (4.0)	93.7 (11.5)	98.1 (5.5)	97.9 (13.5)	106.7 (3.5)
walker2d-expert	96.6 (17.8)	121.5 (4.7)	102.5 (9.6)	131.0 (2.5)	99.4 (10.5)	**133.3 (3.0)**	94.8 (9.1)	131.5 (1.9)	100.8 (7.4)
walker2d-medium-expert	51.4 (21.2)	**131.5 (2.8)**	98.4 (13.8)	121.3 (2.8)	88.0 (11.8)	128.9 (3.9)	105.7 (6.4)	118.3 (11.7)	100.9 (6.7)
walker2d-medium	93.8 (18.7)	98.8 (15.4)	79.0 (13.6)	**126.3 (3.6)**	89.7 (10.9)	117.5 (6.2)	83.8 (8.2)	109.2 (13.1)	88.7 (7.4)

Table 6: Performance of OTED-SAC using different number of experiments. w\ and w\o correspond to with and without our proposed model selection, respectively. The proposed model selection gives overall better results.

I SENSITIVITY OF ONLINE POLICY WITH SIMULATOR INACCURACIES

It is important that we understand the sensitivity of training an online policy in an inaccurate simulator to understand the benefit of tuning a simulator better. For this, we performed controlled experiments where we sampled simulator parameters from a manually curated grid and trained online policy for every parameter combination. More specifically, let's define a delta matrix as

$$\Delta = \begin{bmatrix} -0.5 & 0.0 & 0.5 \\ -0.2 & 0.0 & 0.2 \\ -3.27 & 0.0 & 3.27 \end{bmatrix} \tag{6}$$

where columns correspond to parameter values and rows correspond to geom-friction, actuator-prm, and gravity parameters, respectively. We sample a delta from each row and set the simulator parameter as ground truth parameter plus the delta. This allows us to measure the performance of an online policy with varying design errors. We show that even small errors in the simulator can give a considerable drop in online policy performance (see Figure 11). When all parameters are set incorrectly, we see that the policy doesn't learn at all. This motivates the need for tuning simulator parameters for better policy performance.

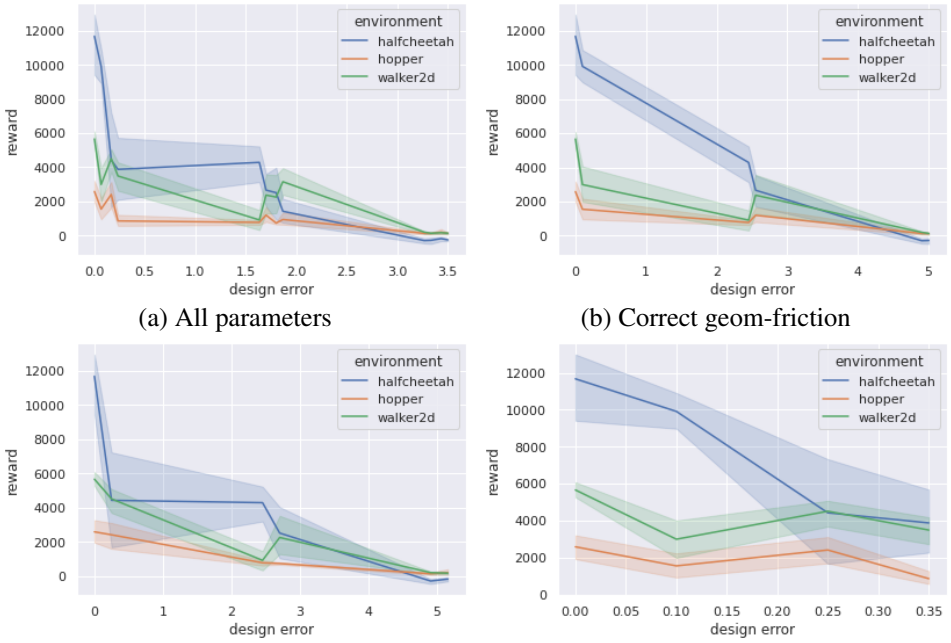
J MODEL SELECTION WITH DESIGN DEVIATION

We would like to first clarify that all model selection criteria in Table 7 are introduced by our work and they are not alternatives to OTED but merely different ways of choosing a designer policy. All these criteria require a well trained OTED and they can be collected with no additional cost.

We ran additional experiments where we chose designers using design deviation criteria instead of log ratios. We observe similar performance between the two (see Table 7). But, we want to point out a significant drawback of design deviation; it is agnostic to any shifts in design distribution. For example, take two different design distributions where means are the same but one has a larger variance. In this case, the design deviation would choose the distribution with lower variance. Now, if we shift the chosen distribution by a large constant where mean gets farther away from the ground truth, design deviation would still choose this distribution as it still has lower variance. In contrast, log ratios is a more principled metric that we aim to optimize in Eq. 3 to train OTED.

K ADDITIONAL EXPERIMENTS ON MINIGRID

In Figure 12, we present the performance of behavior policy in MiniGrid environments. We observe that the behavior policy is very close to the unknown target policy. When the simulator is inaccurate and target parameter can't be perfectly learned (target parameter 0.26 is not within the space of simulator parameters), we observe higher divergence and variance.



(a) All parameters (b) Correct geom-friction

(c) Correct actuator-gainprm (d) Correct gravity

Figure 11: Performance of online policy with varying design errors sampled from a grid. Even small errors can lead to drops in performance (a). We observe a similar trend when one of the parameters is set to the ground truth value (b,c,d).

	Design Deviation	Log Ratios (OTED)
halfcheetah-expert	111.2 (1.8)	111.0 (1.7)
halfcheetah-medium-expert	105.0 (1.7)	109.0 (3.2)
halfcheetah-medium	108.2 (2.7)	111.5 (2.2)
hopper-expert	27.1 (4.2)	45.8 (4.7)
hopper-medium-expert	84.9 (6.4)	60.0 (4.4)
hopper-medium	95.1 (7.8)	109.0 (2.7)
walker2d-expert	106.8 (9.4)	96.6 (17.8)
walker2d-medium-expert	99.7 (9.5)	51.4 (21.2)
walker2d-medium	83.8 (9.9)	93.8 (18.7)

Table 7: Performance of OTED-SAC with different model selection methods. Design deviation also gives very strong results, outperforming log ratios on Walker2d *medium-expert* data. Log ratios gives better results on Hopper *expert*. For a fair comparison, we report performance with 5 seeds.

L LIMITATIONS

OTED has several limitations: (i) A simulator needs to have an interface with a sufficient coverage over parameters, (ii) online data in the simulator should be diverse enough for the discriminator to train well, and (iii) complexity of the parameter space can adversely affect OTED's performance. In our experiments, we showed that setting hidden parameters incorrectly can degrade OTED's performance. Our initial empirical examinations also showed that if online data is not diverse enough, the discriminator overfits which gives poor log ratio estimations. We increase diversity of online episodes by truncating them early while keeping the size of the replay the same.

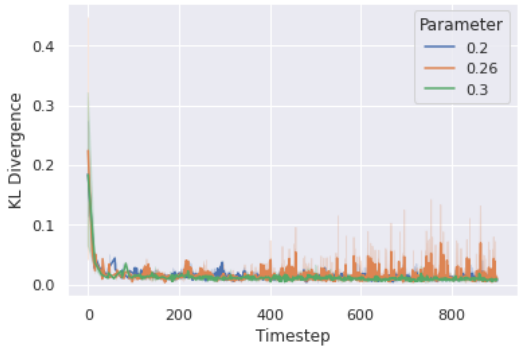


Figure 12: KL-divergence between behavior policy and target ground truth policy in MiniGrid.