Learning from Ambiguous Demonstrations with Self-Explanation Guided Reinforcement Learning[*]

Yantian Zha[†], Lin Guan[†], and Subbarao Kambhampati

School of Computing & AI, Arizona State University, United States of America

Abstract

Our work aims at efficiently leveraging ambiguous demonstrations for the training of a reinforcement learning (RL) agent. An ambiguous demonstration can usually be interpreted in multiple ways, which severely hinders RL-Agents from learning stably and efficiently. Since an optimal demonstration may also suffer from being ambiguous, previous works that combine RL and learning from demonstration (RLfD works) may not work well. Inspired by how humans handle such situations, we propose to use self-explanation (an agent generates explanations for itself) to recognize valuable high-level relational features as an interpretation of why a successful trajectory is successful. This way, the agent can provide some guidance for its RL learning. Our main contribution is to propose the Self-Explanation for RL from Demonstrations (SERLfD) framework, which can overcome the limitations of traditional RLfD works. Our experimental results show that an RLfD model can be improved by using our SERLfD framework in terms of training stability and performance. Some additional details of this paper are in a linked supplemental material [1]

1 INTRODUCTION

Ambiguities commonly exist in communications among people. Due to possible vocabulary differences, when a human teacher attempts to train a robot by showing demonstrations (Robot Learning from Demonstration or LfD), such demonstrations may also suffer from being ambiguous. Maybe from the human's perspective, what he or she intends to demonstrate is clear, but the robot may still make sense of it in a different way. Consider an example illustrated in Fig. 1.1: a robot needs to push two components into two different target regions (L1 and L2) that have two different colors (blue and yellow). Imagine a human teleoperates the robot to push the ring object into L1 region in blue, and block object into L2 in yellow. The robot could interpret this demonstration as pushing the block into the yellow region and ring into the blue region or pushing the block into the region L2 and ring into L1.

If humans were in a robot's position, they may ask themselves: What are the features according to the other person that are more relevant to a good decision? After seeing the pushing demonstration, the human may start from a random hypothesize that the object-location relation (e.g. ring_at_L1) is more useful than the object-color relation (e.g. block_at_yellow). Then they would push ring and block always to L1 and L2 respectively, no matter which one is marked in blue or yellow. If the human fails the task once the yellow and blue colors are exchanged (L1 and L2 in yellow and blue respectively), they may consider the object-color relations (e.g. ring_at_blue and block_at_yellow) to be more relevant to a decision. This procedure is essentially self-explanation, which can be used to guide a learner to learn better (Siegler et al. 2002). According to (Siegler et al. 2002), self-explanations are inferences about relational connections among objects and events, like how procedures cause their effects, and how different structural components can affect a system.

Note that humans do not look at the world at the level of pixels, but could at more abstract cognition levels, like concepts ((Lake, Salakhutdinov, and Tenenbaum 2015; Sreedharan et al. 2020)), factors ((Körding et al. 2007)), and symbols ((Oaksford, Chater et al. 2007)). As for robots, since they take human advice (demonstrations), intuitively it would be beneficial if robots do disambiguation reasoning at a similar level to humans so that control-level learning can be guided. Therefore, we assume the robot has some background knowledge like classifiers for extracting important relations that can map continuous states to a set of predefined predicate symbols, which describe various relational connections among objects and/or events. A robotics expert could provide such background knowledge that helps solve

[*]This research is supported in part by ONR grants N00014-19-1-2119, N00014-16-1-2892, N00014-18-1-2442, N00014-18-1-2840, N00014-9-1-2119, AFOSR grant FA9550-18-1-0067, and a JP Morgan AI Faculty Research grant. The first author thanks Prof. Yezhou Yang for a helpful discussion and Prof. Chitta Baral for computing resource supports.

[†]equal contribution. Correspondence to yzha3@asu.edu

[1] https://drive.google.com/drive/folders/1DESh5R85Hk3t2U2vu_5OHxoztbi2mTcX?usp=sharing

a class of tasks (e.g. Pushing).

Therefore, the key insight behind our approach is: **by identifying which object-event relations from each interaction are likely to be more relevant to a decision, an agent can interpret why a trajectory is successful or unsuccessful so that its learning would be guided and improved**.

Reinforcement-**L**earning **f**rom **D**emonstration (**RLfD**) approaches (e.g. (Hester et al. 2017; Vecerik et al. 2017; Salimans and Chen 2018; Rajeswaran et al. 2017)) that use demonstrations to accelerate RL training have the benefits of only requiring a few demonstrations. However, traditional RLfD approaches may not be able to handle ambiguous demonstrations efficiently. An optimal trajectory from humans may also suffer from being ambiguous. This could make learners fail the task during training by simply "copying" the demonstration due to varying initial states, which is common in robotic tasks.

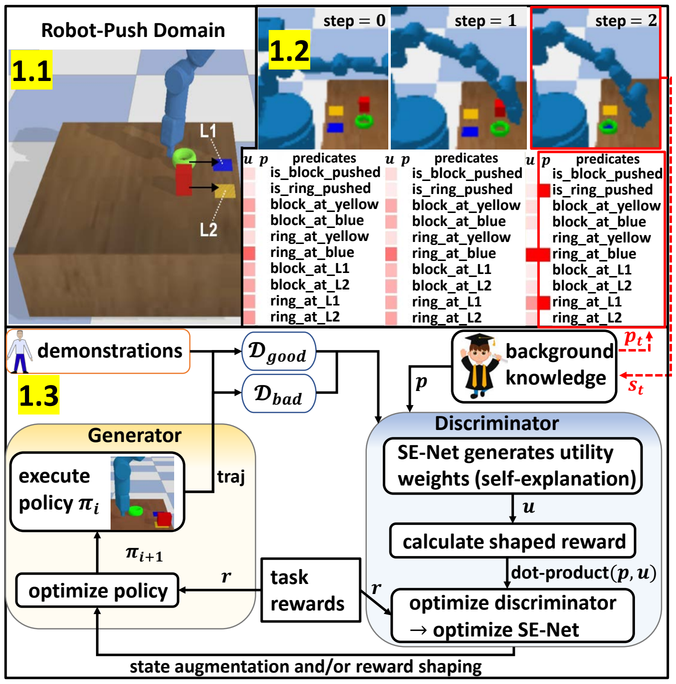

Figure 1: Sub-figure (1): the setting of the robot-push task. There are two target regions indexed by L1 and L2. L1 and L2 are also randomly assigned with colors either blue-yellow or yellow-blue. To finish the task, the ring and block should be pushed into the blue and yellow region respectively. Sub-figure (2): an example of a three-step robot execution with grounded predicates (**p**) and self-explanations (**u**) per step that is generated from our model. Sub-figure (3): The SERLfD framework that couples the learning of self-explaining (inside the blue region) and an RL-Agent (inside the yellow region). The learning of self-explanation is integrated into a discriminator. The policy can be viewed as a generator. A robotics expert provides background knowledge like predicates grounding procedures for robots, which allow robots to disambiguate demonstrations from non-experts.

To mitigate this issue, we (robotics experts) give robots the aforementioned background knowledge so that robots can understand non-expert demonstrations with human-relevant knowledge. Robots would then need to identify salient relations. We introduce a neural network called self-explanation network (SE-Net) to explicitly output a set of predicate utility weights per step. Each utility weight indicates how utilizable a ground predicate is for interpreting a decision. Therefore, the utility weights serve as an explanation hypothesis that self-explains which relations are important per step (e.g. the "u" columns in Fig. 1.2). Such information could guide RL-Agents to learn better. At the step 2 in the Fig. 1.2, while both "ring_at_blue" and "ring_at_L1" are satisfied at this time, weighting them differently can help the agent to connect only useful relations to its decision-making. This also shows that using task-agnostic background knowledge (e.g. human-understandable predicates for a class of tasks) would not remove ambiguity.

However, it is not straightforward to learn to predict proper utility weights for each predicate by using background knowledge. While a human demonstrator provides demonstrations for a specific task, we assume that she/he would not directly tell robots which predicates are important. In other words, we do not have such labels. Inspired by the aforementioned example of how a human may solve a similar problem, one promising direction is to motivate robots to contrastively discover important predicates by distinguishing between successful and unsuccessful trajectories. This requires integrating the ability of self-explaning and that of practising with the current self-explanation hypothesis. To this end, we propose a **S**elf-**E**xplaination for **R**einforcement-**L**earning **f**rom **D**emonstration (**SERLfD**) framework based on Generative Adversarial Inverse Reinforcement Learning (GAN-IRL) methods (Finn et al. 2016; Fu, Luo, and Levine 2017). In the GAN-IRL works, the learning of a non-linear reward predictor and an RL-Agent can guide each other in a generative-adversarial fashion. In our approach, as illustrated in Fig. 1.3, we modify the reward predictor in GAN-IRL by inserting into it a self-explanation network (SE-Net) that generates predicate utilities. While the whole reward predictor can be trained as usual via a discriminator loss, we essentially use the predicted utility weights (i.e. self-explanations) to guide the learning of RL-Agents.

Our main contribution is the SERLfD framework that contrastively and iteratively learns to use background knowledge to self-explain which high-level predicates are task-relevant and meanwhile perform the task. Our SERLfD algorithm is essentially a modification of traditional GAN-IRL methods that were used for Imitation Learning (or LfD). Our SERLfD algorithm combines the benefits of both worlds (RLfD and GAN-IRL). We extensively evaluate our SERLfD framework with multiple candidate RL-Agents in four continuous robotics domains and one discrete Pacman domain. Our evaluation demonstrates that SERLfD clearly outperforms traditional RLfD and GAN-IRL methods with better learning stability and higher scores even in challenging continuous control domains. To the best of our knowledge, our work opens the direction of learning to self-explain potentially ambiguous demonstrations to support RLfD.

2 RELATED WORKS

Deep RL from Demonstrations Many works have investigated the benefits of using RLfD frameworks. The work (Salimans and Chen 2018) uses states in demonstrations as starting points to train DRL with short-term interactions. As such, explorations only happen in a local region around a good state from demonstrations. (Hester et al. 2017) and (Vecerik et al. 2017) propose to use demonstrations to initialize the replay buffer so that the training of a deep Q-Net ((Hester et al. 2017)) or a deep deterministic policy gradient (DDPG) network ((Vecerik et al. 2017)) can be benefited. (Cruz Jr, Du, and Taylor 2017; Rajeswaran et al. 2017; Pfeiffer et al. 2018) propose to directly use demonstrations to initialize neural network parameters by pretraining it with an imitation learning objective. Besides pretraining a DRL model, imitation learning can also be used to construct an auxiliary imitation learning loss ((Vecerik et al. 2017; Nair et al. 2018)). The work (Gao et al. 2018) addresses the problem of DRL from imperfect (noisy and corrupted) demonstrations. (Gao et al. 2018) relies on reward information to perform a Q-function normalization over actions. However, learning from ambiguous demonstrations is an orthogonal issue: Such demonstrations could lead to the highest reward, may not necessarily be corrupted, but confuses the learner.

Imitation Learning from Ambiguous Demonstrations So far only a few works have attempted to do robot LfD from ambiguous demonstrations as in (Breazeal et al. 2006; Bensch and Hellström 2010; Morales and De la Rosa 2013; de Haan, Jayaraman, and Levine 2019). (Breazeal et al. 2006) and (Bensch and Hellström 2010) model demonstration ambiguity as the intention differences of human and robot. (Breazeal et al. 2006) proposes to let a human teacher and robot learner gradually improve the learning difficulty and quality. By explicitly modeling human's intention and belief in a Bayesian inference framework, the robot can discover conflicts, query humans, and learn better. (Bensch and Hellström 2010) models the ambiguity as a joint hypothesis space of multiple categories of concepts (e.g. colors, shapes). To disambiguate the demonstration is to reduce the hypothesis space. (Bensch and Hellström 2010) then proposes a concept learning approach to reduce the hypothesis space by using new demonstrations step by step. (Morales and De la Rosa 2013) models the demonstration ambiguity as a difference in demonstrated actions at the same (or similar) state. To disambiguate demonstrations, (Morales and De la Rosa 2013) adopts clustering algorithms and proposes a two-level clustering approach to simultaneously categorize similar situations and ambiguous actions in each situation. Another work (Brown, Niekum, and Petrik 2020) proposes to use Bayesian optimization to infer reward uncertainty learned by an IRL algorithm, which has the potential of doing imitation learning from ambiguous demonstrations. However, (Brown, Niekum, and Petrik 2020) mainly focuses on the IRL side and evaluates in simple domains. The work (de Haan, Jayaraman, and Levine 2019) addresses distribution drift problems (Ross and Bagnell 2010; Ross, Gordon, and Bagnell 2011) in imitation learning by avoiding causal misidentification caused by using ambiguous demonstrations. While (de Haan, Jayaraman, and Levine 2019) touches on the RL, it is not an RLfD approach because the task (environment) rewards are not used. In contrast, according to (Rajeswaran et al. 2017) the RLfD approaches and our work can naturally handle distribution drift problems since we directly use task rewards.

3 BACKGROUND

Reinforcement Learning and Reward Shaping In our work, we consider a finite-horizon and discounted Markov decision process (MDP) model that can be learned by RL methods. We further assume that the reward function $r(s_t, a_t)$ is sparse, i.e., $r(s_t, a_t) = 0$ in most of the states $s \in S$. Training RL agents in sparse rewards environments could be challenging due to the delayed training signals from effective feedback. One extensively adopted way to ameliorate such training is by adding reward shaping, which provides denser training signals so that the agent could obtain valuable feedback much sooner. However, we need to be careful at providing shaped rewards. As demonstrated in (Ng, Harada, and Russell 1999), a poorly-designed reward shaping function may cause the converged optimal policy to shift as against the one under original rewards. (Ng, Harada, and Russell 1999) proves that potential-based reward shaping function, which follows the form in Eq. 1, is the only class of reward shaping function that can guarantee the invariance of optimal policies.

$$\hat{r}(s_t, a_t) = r(s_t, a_t) + \lambda\Phi(s_{t+1}) - \Phi(s_t) \tag{1}$$

where $\hat{r}(s_t, a_t)$ and $r(s_t, a_t)$ denote the shaped and original reward respectively, λ is an adjustment parameter, Φ denotes any real-valued function, and $\lambda\Phi(s_{t+1}) - \Phi(s_t)$ is the reward shaping term. Note that **we set λ to 1 in the rest of paper**.

Inverse Reinforcement Learning (IRL) IRL is about the problem that the reward function in an MDP model is unknown and needs to be discovered from expert demonstrations. Typically in an IRL framework, the unknown reward function is modeled by a parameterized function of certain features (Ziebart et al. 2008). The design of our SERLfD framework is based on GAN-IRL methods like the Guided Cost Learning (GCL (Finn, Levine, and Abbeel 2016)) optimized by Generative-Adversarial Networks (GAN-GCL (Finn et al. 2016)). GCL and GAN-GCL propose to couple IRL and RL together to do continuous control learning with unknown rewards. Such frameworks allow learning nonlinear rewards that help under complex and unknown dynamics (essentially a model-free IRL).

GAN-GCL integrates IRL and RL by viewing the RL model as a generator and the IRL model as a discriminator that are trained in GAN formulation. The IRL model provides rewards for the RL model. The RL model is trained to gradually shift its sampling distribution to match that of demonstrations. The IRL model is trained to distinguish sampled trajectories from demonstration trajectories, by using the binary cross-entropy loss in Eq. 2. Using such a discriminator to provide rewards for policy learning can solve their Imitation Learning problems.

$$L_{irl}(D_\theta) = E_{\tau\sim p}[-logD_\theta(\tau)] + E_{\tau\sim q}[-log(1 - D_\theta(\tau))] \tag{2}$$

where D_θ denotes a discriminator model. τ denotes a trajectory. The work Adversarial IRL (Fu, Luo, and Levine 2017) further proposes to replace trajectory τ with state-action pairs, which makes the training more stable:

$$D_\theta(s, a) = \frac{exp\{\hat{r}_\theta(s, a)\}}{exp\{\hat{r}_\theta(s, a)\} + q(a|s)} \tag{3}$$

where $\hat{r}_\theta(s, a)$ denotes the estimation of reward feature on a pair of state and action. We refer to this version of GAN-GCL as State-Action-GAN-GCL (**SA-GAN-GCL**).

4 THE SERLfD FRAMEWORK

In this section, we introduce our SERLfD framework (Fig. 1.3) which couples two objectives together. One is to learn self-explanations that identify task-relevant relations by distinguishing between successful and unsuccessful trajectories. The other is to encourage RL-Agents to use self-explanations so that RL training can be improved. We introduce a self-explanation network (SE-Net) that maps a state to a set of utility weights $\boldsymbol{u}$ (self-explanation) that corresponds to a set of grounded predicates values $\boldsymbol{p}$. The predicates and their recognition classifiers can be viewed as background knowledge that a robotics expert offered for helping a robot to handle a class of tasks. We also propose to improve an RLfD agent with self-explanations by either 1) state augmentation, i.e. $\langle s \rangle$ becomes $\langle s, p, u \rangle$; or 2) reward augmentation, i.e. $\hat{r}_\theta(s, u, a, r(s, a))$ where $r(s, a)$ is task reward function that comes from an RL environment. The task reward also serves as an indicator that decides whether a trajectory is successful or not. Since we rely on task rewards for learning both SE-Net and RL-Agent, we do not break RLfD assumptions that demonstrations are used to guide RL to maximize accumulative task rewards. Our RL-Agents can be any deep RL algorithm that is adjusted to learn from demonstrations a policy that optimizes accumulative task rewards, e.g. (Hester et al. 2017; Vecerik et al. 2017). To contrastively learn the SE-Net, we merge the objective of learning to self-explain into the learning of GAN-IRL frameworks ((Finn et al. 2016; Fu, Luo, and Levine 2017)). The discriminators in GAN-IRL frameworks are trained by distinguishing between two different distributions of trajectories: demonstrations or generated samples. Unlike GAN-IRL, the discriminator in SERLfD includes the SE-Net and is trained differently by distinguishing between successful and unsuccessful trajectories. We will explain what are grounded predicate values (Sec. 4.1), how to learn to "self-explain" (Sec. 4.2), how RLfD learning could be benefited from self-explanations while RL-Agents also adapts the learning of SE-Nets further by providing new samples (Sec. 4.3).

4.1 Grounded Predicate Values

A predicate is a logical formulation that describes a relationship among several objects. The objects that a predicate concerns would form the arguments of the predicate. Take for instance a predicate "pushed(obj)" in the aforementioned robot-pushing domain, "obj" is an argument that can be grounded with "block" or "ring". When the argument is grounded, the predicate becomes a grounded predicate, e.g. "pushed(block)". In our work, we use a **binary predicate variable** to represent if a grounded predicate is satisfied or not. For example, "pushed(block)" can be represented by a binary predicate variable "is_block_pushed". When it is satisfied, we assign this binary predicate variable with a value of 1, otherwise -1. We may have a set of grounded predicates (and thus a set of binary predicate variables), according to background knowledge, to describe the most important relations. We assume that the classifiers for detecting the satisfaction of such predicates are available. The rapidly growing computer vision community is already able to provide high-quality visual recognizers that detect objects and relations (e.g. through scene graph analysis (Yang et al. 2018b)). There are also works that improve RL by leveraging ground-truth symbolic high-level knowledge, e.g. (Lyu et al. 2019; Toro Icarte et al. 2018; Camacho et al. 2019; Yang et al. 2018a).

4.2 Training the SE-Net in a Discriminator

The input to the Discriminator (in Fig. 1.3) is obtained by sampling a batch of experiences and grounded predicate values from buffers $\mathcal{D}_{good}$ and $\mathcal{D}_{bad}$. With an experience, SE-Net takes a state s as input and generates a set of **predicate utility weights** $\boldsymbol{u}$: $u = SE{-}Net(s)$. A predicate utility weight serves as a quantitative estimation of how utilizable the corresponding predicate variable is to interpret why making a decision is promising, given a state. The values in $\boldsymbol{u}$ may highlight some predicate variables and play down the others. This way, we identify salient relations for a decision, since a predicate is an abstracted representation of a relation. Even in extreme cases that all predicates have satisfied/unsatisfied groundings, probably only a subset of them are recognized to be useful. Then the utility weights for those predicates would probably have larger absolute values.

Self-explaining can be seen as finding a set $\boldsymbol{u}$ of utility weights. The number of output utility weights per step should match the number of grounded predicate values. The neural network architecture of SE-Net is flexible and should match the state-space. In some of our experiments, our state-space consists of features like object poses. Then SE-Net could be implemented by using a multilayer perceptron.

By taking the dot-product of $\boldsymbol{u}$ and grounded predicate values, we can obtain a scaler value $h_\theta(s)$ that we would use later:

$$h_\theta(s, u) = \sum_{i=0}^{k-1} u_i \cdot P(s)_i \tag{4}$$

where $P(s)$ denotes the grounded predicate values extracted from state s, and k denotes the number of predicates. We explain how we obtain the function P in the **Sec. 2** of linked **supplemental manuscript** (footnote 1).

Recall that GAN-GCL (Finn et al. 2016) and SA-GAN-GCL (Fu, Luo, and Levine 2017) train IRL and RL models together as a GAN (a coupling of discriminator and generator). Traditionally, the IRL model is trained by differentiating demonstration and sampled trajectories. The IRL model (discriminator) also provides rewards for training the RL-Agent to **imitate** the demonstrations. However, solving a SERLfD problem is closer to solving an RLfD problem as in (Vecerik et al. 2017; Hester et al. 2017; Salimans and Chen 2018): They use demonstrations to initialize a replay buffer and bootstrap the learning **of maximizing task rewards.**

Now an important question is how can we integrate the learning of the SE-Net in a discriminator and an RL-Agent. Inspired by the difference between RLfD that requires task rewards and GAN-GCL based Imitation Learning that only needs a discriminator (discussed above Eq. 2), we decide to use the predicted utility weights ***u*** to shape the task rewards. The value $h_\theta(s)$ in Eq. 4 is essentially a reward shaping term that rely on both SE-Net and background knowledge (predicates and $P(s)$). Then we can compute the prediction of shaped rewards $\hat{r}$ as:

$$\begin{aligned}\hat{r}_\theta(s_t, u_t, a_t, s_{t+1}, u_{t+1}, r) = \\ r(s_t, a_t) + h_\theta(s_{t+1}, u_{t+1}) - h_\theta(s_t, u_t)\end{aligned} \quad (5)$$

where $\hat{r}_\theta()$ models the prediction of shaped rewards $\hat{r}(s_t, a_t)$, and $h_\theta(s)$ is computed in Eq. 4. $r(s_t, a_t)$ is the task rewards provided by an RL environment. The formulation of Eq. 5 is based on Eq. 1 and shares similarities with the prediction of shaped rewards in work (Fu, Luo, and Levine 2017). However, different from (Fu, Luo, and Levine 2017), we include task reward r to train the agent to accomplish the task instead of merely imitating. In the rest of this paper, we **shorten the** $\hat{r}_\theta()$ **with long lists of inputs to** $\hat{r}_\theta(s, u, a, r)$.

Since our self-explanation ***u*** only "augments" the task rewards, our discriminator is trained to distinguish between successful and unsuccessful trajectories. In the traditional GAN-IRL works, the task reward is missing. All demonstrations are labeled as good and all sampled trajectories are labeled as bad. But incorporating task rewards can better motivate Self-Explainer to learn to discriminatively recognize which predicates are more useful for solving a task. Sampled trajectories that can accomplish a task, in our problem, should be treated the same as demonstration trajectories. Remember in Fig. 1.3, we store a sampled trajectory into either $\mathcal{D}_{good}$ or $\mathcal{D}_{bad}$ depending on if it accomplishes a task. The discriminator loss L_{SE} for training the SE-Net can be formulated as:

$$\mathrm{L}_{SE} = E_{(s,a)\sim\mathcal{D}_{good}}[-logD(\hat{r}_\theta(s, u, a, r))] + E_{(s,a)\sim\mathcal{D}_{bad}}[-log(1 - D(\hat{r}_\theta(s, u, a, r)))]$$

where $u = SE{-}Net(s)$, the discriminator function $D()$ is formulated in Eq. 3, and $\hat{r}_\theta(s, u, a, r)$ in Eq. 5.

4.3 Improving RL-Agent (Generator) with Self-Explanation

The yellow area in Fig. 1.3 depicts how we train the RL-Agent with the help of predicted self-explanations. Note that the SE-Net learns a complex non-linear function to "explain" a state. SE-Nets also provide more detailed guidance (predicate utilities) rather than merely a numerical reward prediction. In our work, we propose two ways for an RL-Agent to take advantage of SE-Nets. One way is to augment RL states by concatenating them with grounded predicate values and predicted utility weights (self-explanation): $\langle s \rangle$ becomes $\langle s, p, u \rangle$. Then policy network in RL-Agents becomes: $a \sim \pi_\theta(s, p, u)$. Another way is to estimate a shaped reward $\hat{r}_\theta(s, u, a, r)$ by using Eq. 4 and 5. The former provides more detailed guidance whereas the (shaped) rewards are more direct learning signals for RL-Agents. Intuitively, a good explanation provides better guidance for the RL-Agent to sample a trajectory that in turn makes the discriminator harder to distinguish, which adapts the SE-Net further.

The SERLfD learning is summarized in **Algorithm. 1**.

5 EVALUATION

Since self-explanations should play a general role in improving an RL-Agent that is trained with ambiguous demonstrations, we evaluate our SERLfD framework in multiple domains and use different candidate deep RL models as the RL-Agent. We evaluate the RL learning performance in this section and the predicted self-explanations in our **supplemental video** (footnote 1). We design our evaluation to answer the following questions: 1) Can SERLfD outperform RLfD? 2) Which of the two ways of using self-explanation to guide an RL-Agent (state or reward augmentation) is more helpful? 3) What could happen if the self-explanation is used to define the entire reward rather than just the reward shaping? 4) Do self-explanations play a general role in supporting an RL-Agent to learn from ambiguous demonstrations or self-explanations are only effective for certain of the RL models? 5) Since our SERLfD combines the benefits of RLfD and GAN-IRL, does our SERLfD outperform a state-of-the-art GAN-IRL as an Imitation Learning method? 6) Does SERLfD help in both continuous and discrete domains?

To answer 1, we compare the performance of an RLfD model and the same one supported by SE-Net. To answer 2, we do extensive studies of how RL agents can use self-explanations by removing the predicate utility weights from the input of an RL-Agent (RLfD+SE+**nu**), or by removing the reward shaping terms (RLfD+SE+**nrs**). To answer 3, we remove task rewards and only use the dot-product of predicate utility weights and predicate values as rewards to train RL agents (RLfD+SE+**ntr**). To answer 4, we investigate a diverse set of RL-Agents. To answer 5, we compare SERLfD with the SA-GAN-GCL (for Imitation Learning) proposed in (Fu, Luo, and Levine 2017). To answer 6, we evaluate our models in three continuous robotic control domains and one discrete Pacman domain.

5.1 Experiments in Continuous Domain

We start by evaluating SERLfD in a **Robot-Push** domain (as described in Fig. 1) that a continuous control model would need to be learned. We used a Fetch Mobile Manipulator (Wise et al. 2016) that has a 7-DoF arm in PyBullet simulator (Coumans and Bai 2016). We also fixed its mobile base in experiments. In this Robot-Push domain, there are two target regions indexed by L1 and L2. **Note that this indexing is static**. By saying a "region", we mean a square that each side is approximately 0.1m. The region L1 and L2 are assigned with either blue-yellow or yellow-blue colors. The predicate variables are: {is_block_pushed, is_ring_pushed, block_at_yellow, block_at_blue, ring_at_yellow, ring_at_blue, block_at_L1, block_at_L2, ring_at_L1, ring_at_L2}. The task

Algorithm 1: The SERLfD Learning Algorithm

INPUT: A dataset of human demonstrations, an environment with reward function r, state space S, action space A, and all hyper-parameters

```
1: Initialize the weights of an RL-Agent and a Self-Explainer
2: Initialize buffers $\mathcal{D}_{good}$ and $\mathcal{D}_{bad}$ for training the Self-Explainer and $D_{RL}$ for RL from Demonstrations as in (Hester et al. 2017; Vecerik et al. 2017)
3: Store expert experiences into $\mathcal{D}_{good}$ and $\mathcal{D}_{RL}$. Pretrain the RL-Agent with experiences sampled from $\mathcal{D}_{RL}$
4: Sample $K$ trajectories with a random policy and add them to $D_{RL}$. Also add successful and unsuccessful trajectories to $\mathcal{D}_{good}$ and $\mathcal{D}_{bad}$ respectively.
5: for $episode = 1; episode \leq N; episode++$ do
6:     Sample experiences (including grounded predicate values) from $\mathcal{D}_{good}$ and $\mathcal{D}_{bad}$    ▷ Train Self-Explainer
7:     Compute utility weights $u$ and shaped reward prediction $\hat{r}_\theta(s_t, u_t, a_t, s_{t+1}, u_{t+1}, r)$ by using SE-Net, Eqs. 4 and 5
8:     Update the SE-Net via binary cross entropy loss $L_{SE}$ to distinguish successful experiences from unsuccessful experiences
9:     Sample experiences with grounded predicate values from $\mathcal{D}_{RL}$    ▷ Train RL-Agent
10:    Run SE-Net on sampled states to obtain utility weights $u$
11:    Augment input states with grounded predicate values and utility weight values
12:    Augment rewards with predicted shaped reward by using Eq. 5
13:    Update RL-Agent with the augmented experiences
14:    Use RL-Agent to sample a new trajectory and add it to $D_{RL}$. If the trajectory is successful, it would also be added to $\mathcal{D}_{good}$. Otherwise, it would also be added to $\mathcal{D}_{bad}$    ▷ Sample a new trajectory
15: end for
16: return a trained RL-Agent and SE-Net
```

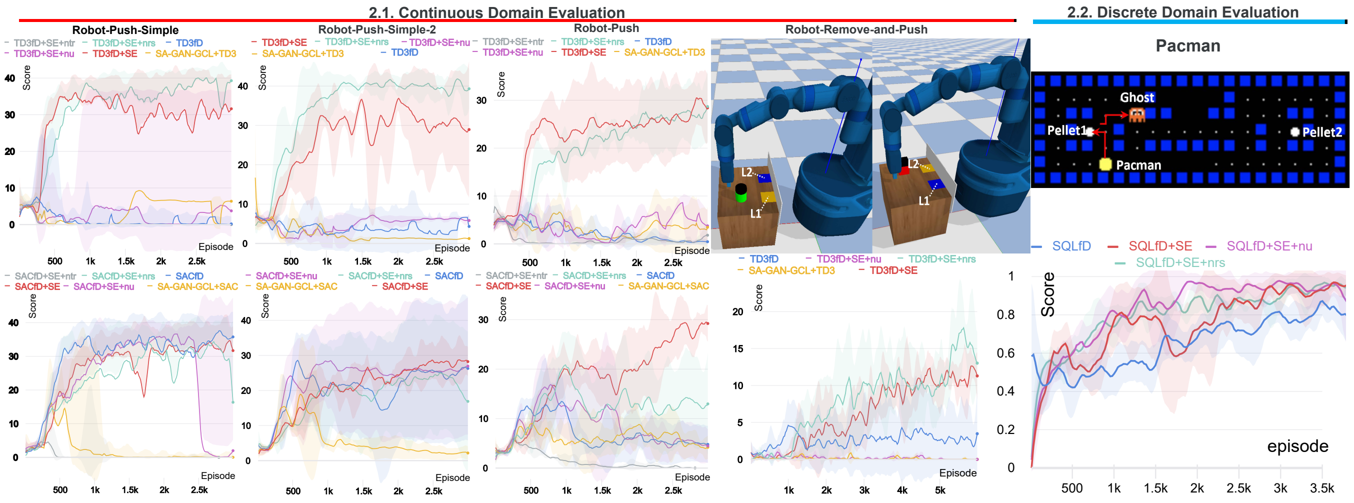

Figure 2: 2.1. Learning curves of training the baseline RLfD agents (TD3fD/SACfD), RLfD with SE-Nets (RLfD+SE), the extensive studies of how RLfD can use self-explanations (RLfD+SE+**nu** and RLfD+SE+**nrs**), using the self-explanation to form the entire rewards for RL-Agents (RLfD+SE+**ntr**), and an Imitation Learning agent built by using RL in the original SA-GAN-GCL framework (Fu, Luo, and Levine 2017); and 2.2. The Pacman domain and the learning curves with the RL-Agent SQLfD. For each curve, we run three times of each algorithm and report the mean and standard-deviation, which are plotted in the bold and lighter color region respectively. y-axis values are scores that each is measured as an average over 100 episodes. x-axis values are episodes.

is to push the block and ring into yellow and blue regions respectively, no matter which is L1 or L2. Our design of state-space follows the FetchPush-v0 environment made by OpenAI-Gym (OpenAI). The state-space consists of the positions and orientations of the objects in domain w.r.t the world frame. Such objects include the end-effector, ring, block, regions in blue and yellow colors, and regions L1 and L2. The action space is defined as a 4-tuple: [translation_x, translation_y, translation_z, yaw_angle] of the end-effector w.r.t world frame. Once the block is pushed to the yellow region and ring to the blue region, the robot finishes a task. The task reward function is sparse: In most situations, $r(s, a)$ is zero; If robot pushes an object towards its target location to be δd closer, it receives a reward of $100 * \delta d$; If either ring or block is pushed into its target region, the robot receives a reward of 25; If both ring and block are pushed into their target regions, the robot receives a reward of 50.

We also designed two simplified versions of the aforementioned Robot-Push domain, named **Robot-Push-Simple** and **Robot-Push-Simple-2**. In the Robot-Push-Simple domain, the predicate variables are: {is_block_pushed, is_ring_pushed, block_at_yellow, block_at_blue, ring_at_yellow, ring_at_blue}. The region L1 and L2 are always in blue and yellow. Other details are the same as the Robot-Push domain. The Robot-Push-Simple-2 domain is identical to the Robot-Push-Simple domain except for that it has a larger predicate set that is the same as the Robot-Push domain.

We use two state-of-the-art RLfD baselines for continuous control tasks: Twin-Delayed DDPG (Fujimoto, Van Hoof, and Meger 2018) from Demonstrations (TD3fD), and Soft-Actor Critic (Haarnoja et al. 2018) from Demonstrations (SACfD). To collect demonstrations, we use a keyboard to input control commands. We collected 8 trajectories (averagely 17 steps each) for the Robot-Push-Simple and Robot-Push-Simple-2 domain and 15 trajectories (averagely 19 steps each) for the Robot-Push domain.

Robot-Push-Simple, Robot-Push-Simple-2, and Robot-Push The results are reported in Fig. 2.1. We extensively study what is the best way for an RL-Agent to use self-explanations by introducing the models RLfD+SE+**nrs** and RLfD+SE+**nu** along with our main model RLfD+SE. Please note that all of the three models that use our self-explainer – either for reward shaping, state augmentation, or both – are our models that require a SERLfD framework and SE-Net. The results present multiple interesting findings. First, the RL-Agents with SE-Nets generally can be trained more stably and achieve higher scores within a shorter time. One exception is when we use SACfD as the baseline RL-Agent in Robot-Push-Simple and Robot-Push-Simple-2. This is because the two domains have a lower degree of ambiguity. The entropy-driven exploration in SACfD can find the optimal policy without the help of self-explanations. Second, the results show that using self-explanations to augment states is better than constructing shaped rewards with self-explanations. The reason could be 1) the state augmentation provides detailed information of self-explanation to RL-Agents, and 2) due to RL-Agents' explorations, the reward shaping that contains self-explanations would not be fully used at the beginning stage of training. We also evaluate an incorrect way of using self-explanation to form the entire rewards (RLfD+SE+**ntr**). Theoretically, RLfD assumes the task/extrinsic reward is available. Demonstrations are used to guide RL agents to find a policy that maximizes the accumulative extrinsic reward faster. Using the self-explanation to form the entire reward breaks this assumption of RLfD. Third, the domain Robot-Push is harder than Robot-Push-Simple-2 but they have the same predicates. Our results for the two domains show that the RLfD agents with SE-Nets can maintain good performance even for harder tasks.

Robot-Remove-and-Push Based on the aforementioned results, we evaluate some interesting models in this more complex robot domain. There are two target regions indexed with L1 and L2. The region L1 and L2 are assigned with either blue-yellow or yellow-blue colors. L1 and L2 are fixed whereas blue and yellow are exchangeable. In each episode, either a block or a cylinder would show up. Both have a black cover at the top. If their initial poses are on the left side of the table, the robot needs to push them to their target regions **with the black cover removed**. The target regions for the block and cylinder are blue and yellow regions respectively. Robots only get a reward of +50 when they accomplish the task. This domain supports a more complex task of 20 predicates (Sec. IV in the supplemental material in footnote 1). We collected 16 demonstrations (averagely 5 steps each). We use TD3fD as a representative of RLfD models. We report the results in the last column of Fig. 2.1 which shows clear benefits from learning self-explainers.

5.2 Experiments in Discrete Domain

We report our results in a discrete **Pacman** Domain (Fig. 2.2) to demonstrate if self-explanation is helpful with discrete state and action spaces. After taking the power pellet, the Pacman should try to eat ghosts within a fixed amount of time. Once the Pacman eats all of the randomly-wandering ghosts it completes the task. Then it gets a reward of 1 and the current episode ends, otherwise, the reward is 0. Two predicate variables, {ghost_nearby, eat_capsule}, are provided to describe whether the ghost is close to the Pacman and whether the Pacman has eaten a pellet. The ambiguity in this domain lies in the proper time to eat a pellet. In the demonstrations (e.g. the red trajectory in Fig. 2.2), the agent rushed to eat the pellet because the ghost was coincidentally nearby. But normally, the agent should wait to eat the pellet until the ghost approaches. Our RL-Agent is Soft Q-Learning (Haarnoja et al. 2017) from Demonstration (SQLfD). We collected 5 demonstrations (averagely 18 steps each). In our training, each episode has at most 2000 steps. From the results, we can conclude that the RL-Agents with SE-Nets perform better.

6 CONCLUSION

In this work, we propose the **S**elf-**E**xplaination for **R**einforcement-**L**earning **f**rom **D**emonstration (**SERLfD**) framework for allowing RLfD agents to efficiently use even ambiguous demonstrations by doing self-explanations. Our extensive evaluation shows appealing benefits of training RLfD from learning self-explanations in one discrete Pacman and four challenging robotics domains. The experimental advantage also suggests that our SERLfD could lead to compelling practical applications since ambiguous demonstrations could frequently happen when robots cohabitate with non-practitioner humans in real-worlds. In this work, we open the direction of using self-explanation to help RLfD learning. We believe there are extensive benefits of learning self-explanations for solving other robot learning problems, which could be investigated in future works. Future extensions may also consider improving the self-explanation mechanism that works under more challenging settings.

References

Bensch, S.; and Hellström, T. 2010. On ambiguity in robot learning from demonstration. In *Intelligent autonomous systems*, 47–56. Citeseer.

Breazeal, C.; Berlin, M.; Brooks, A.; Gray, J.; and Thomaz, A. L. 2006. Using perspective taking to learn from ambiguous demonstrations. *Robotics and autonomous systems*, 54(5): 385–393.

Brown, D. S.; Niekum, S.; and Petrik, M. 2020. Bayesian Robust Optimization for Imitation Learning. *arXiv preprint arXiv:2007.12315*.

Camacho, A.; Icarte, R. T.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *IJCAI*, volume 19, 6065–6073.

Coumans, E.; and Bai, Y. 2016. Pybullet, a python module for physics simulation for games, robotics and machine learning.

Cruz Jr, G. V.; Du, Y.; and Taylor, M. E. 2017. Pre-training neural networks with human demonstrations for deep reinforcement learning. *arXiv preprint arXiv:1709.04083*.

de Haan, P.; Jayaraman, D.; and Levine, S. 2019. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, 11698–11709.

Finn, C.; Christiano, P.; Abbeel, P.; and Levine, S. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*.

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

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

Fujimoto, S.; Van Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*.

Gao, Y.; Xu, H.; Lin, J.; Yu, F.; Levine, S.; and Darrell, T. 2018. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*.

Haarnoja, T.; Tang, H.; Abbeel, P.; and Levine, S. 2017. Reinforcement learning with deep energy-based policies. In *ICML'17 Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 1352–1361.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Dulac-Arnold, G.; et al. 2017. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*.

Körding, K. P.; Beierholm, U.; Ma, W. J.; Quartz, S.; Tenenbaum, J. B.; and Shams, L. 2007. Causal inference in multisensory perception. *PLoS one*, 2(9): e943.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338.

Lyu, D.; Yang, F.; Liu, B.; and Gustafson, S. 2019. SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2970–2977.

Morales, O. C. F.; and De la Rosa, R. F. 2013. Ambiguity analysis in learning from demonstration applications for mobile robots. In *2013 16th International Conference on Advanced Robotics (ICAR)*, 1–6. IEEE.

Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6292–6299. IEEE.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Oaksford, M.; Chater, N.; et al. 2007. *Bayesian rationality: The probabilistic approach to human reasoning*. Oxford University Press.

OpenAI. ????

Pfeiffer, M.; Shukla, S.; Turchetta, M.; Cadena, C.; Krause, A.; Siegwart, R.; and Nieto, J. 2018. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3(4): 4423–4430.

Rajeswaran, A.; Kumar, V.; Gupta, A.; Vezzani, G.; Schulman, J.; Todorov, E.; and Levine, S. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

Ross, S.; and Bagnell, D. 2010. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 661–668.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. 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*, 627–635.

Salimans, T.; and Chen, R. 2018. Learning Montezuma's Revenge from a Single Demonstration. *arXiv preprint arXiv:1812.03381*.

Siegler, R. S.; et al. 2002. Microgenetic studies of self-explanation. *Microdevelopment: Transition processes in development and learning*, 31–58.

Sreedharan, S.; Soni, U.; Verma, M.; Srivastava, S.; and Kambhampati, S. 2020. Bridging the Gap: Providing Post-Hoc Symbolic Explanations for Sequential Decision-Making Problems with Black Box Simulators. *arXiv preprint arXiv:2002.01080*.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2018. Teaching multiple tasks to an RL agent using LTL. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 452–461.

Vecerik, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; and Riedmiller, M. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*.

Wise, M.; Ferguson, M.; King, D.; Diehr, E.; and Dymesich, D. 2016. Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots*.

Yang, F.; Lyu, D.; Liu, B.; and Gustafson, S. 2018a. Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. *arXiv preprint arXiv:1804.07779*.

Yang, J.; Lu, J.; Lee, S.; Batra, D.; and Parikh, D. 2018b. Graph R-CNN for Scene Graph Generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, 1433–1438. Chicago, IL, USA.