Appendix

A. SpriteWorld Experimental Details

A.1. Algoritmic Details

The input to our reward function for all experiments in this domain is a 80×80 RGB image, with an output space of 400 in the underlying MDP state space. We parameterize the reward function for all methods starting from the same base learner whose architecture we summarize in Table 2.

Our LSTM (Hochreiter & Schmidhuber, 1997) implementation is based on the variant used in Zaremba et al. (2014). The input to the LSTM at each time step is the location of the agent, embedded as the (x, y)-coordinates. This is used to predict an spatial map fed as input to the base CNN. We also experimented with conditioning the initial hidden state on image features from a separate CNN, but found that this did not improve performance.

In our demo conditional model, we preserve the spatial information of the demonstrations by feeding in the state visitation map as a image-grid, upsampled with bi-linear interpolation, as an additional channel to the image. In our setup, both the demo-conditional models share the same convolutional architecture, but differ only in how they encode condition on the demonstrations.

For all our methods, we optimized our model with Adam (Kingma & Ba, 2014). We tuned over the learning rate α, the inner learning rate β and ℓ_2 weight decay on the initial parameters. We initialize our models with the Glorot initialization (Glorot & Bengio, 2010). In our LSTM learner, we tuned over embedding sizes and dimensionality. A negative result we found was that bias transformation (Finn et al., 2017b) did not help in our experimental setting.

Table 2. Hyperparameter summary on Spriteworld environment. Curly brackets indicate the parameter was chosen from that set.

Hyperparameters	Value
Architecture	Conv$(256 - 8 \times 8 - 2)$ Conv$(128 - 4 \times 4 - 2)$ Conv$(64 - 3 \times 3 - 1)$ Conv$(64 - 3 \times 3 - 1)$ Conv$(1 - 1 \times 1 - 1)$
Learning rate α	$\{0.0001, 0.00001\}$
Inner learning rate β	$\{0.001, 0.0005\}$
Weight decay ℓ_2	$\{0, 0.0001\}$
Inner gradient steps	$\{1, 3\}$
Max meta-test gradient steps	$\{20\}$
LSTM hidden dimension	$\{128, 256\}$
LSTM embedding sizes	$\{64, 128\}$
Batch size	16
Total meta-training environments	1000
Total meta-val/test environments	32
Maximum horizon (T)	15

A.2. Environment Details

The underlying MDP structure of SpriteWorld is a grid, where the states are each of the grid cells, and the actions enable the agent to move to any one of its 8-connected neighbors. The task visuals are inspired by Starcraft (e.g. (Synnaeve et al., 2016)), although we do not use the game engine. The sprites in our environment are extracted directly from the StarCraft files. We used in total 100 random units for meta-training. Evaluation on new objects was performed with 5 randomly selected sprites. For computational efficiency, we create a meta-training set of 1000 tasks and cache the optimal policy and state visitations under the true cost. Our evaluation is over 32 tasks. Our set of sprites was divided into two categories: buildings and characters. Each characters had multiple poses (taken from different frames of animation, such as walking/running/flying), whereas buildings only had a single pose. During meta-training the units were randomly placed, but to avoid the possibility that the agent would not need to actively avoid obstacles, the units were placed away from the boundary of the image in both the meta-validation and meta-test set.

The terrain in each environment was randomly generated using a set of tiles, each belonging to a specific category (e.g. grass, dirt, water). For each tile, we also specified a set of possible tiles for each of the 4-neighbors. Using these constraints on the neighbors, we generated random environment terrains using a graph traversal algorithm, where successor tiles were sampled randomly from this set of possible tiles. This process resulted in randomly generated, seamless environments. The expert demonstrations were generated using a cost (negative reward) of 8 for the obstacles, 2 for any grass tile, and 1 for any dirt tile. The names of the units used in our experiments are as follows (names are from the original game files):

The list of buildings used is: academy, assim, barrack, beacon, cerebrat, chemlab, chrysal, cocoon, comsat, control, depot, drydock, egg, extract, factory, fcolony, forge, gateway, genelab, geyser, hatchery, hive, infest, lair, larva, mutapit, nest, nexus, nukesilo, nydustpit, overlord, physics, probe, pylon, prism, pillbox, queen, rcluster, refinery, research, robotic, sbattery, scolony, spire, starbase, stargate, starport, temple, warm, weaponpl, wessel.

The list of characters used is: acritter, arbiter, archives, archon, avenger, battlecr, brood, bugguy, carrier, civilian, defiler, dragoon, drone, dropship, firebat, gencore, ghost, guardian, hydra, intercep, jcritter, lurker, marine, missile, mutacham, mutalid, sapper, scout, scv, shuttle, snakey, spider, stank, tank, templar, trilob, ucereb, uikerr, ultra, vulture, witness, zealot, zergling.

B. SUNCG Experimental Details

B.1. Algorithmic Details

Table 3. Hyperparamters on the SUNCG environment. Curly brackets indicate that the the parameter was chosen from that set.

Hyperparameters	Value

Architecture	Conv($16 - 5 \times 5 - 1$)
	Conv($32 - 3 \times 3 - 1$)
	MLP(32)
	MLP(1)

Max number of training steps	15000000
Number of seed	3
Learning rate α	$\{0.1, 0.01, 0.001, 0.0001\}$
Inner learning rate β	$\{0.15, 0.1, 0.01, 0.0001\}$
Inner gradient steps	$\{3, 5\}$
Max meta-test gradient steps	$\{10\}$
Momentum	$\{0.9, 0.95, 0.99\}$

Our per task MaxEnt IRL baseline is learned by using the same base architecture. To provide a fair comparison, we do not use an inner learning rule in the inner loop of ManDRIL such as Adam (Kingma & Ba, 2014) and use regular SGD. For our baseline however, we include a momentum term over which we tune. We tune over the number of training steps, learning rate and momentum parameters. We use SGD with momentum. For ManDRIL, we tune over the inner learning rate β and learning rate α and number of gradient steps. At meta-test time, we experimented with taking up to 10 gradient steps. For pretraining IRL, we first train for 150,000 steps, freeze the weights, and fine tune them for every separate task. For training from scratch, we use the Glorot uniform initialization in the the convolutional layers (Glorot & Bengio, 2010).

B.2. Environment Details

Table 4. Summary of SUNCG environment setup.

Hyperparameters	Value
Discount (γ)	0.99
Maximum horizon (T)	40
Initial random steps	30
Number of demonstrations	5
Training environments	1004
Test environments	236
Test-house environments	173
(PICK/NAV) split:	716/697

The MDP in each environment is discretized into a grid where the state is defined by the grid coordinates plus the agent's orientation (N,S,E,W). The agent receives an observation which is a first-person panoramic view. The panoramic view consists of four 32×24 semantic image observations containing 61 channels.

The only departure for the task setup of Fu et al. (2019) that we make is to randomize the agent's start location by executing a random walk at the beginning of each episode. In Fu et al. (2019), the agent's start location was previously deterministic which allows a trivial solution of memorizing the provided demonstrations.

C. SpriteWorld Meta-Test Training Performance

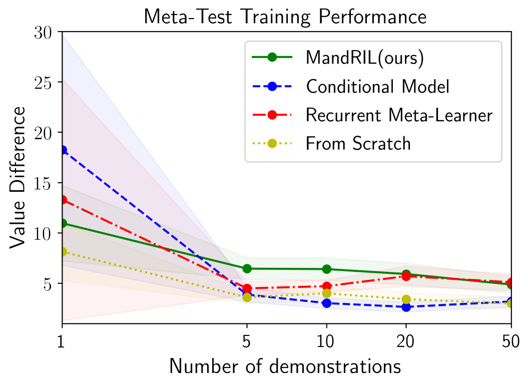

Figure 7. Meta-test "training" performance with varying numbers of demonstrations (lower is better). This is the performance on the environment for which demonstrations are provided for adaptation. As the number of demonstrations increase, all methods are able to perform well in terms of training performance as they can simply overfit to the training environment without acquiring the right visual cues that allow them to generalize. However, we find comes at the cost comes of considerable overfitting as we discuss in Section. 5.

D. SUNCG DAgger Performance

Table 5. DAgger success rate (%) on heldout tasks with 5 demonstrations. ManDRIL values are repeat for viewing convenience. Results are averaged over 3 random seeds.

Method	Test			Unseen Houses		
	Pick	Nav	Total	Pick	Nav	Total
DAgger	1.0	12.8	7.5	7.4	15.5	11.8
ManDRIL(ours)	**52.3**	**90.7**	**77.3**	**56.3**	**91.0**	**82.6**

Here we show the performance of DAgger (Ross et al., 2011), in the setting where the number of samples that is equal to the number of demonstrations. Overall, while DAgger slightly improves performance over behavioral cloning, the performance still lags significantly behind ManDRIL and other IRL methods.

E. Detailed Meta-Objective Derivation

We define the quality of reward function $r_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^k$ on task $\mathcal{T}$ with the MaxEnt IRL loss, $\mathcal{L}^{\mathcal{T}}_{\text{IRL}}(\boldsymbol{\theta})$, described in Section 4. The corresponding gradient is

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{IRL}}(\boldsymbol{\theta}) = \frac{\partial\, r_{\boldsymbol{\theta}}}{\partial\, \boldsymbol{\theta}} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}_\mathcal{T}}), \tag{9}$$

where $\partial\, r_{\boldsymbol{\theta}} / \partial\, \boldsymbol{\theta}$ is the $k \times |\mathcal{S}||\mathcal{A}|$-dimensional Jacobian matrix of the reward function $r_{\boldsymbol{\theta}}$ with respect to the parameters $\boldsymbol{\theta}$. Here, $\boldsymbol{\mu}_\tau \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is the vector of *state-action visitations* under the trajectory τ (i.e. the vector whose elements are 1 if the corresponding state-action pair has been visited by the trajectory τ, and 0 otherwise), and $\boldsymbol{\mu}_{\mathcal{D}_\mathcal{T}} = \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{\tau \in \mathcal{D}_\mathcal{T}} \boldsymbol{\mu}_\tau$ is the mean state visitations over all demonstrated trajectories in $\mathcal{D}_\mathcal{T}$. Let $\boldsymbol{\phi}_\mathcal{T} \in \mathbb{R}^k$ be the updated parameters after a single gradient step. Then

$$\boldsymbol{\phi}_\mathcal{T} = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{tr}}_{\mathcal{T}}(\boldsymbol{\theta}). \tag{10}$$

Let $\mathcal{L}^{\text{test}}_{\mathcal{T}}$ be the MaxEnt IRL loss, where the expectation over trajectories is computed with respect to a test set that is *disjoint* from the set of demonstrations used to compute $\mathcal{L}^{\text{test}}_{\mathcal{T}}(\boldsymbol{\theta})$ in Eq. 10. We seek to minimize

$$\sum_{\mathcal{T} \in \mathcal{T}^{\text{test}}} \mathcal{L}^{\text{test}}_{\mathcal{T}}(\boldsymbol{\phi}_\mathcal{T}) \tag{11}$$

over the parameters $\boldsymbol{\theta}$. To do so, we first compute the gradient of Eq. 11, which we derive here. Applying the chain rule

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{test}}_{\mathcal{T}} &= \frac{\partial\, \boldsymbol{\phi}_\mathcal{T}}{\partial\, \boldsymbol{\theta}} \frac{\partial\, r_{\boldsymbol{\phi}_\mathcal{T}}}{\partial\, \boldsymbol{\phi}_\mathcal{T}} \frac{\partial\, \mathcal{L}^{\text{test}}_{\mathcal{T}}}{\partial\, r_{\boldsymbol{\phi}_\mathcal{T}}} \\
&= \frac{\partial}{\partial\, \boldsymbol{\theta}} \left(\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{tr}}_{\mathcal{T}}(\boldsymbol{\theta})\right) \frac{\partial\, r_{\boldsymbol{\phi}_\mathcal{T}}}{\partial\, \boldsymbol{\phi}_\mathcal{T}} \frac{\partial\, \mathcal{L}^{\text{test}}_{\mathcal{T}}}{\partial\, r_{\boldsymbol{\phi}_\mathcal{T}}} \\
&= \left(\mathbf{I} - \alpha \frac{\partial}{\partial\, \boldsymbol{\theta}} \left(\frac{\partial\, r_{\boldsymbol{\theta}}}{\partial\, \boldsymbol{\theta}} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}_\mathcal{T}})\right)\right) \frac{\partial\, r_{\boldsymbol{\phi}_\mathcal{T}}}{\partial\, \boldsymbol{\phi}_\mathcal{T}} \frac{\partial\, \mathcal{L}^{\text{test}}_{\mathcal{T}}}{\partial\, r_{\boldsymbol{\phi}_\mathcal{T}}}
\end{aligned} \tag{12}$$

where in the last equation we substitute in the gradient of the MaxEnt IRL loss in Eq. 9 for $\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{tr}}_{\mathcal{T}}(\boldsymbol{\theta})$. In Eq. 12, we use the following notation:

- $\partial\, \boldsymbol{\phi}_\mathcal{T} / \partial\, \boldsymbol{\theta}$ denotes the $k \times k$-dimensional vector of partial derivatives $\partial\, \boldsymbol{\phi}_{\mathcal{T},i} / \partial\, \boldsymbol{\theta}_j$,
- $\partial\, r_{\boldsymbol{\phi}_\mathcal{T}} / \partial\, \boldsymbol{\phi}_\mathcal{T}$ denotes the $k \times |\mathcal{S}||\mathcal{A}|$-dimensional matrix of partial derivatives $\partial\, r_{\boldsymbol{\phi}_{\mathcal{T},i}} / \partial\, \boldsymbol{\phi}_{\mathcal{T},j}$,
- and, $\partial\, \mathcal{L}^{\text{test}}_{\mathcal{T}} / \partial\, r_{\boldsymbol{\phi}_\mathcal{T}}$ denotes the k-dimensional gradient vector of $\mathcal{L}^{\text{test}}_{\mathcal{T}}$ with respect to $r_{\boldsymbol{\phi}_\mathcal{T}}$.

We will now focus on the term inside of the parentheses in Eq. 12, which is a $k \times k$-dimensional matrix of partial derivatives.

$$\begin{aligned}
&\frac{\partial}{\partial\, \boldsymbol{\theta}} \left(\frac{\partial\, r_{\boldsymbol{\theta}}}{\partial\, \boldsymbol{\theta}} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}_\mathcal{T}})\right) \\
&= \sum_{i=1}^{|\mathcal{S}||\mathcal{A}|} \left[\frac{\partial^2\, r_{\boldsymbol{\theta}}}{\partial\, \boldsymbol{\theta}^2} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}_\mathcal{T}})_i + \frac{\partial}{\partial\, \boldsymbol{\theta}} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau])_i \left(\frac{\partial\, r_{\boldsymbol{\theta},i}}{\partial\, \boldsymbol{\theta}}\right)^\top \right] \\
&= \sum_{i=1}^{|\mathcal{S}||\mathcal{A}|} \left[\frac{\partial^2\, r_{\boldsymbol{\theta}}}{\partial\, \boldsymbol{\theta}^2} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}_\mathcal{T}})_i + \right. \\
&\left. \left(\frac{\partial\, r_{\boldsymbol{\theta},i}}{\partial\, \boldsymbol{\theta}}\right) \left(\frac{\partial}{\partial\, r_{\boldsymbol{\theta},i}} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau])_i\right) \left(\frac{\partial\, r_{\boldsymbol{\theta},i}}{\partial\, \boldsymbol{\theta}}\right)^\top \right]
\end{aligned}$$

where between the first and second lines, we apply the chain rule to expand the second term. In this expression, we make use of the following notation:

- $\partial^2\, r_{\boldsymbol{\theta}} / \partial\, \boldsymbol{\theta}^2$ denotes the $k \times |\mathcal{S}||\mathcal{A}|$-dimensional matrix of second-order partial derivatives of the form $\partial^2\, r_{\boldsymbol{\theta},i} / \partial\, \boldsymbol{\theta}_j^2$,
- $(\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}_\mathcal{T}})_i$ denotes the ith element of the $|\mathcal{S}||\mathcal{A}|$-dimensional vector $(\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}_\mathcal{T}})_i$,
- $\partial\, r_{\boldsymbol{\theta},i} / \partial\, \boldsymbol{\theta}$ denotes the k-dimensional matrix of partial derivatives of the form $\partial\, r_{\boldsymbol{\theta},i} / \partial\, \boldsymbol{\theta}_j$ for $j = 1, 2, \ldots, k$,
- and, $\frac{\partial}{\partial\, r_{\boldsymbol{\theta},i}} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau])_i$ is the partial derivative of the ith element of the $|\mathcal{S}||\mathcal{A}|$-dimensional vector $\mathbb{E}_\tau[\boldsymbol{\mu}_\tau]$ with respect to the ith element of the $|\mathcal{S}||\mathcal{A}|$-dimensional vector $r_{\boldsymbol{\theta}}$ of reward (i.e. the reward function).

When substituted back into Eq. 12, the resulting gradient is equivalent to that in Eq. 8 in Section 4. In particular, defining the $|\mathcal{S}||\mathcal{A}|$-dimensional diagonal matrix D as

$$D := \text{diag}\left(\left\{\frac{\partial}{\partial\, r_{\boldsymbol{\theta},i}} (\mathbb{E}_\tau[\mu_\tau])_i\right\}_{i=1}^{|\mathcal{S}||\mathcal{A}|}\right)$$

then the final term can be simplified to

$$\begin{aligned}
&\sum_{i=1}^{|\mathcal{S}||\mathcal{A}|} \left(\frac{\partial\, r_{\boldsymbol{\theta},i}}{\partial\, \boldsymbol{\theta}}\right) \left(\frac{\partial}{\partial\, r_{\boldsymbol{\theta},i}} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau])_i\right) \left(\frac{\partial\, r_{\boldsymbol{\theta},i}}{\partial\, \boldsymbol{\theta}}\right)^\top \\
&= \left(\frac{\partial\, r_{\boldsymbol{\theta}}}{\partial\, \boldsymbol{\theta}}\right) D \left(\frac{\partial\, r_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}\right)^\top .
\end{aligned}$$

In order to compute this gradient, however, we must take the gradient of the expectation $\mathbb{E}_\tau[\boldsymbol{\mu}_\tau]$ with respect to the reward function $r_{\boldsymbol{\theta}}$. This can be done by expanding the expectation as follows

$$
\begin{aligned}
&\frac{\partial}{\partial r_{\boldsymbol{\theta}}}\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] = \frac{\partial}{\partial r_{\boldsymbol{\theta}}}\sum_\tau \left(\frac{\exp(\boldsymbol{\mu}_\tau^\top r_{\boldsymbol{\theta}})}{\sum_{\tau'}\exp(\boldsymbol{\mu}_{\tau'}^\top r_{\boldsymbol{\theta}})}\right)\boldsymbol{\mu}_\tau \\
&= \sum_\tau \left(\left(\frac{\exp(\boldsymbol{\mu}_\tau^\top r_{\boldsymbol{\theta}})}{\sum_{\tau'}\exp[\boldsymbol{\mu}_{\tau'}^\top r_{\boldsymbol{\theta}})}\right)(\boldsymbol{\mu}_\tau\boldsymbol{\mu}_\tau^\top) - \frac{\exp(\boldsymbol{\mu}_\tau^\top r_{\boldsymbol{\theta}})}{(\sum_{\tau'}\exp(\boldsymbol{\mu}_{\tau'}^\top r_{\boldsymbol{\theta}}))^2}\sum_{\tau'}(\boldsymbol{\mu}_{\tau'}\boldsymbol{\mu}_\tau^\top)\exp(\boldsymbol{\mu}_{\tau'}^\top r_{\boldsymbol{\theta}})\right) \\
&= \sum_\tau P(\tau \mid r_{\boldsymbol{\theta}})(\boldsymbol{\mu}_\tau\boldsymbol{\mu}_\tau^\top) - \sum_\tau P(\tau|r_{\boldsymbol{\theta}})\sum_{\tau'}P(\tau' \mid r_{\boldsymbol{\theta}})(\boldsymbol{\mu}_{\tau'}\boldsymbol{\mu}_\tau^\top) \\
&= \mathbb{E}_\tau\left[(\boldsymbol{\mu}_\tau\boldsymbol{\mu}_\tau^\top) - \sum_{\tau'}P(\tau' \mid r_{\boldsymbol{\theta}})(\boldsymbol{\mu}_{\tau'}\boldsymbol{\mu}_\tau^\top)\right] \\
&= \mathbb{E}_\tau[\boldsymbol{\mu}_\tau\boldsymbol{\mu}_\tau^\top] - \mathbb{E}_{\tau',\tau}[\boldsymbol{\mu}_{\tau'}\boldsymbol{\mu}_\tau^\top] \\
&= \mathbb{E}_\tau[\boldsymbol{\mu}_\tau\boldsymbol{\mu}_\tau^\top] - \mathbb{E}_\tau[\boldsymbol{\mu}_\tau](\mathbb{E}_\tau[\boldsymbol{\mu}_\tau])^\top \\
&= \mathrm{Cov}[\boldsymbol{\mu}_\tau].
\end{aligned}
$$