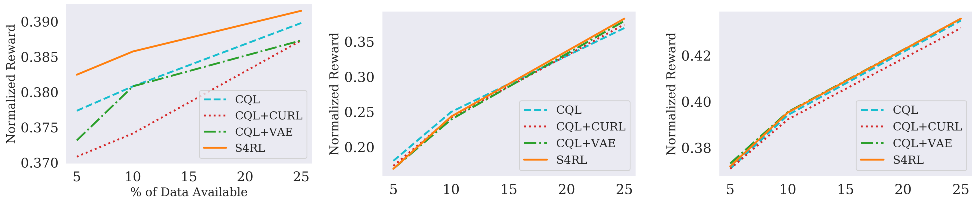
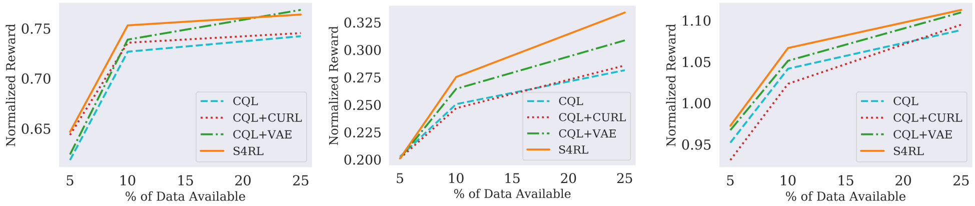
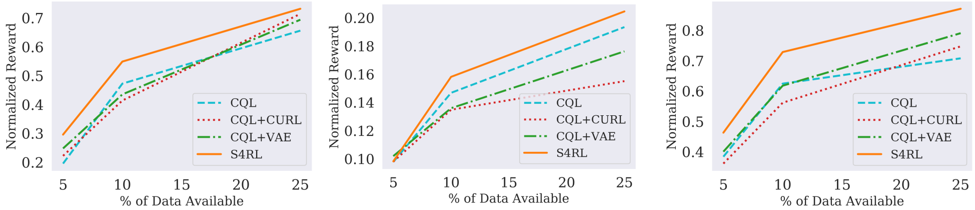


(a) cheetah-medium **(b)** cheetah-medium-replay **(c)** cheetah-medium-expert



(d) hopper-medium **(e)** hopper-medium-replay **(f)** hopper-medium-expert



(g) walker2d-medium **(h)** walker2d-medium-replay **(i)** walker2d-medium-expert

Figure 5: Learning from limited data. We train an S4RL agent and the baselines on a small 5%, 10%, and 25% of the different D4RL data splits. We see that S4RL is consistently the best performing algorithm when learning from limited data.

A Learning with limited data

Its possible that only a small amount of offline data is available to train an offline agent, which motivates us to test the agents on learning from limited data. We do this by randomly selecting 5%, 10%, and 25% of the datasets in each data split for the 3 OpenAI Gym tasks on 3 different data splits: "-medium", "-medium-replay" and "-medium-expert". By doing so, we significantly limit the amount of information the agent has about the environment thereby increasing the difficulty of learning a good representation from data. We present the normalized reward (y-axis) over the different data % available (x-axis) in Figure 5. On the Hopper and Walker2d data splits, the S4RL-$\mathcal{N}$ agent is able to learn significantly outperform all the baselines on both environments in each data split and data-percent. Despite the simplicity and ease of implementation, the agent is able to learn better representations than CQL+CURL and CQL+VAE.

B Online RL with S4RL

	SAC	SAC+S4RL-$\mathcal{N}$	SAC+S4RL-Adv
HalfCheetah-v2	9302	**11039**	10483
Hopper-v2	3341	**3891**	3614
Walker2d-v2	3620	**4367**	4007

Table 3: Results on online RL over 3 benchmark OpenAI Gym environments

We further experiment with online RL using a base SAC agent [17] on 3 benchmark OpenAI Gym environments (the same ones included in the D4RL benchmark [4]). We report the episodic reward after 1M timesteps in Table 3, where we continue to see that S4RL significantly outperforms the baseline SAC agent.

C Full Robotic Experiments

Domain	**Task**	**Behaviour policy at 1M**	**CQL**	**CQL+CURL**	**CQL+VAE**	**CQL +S4RL-$\mathcal{N}$**	**CQL +S4RL-Adv**
MetaWorld	reach-v1	95%	77%	78%	79%	**84%**	**93%**
	reach-wall-v1	90%	54%	57%	67%	**78%**	**88%**
	sweep-into-v1	75%	32%	33%	41%	**63%**	**79%**
	door-close-v1	35%	21%	25%	**30%**	**30%**	**39%**
	push-v1	30%	12%	12%	18%	**27%**	**35%**
RoboSuite	block-lifting	-	28%	28%	37%	**45%**	**53%**
	pick-and-place-can	-	20%	16%	29%	33%	**44%**

Table 4: Experiments on robotic manipulation in the Metaworld environment [1]. We report the % goals completed by the behaviour policy at 1M steps for MetaWorld environments. When the offline policy outperforms the behaviour policy, we highlight that in **blue text** and when the S4RL agent outperforms the baselines we highlight it in **bold**. These are the numerical results used in Figure 4.

D Q-network regularization

Task	**CQL**	**CQL +Dropout**	**CQL +L2-Reg**	**CQL +S4RL-Adv**
reach-v1	77%	68%	79%	**93%**
reach-wall-v1	54%	43%	51%	**88%**
sweep-into-v1	32%	22%	34%	**79%**
door-close-v1	21%	18%	21%	**39%**
push-v1	12%	4%	9%	**35%**

Table 5: Effect of different regularization schemes on the performance over 5 difficult robotic tasks in the MetaWorld environments [1]. We use $p = 0.3$ for Dropout and weight decay value of 1×10^{-4} for L2-Regularization.

Along with self-supervision, in deep learning literature there exist other forms of regularization methods for training neural networks. We include experiments with two such methods: Monte Carlo dropout mask (MC-Dropout) [43] and L2-regularization over Q-network weights. The results are shown in Table 5.

E Ablation over Hyperparamaters

The results for ablation over values of σ and ϵ for the two best performing methods (S4RL-$\mathcal{N}$ and S4RL-Adv) are presented in Table 6. We see that large values of ϵ or σ (10^{-2})significantly hurt the performance of S4RL, whereas the model is relatively stable within the range of 10^{-3} and 10^{-4}

Task	**CQL+S4RL-$\mathcal{N}$** $(\sigma = 10^{-2})$	**CQL+S4RL-$\mathcal{N}$** $(\sigma = 10^{-3})$	**CQL+S4RL-$\mathcal{N}$** $(\epsilon = 10^{-4})$	**CQL+S4RL-Adv** $(\epsilon = 10^{-2})$	**CQL+S4RL-Adv** $(\epsilon = 10^{-3})$	**CQL+S4RL-Adv** $(\epsilon = 10^{-4})$
reach-v1	53%	80%	**85%**	50%	88%	**93%**
reach-wall-v1	49%	70%	**72%**	44%	79%	**88%**
sweep-into-v1	47%	**64%**	61%	36%	65%	**79%**
door-close-v1	8%	23%	**26%**	19%	32%	**39%**
push-v1	5%	22%	**27%**	13%	27%	**35%**

Table 6: Hyperparameter study on the Metaworld environment [1]. We see that S4RL performs best when only local perturbations are applied to the states. As noted in Section 5, we use $\sigma = 3 \times 10^{-4}$ and $\epsilon = 1 \times 10^{-4}$ for S4RL-$\mathcal{N}$ and S4RL-Adv, respectively.

suggesting the need for local perturbations to the proprioceptive information. A large variance or a large adversarial sample hurts the performance since the new states $T(s_t)$ may be semantically different than the original state, since proprioceptive inputs are continuous unlike pixel-based input.

F Implementation details on state-switch

The implementation of the state-switch experiments is done by using ad-hoc rules for each environment where we only replace dimensions which are similar to each other. Such as a joint-angle is only replaced by another angle, and a joint velocity is only replaced by another velocity. Using this scheme ensures that we remain approximately in the same bounds of what is physically realizable, since its possible that velocities and angles do not work on the same scale. Since state-switch needs significantly more oracle knowledge of the environment and the state space, it may not be the best choice of augmentation for environments where the state space is not known. Furthermore, we see that in practice, state-switch is unable to perform well on most baselines as shown in Table 1, which consolidates the relative ineffectiveness of the augmentation choice.

G Effect of network width and performance

	network width=256		**network width=512**		**network width=1024**	
	CQL	**CQL+S4RL-Adv**	**CQL**	**CQL+S4RL-Adv**	**CQL**	**CQL+S4RL-Adv**
push	12%	**35%**	11%	**33%**	4%	**33%**
door-close	21%	**39%**	19%	**37%**	12%	**36%**
sweep-into	32%	**79%**	27%	**74%**	11%	**75%**

Table 7: Effect of network width of the Q-functions and its effect on performance on the MetaWorld suite of tasks.

Recent work has tried to shown a relation between the network width and its effect on overfitting, showing that as the network width increases, the vanilla Q-learning methods start to overfit [15]. We show similar results in offline RL in Table 7, where we show how similar effect can be seen with the base CQL algorithm, but using S4RL-Adv, we are able to greatly reduce the performance degradation as the network width is increased. We note that there is no general consensus on overfitting and how to measure robustness to overfitting in the community. We simply add these additional experiments, in accordance with the recent literature, as done by [15], without concluding that S4RL prevents overfitting to the offline data.

H Measurement noise

We add additional results in measurement model of the robot. During testing, we randomly perturb the states by using an additive Gaussian noise of the form $\mathcal{N}(0, 0.001)$. The results are reported in Table 8. This experiment further shows that using S4RL, we are able to learn policies that are more robust to measurement noise when deployed in the environment.

	CQL	**CQL+S4RL-$\mathcal{N}$**	**CQL+S4RL-Adv**
push	49%	69%	**73%**
door-close	38%	65%	**67%**
sweep-into	21%	50%	**59%**
reach-wall	15%	19%	**28%**
reach	10%	13%	**22%**

Table 8: Experiments with measurement noise during evaluation. We add an additive Gaussian noise during evaluation. The noise is randomly sampled from $\mathcal{N}(0, 0.001)$ and added to the states input to the policy.

I Details about Robosuite experiments

Robosuite [2] datasets were collected by first training a Soft Actor Critic [44] RL agent from scratch on the Lift and Can tasks. Agent checkpoints were saved regularly during training – 5 checkpoints for the Lift task, and 17 checkpoints for the Can task. For each checkpoint, 300 agent rollouts were collected (with horizon 150) for a total of 1500 Lift trajectories and 5100 Can trajectories. Consequently, these datasets contain a mixture of expert and suboptimal trajectories, and resemble datasets from common offline RL benchmarks [4, 9].