Imitation Learning by Estimating Expertise of Demonstrators

Mark Beliaev [*1] **Andy Shih** [*2] **Stefano Ermon** [2] **Dorsa Sadigh** [2] **Ramtin Pedarsani** [1]

Abstract

Many existing imitation learning datasets are collected from multiple demonstrators, each with different expertise at different parts of the environment. Yet, standard imitation learning algorithms typically treat all demonstrators as homogeneous, regardless of their expertise, absorbing the weaknesses of any suboptimal demonstrators. In this work, we show that unsupervised learning over demonstrator expertise can lead to a consistent boost in the performance of imitation learning algorithms. We develop and optimize a joint model over a learned policy and expertise levels of the demonstrators. This enables our model to learn from the optimal behavior and filter out the suboptimal behavior of each demonstrator. Our model learns a single policy that can outperform even the best demonstrator, and can be used to estimate the expertise of any demonstrator at any state. We illustrate our findings on real-robotic continuous control tasks from Robomimic and discrete environments such as MiniGrid and chess, outperforming competing methods in 21 out of 23 settings, with an average of 7% and up to 60% improvement in terms of the final reward.

1. Introduction

Reinforcement learning provides a powerful and general framework for tasks such as autonomous vehicles, assistive robots, or conversation agents, by optimizing behavior with respect to user-specified reward functions. However, online interaction with the environment can be costly or even unsafe (Sutton & Barto, 2018; Mihatsch & Neuneier, 2002; Hans et al., 2008; Garcıa & Fernández, 2015), and specifying reward functions can be difficult in practice (Hadfield-Menell et al., 2017). Instead, one can mitigate these issues by viewing the problem through the lens of offline learning, where the learner either has access to demonstrations of the task along with the corresponding reward values as in offline reinforcement learning (Levine et al., 2020), or only has access to expert demonstrations without any reward information as in imitation learning (Pomerleau, 1991; Argall et al., 2009). In this work, we focus on the imitation learning setting—only assuming access to demonstrations.

The success of offline methods crucially depends on the availability of a large and diverse dataset (Pinto & Gupta, 2016; Fu et al., 2020) and as such, there has been a flurry of work in collecting vast amounts of expert demonstrations in various domains (Sharma et al., 2018; Zhang et al., 2018; Mandlekar et al., 2019; Fu et al., 2020; Mandlekar et al., 2021). A common finding from these works is the need for crowd-sourced data collection, both for scale and for diversity (Sharma et al., 2018). For example, Roboturk (Mandlekar et al., 2019) reports data-collection from 54 different humans, and Robomimic (Mandlekar et al., 2021) organizes its data into 6 different types of demonstrators with varying levels of expertise. Other works such as CARLA (Dosovitskiy et al., 2017) use a mix of human and autonomous agents for collecting demonstrations.

These crowd-sourced data collection pipelines inevitably generate a diverse dataset of behavior from users with varying levels of expertise. *Yet, current imitation learning algorithms typically treat these datasets as homogeneous.* By assuming all demonstrations are equally optimal, these algorithms may be blindly learning from the weaknesses of suboptimal demonstrators. Instead, our work asks the question: can we make use of the knowledge that the trajectories come from different demonstrators with various levels of suboptimality? More specifically, can we use the information of which demonstrator provided which demonstration to improve learning?

Our key insight is that imitation learning frameworks should account for the varying levels of suboptimality in large offline datasets by leveraging information about demonstrator identities. For example, say we learn to play chess by imitating from a large dataset of games. We do not know the expertise levels of the players, but we do know which player

[*]Equal contribution [1]Department of Computer Science, University of California, Santa Barbara [2]Department of Computer Science, Stanford University. Correspondence to: Mark Beliaev <mbeliaev@ucsb.edu>, Andy Shih <andyshih@cs.stanford.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

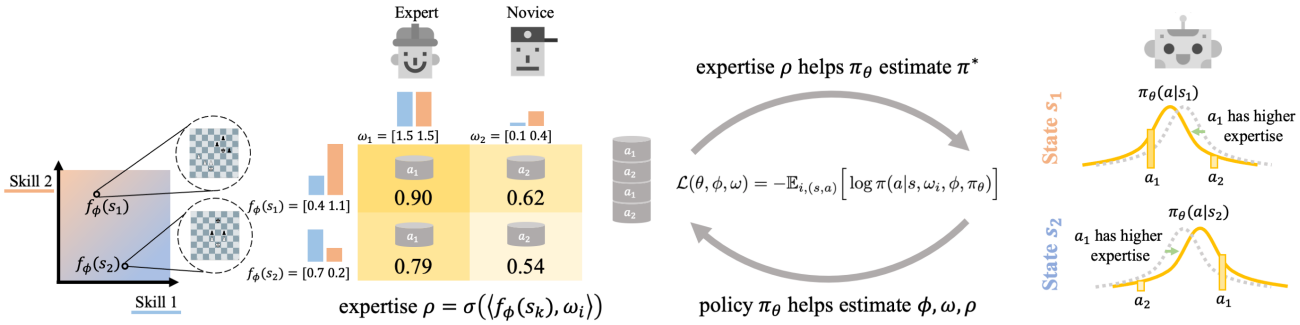


Figure 1. Left: State embeddings encode the skills associated with states of the environment. Middle: A demonstrator's expertise ρ at a state is a function of the state embedding and the demonstrator embedding. Right: Using the expertise levels, the model improves the learned policy π_θ, which in turn helps better estimate the state/demonstrator embeddings (ϕ and ω) and the expertise level (ρ).

played which games. Some players could be highly skilled Grandmasters, in which case we should treat their demonstrations seriously, but some could be novices, in which case we might ignore their demonstrations. Although we are not given their expertise levels, we can rely on information about which player played which games to estimate their expertise levels in an unsupervised manner. These estimated expertise levels can then be used to more effectively learn a chess-playing policy.

We propose ILEED, *Imitation Learning by Estimating Expertise of Demonstrators*, to imitate the trajectories in the dataset, while simultaneously learning to account for the different demonstrators' suboptimalities without any prior knowledge of their expertise (Fig. 1). ILEED optimizes a joint model over a learned policy and expertise levels, and recovers not just a single expertise value for each demonstrator, but a state-dependent expertise value that reveals which demonstrators are better at acting in specific states. We provide our implementation of ILEED online (Beliaev & Shih, 2022).

Our main contributions are as follows:

- We develop an imitation learning algorithm that jointly optimizes for an imitating policy and the expertise levels of the demonstrators. The joint model can estimate *state-dependent* expertise of demonstrators, identifying which demonstrator can perform well in which states.
- We theoretically show that our model generalizes standard behavioral cloning, and that our algorithm can recover the optimal policy via maximum likelihood if the suboptimal demonstrations align with our model's generative process.
- We experimentally show the success of our method compared to standard baselines on 1) simulated datasets for grid-world, 2) human datasets for continuous control, and 3) human datasets for chess endgames. We empirically demonstrate that our learned policy outperforms policies trained without taking into account demonstrator identities, and is comparable to policies trained only on high-quality demonstrations.

2. Related Work

First we discuss imitation learning, an instance of offline learning without the use of reward information. Since our work is concerned with learning from suboptimal demonstrations, we then discuss connections to modeling expertise in more general supervised learning settings.

Imitation Learning. There is a large body of work which approaches offline learning by relying on expert trajectories composed solely of state-action pairs, avoiding the need for labeling with a reward signal (Argall et al., 2009; Pomerleau, 1991; Ross et al., 2011; Finn et al., 2016; Ho & Ermon, 2016; Ding et al., 2019). The main challenge for IL approaches is the reliance on access to large amounts of expert demonstrations. In addition to this challenge, many real-world applications require the execution of multiple skills, where expert demonstrations are even more limited. Several lines of work aim to solve this: multi-task and meta imitation learning (Babes et al., 2011; Dimitrakakis & Rothkopf, 2011; Hausman et al., 2017; Li et al., 2017) as well as few shot learning (Duan et al., 2017; James et al., 2018; Singh et al., 2020) tackle the data efficiency problem in IL by considering both transfer learning to unseen skills and learning diverse sets of skills from multiple expert policies. Although similar to our setting due to the dependence on multiple demonstrators, unlike our method these approaches often assume oracle demonstrations, and in some cases access to online fine-tuning.

Stepping away from traditional imitation learning work that assumes access to oracle demonstrations, we are specifically concerned with learning from crowd-sourced data, where suboptimal demonstrations are unavoidable. Several works have considered this setting (Brown et al., 2019;

2020; Chen et al., 2020; Zhang et al., 2021; Cao & Sadigh, 2021), analyzing the impact of suboptimal demonstrations, and developing novel methods which can relax the amount of supervision required. Unlike our method, all of these approaches require environment dynamics to train on the attained reward signal, and in some cases knowledge about rankings over the set of demonstrations. We emphasize that our method does not rely on knowledge of environment dynamics or expertise rankings, and we leverage only the demonstrator identity of each demonstration.

We discuss two recent works that also address the suboptimal setting without utilizing environment dynamics. The first uses behavioral cloning (BC) to learn an ensemble policy directly from noisy demonstrations (Sasaki & Yamashina, 2021). Unlike our work that learns individual expertise levels of demonstrators, this work is mainly concerned with learning the best policy over noise-injected demonstrations without modeling the demonstrator identity or expertise. The second work (TRAIL) (Yang et al., 2021) tackles this setting by leveraging suboptimal data to extract a latent action space, which is used alongside standard BC to train a policy on a small set of "near-optimal" expert demonstrations. In contrast, our approach does not rely on access to such near-optimal expert demonstrations.

Unsupervised Estimation of Expertise. The problem of inferring ground truth labels from crowd-sourced human data has been studied in biostatistics, education, and more recently computer vision, and NLP. These works generally tackle the problem using the Expectation Maximization (EM) algorithm to solve for the individual error rates of the human annotators (Dawid & Skene, 1979). Furthermore, similar approaches have been applied to the crowd sourcing problem of labeling large image datasets (Whitehill et al., 2009; Raykar et al., 2010; Welinder et al., 2010), learning a model over annotators to generate more accurate estimates. Specifically, one paper models each annotator and task using multidimensional variables representing difficulty, competence, expertise, and bias (Welinder et al., 2010). Inspired by this, we apply a similar formulation to IL, addressing several challenges that go beyond the scope of supervised learning. More precisely, in the image domain, one can collect multiple annotator labels for many images and compare them to the ground truth. In our imitation learning setting, on the other hand, demonstrators may not visit the same states, states are intertwined through dynamics, and the optimal policy may give action distributions instead of a single optimal action.

3. Joint Estimation of Policy and Expertise

In this section we describe a joint model that learns from a dataset consisting of a mixture of demonstrations from demonstrators with varying, but unknown, levels of expertise. Our model both infers state-dependent expertise levels of the different demonstrators in the dataset, and recovers a single policy learned from all the demonstrations.

3.1. Problem Setup

Dataset. We collect a set $\mathcal{D}_i$ of trajectories $\tau = (s_0, a_0, \ldots, s_t, a_t)$ of varying length from each demonstrator i. We assume the trajectories from demonstrator i are sampled from some fixed underlying policy π_i. The full dataset $\mathcal{D} = \{(i, \mathcal{D}_i)\}_{i=1}^m$ is the union of the dataset from each of the m demonstrators, labeled by the demonstrator index. Generally speaking, each trajectory could exhibit close to random behavior, but could also come from a demonstrator with high expertise. We would like a model that identifies when demonstrations are suboptimal, to better learn a single policy from the mixed bag of demonstrations.

Demonstrator Model. Our demonstrator model should be able to express state-dependent expertise. For example, demonstrator A may be adept at washing the dishes, while demonstrator B may be adept at vacuuming the floor, and modeling this state-dependent expertise can allow us to recognize and combine their strengths in different states. To model such suboptimal policies, we will define two main components: 1) the expertise level of a demonstrator *at a given state* and 2) the demonstrator's action distribution at a state as a function of their expertise level and the optimal action distribution at that state.

1) Expertise Levels. Drawing inspiration from annotator models (Welinder et al., 2010), we model expertise levels using two embeddings: a d–dimensional state embedding using a deterministic map $f_\phi : \mathcal{S} \to \mathbb{R}^d$ (parameterized by ϕ) from states to embeddings, and demonstrator embeddings $\omega \in \mathbb{R}^{m \times d}$, where ω_i is a d–dimensional vector capturing the aptitude of demonstrator i. Using these embeddings, we quantify the expertise level of demonstrator i at state s as:

$$\rho_\phi(s, \omega_i) = \sigma(\langle f_\phi(s), \omega_i \rangle), \tag{1}$$

where $\sigma : \mathbb{R} \to (0, 1)$ is the sigmoid function and $\langle \cdot, \cdot \rangle$ denotes the inner product.

We can interpret each dimension of the embedding vector $f_\phi(s)$ as a weighting of how relevant a latent skill is in acting correctly at that state s. A demonstrator's skill set is the d-dimensional embedding ω_i that expresses how adept the demonstrator is at each skill. This way we can measure how qualified demonstrator i is at *acting in* state s by computing the inner product $\langle f_\phi(s), \omega_i \rangle$ between the task encoding of the state, and the demonstrator's skill set ω_i.

2) Demonstrator's Action Distribution. We now define the demonstrator's suboptimal policy as a function of their expertise level and the optimal policy $\pi_{\theta^\star}$.

We would like the expertise level $0 \leq \rho_\phi(s, \omega_i) \leq 1$ of

demonstrator i at state s to be correlated with how close their action distribution is to the true action distribution $\pi_{\theta^\star}(a|s)$, with $\rho_\phi(s, \omega_i) = 1$ corresponding to exactly $\pi_{\theta^\star}(a|s)$ and $\rho_\phi(s, \omega_i) = 0$ corresponding to a uniformly random distribution. We can satisfy this desiderata, separately for discrete and continuous action spaces, using the following models.

Discrete Action Space. When the action space is discrete, we use $\rho_\phi(s, \omega_i)$ to interpolate between the optimal policy and the uniformly random policy which assigns probability $1/|\mathcal{A}|$ to each action.

$$\pi(a|s, \omega_i, \phi, \pi_{\theta^\star}) = \rho_\phi(s, \omega_i)\pi_{\theta^\star}(a|s) + \frac{1-\rho_\phi(s,\omega_i)}{|\mathcal{A}|} \quad (2)$$

Continuous Action Space. For continuous action spaces, we will focus on Gaussian Mixture Model (GMM) action distributions, as in Mandlekar et al. (2021). Specifically, at a given state s the optimal policy $\pi_{\theta^\star}(a|s)$ outputs a probability distribution over actions $a \in \mathcal{A}$ in the form of a GMM with k mixtures $\pi_{\theta^\star}(a|s) = \sum_{j=1}^{k} \alpha_j \mathcal{N}(a; \mu_j^\star(s), \sigma_j^\star(s))$. Then, given a demonstrator with expertise level $\rho_\phi(s, \omega_i)$ at state s, we simply scale the variance of the optimal policy's GMM (equally for each component) by $1/\rho_\phi(s, \omega_i)$. Thus, the probability of the demonstrator actions $\pi(a|s, \omega_i, \phi)$ modifies $\pi_{\theta^\star}$ as follows:

$$\pi(a|s, \omega_i, \phi, \pi_{\theta^\star}) = \sum_{j=1}^{k} \alpha_j \mathcal{N}(a; \mu_j^\star(s), \sigma_j^\star(s)/\rho_\phi(s, \omega_i)) \quad (3)$$

An expertise level of 1 corresponds to an expert whose recommendations align with the optimal policy, whereas an expertise level approaching 0 corresponds to an expert with close to uniformly random actions. More complex models of expertise can be explored, but we find that our simple single-valued expertise model already gives good improvements.

3.2. Learning the Optimal Policy and Expertise Levels

So far we have defined the model of demonstrators with respect to some optimal policy $\pi_{\theta^\star}(a|s)$, i.e., a demonstrator's expertise level correlates with how close their policy is to $\pi_{\theta^\star}(a|s)$. However, we do not have access to $\pi_{\theta^\star}(a|s)$. Hence, we will define a parametric family of policies $\{\pi_\theta : \theta \in \Theta\}$, and try to recover $\pi_{\theta^\star}(a|s)$. For the analysis in the rest of the section we will assume that Θ is well-specified (i.e. $\theta^\star \in \Theta$), and that all demonstrators explore all states with non-zero probability. Put together, we will be jointly learning θ, ϕ, ω: the optimal policy, the state embedding network, and the demonstrator embeddings. Using a maximum likelihood approach, we optimize these variables with the loss corresponding to the negative log-likelihood (NLL) of data.

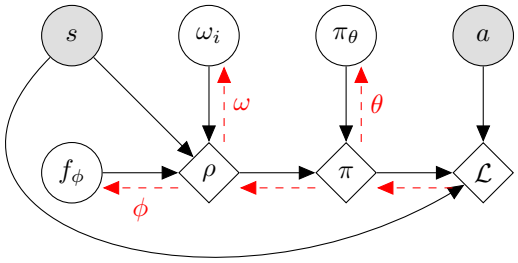


Figure 2. We observe the state, action, and demonstrator index i. Diamond nodes are deterministically computed. From the state embedding $f_\phi(s)$ and the demonstrator embedding ω_i, we compute the expertise level ρ. We combine ρ with the estimate π_θ of the optimal policy to obtain the demonstrator policy π. The loss $\mathcal{L}$ (Eq. 4) of π on state s and action a is back-propagated to update the parameters θ, ϕ, ω.

$$\mathcal{L}(\theta, \phi, \omega) = -\mathbb{E}_{i,(s,a)}\Big[\log \pi(a|s, \omega_i, \phi, \pi_\theta)\Big] \quad (4)$$

$$\approx -\frac{1}{|\mathcal{D}|}\sum_i \sum_{(s,a)\in\mathcal{D}_i} \log \pi(a|s, \omega_i, \phi, \pi_\theta) \quad (5)$$

Although we can simply rely on the loss in Eq. 4 to learn the state embedding f_ϕ (used in Eq. 1), it can be beneficial to consider the dynamics of the MDP environment as well, which may reveal more about the difficulty of each state. One popular approach is the DeepMDP framework (Gelada et al., 2019), which uses an auxiliary loss to predict the environment dynamics in latent space. At a high level, this process uses the trajectories in our dataset as samples of the MDP dynamics to help learn a better state embedding f_ϕ. The details of this framework is described in Appendix A.

The overall learning framework can be seen in Fig. 2, where for demonstrator i, we compute an expertise level $\rho_\phi(s, \omega_i)$, that is then combined with our estimate π_θ of the optimal policy to derive the demonstrator policy $\pi(a|s, \omega_i, \phi, \pi_\theta)$ in Eq. 4. We update all the parameters (θ, ϕ, ω) to optimize our loss function.

To see why our loss function is suitable, we can rewrite the joint optimization equivalently only over θ, and show that the following objective serves as a *proper* loss function.

$$\mathcal{L}(\theta) = -\max_{\phi,\omega} \mathbb{E}_{i,(s,a)}\Big[\log \pi(a|s, \omega_i, \phi, \pi_\theta)\Big] \quad (6)$$

In other words, $\theta^\star$ is the (non-unique) minimizer of $\mathcal{L}(\theta)$.

Proposition 3.1. *$\mathcal{L}(\theta)$ is a **proper** loss function.*

In addition, it is easy to see that our framework generalizes standard behavioral cloning. If we set the embedding vectors to large positive values everywhere, then the expertise levels $\rho_\phi(s, \omega_i)$ for all states and all demonstrators will approach 1, in which case $\mathcal{L}(\theta)$ approaches $\mathcal{L}_{BC}(\theta)$.

Remark 3.2. *ILEED recovers the standard behavioral cloning framework by setting all expertise levels to* 1.

Moreover, we will recover a different policy than behavioral cloning unless all demonstrators have identical policies.

Proposition 3.3. *We have that* $\min_\theta \mathcal{L}_{BC}(\theta) > \min_\theta \mathcal{L}(\theta)$ *unless all demonstrators have identical policies.*

Hence, in the presence of different demonstrators, our framework will incorporate their varying expertise levels into its estimate of the optimal policy.

We have shown several nice properties of our loss function: 1) $\theta^\star$ minimizes $\mathcal{L}(\theta)$, and 2) $\mathcal{L}(\theta, \phi, \omega)$ generalizes $\mathcal{L}_{BC}(\theta)$, and 3) our framework incorporates varying expertise levels. However, we have not shown that $\mathcal{L}(\theta)$ is *strictly proper*, i.e., $\theta^\star$ may not be the unique minimizer and therefore it is unclear if we will recover the optimal policy. To this end, we show that if we have knowledge of the state embedding $f_\phi(s)$, then under some assumptions, we can uniquely recover the optimal policy $\pi_{\theta^\star}$.

Lemma 3.4. *Let* ϕ *be the ground truth state embedding parameters, and let* $\mathcal{L}_\phi(\theta)$ *be the loss in Eq. 6 using fixed* ϕ*.*

$$\mathcal{L}(\theta) = -\max_{\omega} \mathbb{E}_{i,(s,a)}\Big[\log \pi(a|s, \omega_i, \phi, \pi_\theta)\Big]$$

Under both the discrete and continuous action model (Eq. 2 & 3), $\mathcal{L}_\phi(\theta)$ *is a* **strictly proper** *loss function if for all states* s_0*, there exists a set of other states* $s_{1:r}$ *such that*

$$f_\phi(s_0) = \alpha_1 f_\phi(s_1) + \alpha_2 f_\phi(s_2) + \ldots + \alpha_r f_\phi(s_r) \quad (7)$$

and no set of constants $C_{0:r}$*, with* $C_0 \neq 1$*, satisfies the following conditions.* $\forall i \in \{1, \ldots, m\}$*:*

$$\sigma^{-1}(\rho_\phi(s_0, \omega_i)/C_0) = \sum_{k=1}^{r} \alpha_k \sigma^{-1}(\rho_\phi(s_k, \omega_i))/C_k).$$

Intuitively, the challenge in uniquely recovering the optimal policy is that the true action distribution $\pi_{\theta^\star}(\cdot|s)$ at a state may be expressed as a low-expertise version of another distribution $\tilde{\pi}(\cdot|s)$. Our model might incorrectly recover $\tilde{\pi}(\cdot|s)$, and compensate by decreasing the expertise levels of all demonstrators in a precise way. If all the state embeddings are linearly independent (e.g. basis vectors in high-dimensions), then our model can fully control the demonstrator expertise levels $\rho_\phi(s, \omega_i)$ by tweaking the demonstrator embeddings ω. On the other hand, the above result says that if the state embeddings are intertwined (Eq. 7), then the model cannot fully control $\rho_\phi(s, \omega_i)$. Therefore the model cannot mistake $\pi_{\theta^\star}(\cdot|s)$ for $\tilde{\pi}(\cdot|s)$ since it cannot compensate for the different expertise levels. We include the proofs of these results in Appendix D.

Moreover, note that the constraints on $C_{0:r}$ are less likely to hold as the number of demonstrators m grows. In other words, assuming we have learned the correct state embeddings, if the embeddings are intertwined enough then ILEED can recover the optimal policy. On the other hand, BC will be unable to recover the optimal policy in the presence of suboptimality. We include a concrete example in Appendix B showing the improvement of our method in the presence of demonstrators with varying optimalities.

Summary. We defined a model of suboptimal demonstrators, where the action distribution of demonstrator i at a state s is determined by the state embedding $f_\phi(s)$, the expertise embedding ω_i, and the optimal action distribution $\pi_{\theta^\star}(a|s)$ (Eq. 2, 3). Our method generalizes the standard BC framework to the case of demonstrators with varying expertise levels, enabling us to better handle demonstrators with varying state-dependent optimalities. Finally, we show the recoverability of the optimal policy when we have knowledge of the state embeddings, and integrate unsupervised techniques for learning these state embeddings.

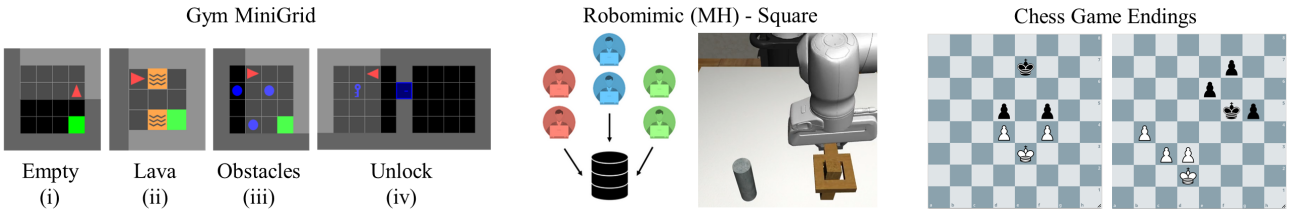
4. Experiments

We will test if our algorithm can: (1) learn a policy from a mixture of *simulated* demonstrations with varying levels of proficiency, (2) learn a policy from a mixture of *human* demonstrations with varying levels of proficiency, (3) recover *human* expertise levels even when learning an optimal policy is too challenging, and (4) learn a policy for multiple skills from a mixture of *simulated* demonstrations with state-dependent noise.

Note that we can use ILEED to learn either state-*dependent* expertise levels, or state-*independent* expertise levels where we assume a demonstrator has a single expertise value that is the same at all states. Out of the four aforementioned experiments in the paper, only the last experiment learns state-dependent expertise levels. For the first three experiments, we found that adding state-dependence did not improve performance. For the fourth experiment, state-dependent expertise was important due to the multi-skill nature of the task, and a state-independent approach did poorly. We show this by including an ablation along with the last experiment that studies the individual effect of state embeddings and demonstrator identities on ILEED's performance. Before going over our results, we briefly detail the specific environments and datasets used in our experiments.

4.1. Environments and Datasets

For the first and last experiment we rely on simulated data, using 4 MiniGrid (Chevalier-Boisvert et al., 2018) environments along with pre-trained policies with various levels of injected noise. All 4 environments use a partially observ-



(a) (i) *Empty* - The agent starts at a random position. The objective is to reach the green square.
(ii) *Lava* - The agent must pass through a narrow gap in a vertical strip of lava to. Touching the lava terminates the episode with a zero reward, making this environment useful for studying safe exploration.
(iii) *Obstacles* - A large penalty is subtracted if the agent collides with an obstacle and the episode finishes, resulting in a possible negative reward.
(iv) *Unlock* - The agent must unlock the door by first picking up a key and then entering the door.

(b) We utilized the low dimension Multi-Human (MH) Robomimic dataset which contains 50 demonstrations from 6 humans, split up into three groups based on their proficiency. We used the Square environment, a continuous control task in which a robotic arm must pick up a square nut and place it on a rod, and a binary reward is awarded for completing the task in the allotted time. The images shown were directly taken from the study (Mandlekar et al., 2021).

(c) We examine games from the lichess database (McIlroy-Young et al., 2020), consisting of ∼10M games in 2019 from players with rating ranging from 1k to 2k. We limit ourselves to endgame positions with only kings and pawns. We encode the state into a discrete vector and use a 3-layer MLP policy network and 2-dimensional state and expertise embeddings. We divide the dataset into 5 bins based on rating percentile (e.g. bottom 20% of players in the first bin).

Figure 3. Environments used throughout our work: MiniGrid, Robomimic, and chess.

able view with 3 input values per visible grid cell, and a maximum reward of one is given if the objective is reached with a small penalty subtracted for the number of steps to reach the goal. The environments are depicted in Fig. 3(a), along with a brief description for each. For the second experiment, which relies on suboptimal human data, we use the Robomimic dataset and codebase (Mandlekar et al., 2021) which consists of various continuous control robotics environments along with corresponding sets of suboptimal human data. We depict the environment used and briefly describe the dataset in Fig. 3(b). For the third experiment we derive player rankings using human chess game-ending data provided by the lichess database (McIlroy-Young et al., 2020), which is briefly explain in Fig. 3(c).

4.2. Baselines

Throughout the experiments we compare our model with 3 other IL algorithms (BC, BC-RNN, GAIL), as well as one recent offline learning algorithms IRIS (Mandlekar et al., 2020). To the best of our knowledge, there are no IL algorithms which show good performance on suboptimal human datasets besides BC-based approaches. Current approaches that tackle this setting either rely on the reward signal (Fujimoto et al., 2019; Kumar et al., 2020; Mandlekar et al., 2020) (BCQ, CQL, IRIS), or break the offline assumption by using environment simulations (Ho & Ermon, 2016; Brown et al., 2019; 2020; Chen et al., 2020; Fu et al., 2017) (GAIL, D-REX, T-REX, SSRR, AIRL). In addition, recent work has shown that BC-based approaches perform better compared to other offline learning techniques in settings with suboptimal demonstrations (Mandlekar et al., 2021; Florence et al., 2021). We thus treat BC-based approaches as our main baselines for the simulated experiments. To further test our model against the aforementioned recent algorithms, we directly compare with the results from Robomimic.

4.3. Learning from Simulated Data

We first study how our algorithm performs on simulated suboptimal data, where we vary the optimality level by injecting noise into pre-trained policies. When simulating data, we use a set of m state-independent expertise levels β_i for $i \in \{1, \dots, m\}$, and collect a fixed number of state-action pairs from each. Specifically $\beta_i = \rho_\phi(s, \omega_i)\ \forall s \in \mathcal{S}$, where we use the discrete action space model defined in Eq. 2 to interpolate between a random policy $\beta = 0$ and the pre-trained policy $\beta = 1$. All of the MiniGrid experiments use fully-connected neural networks with the Adam optimizer, with specific parameters left to Appendix E. For the four aforementioned environments, the respective performance of the pre-trained policies along with their noised version are listed in Table 5 in Appendix C.

Before moving on to our first experiment, we tested the relationship between the learned policy's performance and the corresponding NLL defined in Eq. 4. By varying the number of restarts and choosing the policy with the highest likelihood, we can empirically test if our defined loss in Eq. 4 can also serve as a good validation metric for the final policy's performance. We display this in Table 4 of Appendix C.1, where we see that as the number of restarts increases, the policy's performance improves as well. Based on this insight, we set the number of restarts to 20. As for the two baselines in this experiment, BC and GAIL, we also restart BC as many times as ILEED, choosing the policy

Table 1. Effect of Varying Population Expertise β
We compute mean episodic reward of policies from different IL algorithms, over 20 trials (for BC and ILEED).

β	Empty			Obstacles		
	BC	**GAIL**	**ILEED**	**BC**	**GAIL**	**ILEED**
1	0.81	0.96	**0.97**	0.18	−0.82	**0.91**
5	0.97	0.96	0.97	0.66	−0.77	**0.94**
10	0.97	0.96	0.97	0.63	−0.01	**0.94**
unif	0.97	0.96	0.97	0.80	−0.84	**0.90**

with the lowest loss. In contrast, we only ran GAIL once, because unlike BC and ILEED, GAIL optimizes for a saddle point as opposed to a minimum, so we cannot use the lowest loss as a validation metric for choosing between restarts.

We want to study how varying the population of demonstrators using simulated noise affects the performance of selected algorithms. To test this, we chose several sets of m demonstrators, each with a different predefined set of expertise levels $\beta = \{\beta_1, \ldots, \beta_M\}$. The four chosen sets are denoted as: β-*1*:$\{\beta_1 = 0.99, \beta_{2:10} = 0.01\}$, β-*5*:$\{\beta_{1:5} = 0.99, \beta_{6:10} = 0.01\}$, β-*10*:$\{\beta_{1:10} = 0.99\}$, β-*unif*:$\{\beta_i = 0.05 + 0.1(i-1)\}$. By keeping the noise levels constant and re-sampling state-action pairs for each trial, we study how different distributions of expertise levels affect performance, showing this result in Table 1.

Our algorithm outperforms both BC and GAIL in all 4 population settings when testing on the *Empty* and *Obstacles* MiniGrid environments. For the simpler *Empty* environment, we see all algorithms were able to match the pre-trained policy's performance, though the BC struggled in the β-*1* population shown on the first row, where only 1 of the 10 demonstrators is competent. In the more challenging *Obstacles* environment, we see both GAIL and BC are unable to imitate the policy even when noise is diminished to 1% in the β-*10* population shown on the third row, where all 10 demonstrators are fairly competent. Meanwhile ILEED is able to achieve consistent performance even for the noisier populations. Overall GAIL showed inconsistent results and did not perform well in the *Obstacles* environment, and we note again that GAIL also assumes access to environment dynamics. We include these comparisons for *Lava* and *Unlock* in Appendix C.2. Note that the rankings derived from the estimated expertise levels of the demonstrators correlate with the performance of the final policy as we show in Appendix C.3.

4.4. Learning from Suboptimal Human Data

Next, we test our model on the continuous control task *Square* depicted in Fig. 3(b), where the dataset provided contains suboptimal human demonstrations. Specifically, the demonstrations used consists of three subsets of 100 demonstrations provided by: two "better" quality operators, two "okay" operators, and two "worse" operators. Like the original study, we use different combinations of the smaller subsets to investigate how suboptimal human data affects performance. Using this dataset, we are able to compare our method with three IL algorithms (BC, BC-RNN, HBC), as well as three recent offline learning algorithms (BCQ, CQL, IRIS) which differ from the IL setup by also utilizing reward information. Nonetheless, we are able to outperform all six methods for every combination of the suboptimal dataset, and hence only include results for the strongest baselines of each group: BC-RNN and IRIS. This result is displayed in Table 2, where we note that no restarts were used in our model for fair comparison with results reported from (Mandlekar et al., 2021).

Table 2. Suboptimal Human Data for Continuous Control
Our method outperforms all other methods in all settings. We copy results for the two strongest methods (Mandlekar et al., 2021), and average ILEED over 3 trials as done in the Robomimic study.

Dataset	**BC-RNN**	**IRIS**	**ILEED** (ours)
All	$\mathbf{78.0 \pm 4.3}$	52.7 ± 5.0	$\mathbf{78.0 \pm 1.6}$
Worse	39.3 ± 3.8	38.7 ± 0.9	$\mathbf{46.7 \pm 4.7}$
Okay	45.3 ± 2.5	42.0 ± 3.3	$\mathbf{53.3 \pm 2.5}$
Better	66.0 ± 2.8	60.0 ± 1.6	$\mathbf{72.7 \pm 3.8}$
Worse-Okay	55.3 ± 0.9	43.3 ± 2.5	$\mathbf{59.3 \pm 3.8}$
Worse-Better	73.3 ± 6.2	56.7 ± 3.4	$\mathbf{77.3 \pm 6.8}$
Okay-Better	74.0 ± 2.8	56.7 ± 3.8	$\mathbf{77.3 \pm 0.9}$
Total	61.6 ± 3.3	50.0 ± 2.9	$\mathbf{66.4 \pm 3.4}$

We can see from the results that modeling human expertise levels as done by ILEED provides consistent improvement over other IL algorithms. As noted by the cited study: BC-RNN is a strong baseline on suboptimal human data, but there is room for improvement. We see from Table 2 that ILEED does in fact improve over BC-RNN as it is better at utilizing suboptimal demonstrations. Finally, we can see that ILEED outperforms BC-RNN in all settings with an average 4.8% increase in final reward. This shows that learning a model for demonstrator's expertise can significantly boost performance by taking more advantage of suboptimal data compared to methods that ignore demonstrator expertise.

4.5. Estimating Expertise from Human Data

For our second human experiment, we explore the ambitious task of learning to play chess endgames purely from data (McIlroy-Young et al., 2020), without access to the environment (i.e., without knowing the rules of the game or accessing rewards). This task is extremely difficult, since most chess-playing agents assume access to the game's rules and rely on some form of self-play or tree-search to reach good performance (Pascutto et al., 2018; Romstad et al.,

2021). Nevertheless, this task serves as a good benchmark for validating our model's estimation of the expertise level of demonstrators. After splitting our dataset into 5 bins based on rating percentile (Fig. 3(c)), we recover the expertise ρ of the bins.

Bins	1	2	3	4	5
Expertise	.9207	.9274	.9298	.9328	.9329

In the above table, we see that our predicted expertise levels (averaged over states in the dataset) of the bins align with the ground truth skill levels of the bins – the higher the ground truth rating of the bins, the higher our estimated expertise level. This suggests that our framework can serve a dual purpose as an estimator of demonstrator expertise levels. Though these expertise values seem close in absolute terms, we note that a difference of a few percentage points in accuracy can lead to a large rating difference (cf. Fig. 3 in (McIlroy-Young et al., 2020)), and that the takeaway of our experiment is the monotonicity of our recovered expertise values. Lastly, in Table 8 (in Appendix C.4), we check that our method also outperforms BC in terms of the performance of the learned policy, as measured by treating StockFish evaluations as the reward in the environment. Though we did not expect to recover a good policy, given the difficulty of the task, the improvement over BC may be interesting for further investigation.

4.6. Learning from Expertise in Different Skills

So far, we have not used our model's state-dependent components, namely the state embedding $f_\phi(s)$ and the auxiliary loss provided by the DeepMDP framework. In fact, we note that surprisingly, the simulated MiniGrid experiments performed in Section 4.3 produced near-identical results with and without the auxiliary loss. Thus before concluding, we study the state dependent components by simulating an environment with multiple skills, where each skill can be seen as achieving an independent task within the environment.

In this experiment, we utilize the environments *Unlock*, *Lava*, and *Empty*. We note that these three environments exhibit the need for independent skills, i.e., a policy trained on one of them does not necessarily do well on the other two. The reason we focus on independent skills is to ensure training a policy on each and all task will lead to full coverage over all the desired states and skills; however, our method can also generalize to settings with non-independent skills as well. We place the 3 environments in succession, such that an agent must successfully perform all 3 tasks to receive a reward of 1. When collecting trajectories, we allow continuation to the next environment even if the policy failed in the current environment. For evaluation, we average the agents' performance on all environments. For the datasets we use 3 demonstrators, one being an expert in each skill.

Table 3. Learning Multiple Skills from Suboptimal Data
Mean and standard deviation of episodic reward (over 100 trials) for all demonstrators, the best demonstrator and ILEED.

β	**All Demons.**	**Best Demons.**	**ILEED**
0.01	0.40 ± 0.04	0.44 ± 0.04	$\mathbf{0.70 \pm 0.06}$
0.10	0.52 ± 0.05	0.58 ± 0.05	$\mathbf{0.80 \pm 0.06}$
0.20	0.67 ± 0.04	0.76 ± 0.04	$\mathbf{0.86 \pm 0.05}$
0.50	0.85 ± 0.03	0.90 ± 0.01	0.86 ± 0.04
1.00	0.94 ± 0.01	0.94 ± 0.01	0.89 ± 0.04

Specifically, we collect 10000 state action pairs from all 3 demonstrators, where they act according to the optimal (pre-trained) policy in the environments they are *skilled* in, and suboptimally in the other two environments. We control the level of suboptimality by β, which refers to the probability that the demonstrator acts optimally for the environments they are *unskilled* in. For example, a demonstrator skilled in *Unlock* with $\beta = 0.1$ will always act according to the pre-trained policy when in the *Unlock* environment, while only following the corresponding pre-trained policies 10% of the time in the other two environments (acting randomly otherwise). This way, the noisiest dataset at $\beta = 0$ still contains some optimal demonstrations for all 3 environments, and as we increase β to 1, all demonstrations come from the corresponding pre-trained policy. We show the result for 5 values of expertise β in Table 3.

We can see that our model consistently outperforms the average demonstrator for $\beta < 1$, and even the best demonstrator for $\beta < 0.5$. This is expected for lower β, as even the best demonstrator can only be successful at one of the tasks due to the independence of tasks. We note that the optimal dataset at $\beta = 1$ contains no noisy demonstrations, in which case our model is able to come within 5% of the best demonstrator. As we decrease β, each demonstrator provides more suboptimal state-action pairs for 2 of the 3 environments, but our model is still able to combine their expertise and learn a policy which is skilled in multiple environments. We show trajectories in Fig. 4 of the Appendix, noting that for all settings our learned policy adequately performs on 2 environments, failing mostly in *Unlock*. Additionally, we provide an ablation study in Appendix C.5 to test the individual effect of state embeddings and demonstrator identities on ILEED's performance, showing that both components contribute to the high performance of ILEED in this multi-skill experiment.

5. Discussion

Summary. We present ILEED – an approach for IL from demonstrators with varying but unknown state-dependent expertise. By jointly optimizing for the optimal policy and

demonstrator expertise, we learn a better policy as compared to imitation learning baselines, and also are able to recover accurate estimates of the ground-truth expertise levels.

Our framework highlights the important problem of IL from datasets with multiple demonstrators. Datasets collected from different demonstrators are already prevalent across continuous control and discrete tasks (Mandlekar et al., 2021; McIlroy-Young et al., 2020), and will only increase in relevance as the need for large and diverse datasets inevitably grows (Sharma et al., 2018). We show that in these settings, unsupervised estimation of the demonstrator expertise gives a large boost in performance.

Limitation and Future Work. Our work is limited in a number of ways. First, some of our model's theoretical properties and predictive power rely on recovering effective state embeddings, which can be challenging in practice, e.g., when the state space is not fully explored by the demonstrators. Moreover, we have yet to analyze the relationship between recovered expertise values ρ and demonstrators' true expertise levels for environments with state-dependent expertise. Finally, our method currently has a simple model of suboptimality that uses a single-valued expertise level at each state. More complex demonstrator models can be explored to capture different modes of suboptimality in demonstrators.

Nevertheless, we believe our framework provides a novel and effective approach for addressing the prevalent problem of learning from demonstrators with varying levels of expertise. Our framework is general, and lends itself to many possible directions of future work. As part of future work, we plan to model the uncertainty over the expertise levels and consider different notions of suboptimality beyond noisy action distributions.

Acknowledgements

This research was supported by NSF (1941722, 2006388, 2125511, 2003035, 1952920, 1651565), ARO (W911NF2110125), AFOSR, ONR, and Ford.

References

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Babes, M., Marivate, V. N., Subramanian, K., and Littman, M. L. Apprenticeship learning about multiple intentions. In *ICML*, 2011.

Beliaev, M. and Shih, A. ILEED, 6 2022. URL https://github.com/Stanford-ILIAD/ILEED.

Brown, D., Goo, W., Nagarajan, P., and Niekum, S. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pp. 783–792. PMLR, 2019.

Brown, D. S., Goo, W., and Niekum, S. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pp. 330–359. PMLR, 2020.

Cao, Z. and Sadigh, D. Learning from imperfect demonstrations from agents with varying dynamics. *IEEE Robotics and Automation Letters*, 6(3):5231–5238, 2021.

Chen, L., Paleja, R., and Gombolay, M. Learning from suboptimal demonstration via self-supervised reward regression. *arXiv preprint arXiv:2010.11723*, 2020.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Dawid, A. P. and Skene, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

Dimitrakakis, C. and Rothkopf, C. A. Bayesian multitask inverse reinforcement learning. In *European workshop on reinforcement learning*, pp. 273–284. Springer, 2011.

Ding, Y., Florensa, C., Phielipp, M., and Abbeel, P. Goal-conditioned imitation learning. *arXiv preprint arXiv:1906.05838*, 2019.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control, 2016.

Duan, Y., Andrychowicz, M., Stadie, B., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1087–1098, 2017.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pp. 49–58. PMLR, 2016.

Florence, P., Lynch, C., Zeng, A., Ramirez, O., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. Implicit behavioral cloning. *arXiv preprint arXiv:2109.00137*, 2021.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, 2019.

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S. J., and Dragan, A. D. Inverse reward design. In *Advances in Neural Information Processing Systems 30*, 2017.

Hans, A., Schneegaß, D., Schäfer, A. M., and Udluft, S. Safe exploration for reinforcement learning. In *ESANN*, pp. 143–148. Citeseer, 2008.

Hausman, K., Chebotar, Y., Schaal, S., Sukhatme, G., and Lim, J. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *arXiv preprint arXiv:1705.10479*, 2017.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

James, S., Bloesch, M., and Davison, A. J. Task-embedded control networks for few-shot imitation learning. In *Conference on Robot Learning*, pp. 783–795. PMLR, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Li, Y., Song, J., and Ermon, S. Infogail: Interpretable imitation learning from visual demonstrations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 3815–3825, 2017.

Mandlekar, A., Booher, J., Spero, M., Tung, A., Gupta, A., Zhu, Y., Garg, A., Savarese, S., and Fei-Fei, L. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1048–1055. IEEE, 2019.

Mandlekar, A., Ramos, F., Boots, B., Savarese, S., Fei-Fei, L., Garg, A., and Fox, D. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4414–4420. IEEE, 2020.

Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and Martín-Martín, R. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on robot learning*, 2021.

McIlroy-Young, R., Sen, S., Kleinberg, J., and Anderson, A. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1677–1687, 2020.

Mihatsch, O. and Neuneier, R. Risk-sensitive reinforcement learning. *Machine learning*, 49(2):267–290, 2002.

Pascutto, G.-C., Linscott, G., Lyashuk, A., Huizinga, F., et al. Leela Chess Zero, 2018. URL https://lczero.org.

Pinto, L. and Gupta, A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 3406–3413. IEEE, 2016.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L. Learning from crowds. *Journal of Machine Learning Research*, 11(4), 2010.

Romstad, T., Costalba, M., Kiiski, J., Linscott, G., Nicolet, S., Geschwentner, S., VandeVondele, J., et al. Stockfish, 2021. URL https://stockfishchess.org.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Sasaki, F. and Yamashina, R. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zrT3HcsWSAt.

Sharma, P., Mohan, L., Pinto, L., and Gupta, A. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pp. 906–915. PMLR, 2018.

Singh, A., Jang, E., Irpan, A., Kappler, D., Dalal, M., Levinev, S., Khansari, M., and Finn, C. Scalable multi-task imitation learning with autonomous improvement. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2167–2173. IEEE, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Welinder, P., Branson, S., Perona, P., and Belongie, S. The multidimensional wisdom of crowds. *Advances in neural information processing systems*, 23:2424–2432, 2010.

Whitehill, J., Wu, T.-f., Bergsma, J., Movellan, J., and Ruvolo, P. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems*, 22: 2035–2043, 2009.

Yang, M., Levine, S., and Nachum, O. Trail: Near-optimal imitation learning with suboptimal data. *arXiv preprint arXiv:2110.14770*, 2021.

Zhang, S., Cao, Z., Sadigh, D., and Sui, Y. Confidence-aware imitation learning from demonstrations with varying optimality. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhang, T., McCarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg, K., and Abbeel, P. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635. IEEE, 2018.

A. Learning State Embeddings from Transitions

Although we can simply rely on the NLL loss defined in Eq. 4 to learn the state embedding f_ϕ (used in Eq. 1), it can be beneficial to consider the dynamics of the MDP environment as well, which may reveal more about the difficulty of each state. We do not have access to the MDP dynamics, but we can use the provided state transitions as samples from the dynamics to better learn state embeddings.

One popular approach is the DeepMDP framework (Gelada et al., 2019), which attempts to predict the environment dynamics in latent space. Using DeepMDP as an auxiliary task in the Atari 2600 domain has shown large performance improvements over model-free RL (Gelada et al., 2019). DeepMDP trains an embedding function by minimizing two losses: prediction of rewards and prediction of the distribution over next latent states. In our case, we do not have access to the rewards, so instead we replace the DeepMDP reward loss with our log-likelihood loss in Eq. 4.

First, we define a latent transition network $g_\psi : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}^d$ parametrized by ψ, which takes as input the current state embedding $f_\phi(s)$ and the action a, and outputs the predicted next-state embedding. Then, we take a transition tuple (s, a, s') and minimize the distance between the predicted next-state embedding $g_\psi(f_\phi(s), a)$ and the true next-state embedding $f_\phi(s')$ on a metric D, which we choose to be the smooth $L1$ metric.

$$\mathcal{L}(\psi, \phi) = \mathbb{E}_{(s,a,s')}\Big[D\big(g_\psi(f_\phi(s), a), f_\phi(s')\big)\Big] \quad (8)$$

We augment our loss in 4 with this auxiliary loss $\mathcal{L}(\psi, \phi)$ to help us learn the parameters ϕ used for the state embedding f_ϕ. Intuitively, $\mathcal{L}(\psi, \phi)$ encourages the learned embeddings f_ϕ to admit predictable transitions, which hopefully pushes the embeddings of similar states close together. At a high level, this process uses the trajectories in our dataset as samples of the MDP dynamics to help learn a better state embedding f_ϕ.

B. Concrete Example of Embedding Values

We show a concrete example on a simple 3-state and 3-action task. First, we show the state embedding and the optimal action distribution for the three states, one per row.

State Embedding and Optimal Policy

$f_\phi(s)$	a_1	a_2	a_3
{1,0}	.8	.1	.1
{1,1}	.0	.5	.5
{0,1}	.1	.1	.8

Next, we assume that we have two demonstrators, who are suboptimal w.r.t. the optimal action distribution. Note that demonstrator 1 is optimal in the first two states, and demonstrator 2 is optimal in the last two states.

Demonstrator 1

a_1	a_2	a_3
.8	.1	.1
.0	.5	.5
.3	.3	.4

Demonstrator 2

a_1	a_2	a_3
.4	.3	.3
.0	.5	.5
.1	.1	.8

Given a large dataset of demonstrations from demonstrators 1 and 2, ILEED and BC can recover the following policies.

ILEED(left) vs BC (right) recovery

ω_1	a_1	a_2	a_3	a_1	a_2	a_3
{50,-1.79}	.8	.1	.1	.6	.2	.2
ω_2	.0	.5	.5	.0	.5	.5
{-1.79,50}	.1	.1	.8	.2	.2	.6

log-likelihood: -0.807 -0.865

ILEED can learn the demonstrators embeddings (e.g. demonstrator 1 is better at skill 1, and demonstrator 2 is better at skill 2), and account for their suboptimalities to recover the optimal action distribution in all three states. The model can recover this via maximum likelihood, since it gives a better log-likelihood than standard BC. On the other hand, BC will average the demonstrations, and act suboptimally in the first and third state.

C. Additional Experiments

C.1. Relationship between reward and log-likelihood

First we test the relationship between the learned policy's performance and the corresponding NLL defined in Eq. 4. We did this by varying the number of restarts, and evaluating the performance of the policy with the highest likelihood. We ran this over 100 trials, each time collecting a new set of 1000 state-action pairs from 10 independent demonstrators with uniformly generated noise levels ranging $\alpha \sim U(0, 0.5)$. From this we estimated two values, the probability of the policy exceeding the mean demonstrator performance denoted by p, and the probability of the policy exceeding the best demonstrator's performance denoted by p^*. The results are provided in Table 4.

Table 4. Effect of Restarting
Estimated values for p and p^* computed over 100 trials.

Environment	Num. of Restarts		
	1	5	20
Empty	$0.94, 0.82$	$1.00, 1.00$	$1.00, 1.00$
Lava	$0.79, 0.63$	$0.99, 0.87$	$1.00, 0.95$
Obstacles	$0.37, 0.11$	$0.77, 0.31$	$0.84, 0.29$
Unlock	$0.25, 0.00$	$0.48, 0.00$	$0.59, 0.00$

Note that because the pre-trained policy's performance on

Table 5. Pre-trained Policy Performance
Mean episodic reward computed over 1000 runs.

Environment	Expertise level β			
	1.0	0.9	0.5	0.1
Empty	0.97	0.97	0.90	0.44
Lava	0.95	0.88	0.67	0.05
Obstacles	0.95	0.94	0.86	0.31
Unlock	0.87	0.90	0.83	0.27

the Unlock environment is particularly sensitive to noise, as shown by Table 5, we expect our algorithm to do relatively worse on this environment. On top of this, we see the pre-trained policy ($\beta = 1.0$) for *Unlock* does slightly worse than the noisy policy with $\beta = 0.9$. This is because the pre-trained MLP policy does not fully solve *Unlock* as it has no history component, getting stuck at certain states, hence adding a small amount of noise helps the policy escape such states.

As we increase the number of restarts, the policy's performance improves as well. Not only does this result show that our method is able to outperform the best demonstrator, it implies that our defined NLL is a good metric for evaluating the optimal policy. Based on this, we set the number of restarts to 20, where we note that given the small sizes of the datasets we utilize, restarting does not drastically affect computation time. Following this, we compared how our algorithm performed with respect to other baselines.

C.2. MiniGrid Lava and Unlock

We show the remainder of the results from Section 4.3 in Table 6. Overall GAIL showed inconsistent results for the *Unlock* environment, and performed poorly on *Lava*. Overall, ILEED was able to to take advantage of the noisy demonstrations, while BC suffered as β decreased.

Table 6. Effect of Varying Population Expertise β
We compute mean episodic reward of policies from different IL algorithms, 20 trials (for BC and ILEED).

β	Lava			Unlock		
	BC	**GAIL**	**ILEED**	**BC**	**GAIL**	**ILEED**
1	0.95	0.00	0.95	0.15	0.96	0.57
5	0.95	0.00	0.95	0.75	0.18	0.81
10	0.95	0.07	0.95	0.49	0.74	0.79
unif	0.95	0.00	0.95	0.79	0.01	0.78

C.3. MiniGrid Recovered Rankings

Our third experiment described in Section 4.5 of the paper showed that ILEED can learn to rank demonstrators. Here, we examine if a good ranking of the demonstrators correlates with good performance of the final policy. Using grid-world *Obstacles* as setup in Section 4.3, we designed two demonstrator populations with the same average expertise, but one has low range (0.15 to 0.85) (harder to rank) and one has high range (0.05 to 0.95) (easier to rank). Table 7 suggests that lower ranking loss relates to better reward, which is interesting since the average expertise of the two populations are identical.

Table 7. Pre-trained Policy Performance
Mean episodic reward computed over 1000 runs.

Env: Obstacles		
Range	**Reward**	**Rank Loss**
low	0.73	0.27
high	0.86	0.00

C.4. Chess policy

In Table 8 we show results for the chess-playing policy learned via imitation learning on the lichess database. As mentioned before, learning to play chess without knowing the rules of the game is extremely challenging, since we cannot improve via self-play. As a result, both BC and ILEEDlearn relatively poor policies (though still better than random). To interpret the results, note that we took the chess endgame positions from the database and polled a chess move from the learned policies. Then, we measured the difference in StockFish evaluation (ran for 2 seconds for each position) between the starting and ending positions. We set the evaluation of positions with inevitable mate to -100. One very crude way to interpret the results is that ILEED will blunder into an inevitable mate around 3% of the time, compared to 4% for BC and 12% for random. We note that this is not the most precise interpretation, since it ignores the change in evaluation of non-mating positions.

Table 8. Chess Policy Evaluation: average pawn loss per move.

Random	**BC**	**ILEED**
-12.30 ± 0.11	-4.00 ± 0.11	$\mathbf{-3.27 \pm 0.09}$

C.5. Ablation Studies

Lastly, we show our ablation studies discussed in Section 4.6. Using the same experimental setup as Table 3, we conducted an additional study to test the individual effect of state embeddings and demonstrator identities on ILEED's performance. In Table 9, SInd is the state-independent variant of ILEED that does not use the state embeddings, while DInd is the demonstrator-independent variant of ILEED that uses state embeddings but removes the demonstrator identities

treating them uniformly. As shown in the table, for varying levels of β (demonstrator suboptimality), both SInd and DInd are worse but each contribute to the high performance of ILEED in this multi-skill experiment.

Table 9. Performance of ILEED Ablations
Mean of episodic reward over 20 trials for the best demonstrator (BestD), BC, ILEED, as well as the state independent (SInd) and demonstrator independent (DInd) versions of ILEED.

β	**BestD**	**BC**	**SInd**	**DInd**	**ILEED**
.01	0.44	0.10	0.19	0.10	**0.70**
.20	0.76	0.68	0.74	0.64	**0.86**
1.0	**0.94**	0.88	0.89	0.87	0.89

D. Proofs

Proof of Proposition 3.1

Proof. We write the expectation as sampling first a demonstrator i and a state s, and then sampling an action from the ground truth policy $\pi(a|s, \omega_i^\star, \phi^\star, \pi_{\theta^\star})$:

$$\mathcal{L}(\theta) = -\max_{\phi,\omega} \mathbb{E}_{i,s}\mathbb{E}_{\pi(a|s,\omega_i^\star,\phi^\star,\pi_{\theta^\star})}\Big[\log \pi(a|s,\omega_i,\phi,\pi_\theta)\Big]$$

The inner expectation corresponds to the log-loss scoring rule, which we know is strictly proper. Hence, $\omega_i^\star, \phi^\star, \theta^\star$ maximizes the log term of each inner expectation. Therefore, by taking the max over ϕ, ω, we have that $\theta^\star$ is a minimizer of $\mathcal{L}(\theta)$. (But may not be the unique minimizer since ϕ and ω do not necessarily have to take on the values $\phi^\star, \omega^\star$). □

Proof of Proposition 3.3

Proof. Since $\mathcal{L}(\theta)$ is proper (Proposition 3.1) and Θ is well-specified, we know that at $\min_\theta \mathcal{L}(\theta)$ we have:

$$\min_\theta \mathcal{L}(\theta) = -\mathbb{E}_{i,s}\mathbb{E}_{\pi(a|s,\omega_i^\star,\phi^\star,\pi_{\theta^\star})}\Big[\log \pi(a|s,\omega_i^\star,\phi^\star,\pi_{\theta^\star})\Big]$$

On the contrary, behavioral cloning uses a single policy to model all demonstrators.

$$\mathcal{L}(\theta_{BC}) = -\mathbb{E}_{i,s}\mathbb{E}_{\pi(a|s,\omega_i^\star,\phi^\star,\pi_{\theta^\star})}\Big[\log \pi_{\theta_{BC}}(a|s)\Big]$$

Since the inner log-loss is a strictly proper loss function, the equality $\min_\theta \mathcal{L}(\theta) = \mathcal{L}(\theta_{BC})$ only holds when $\pi_{\theta_{BC}}(a|s) = \pi(a|s, \omega_i^\star, \phi^\star, \pi_{\theta^\star})$ for all i, meaning that all the demonstrators must have identical policies.

□

Proof of Lemma 3.4

Proof. We need to show that under the conditions in Lemma 3.4,

$$\mathcal{L}_\phi(\theta^\star) = -\max_\omega \mathbb{E}_{i,(s,a)}\Big[\log \pi(a|s,\omega_i,\phi,\pi_{\theta^\star})\Big]$$
$$< \mathcal{L}_\phi(\theta') = -\max_\omega \mathbb{E}_{i,(s,a)}\Big[\log \pi(a|s,\omega_i,\phi,\pi_{\theta'})\Big]$$

for all $\theta' \neq \theta^\star$, where demonstrations (s, a) for demonstrator i are drawn from $\pi(a|s, \omega_i, \phi, \pi_{\theta^\star})$.

Again using the fact that log-loss is strictly proper, the equality $\mathcal{L}_\phi(\theta^\star) = \mathcal{L}_\phi(\theta')$ only holds when the inner policies are equivalent for all states, i.e., $\pi(a|s, \omega_i, \phi, \pi_{\theta^\star}) = \pi(a|s, \omega_i', \phi, \pi_{\theta'})$ for all i, s, a.

Recall that in the discrete action model:

$$\pi(a|s,\omega_i,\phi,\pi_{\theta^\star}) = \rho_\phi(s,\omega_i)\pi_{\theta^\star}(a|s) + \frac{1-\rho_\phi(s,\omega_i)}{|\mathcal{A}|}$$

and in the continuous action model:

$$\pi(a|s,\omega_i,\phi,\pi_{\theta^\star}) = \sum_{j=1}^{k} \alpha_j \mathcal{N}(a;\mu_j(s),\sigma_j(s)/\rho_\phi(s,\omega_i))$$

To simplify notation, we will write the policy $\pi(\cdot|\cdot, \omega_i, \phi, \pi)$ as $\text{NOISE}(\pi, \rho_\phi(s, \omega_i))$. Conveniently, from the form of either the discrete/continuous action model, we see that $\text{NOISE}(\pi_{\theta^\star}, \rho_\phi(s, \omega_i)) = \text{NOISE}(\pi_{\theta'}, \rho_\phi(s, \omega_i'))$ if and only if $\pi_{\theta'} = \text{NOISE}(\pi_{\theta^\star}, \frac{\rho_\phi(s,\omega_i)}{\rho_\phi(s,\omega_i')})$, since the noise is multiplicative. In other words, an incorrect policy $\pi_{\theta'}$ can achieve optimal loss iff it is a noised version of the true policy $\pi_{\theta^\star}$, and that the ratio of the expertise levels correspond to the noise of $\pi_{\theta'}$ relative to $\pi_{\theta^\star}$. Moreover, this also tells us that the ratio must be the same for all demonstrators, so $\frac{\rho_\phi(s,\omega_i)}{\rho_\phi(s,\omega_i')} = \frac{\rho_\phi(s,\omega_j)}{\rho_\phi(s,\omega_j')}$ for all i, j.

However, when the expertise levels are intertwined, it may not be possible to set each expertise level to correspond to the desired noise. Recall that the expertise levels $\rho_\phi(s, \omega) = \sigma(\langle f_\phi(s), \omega\rangle)$.

Suppose that at a state s

$$f_\phi(s_0) = \alpha_1 f_\phi(s_1) + \alpha_2 f_\phi(s_2) + \ldots + \alpha_r f_\phi(s_r)$$

Then if $\text{NOISE}(\pi_\theta, \rho_\phi(s, \omega)) = \text{NOISE}(\pi_{\theta'}, \rho_\phi(s, \omega'))$ for all states $s_{0:m}$, we have that

$$\forall i, j, k : \frac{\rho_\phi(s_k,\omega_i)}{\rho_\phi(s_k,\omega_i')} = \frac{\rho_\phi(s_k,\omega_j)}{\rho_\phi(s_k,\omega_j')}$$

Or equivalently,

$$\forall i, k : \frac{\rho_\phi(s_k,\omega_i)}{\rho_\phi(s_k,\omega_i')} = C_k$$

Next we will rewrite to isolate the inner product.

$$\rho_\phi(s_k, \omega_i') = \rho_\phi(s_k, \omega_i)/C_k$$
$$\sigma(\langle f_\phi(s_k), \omega_i' \rangle) = \rho_\phi(s_k, \omega_i)/C_k$$
$$\langle f_\phi(s_k), \omega_i' \rangle = \sigma^{-1}(\rho_\phi(s_k, \omega_i)/C_k)$$

Now we can apply the linear dependence between s_0 and $s_{1:r}$ to get a relationship between the constants $C_{0:r}$. In particular, for all i:

$$\langle f_\phi(s_0), \omega_i' \rangle = \sum_{k=1}^{r} \alpha_k \langle f_\phi(s_k), \omega_i' \rangle$$
$$\sigma^{-1}(\rho_\phi(s_0, \omega_i)/C_0) = \sum_{k=1}^{r} \alpha_k \sigma^{-1}(\rho_\phi(s_k, \omega_i))/C_k)$$

Therefore, if no settings of the constants $C_{0:r}$ can satisfy the above conditions, then we cannot set the expertise levels to accommodate a noised version of the true optimal policy $\pi_{\theta^\star}$. In such a case, $\theta^\star$ is the unique minimizer of $\mathcal{L}(\theta)$.

□

E. Implementation Details

Overall we performed 3 sets of experiments corresponding to the simulated MiniGrid environments, the continuous control task Square from the Robomimic study, and chess. We go over these three sets of experiments, detailing the framework and parameter setup used to run our method ILEED as well as any other baselines we used for comparison. We note that working implementations of ILEED and BC are provided in our supplementary material, with code that can be used to reproduce the MiniGrid results. For GAIL, we utilized an implementation built on top of rllab's codebase (Duan et al., 2016), where the specific installation instruction are provided in the supplementary material. For experiments utilizing the Robomimic study we refer readers to the original codebase (Mandlekar et al., 2021), noting that to implement ILEED on top of their GMM policy class required minor changes to their framework which we detail below.

MiniGrid Experiments

For all MiniGrid experiments we relied on the provided Gym implementation (Chevalier-Boisvert et al., 2018) to simulate the environments, and used the well known stable baselines library (Raffin et al., 2021) to train our policies with PPO. We list the specific hyperparameters in Table 10, where we note that we used flattened observations for all MiniGrid experiments, relying on the standard 'MlpPolicy' class provided by stable baselines. To run ILEED and BC on the MiniGrid environments as discussed in Section 4.3 and 4.6 we utilized our own policy class consisting of a 3–layer MLP with 4 neurons in the hidden layer, ReLU nonlinearities between, and a softmax at the output. For training, we ran 2000 iterations utilizing two Adam (Kingma & Ba, 2017) optimizers, one for parameters θ, ϕ, and ψ with a learning rate of 1e-3, and the other for the expertise parameters ω with a smaller learning rate of 1e-2. For all experiments we relied on a two dimensional state-embedding f_ϕ which was paremetrized by a 3–layer MLP just like the policy. We list these parameters in Table 11 below, noting that our implementation is provided as part of the supplementary material. The parameters for GAIL are listed separately in Table 12, where we note that we utilized the recommended set of parameters provided by the implementation.

Table 10. Parameters used for PPO
Default implementation provided by stable baselines 3 (Raffin et al., 2021), where below we list the specific parameters we changed.

Parameter	Description
Policy Class	MlpPolicy
Update Steps	128
Num. of Environments	8
Batch Size	4
Learning Rate	0.00025
Timesteps	200000

Table 11. Parameters used for ILEED
Implementation parameters for ILEED are listed below. For BC we utilized the same parameters, removing the unnecessary components.

Parameter	Description
Policy	3–layer MLP + Softmax
Activation	ReLU
Hidden Size(s)	4
Num. Iterations	2000
Learning Rate θ, ϕ, ψ	0.001
Learning Rate ω	0.01
State Embedding Dimension	2

Robomimic Experiments

As stated in the main text, we utilized the implementation provided by the original study (Mandlekar et al., 2021), creating an instance of ILEED by changing the RNN-GMM policy class to match the annotator model we defined in Eq. 3. Specifically, we edited the file *robomimic/models/policy_nets.py*, changing the *RNNGMMActorNetwork* class by adding an optional scaling parameter to the variance which is outputted by the policy. We then utilized a separate Adam optimizer with a learning rate of 1e-4 to learn this parameter as done in the MiniGrid implementation. Although we do not include this specific implementation of ILEED in our supplementary material due to the size of the Robomimic library, we plan to upload our algorithm directly to the Robomimic codebase. To

Table 12. Parameters used for GAIL
Implementation parameters for GAIL are listed below.

Parameter	Description
Policy	Categorical MLP
Hidden Size(s)	$32, 32$
Latent Dimension	2
Batch Size	8000
Critic	Wassertstein
Critic Epochs	50
Critic Learning Rate	0.0001
Critic Dropout Prob.	0.6
Critic Penalty	1
Critic Gradient Norm	50
Recognition Epochs	50
Recognition Learning Rate	$.0001$
Scheduler k	20
TRPO Step Size	0.01

run our experiments, we utilize the exact same setup as the original study.

Chess Experiments

For the experiments on chess, we detail the architecture and parameters in Table 13 below.

Table 13. Parameters used for ILEED on chess
Implementation parameters for ILEED when used with chess are listed below. For BC we utilized the same parameters, removing the unnecessary components.

Parameter	Description
Policy	3–layer MLP + Softmax
Activation	ReLU
Hidden Size(s)	8
Num. Iterations	4000
Learning Rate θ, ϕ, ψ	0.001
Learning Rate ω	0.01
State Embedding Dimension	2

Computational Resources

All of the experiments were performed on a machine with the 8C/16T Intel-9900K CPU, 32GB RAM, and an RTX-3080 GPU.

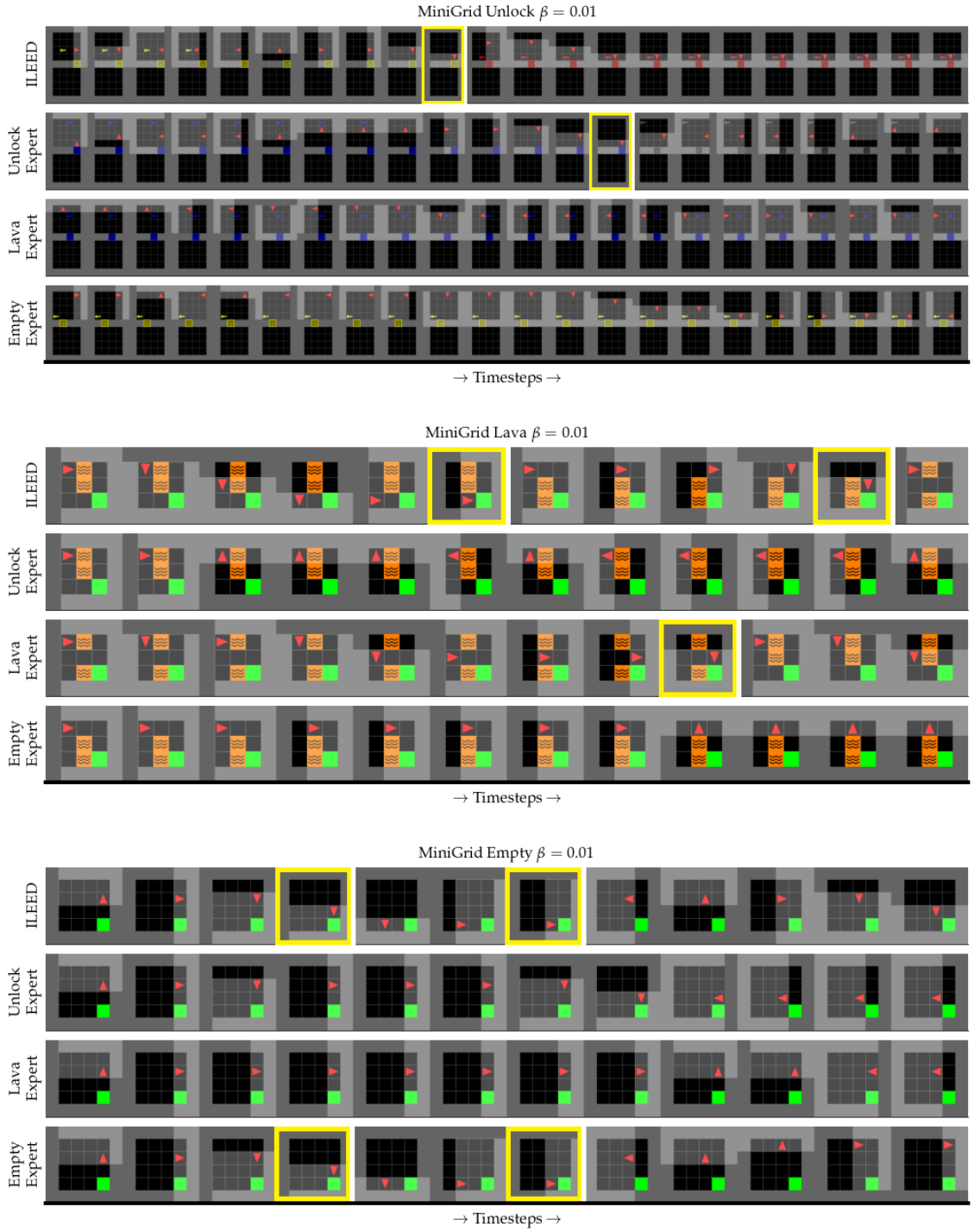


Figure 4. Trajectories corresponding to the MiniGrid experiments performed in Section 4.6. For each environment, we show the trajectory of the policy learned by ILEED on top and place the demonstrator trajectories below. The yellow bordered frames signify when the agent has successfully reached the goal. Since $\beta = 0.01$ corresponds to the nosiest setting, we see the demonstrators only act optimally in the environment they are skilled in. For example, we see that other than ILEED, the *Lava* expert is the only one to succeed in *Lava*. On the other hand, we see ILEED performing adequately in *Lava* and *Empty*, even showing some performance in the more challenging *Unlock* environment despite the poor quality of the dataset. Specifically we see for the *Unlock* environment, the policy trained by ILEED is able to unlock the yellow door, but fails to unlock the red door. We emphasize again that the results shown are for the worst setting of $\beta = 0.01$. As we increase β the demonstration quality improves, making it easier for ILEED to imitate multiple skills as shown in Table 3.