A Experiment Details

Due to the relatively small scale of human-human interactions to train our policies, we manually represent the state as a 64- dimensional vector of hand-crafted features. We use the same features as in Carroll et al. [6], which contains relative positions of the ego agent to: the other player, the closest onion/tomato, plate, soup, onion/tomato source, plate dispenser, serving location, and pot. The featurization also contains contains a boolean encoding of the agent's orientation and whether the agent is adjacent to empty counters. Such featurization should enable more efficient learning of policies on the dataset.

Latent strategy learning. Note that while the human partner's latent strategy $\mathbf{z}$ influences the ego agent's next state $\mathbf{s}'$ together with their own state $\mathbf{s}$ and action $\mathbf{a}$, the latent strategy $\mathbf{z}$ may actually provide only a small amount of information in reconstructing the next state. Specifically, in *Overcooked*, the latent strategy $\mathbf{z}$ only influences where the human is relative to the ego agent. Because of this, we propose a simpler learning framework for the encoder. Namely, instead of reconstructing the next state via $d(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \mathbf{z})$, the latent strategy is only used to predict the human's action via an alternative decoder $\tilde{d}(\tilde{\mathbf{a}}|\mathbf{z})$, where the human's action $\tilde{\mathbf{a}} \in \mathcal{A}$ can be easily inferred from taking the difference between $\mathbf{s}'$ and $\mathbf{s}$.

The input to the encoder are a sequence of the previous 4 states observed by the ego agent. The encoder is parameterized as an LSTM with hidden dimension 256. The output of the LSTM is fed through a fully-connected layer to generated an 8-dimensional vector. Then, the decoder is simply parameterized as a 2-layer MLP with hidden dimension 16, whose output is a distribution over the 6 actions the human could take.

Policy learning. We parameterize the Q-function as a 3-layer fully-connected MLP with hidden dimension 256. The output of the policy is a 6-dimensional vector containing the Q-values for each of the 6 actions. Since the action space is discrete, the policy is simply choosing the action with the maximum Q-value, *i.e.*, $\pi(\mathbf{a} \mid \mathbf{s}) = \mathbb{I}\{\mathbf{a} = \max_{\mathbf{a}'} Q(\mathbf{s}, \mathbf{a}')\}$ We use CQL [18] to train our Q-function.

$$\begin{aligned}
\hat{Q}^{k+1} = \arg\min_{Q} \mathbb{E}_{\substack{\mathbf{s},\mathbf{a},h\sim\mathcal{D} \\ \mathbf{z}\sim f(\mathbf{z}|h)}} \left[\left(Q(\mathbf{s},\mathbf{a},\mathbf{z}) - \hat{\mathcal{B}}^{\hat{\pi}_k} \hat{Q}^k(\mathbf{s},\mathbf{a},\mathbf{z}) \right)^2 \right] + \\
\alpha \left(\mathbb{E}_{\substack{\mathbf{s},h\sim\mathcal{D} \\ \mathbf{z}\sim f(\mathbf{z}|h)}} \left[\log \sum_{\mathbf{a}} \exp(Q(\mathbf{s},\mathbf{a},\mathbf{z})) \right] - \mathbb{E}_{\substack{\mathbf{s},\mathbf{a},h\sim\mathcal{D} \\ \mathbf{z}\sim f(\mathbf{z}|h)}} \left[Q(\mathbf{s},\mathbf{a},\mathbf{z}) \right] \right) \\
\hat{\pi}^{k+1} = \arg\max_{\pi} \mathbb{E}_{\substack{\mathbf{s},h\sim\mathcal{D}\mathbf{z}\sim f(\mathbf{z}|h), \\ \mathbf{a}\sim\pi_k(\mathbf{a}|\mathbf{s},\mathbf{z})}} \left[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a},\mathbf{z}) \right] . \qquad (2)
\end{aligned}$$

Hyperparameters. We use the hyperparameters reported in Table 1 across all layouts.

Hyperparameter	Setting (for all layouts)
Discount factor	0.99
Batch size	256
Target network update period	every 500 updates
Number of updates per iteration	200
Number of iterations	100
Learning rate	3e-4

Table 1: Hyperparameters used during training.