Appendix

We divide the appendix into four different sections following the results section. Each section additionally provides hyper-parameters used for IPL in that section. The first section, setup, contains details information on the experimental setup and hyper-parameters used. The second section on benchmark results gives full learning curves for the experiments in Section 4.2. The third section provides full learning curves for the MetaWorld and Data-scaling experiments. The final Appendix section provides extended ablations.

A Setup

Here we provide the full algorithmic outline of IPL using Implicit Q-Learning [27] that mimics our implementation. While in practice the policy π could be extracted at the end of training, we do it simultaneously as in [27] in order to construct learning curves.

Algorithm 2: IPL Algorithm (IQL Variant)

Input : $\mathcal{D}_p$, $\mathcal{D}_o$, λ, α
for $i = 1, 2, 3, \ldots$ **do**
  Sample batches $B_p \sim \mathcal{D}_p, B_o \sim \mathcal{D}_o$
  Update Q: $\min_Q \mathbb{E}_{B_p}[\mathcal{L}_p(Q)] + \lambda \mathbb{E}_{B_p \cup B_o}[\mathcal{L}_r(Q)]$
  Update V: $\min_V \mathbb{E}_{B_p \cup B_o}\left[|\tau - \mathbb{1}(Q(s,a) - V(s))|\,(Q(s,a) - V(s))^2\right]$
  Update π: $\max_\pi \mathbb{E}_{\mathcal{D}_p \cup \mathcal{D}_o}[e^{\beta(Q(s,a)-V(s))} \log \pi(a|s)]$

Note that above we write the temperature parameter β as done in IQL, instead of how it is usually done, using α in the denominator [18, 42].

When sampling batches of preference data $B_p \sim \mathcal{D}_p$, we take sub-samples of each segment σ of length s. For a sampled data point $(\sigma^{(1)}, \sigma^{(2)}, y)$, we sample start $\sim$ Unif$[0, 1, 2, \ldots k - s]$ and then let take $\sigma = s_{\text{start}}, a_{\text{start}}, \ldots, s_{\text{start}+s}$. We use the same start value across the entire batch.

Given that we run experiments using MLPs, all of our experiments were run on CPU compute resources. Each seed for each method requires one CPU core (two hyper-threads) and 8 Gb of memory.

B Benchmark Results

Here we provide details for our experiments on the preference-based RL benchmark from Kim et al. [25]. We use the same hyperparameters as Kim et al. [25] and Kostrikov et al. [27] where applicable as shown in Table 4.

Gym-Mujoco Locomotion. Hopper and Walker2D agents are tasked with learning locomotion policies from datasets of varying qualities taken from the D4RL [16] benchmark. Preference datasets were constructed by Kim et al. [25] by uniformly sampling queries and labeling a subset of them. For all locomotion tasks the segment length. Preference datasets for "medium" quality offline datasets contain 500 queries, while preference datasets for "expert" quality offline datasets contain 100 queries. Segment lengths $k = 100$ for all datasets, and were subsampled to length $s = 64$ by IPL and our MR (reimplementation). Evaluation was preformed over 10 episodes every 5000 steps. Full learning curves are shown in Fig. 3.

RoboMimic. The RoboMimic datasets contain interaction data of two types: ph — proficient human and mh – multihuman. The multi-human data was collected from human demonstrators of mixed quality. The robot is tasked with learning how to lift a cube (lift) or pick and place a can (can). Preference datasets were again taken directly from Kim et al. [25]. Preference datasets of size 100 with segment lengths $k = 50$, randomly sub-sampled to length $s = 32$ were used for the ph datasets. Preference datasets of size 500 with segment lengths $k = 100$, randomly sub-sampled to length $s = 64$ were used for the mh datasets. Evaluation was performed over 25 episodes every 50000 steps. Full learning curves are shown in Fig. 4.

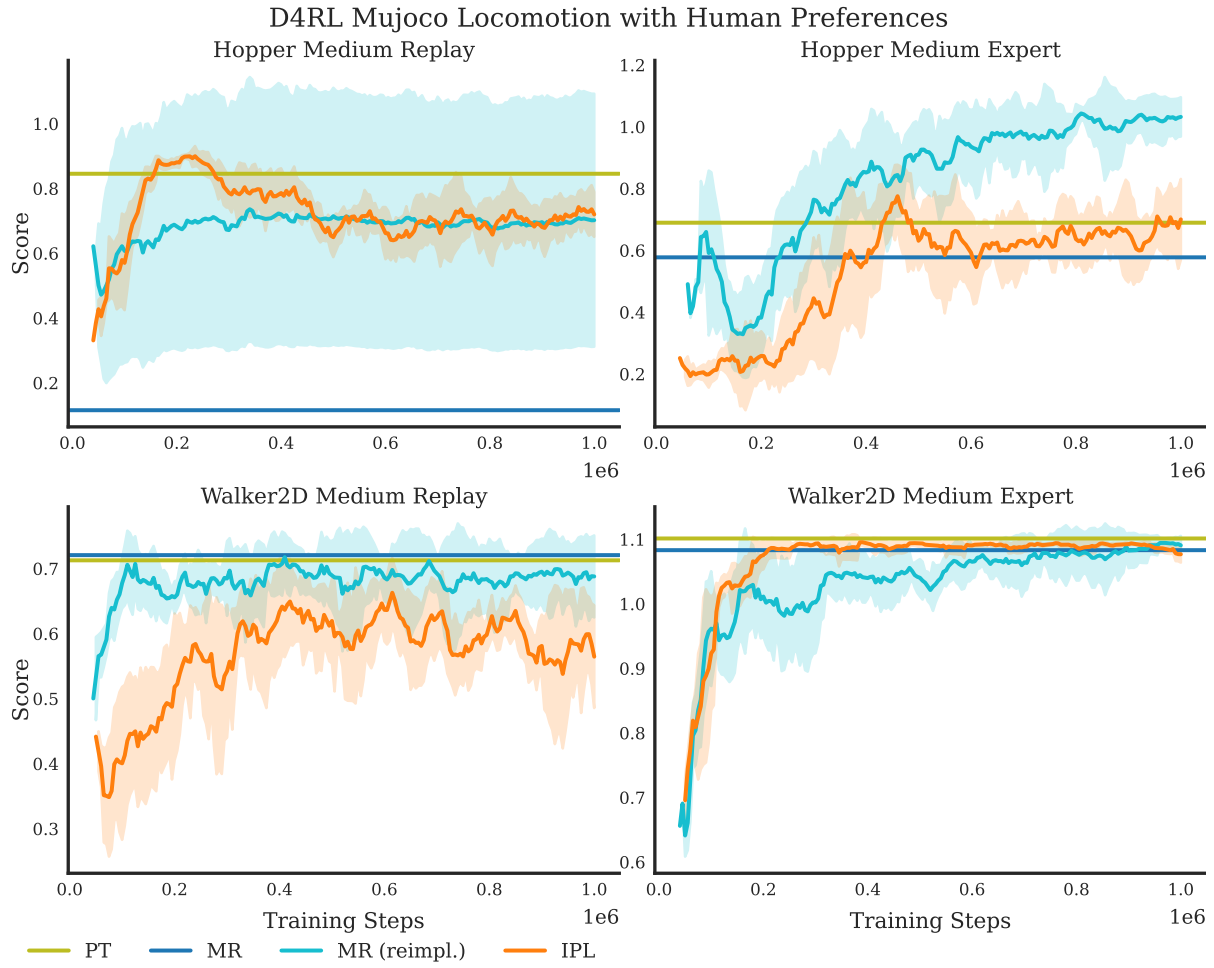

Figure 3: Full learning curves on the D4RL locomotion benchmark with human preferences.

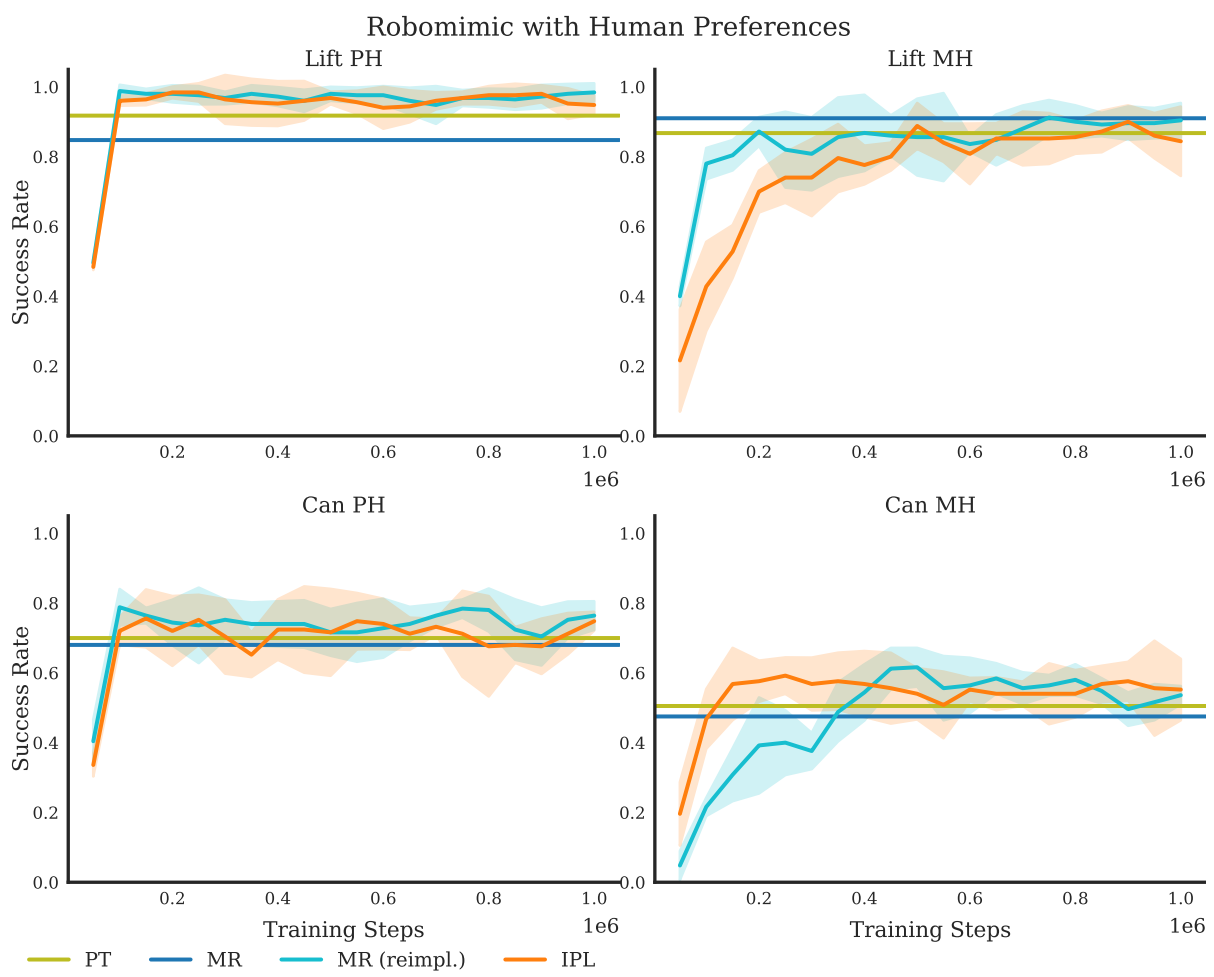

Figure 4: Full learning curves on the RoboMimic benchmark with human preferences.

Common Hyperparameters

Parameter	Locomotion	Robomimic
Q, V, π Arch	2x 256d	2x 256d
Learning Rate	0.0003	0.0003
Optimizer	Adam	Adam
β	3.0	0.5
τ	0.7	0.7
$\mathcal{D}_o$ Batch Size	256	256
$\mathcal{D}_p$ Batch Size	8	8
Training Steps	1 Mil	1 Mil
k	100	100, 50
Subsample s	64	64, 32

MR Hyperparameters

Parameter	Locomotion	Robomimic
r_θ Arch	2x 256d	2x 256d
r_θ LR	0.0003	0.0003
r_θ Optimizer	Adam	Adam
r_θ Steps	20k	20k

IPL Hyperparameters

Parameter	Locomotion	Robomimic
λ	0.5	4

Table 4: Hyperparameters used for the benchmark experiments. We can see that IPL has fewer hyperparameters.

Common Hyperparameters

Parameter	Value
Q, V, π Arch	3x 256d
Learning Rate	0.0003
Optimizer	Adam
β	4.0
τ	0.7
$\mathcal{D}_p$ Batch Size	16
Training Steps	200k
k	25
Subsample s	16

MR Hyperparameters

Parameter	Value
r_θ Arch	3x 256d
r_θ LR	0.0003
r_θ Optimizer	Adam
r_θ Steps	20k

IPL Hyperparameters

Parameter	Locomotion
λ	0.5

Table 5: Hyper-parameters used in the MetaWorld data scaling experiments.

C Data Scaling Results

Experiments for data scaling were conducted on the MetaWorld benchmark from Yu et al. [52]. Offline datasets for five different MetaWorld tasks were constructed as follows: Collect 100 trajectories of expert data on the target task using the built in ground truth policies with the addition of Gaussian noise of standard deviation 1.0. Collect 100 trajectories of sub-optimal data by running the ground-truth policy for a different randomization of the target task with Gaussian noise 1.0. Collect 100 trajectories of even more sub-optimal data by running the ground truth policy *of a different task* with Gaussian noise standard deviation 1.0 in the target domain. Finally, collect 100 trajectories with uniform random actions. As MetaWorld episodes are 500 steps long, this results in 200,000 time-steps of data. We then construct preference datasets by uniformly sampling segments from the offline dataset and assigning labels y according to $\sum_t r(s_t^{(1)}, a_t^{(1)}) > \sum_t r(s_t^{(2)}, a_t^{(2)})$ where r is the ground truth reward provided by metaworld. We then train using only the data from $\mathcal{D}$ General architecture hyper-parameters were taken from Lee et al. [30], Hejna and Sadigh [21] which also use the MetaWorld benchmark, but for online preference-based RL. Full-hyper parameters are shown in Table 5. We run 20 evaluation episodes every 2500 steps. Full learning curves are shown in Fig. 5. When reporting values in Table 2, we choose the maximum point on the learning curves which average across five seeds. This provides results as if early stopping was given by an oracle, which is less optimistic than averaging the maximum of each seed as done in Mandlekar et al. [34].

D Ablations

In this section we provide additional ablations on both the benchmark datasets and MetaWorld datasets. We keep the hyperparameters the same, except for the parameter-efficient experiments. We run hyper-parameter sensitivty results for the human-preference benchmark datasets in Fig. 6. The top row depicts the sensitivity for IPL to the value of λ. The bottom row depicts the sensitivity of MR to the number of timesteps the reward function is trained for.

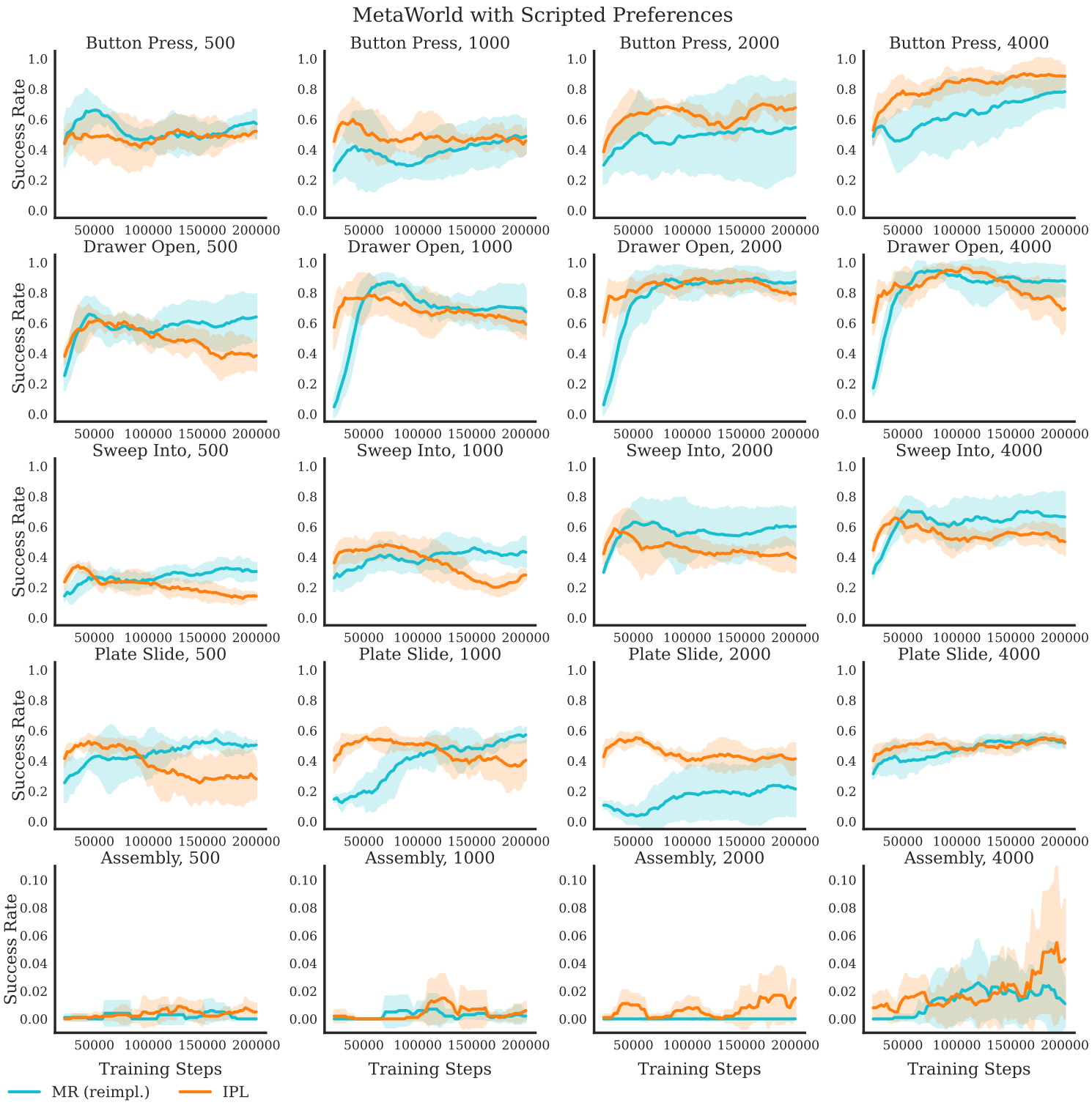

Figure 5: Full learning curves for the MetaWorld data scaling results with scripted preferences.

For the parameter-efficient experiments *only* we use an efficient version of IPL based on AWAC [37] to additionally remove the value network. The outline of this variant is given below

Algorithm 3: IPL Algorithm (AWAC Variant)

Input : $\mathcal{D}_p$, $\mathcal{D}_o$, λ, α
for $i = 1, 2, 3, \ldots$ **do**
    Sample batches $B_p \sim \mathcal{D}_p, B_o \sim \mathcal{D}_o$
    Estimate V as $Q(s, \pi(s))$
    Update Q: $\min_Q \mathbb{E}_{B_p}[\mathcal{L}_p(Q)] + \lambda \mathbb{E}_{B_p \cup B_o}[\mathcal{L}_r(Q)]$
    Update π: $\max_\pi \mathbb{E}_{\mathcal{D}_p \cup \mathcal{D}_o}[e^{\beta(Q(s,a) - Q(s,\pi(s)))} \log \pi(a|s)]$

For this version of IPL, we use $\lambda = 0.5$. All other hyper-parameters remain the same as in Table 6 except the architectures. For the parameter-efficiency experiments only we use MLPs consisting of two dense layers with either dimension 64 or dimension 35. Running MR with a two-layer MLP of dimension 35 has almost exactly the same number of parameters as IPL-AWAC with two-layer MLPs of dimension 64. We include full results for the parameter-efficiency experiments in Table 6. We find that on Drawer Open and Sweep Into, IPL outperforms both MR (64) and MR (35). In these environments, performance increases from MR (35) to MR (64) indicating that the expressiveness of the Q-function and policy are limiting performance. For the same budget, IPL is able to perform

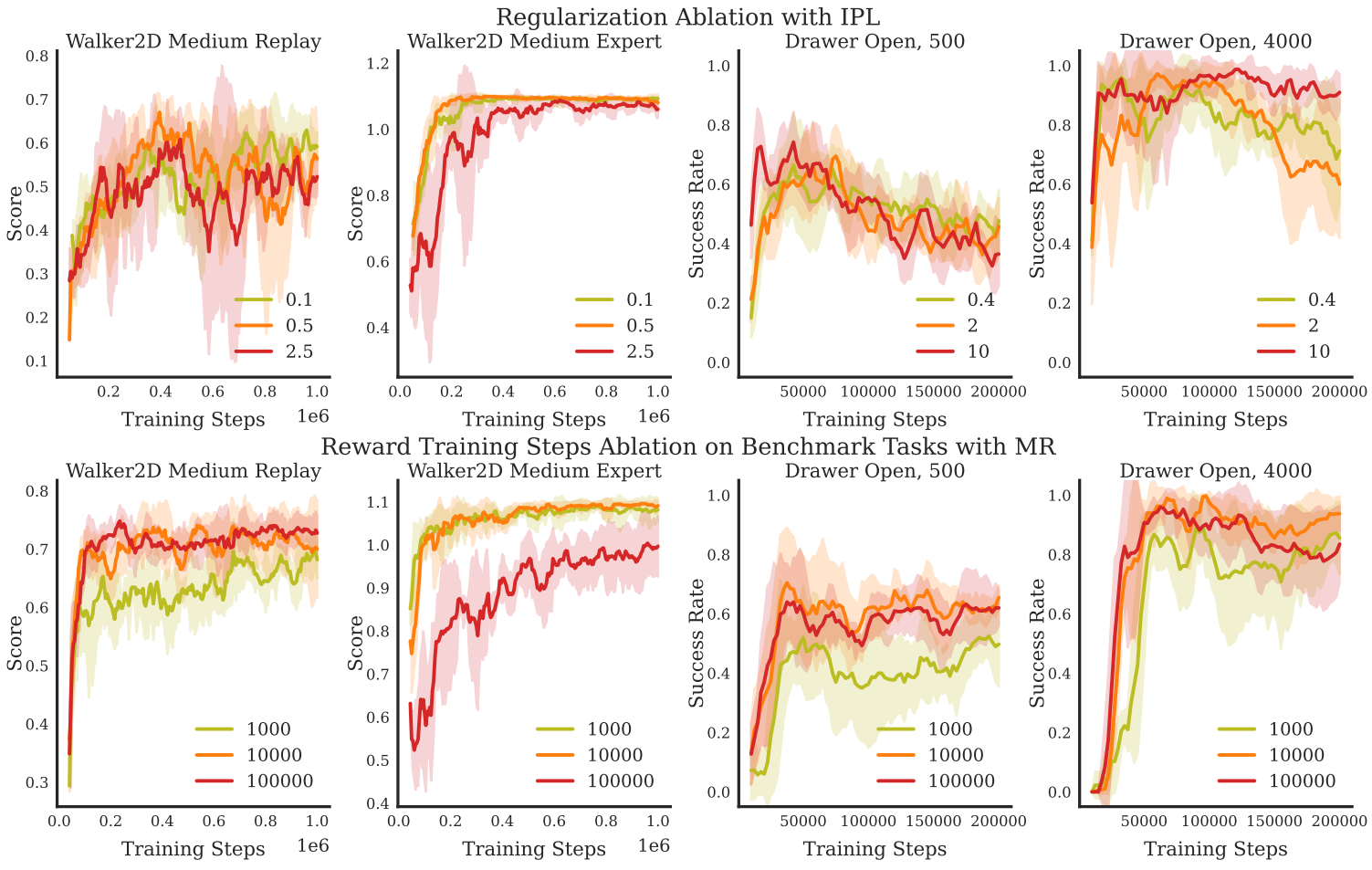

Figure 6: Ablations on regularization strength λ for IPL (top row) and the number of reward steps for MR (bottom row). We see that IPL is relatively consistent across different values of λ. MR on the other hand, can vary greatly if the reward function under or over fits. In Walker2D Medium Replay and Drawer Open, 500, we see that it can easily under-fit. In Walker2D Medium Expert it easily over-fits.

better. In Button Press, the simplest task, we find that MR (64) actually over-fits more than MR (35) and MR (64) ends up performing worse. In Plate Slide, all methods perform similarly independent of parameter count. We omit Assembly because of its low success rate at all data scales.

Preference Queries		500	1000	2000	4000
Button Press	MR (35)	**73.9 ± 8.9**	**86.8 ± 8.2**	**89.9 ± 14.4**	**99.0 ± 1.0**
	MR (64)	54.2 ± 16.1	42.6 ± 33.0	67.1 ± 14.9	43.4 ± 7.4
	IPL (64)	65.8 ± 13.3	79.8 ± 18.1	80.0 ± 17.3	**95.8 ± 5.2**
Drawer Open	MR (35)	13.4 ± 13.9	12.6 ± 21.9	15.5 ± 20.1	18.4 ± 25.6
	MR (64)	13.4 ± 19.0	57.1 ± 31.2	54.5 ± 31.7	78.8 ± 12.2
	IPL (64)	**89.8 ± 11.3**	**93.2 ± 2.5**	**99.5 ± 0.9**	**95.5 ± 3.7**
Sweep Into	MR (35)	35.1 ± 8.9	42.4 ± 9.9	45.9 ± 9.6	35.9 ± 4.1
	MR (64)	31.1 ± 6.4	55.8 ± 5.9	49.6 ± 10.3	56.4 ± 10.3
	IPL (64)	**41.1 ± 14.2**	**63.9 ± 8.0**	**65.0 ± 12.0**	**63.9 ± 11.8**
Plate Slide	MR (35)	**55.2 ± 6.1**	**51.1 ± 4.4**	**53.0 ± 2.0**	**48.9 ± 3.3**
	MR (64)	46.6 ± 21.9	**50.8 ± 0.6**	47.0 ± 2.5	**48.5 ± 4.6**
	IPL (64)	**54.9 ± 3.2**	**49.4 ± 1.6**	45.2 ± 9.0	**48.8 ± 4.9**

Table 6: Performance of different methods on the MetaWorld tasks under a limited parameter budget. MR (35) and IPL (64) have the same number of parameters. The Assembly task is ommited due to low success rate. On Button Press, fewer parameters appears to perform better as, due to the simplicity of the task, its easier for the bigger models to overfit. On Drawer Open and Sweep Into, we see consistent gains from increasing the number of parameters in the network, and IPL performs best overall. On the Plate Slide task, all methods at different parameter scales perform similarly.