Inverse Reinforcement Learning with Natural Language Goals

Li Zhou
Amazon
lizhouml@amazon.com

Kevin Small
Amazon
smakevin@amazon.com

Abstract

Humans generally use natural language to communicate task requirements amongst each other. It is desirable that this would be similar for autonomous machines (e.g. robots) such that humans can convey goals or assign tasks more easily. However, understanding natural language goals and mapping them to sequences of states and actions is challenging. Previous research has encountered difficulty generalizing learned policies to new natural language goals and environments. In this paper, we propose an adversarial inverse reinforcement learning algorithm that learns a language-conditioned policy and reward function. To improve the generalization of the learned policy and reward function, we use a variational goal generator that relabels trajectories and samples diverse goals during training. Our algorithm outperforms baselines by a large margin on a vision-based natural language instruction following dataset, demonstrating a promising advance in providing natural language instructions to agents without reliance on instruction templates.

1 Introduction

Inverse reinforcement learning (IRL) [1, 2, 3], a specific form of imitation learning [4], is the task of learning a reward function and hence a policy based on expert demonstrations. Imitation learning has been successfully applied to a wide range of tasks including robot manipulation [5], autonomous driving [6], human behavior forecasting [7], video game AI [8], etc. However, goals of the tasks are usually specified intrinsically by the environments and an agent is trained for each specific task. To generalize the learned policy to new goals, many goal-conditioned imitation learning [9] and reinforcement learning algorithms [10, 11, 12] have been proposed in which the policy is explicitly conditioned on a goal. Normally, the goals either share the same space with the states or can be easily mapped to the state space. For example, in Ding et al. [9], the goals and states are both coordinates in the environment and the goals are provided to the agent by specifying the goal positions.

Humans use natural language to communicate with each other. A natural way to convey goals or assign tasks to an autonomous machine (e.g. a robot) is also by natural language. For example, in household tasks, when asking an agent to pick up a toolbox from the garage, we do not assign the coordinates of the toolbox to the robot. Instead, we ask the agent *retrieve my toolbox from the garage*. This requires the agent to understand the semantics of the natural language goals, associate states and actions with the natural language goals, and infer whether the natural language goals are achieved or not, all of which are very challenging tasks. Fu et al. [13] and Bahdanau et al. [14] are the most related works to our paper. They propose to learn a language-conditioned reward function under the maximum entropy inverse reinforcement learning [1] and generative adversarial imitation learning framework [15]. However, their natural language goals in the experiments are template-based, and Fu et al. [13]'s policy is optimized exactly in a grid environment with known dynamics. When the policy is optimized by sample-based reinforcement learning algorithms such as deep Q-learning [16], the model performance drops significantly. This reveals the sample efficiency challenge in this problem

setting; that is, when the natural language goals and the environments are complicated, generalization to new goals and new environments becomes a very difficult problem.

Goal relabeling techniques such as hindsight experience replay (HER) [17] and latent goal relabeling [12] have been shown to efficiently improve sample efficiency in reinforcement learning setting. However, when the goals are natural language and are in a different space than the state space, goal relabeling cannot be applied directly, as we cannot easily relabel a state to a natural language goal. Cideron et al. [18] proposes to build such a relabeling function with a sequence-to-sequence model that takes a trajectory as input and output a relabeled natural language goal. However, they require the ground-truth reward function to be accessible, and their experiments are based on simple template-based natural language goals. Moreover, as we will show in this paper, applying HER to IRL with natural language goals doesn't significantly improve performance, as the reward function does not generalize well to relabeled goals.

In this paper, we propose a sample efficient algorithm for IRL with natural language goals. To the best of our knowledge, our work is the first IRL algorithm that works with real human generated natural language goals (as opposed to template-based language goals [3, 14]) in a real world vision-based environment. Our contributions include: 1) proposing a natural language goal conditioned adversarial inverse reinforcement learning algorithm, 2) specifying a variational goal generator that efficiently sample diverse natural language goals given a trajectory, 3) utilizing goal relabeling and sampling strategies to improve the generalization of both the policy and the reward function to new natural language goals and new environments, 4) proposing a self-supervised learning mechanism to further improve the generalization in new environments. Through these innovations, we show that our algorithm significantly outperform baselines.

2 Problem Formulation

A task is defined as a pair (E, G), where E is an environment that the agent can interact with, and G is a natural language goal that the agent has to fulfill. $G = \{w_1, w_2, ..., w_N\}$ consists of N words. For example, in our experiments E is a realistic 3D indoor environment and G is a human-generated navigation instruction. The true reward function of each task is unknown, but there exists a set of expert demonstrations. Each expert demonstration consists of a task (E, G) and a trajectory τ. The trajectory $\tau = \{s_1, a_1, s_2, a_2, ..., s_T, a_T\}$ consists of a sequence of states perceived by the experts and the actions taken by the experts given the states. For example, in our experiments the states are first-person views in a 3D indoor environment and the actions are movements towards a direction in the environment. While our proposed methods can be applied to both continuous and discrete action spaces, we focus on discrete action spaces in this paper. The objective of the algorithm is to learn a policy that imitate expert demonstrations, so that given a new natural language goal in an existing or new environment, the agent can perform a sequence of actions to achieve that goal. In this paper, we use G to represent an arbitrary goal, $\boldsymbol{G}$ to represent the set of goals in expert demonstrations, and G_i to represent the i-th goal in $\boldsymbol{G}$. The same notation style applies to E and τ too.

3 Approach

3.1 Preliminary: Adversarial Inverse Reinforcement Learning (AIRL)

AIRL [3] is based on maximum entropy inverse reinforcement learning (MaxEntIRL) [1]. In MaxEntIRL, the probability of a trajectory is defined as $p_\theta(\tau) = \frac{1}{Z}\exp(f_\theta(\tau))$, where f_θ is the reward function to learn and $Z = \sum_\tau \exp(f_\theta(\tau))$ is the partition function. MaxEntIRL learns the reward function from expert demonstrations by maximizing the likelihood of trajectories in the expert demonstrations: $\max_\theta \mathbb{E}_{\tau\sim\mathcal{D}}(\log p_\theta(\tau))$. Maximizing this likehood is challenging because the partition function Z is hard to estimate. AIRL maximizes this likelihood by using a GAN [19] framework. The GAN generator is the policy π to learn and the GAN discriminator is defined as

$$D_{\theta,\phi}(s, a, s') = \frac{\exp\{f_{\theta,\phi}(s, a, s')\}}{\exp\{f_{\theta,\phi}(s, a, s')\} + \pi(a|s)} \tag{1}$$

where $f_{\theta,\phi}(s, a, s') = g_\theta(s, a) + \gamma h_\phi(s') - h_\phi(s)$, $g_\theta(s, a)$ is the reward function to learn, and $h_\phi(s') - h_\phi(s)$ is the reward shaping term. The policy π and the discriminator are updated alternately.

3.2 Inverse Reinforcement Learning with Natural Language Goals

Our model is based on the AIRL framework. In our problem setting, the policy and the reward function are conditioned on a natural language goal G, so we extend the discriminator in Equation (1) to have $f_{\theta,\phi}(s, a, s', G) = g_\theta(s, a, G) + \gamma h_\phi(s', G) - h_\phi(s, G)$ such that

$$g_\theta(s, a, G) = \text{MLP}([e^s; \text{Att}(s, G); e^a])$$
$$h_\phi(s, G) = \text{MLP}([e^s; \text{Att}(s, G)])$$

where $\text{MLP}(\cdot)$ is a multilayer perceptron, e^s and e^a are the embeddings of state s and action a, respectively, and $\text{Att}(s, G)$ is an attention function. $\text{Att}(s, G) = \sum \alpha_i e_i^w$ where e_i^w is the word embedding of w_i in G, $\alpha_i = (\text{Linear}(e^s) \cdot e_i^w) \,/\, (\sum_i \text{Linear}(e^s) \cdot e_i^w)$ and $\text{Linear}(\cdot)$ is a single-layer perceptron. We use soft actor-critic (SAC) [20], one of the state-of-the-art off-policy reinforcement learning algorithms, to optimize policy π given the reward function $g_\theta(s, a, G)$. SAC includes a policy network to predict an action given a state and a goal, and a Q-network that estimate the Q-value of an action given a state and a goal. We define the policy network as

$$\pi_w(s, a, G) = \text{Softmax}(e^a \cdot \text{MLP}([e^s; \text{Att}(s, G)]))$$

and we define the Q-network $q_\psi(s, a, G)$ as the same network architecture as g_θ. Compared with on-policy algorithms such as TRPO [21], SAC utilizes a replay buffer to re-use sampled trajectories. The replay buffer is beneficial to the training of both the discriminator and the policy. *To update the discriminator*, we sample negative (s, a, s', G) examples from the replay buffer and sample positive (s, a, s', G) examples from the expert demonstrations. *To update the policy*, we sample a batch of (s, a, s', G) from the replay buffer and use g_θ to estimate their rewards; then we update the Q- and policy network using these reward-augmented samples. We modify SAC slightly to support discrete action space. For details about model architecture and optimization, please refer to Appendix B.

3.3 A Variational Goal Generator for Improved Generalization and Sample Efficiency

Although SAC has better sample efficiency than many on-policy RL algorithms, as we will show in our experiments, in environments with complicated natural language goals and high dimensional state spaces (such as vision-based instruction following tasks), AIRL with SAC performs only slightly better than supervised learning based behavior cloning, and the learned policy and discriminator cannot generalize well to new goals or new environments. In this section, however, we show that by learning a variational goal generator, we can enrich the training data for both the discriminator and the policy, which leads to a large improvement on sample efficiency and generalization for both the discriminator and the policy.

A goal generator takes a trajectory τ (sequence of states and actions) as input and outputs a natural language goal G. Given expert demonstrations, a straightforward way of learning a goal generator is to train an encoder-decoder model that encodes a trajectory and decodes a natural language goal. However, in reality, natural language is highly flexible. Given a trajectory, there are many possible ways to describe it using natural language, and the description will likely be biased due to variance in people's expression preferences. For example, consider a trajectory in a vision-based instruction following task, in which the agent goes to the kitchen and washes the dishes. A possible goal to describe this trajectory can be *go to the kitchen and wash the dishes*; another possible goal can be *clean the dishes on the dining table*. To better model the variations of natural language goals and generate more diverse natural language goals, we learn a variational encoder-decoder model as goal generator. The generative process of the variational goal generator is

$$\mu_{prior}, \sigma^2_{prior} = f_{prior}(\tau)$$
$$z \sim \mathcal{N}\left(\mu_{prior}, \sigma^2_{prior}\mathbf{I}\right)$$
$$G = f_{dec}(z, \tau)$$

where f_{prior} is a LSTM-based trajectory encoder and f_{dec} is an attention-based goal decoder. The posterior distribution of latent variable z is approximated by

$$\mu_{posterior}, \sigma^2_{posterior} = f_{posterior}(\tau, G)$$
$$q(z|\tau, G) = \mathcal{N}\left(z|\mu_{posterior}, \sigma^2_{posterior}\mathbf{I}\right)$$

where $f_{posterior}$ is a LSTM-based trajectory and goal encoder. We then maximize the variational lower bound $-D_{KL}(q(z|\tau, G)||p(z)) + \mathbb{E}_{q(z|\tau,G)}[\log p(G|z, \tau)]$. For details about network architecture and optimization, please refer to Appendix B. Given the learned variational goal generator, we propose the following three goal relabeling and sampling strategies.

3.3.1 Expert Goal Relabeling (EGR)

As we discussed above, multiple goals can be mapped to the same trajectory. We propose to augment expert demonstrations by generating N other goals for each expert trajectory τ_i: $G_{i,n} \sim \text{GoalGenerator}(\tau_i),\ n = \{1, 2, ..., N\}$, where $G_{i,n}$ is the n-th generated goal for τ_i, and GoalGenerator$(\cdot)$ is the learned variational goal generator. Our choice of N is 2. We will show model performance under different values of N in the experiments (Appendix D). These newly generated tuples $\{(E_i, G_{i,n}, \tau_i)\}_{n=1}^{N}$ are also treated as expert demonstrations for training. We represent the set of goals in the augmented expert demonstrations by $\boldsymbol{G}^1$ where the superscript of $\boldsymbol{G}$ represents the round of generation which will be described shortly.

3.3.2 Hindsight Goal Relabeling (HGR) for the Discriminator

In the AIRL framework, the quality of the discriminator is crucial to the generalization of the learned policy. A good discriminator learns a good reward function that generalizes well to new goals and new environments such that the learned policy can also well generalize to new goals and new environments [13]. To improve the discriminator, we propose to augment the positive examples of the discriminator by relabeling the goals of the sampled trajectories. More specifically, during the training, given a goal (E_j, G_j^1), the agent interacts with the environment and samples a trajectory τ_j^1. We then use the variational goal generator to sample a goal for τ_j^1:

$$\tau_j^1 \sim \pi(E_j, G_j^1) \tag{2}$$

$$G_j^2 \sim \text{GoalGenerator}(\tau_j^1) \tag{3}$$

The tuples $(\boldsymbol{E}, \boldsymbol{G}^2, \boldsymbol{\tau}^1)$ are treated as positive examples for the discriminator, as a supplement to the positive examples from expert demonstrations $(\boldsymbol{E}, \boldsymbol{G}^1, \boldsymbol{\tau})$.

3.3.3 Hindsight Goal Sampling (HGS) for Policy Optimization

When optimizing the policy with SAC, we have to sample natural language goals to train the policy. One natural way is to sample goals from expert demonstrations. However, expert demonstrations are limited and can be expensive to acquire, and it is hard to cover various goals we would encounter in the test phase. Meanwhile, as we will show in our experiments, training with a diverse set of goals is beneficial to the generalization of the policy. Therefore, we propose to also sample goals from $\boldsymbol{G}^2$ so that the policy can train with goals beyond these in the expert demonstrations.

$$\tau_j^2 \sim \pi(E_j, G_j^2) \tag{4}$$

$$G_j^3 \sim \text{GoalGenerator}(\tau_j^2) \tag{5}$$

Of course, training the policy with $\boldsymbol{G}^2$ relies on the discriminator's generalization ability to provide reasonable reward estimates for states and actions sampled under $\boldsymbol{G}^2$. This is ensured by HGR in Section 3.3.2, as $(\boldsymbol{E}, \boldsymbol{G}^2, \boldsymbol{\tau}^1)$ are provided as positive examples for the discriminator.

The process from Equation (2) to Equation (5) can be seen as using $\boldsymbol{G}^1$ and $\boldsymbol{\tau}$ as seed, and iteratively sample $\boldsymbol{\tau}^v$ from $\boldsymbol{G}^v$ using the policy, and then sample $\boldsymbol{G}^{v+1}$ from $\boldsymbol{\tau}^v$ using the goal generator. We can go deeper in this loop to sample more diverse natural language goals and trajectories for the policy and the discriminator to train on. More specifically, *to train the discriminator*, positive examples are sampled from $(\boldsymbol{E}, \boldsymbol{G}^1, \boldsymbol{\tau})$ with probability 0.5, and are sampled from $\{(\boldsymbol{E}, \boldsymbol{G}^{v+1}, \boldsymbol{\tau}^v) | v \geq 1\}$ with probability 0.5; negative examples are sampled from $\{(\boldsymbol{E}, \boldsymbol{G}^v, \boldsymbol{\tau}^v) | v \geq 1\}$. *To train the policy*, goals are sampled from $(\boldsymbol{E}, \boldsymbol{G}^1)$ with probability 0.5 and are sampled from $\{(\boldsymbol{E}, \boldsymbol{G}^{v+1}) | v \geq 1\}$ with probability 0.5. Algorithm 1 shows the overall description for LangGoalIRL.

3.3.4 Hindsight Experience Replay (HER) for Policy Optimization

A closely related work, hindsight experience replay [17], has been shown to work very well in reinforcement learning with sparse rewards setting. We can easily incorporate HER into our training procedure. That is, when sampling from replay buffer to optimize policy π, we sample batches not only from $\{(\boldsymbol{E}, \boldsymbol{G}^v, \boldsymbol{\tau}^v)\ | v \geq 1\}$, but also from $\{(\boldsymbol{E}, \boldsymbol{G}^{v+1}, \boldsymbol{\tau}^v)\ | v \geq 1\}$. The difference between HER and HGR from Section 3.3.2 is that the former is goal relabeling for the policy while the latter is goal relabeling for the discriminator. HER is most efficient when rewards are sparse; however, in our setting, the rewards provided by the discriminator are not sparse, and we do not observe a boost of performance after applying HER. We will discuss this issue more in the experiments section.

Algorithm 1 Inverse Reinforcement Learning with Natural Language Goals (LangGoalIRL)

Input: GoalGenerator: the variational goal generator from section 3.3; $\mathcal{D}$: expert demonstrations; $\mathcal{R} = \varnothing$: replay buffer; b: batch size; N: number of expert relabeling goals.

$\tilde{\mathcal{D}} = \varnothing, \mathcal{G} = \varnothing$
for each $(E_i, G_i, \tau_i) \in \mathcal{D}$ **do**
  for n = 1, 2, ..., N **do**
    $G_{i,n} \sim$ GoalGenerator(τ_i) ▷ Expert Goal Relabeling (EGR)
  end for
  add (E_i, G_i, τ_i) and $\{(E_i, G_{i,n}, \tau_i)\}_{n=1}^{N}$ to $\tilde{\mathcal{D}}$
end for
while not converged **do**
  $r_1, r_2 \sim$ Uniform$(0, 1)$
  if $r_1 < 0.5$ **then**
    Sample a goal $(E_j, G_j) \sim \tilde{\mathcal{D}}$
  else
    Sample a goal $(E_j, G_j) \sim \mathcal{G}$ ▷ Hindsight Goal Sampling (HGS)
  end if
  Sample a trajectory using the current policy $\tau_j' \sim \pi(E_j, G_j)$
  Sample a relabeled goal $G_j' \sim$ GoalGenerator(τ_j')
  Add $(E_j, G_j, G_j', \tau_j')$ to replay buffer $\mathcal{R}$
  Add (E_j, G_j') to $\mathcal{G}$
  if $r_2 < 0.5$ **then**
    Sample a batch $\mathcal{P}_+ = \{(s_k^t, a_k^t, s_k^{t+1}, G_k)\}_{k=1}^{b} \sim \tilde{\mathcal{D}}$
  else
    Sample a batch $\mathcal{P}_+ = \{(s_k^t, a_k^t, s_k^{t+1}, G_k')\}_{k=1}^{b} \sim \mathcal{R}$ ▷ Hindsight Goal Relabeling (HGR)
  end if
  Sample a batch $\mathcal{P}_- = \{(s_k^t, a_k^t, s_k^{t+1}, G_k)\}_{k=1}^{b} \sim \mathcal{R}$
  Update discriminator parameters with $\mathcal{P}_+$ and $\mathcal{P}_-$ as positive and negative examples, respectively.
  Sample a batch $\mathcal{Q} = \{(s_k^t, a_k^t, s_k^{t+1}, G_k)\}_{k=1}^{b} \sim \mathcal{R}$
  Expand each entry of $\mathcal{Q}$ with reward $r_k^t = g_\theta(s_k^t, a_k^t, G_k)$
  Optimize π using Soft Actor-Critic with $\mathcal{Q}$
end while

3.4 Self-Supervised Learning in New Environments

In this section, we are interested in the scenario where the learned policy is deployed to a new environment. For example, after training an embodied agent to perform tasks in a set of buildings, we may deploy this agent to a new building with different floor plans. We assume that we have access to these new environments but we do not have any expert demonstrations nor any natural language goals in new environments. Note that we can not directly apply natural language goals from existing environments to new environments, because goals are tied to the environments. For example, in instruction-following tasks, an example goal is *go downstairs and walk pass the living room*. However, there may be no stairs in a new environment. As a result, we can not just sample goals from expert demonstrations to train in the new environments.

The high level idea of our method is to first learn a policy π_p that is goal-agnostic. That is, the policy selects an action purely based on the state without conditioning on a goal. This model gives us a prior of how the agent will act given a state. We use behavior cloning to train this policy on expert demonstrations. More details are in Appendix B. We then sample trajectories in the new environments using this policy and sample goals of these trajectories using the goal generator:

$$\tau^{new} \sim \pi_p(E^{new})$$
$$G^{new} \sim \text{GoalGenerator}(\tau^{new})$$

$(\boldsymbol{E}^{new}, \boldsymbol{G}^{new}, \boldsymbol{\tau}^{new})$ are treated as expert demonstrations in the new environments. Then we fine-tune the discriminator and the policy in new environments using Algorithm 1 (w/o expert goal relabeling). The self-supervised learning signals come from both the generated expert demonstrations for the new environments, and the hindsight goal relabeling and sampling proposed in Section 3.3. The generated expert demonstrations can be seen as seeds to bootstrap hindsight goal relabeling and sampling.

4 Experiments

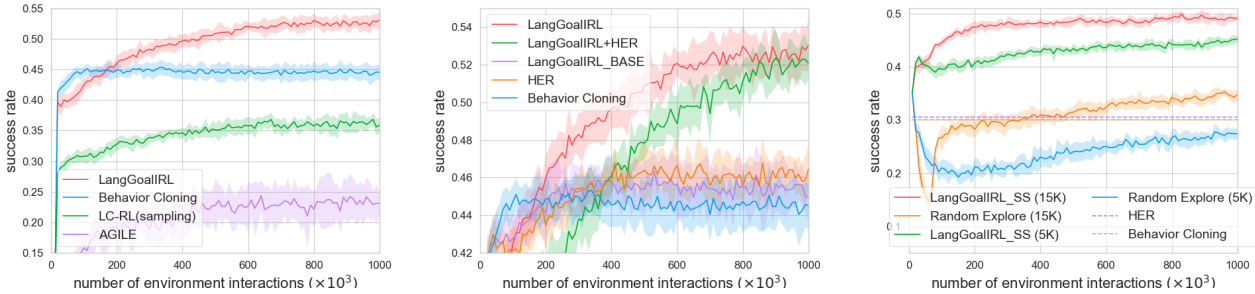
While our proposed method works on any natural language goal based IRL problems, our experiments focuses on a challenging vision-based instruction following problem. We evaluate our model on the Room-2-Room (R2R) dataset [22], a visually-grounded natural language navigation task in realistic 3D indoor environments. The dataset contains $7,189$ routes sampled from 90 real world indoor environments. A route is a sequence of viewpoints in the indoor environments with the agent's first-person camera views. Each route is annotated by humans with 3 navigation instructions. The average length of navigation instructions is 29 words. The dataset is split into train (61 environments and $14,025$ instructions), seen validation (61 environments same as train set, and $1,020$ instructions), unseen validation (11 new environments and $2,349$ instructions), and test (18 new environments and $4,173$ instructions). We don't use the test set for evaluation because the ground-truth routes of the test set are not released. Along with the dataset, a simulator is provided to allow the embodied agent to interact with the environments. The observation of the agent is the first-person camera view, which is a panoramic image. The action of the agent is the nearby viewpoints that the agent can move to. For details about the dataset, please see Appendix A. The R2R dataset has become a very popular testbed for language grounding in visual context [23, 24, 25, 26].

We evaluate the model performance based on the trajectory success rate. Each navigation instruction in the R2R dataset is labeled with a goal position. Following Anderson et al. [22], the agent is considered successfully reaching the goal if the navigation error (the distance between the stop position and the goal position) is less than 3 meters. Note that the goal position is not available to our model, as we use navigation instructions as goals.

We compare our model with the following baselines: (1) **LangGoalIRL_BASE**, which corresponds to our proposed model in Section 3.2 but without the goal relabeling/sampling strategies proposed in Section 3.3. (2) **Behavior Cloning**, which is imitation learning as supervised learning. The model shares the same architecture as the policy network of LangGoalIRL and is trained to minimize cross entropy loss with actions in the expert demonstrations as labels. (3) **LC-RL (Sampling)** [13], which also uses MaxEntIRL [1] to learn a reward function. It optimizes the policy exactly in a grid environment, which is not scalable to our experiment setting, so we use AIRL with SAC for LC-RL to optimize the policy. In this case, LC-RL is very similar to LangGoalIRL_BASE, except that LC-RL simply concatenate state and goal embeddings as input to the policy and reward function, while LangGoalIRL_BASE uses the attention mechanism $\text{Att}(s, G)$ in Section 3.2. (4) **AGILE** [14], which proposes to learn a discriminator (reward function) that predicts whether a state is the goal state for a natural language goal or not. The positive examples for the discriminator are the (natural language goal, goal state) pairs in expert demonstrations, and the negative examples are sampled from the replay buffer of SAC. (5) **HER** [17, 18], which uses our learned variational goal generator to relabel trajectories. This corresponds to the `final` strategy in Andrychowicz et al. [17] applied to LangGoalIRL_BASE. (6) **Upper Bound** [23]. Recently, there are a few vision-language navigation models [23, 24, 25, 26] developed and evaluated on R2R dataset. They assume the goal positions or the optimal actions at any states are known, and assume the environments can be exhaustively searched without sample efficiency considerations in order to do training data augmentation. As a result, their problem settings are different from us, but we include Tan et al. [23]'s result as our upper bound. Note that without the assumptions above, their model performance reverts to our **Behavior Cloning** baseline. We will discuss more about this in Related Works section. For implementation details of our algorithms and the baselines, please see Appendix B.

The performance of our algorithm and baselines are shown in Figure 1, Table 1, and Table 2. From the results, we would like to highlight the following observations.

1. **LangGoalIRL outperforms baselines by a large margin.** From Table 1 and Figure 1(a) we can see that LangGoalIRL achieves a 0.530 success rate on seen validation set, a 47.63% improvement over LC-RL (sampling) and a 129% improvement over AGILE. Rewards learned by AGILE are binary and sparse, which may be one of the main reasons why AGILE performs even worse than Behavior Cloning. The difference between LC-RL (sampling) and LangGoalIRL_BASE is just the lack of attention mechanism over natural language goals, which simply indicates that attention mechanism is important when natural language goals are complicated. We can further explore other powerful attention mechanism such as the Transformer architecture [28]. The success rate of LangGoalIRL is 18.04% and 15.91% higher than LangGoalIRL_BASE on seen validation and unseen validation, respectively, which shows that the three proposed strategies efficiently improve the generalization of



(a) Baseline performance in seen validation.

(b) HER variants performance in seen validation.

(c) Self-supervised learning in unseen validation.

(d) More self-supervised learning in unseen validation.

(e) Ablation study in seen validation.

(f) More ablation study in seen validation.

Figure 1: Model performance on both seen and unseen validation. LangGoalIRL_SS in Figure (c) and (d) represents self-supervised learning in unseen environments. LangGoalIRL_SS (15K) means that $15,000$ demonstrations are sampled by the goal-agnostic policy π_p. Random Explore means that a random policy instead of π_p is used to sample demonstrations. LangGoalIRL_SS_G2 means that the policy does not iteratively sample goals from $\{(\boldsymbol{E}, \boldsymbol{G}^v)|v > 2\}$ as described in Section 3.3.3.

	seen validation	unseen validation
Behavior Cloning	0.445 ± 0.0122	0.300 ± 0.0114
LC-RL (sampling)	0.359 ± 0.0117	0.216 ± 0.0081
AGILE	0.231 ± 0.0173	0.253 ± 0.0269
HER	0.464 ± 0.0126	0.306 ± 0.0173
LangGoalIRL_BASE	0.449 ± 0.0130	0.308 ± 0.0087
LangGoalIRL	$\mathbf{0.530 \pm 0.0138}$	0.357 ± 0.0089
+self-supervised	–	$\mathbf{0.491 \pm 0.0065}$
Upper Bound	0.621	0.645

Table 1: Success rate after 1 million environment interactions. LangGoalIRL+self-supervised is further fine-tuned in unseen environment for 1 million environment interactions (explained in Figure 1(c)).

Expert Goal Relabeling	Hindsight Goal Relabeling	Hindsight Goal Sampling	seen validation success rate
–	–	–	0.449 ± 0.0130
✓	–	–	0.450 ± 0.0122
–	✓	–	0.427 ± 0.0048
–	–	✓	0.399 ± 0.0124
✓	✓	–	0.504 ± 0.0185
✓	–	✓	0.444 ± 0.0134
–	✓	✓	0.503 ± 0.0117
✓	✓	✓	$\mathbf{0.530 \pm 0.0138}$

Table 2: Ablation study of the three proposed strategies on seen validation set. A complete ablation study (which includes HER) is in Appendix D.

the policy to new natural language goals and new environments. From Figure 1(b) we see that, when combined with HER, LangGoalIRL converges slower and performs worse than LangGoalIRL alone. HER has been shown to efficiently deal with reward sparsity; however, the rewards learned by AIRL are not sparse. From Table 1 we see that HER alone barely outperforms LangGoalIRL_BASE. As we will discuss shortly, HER only improves the sample efficiency of the *generator (policy)*, however, the generalization of the *discriminator (reward function)* is what is crucial to the overall performance of the policy. Finally, from Figure 1(b) we can also see that LangGoalIRL_BASE consistently outperforms Behavior Cloning after about 200k interactions, which effectively indicates the benefit of using inverse reinforcement learning over behavior cloning.

2. **Hindsight goal relabeling and expert goal relabeling improve the generalization of the *discriminator (reward function)*; such generalization is *crucial* to the performance of the policy.** From Table 2 and Figure 1(f), we see that if we take out HGR, the success rate drops significantly from 0.530 to 0.444. This shows that HGR plays a key role in learning a better reward function by enriching positive examples of the discriminator. Meanwhile, when applying HGS without HGR, the success rate drops from 0.503 to 0.399. This is because the discriminator cannot generalize well to sampled goals when HGR is not applied, which shows the importance of discriminator generalization. However, from Table 2 and Figure 1(e) we can also see that if we only keep HGR, the success rate drops to 0.427 which is even lower than LangGoalIRL_BASE. We observe that in this case

the goals generated by the variational goal generator only appear in the positive examples of the discriminator, so the discriminator easily identify these goals, and simply assign high rewards to these goals regardless of what action is taking. When EGR and HGR are applied together, relabeled goals appear in both positive and negative examples, and the success rate improves from 0.427 to 0.504.

3. **Self-supervised learning largely improves performance in unseen environments.** Table 1 shows that with self-supervised learning, LangGoalIRL achieves a success rate of 0.491, a 59.42% improvement over LangGoalIRL_BASE and a 37.54% improvement over LangGoalIRL w/o self-supervised learning. This shows that the proposed self-supervised learning algorithm can significantly improve policy performance even though neither expert demonstrations nor natural language goals in new environments are given. The number of trajectories sampled by the goal-agnostic policy π_p in new environments by default is $15,000$. In Figure 1(c), we show the performance of a baseline model where the embodied agent randomly explore the new environments (randomly select an action at any states, but do not visit any visited viewpoints) rather than using π_p to sample trajectories. We set the number of sampled trajectory to $15,000$ and $5,000$. We can see that the performance of the agent drops at the early stage, as randomly explored trajectories are not good seeds to bootstrap Algorithm 1 in new environments. After early stages, the model performance gradually increases as HGR and HGS provide some self-supervised signals. However, overall the performance is lower than our model. More experiment results are in Table 2 of Appendix D. Finally, self-supervised learning can also be applied to existing environments by augmenting the expert demonstrations in seen validation set. While this is not the setting we are focusing on, we include the results in Table 3 of Appendix D.

4. **Hindsight goal sampling improves the generalization of the policy by enabling the policy to explore a more diverse set of goals.** From Table 2 and Figure 1(f) we can see that HGS further improves the policy performance from 0.504 to 0.530 on seen validation set. Figure 1(d) shows the impact of HGS on self-supervised learning in new environments. The baseline, LangGoalIRL_SS_G2 samples goals only from $\boldsymbol{G}^1$ and $\boldsymbol{G}^2$ defined in Section 3.3 and does not iteratively sample goals from $\{(\boldsymbol{E}, \boldsymbol{G}^v) | v > 2\}$. As goals sampled are less diverse, we see that LangGoalIRL_SS_G2 performs worse than LangGoalIRL_SS given $5,000$, $10,000$ and $15,000$ generated expert demonstrations in new environments. More experiments are in Table 2 of Appendix D.

5 Related Works

Goal-conditioned reinforcement learning and imitation learning have been explored by many prior works [11, 29, 12, 30, 9]. Usually, the task is to learn a policy that is parameterized by a goal and can generalize to new goals. In most prior works the goals and states are in a same space, such as positions in a Cartesian coordinate system [30, 9]. However, in this paper we investigate a different setting where the goals are natural language goals. The closest prior works to our algorithm are language-conditioned IRL and reward learning [13, 14, 31, 32, 33]. MacGlashan et al. [31] and Williams et al. [32] learn a semantic parser to map language instructions to reward functions. Goyal et al. [33] propose to train a model to predict whether a language instruction describes an action in a trajectory, and use the predictions to do reward shaping. Bahdanau et al. [14] propose to use GAN [19, 15] to learn a discriminator that predicts whether a state is the goal state of a natural language instruction. Fu et al. [13] proposes to use the MaxEntIRL [1] framework to learn a language-conditioned reward function for instruction following tasks. The language instructions in Bahdanau et al. [14] and Fu et al. [13]'s experiments are generated by templates and much easier to understand compared with our dataset. Meanwhile, our model demonstrates significantly better generalization than these two prior works. There are also many prior works on goal relabeling to improve sample efficiency, but none of them can be directly applied to the IRL with natural language goals setting. Andrychowicz et al. [17] propose hindsight experience replay that samples additional goals for each transition in the replay buffer and can efficiently deal with the reward sparsity problem. Nair et al. [12] propose to sample additional diverse goals from a learned latent space. Ding et al. [9] propose to relabel trajectories using the states within the trajectories. These algorithms do not work with natural language goals. Cideron et al. [18] and Jiang et al. [34] generalize HER to language setting and relabel trajectories by hindsight language instructions. However, in this paper we show that, when doing IRL, simply applying HER to policy optimization does not improve policy performance, and it is critical to improve the generalization of the reward function. Finally, there are quite a few recent works [23, 24, 25, 26] on Room-2-Room dataset, which propose solutions for vision-language navigation [22]. These works do not focus on sample efficient IRL algorithms.

They use student-forcing [22] to train their supervised learning models or pre-train their RL models. With student-forcing, the policy is assumed to always have access to the optimal action at any state it encounters during training. This is a even stronger assumption than knowing the ground-truth reward function. Fried et al. [25] and Tan et al. [23] propose to learn a speaker model that back-translate a given trajectory to a natural language goal, which is similar to our goal generator without latent variables. However, the purpose of the speaker model is to augment training data before the training, rather than relabeling goals during training. To augment training data, they generate the shortest path between any two viewpoints. This requires the topological graph of viewpoints before the training. Moreover, they augment training set with every shortest path that is not included in the original training set. This assumes the environments can be exhaustively searched without sampling efficiency considerations. Other proposed techniques in these papers such as environment dropout [23] and progress monitoring [26] are orthogonal to our work. For a survey of reinforcement learning with natural language, please see Luketina et al. [35].

6 Conclusion

In this paper, we propose a sample efficient algorithm for IRL with natural language goals. We propose to learn a variational goal generator that can relabel trajectories and sample diverse goals. Based on the variational goal generator, we propose three strategies to improve the generalization and sample efficiency of the language-conditioned policy and reward function. Experiment results show that our algorithm outperforms existing baselines by a large margin and generalizes well to new natural language goals and new environments, thus increasing flexibility of expression and domain transfer in providing instructions to autonomous agents.

References

[1] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[2] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[3] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[4] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[5] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58, 2016.

[6] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211. IEEE, 2017.

[7] Nicholas Rhinehart and Kris M Kitani. First-person activity forecasting with online inverse reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3696–3705, 2017.

[8] Aaron Tucker, Adam Gleave, and Stuart Russell. Inverse reinforcement learning for video games. *arXiv preprint arXiv:1810.10593*, 2018.

[9] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. In *Advances in Neural Information Processing Systems*, pages 15298–15309, 2019.

[10] LP KAELBLING. Learning to achieve goals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1094–1098, 1993.

[11] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320, 2015.

[12] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

[13] Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

[14] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. *International Conference on Learning Representations*, 2019.

[15] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[17] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, pages 5048–5058, 2017.

[18] Geoffrey Cideron, Mathieu Seurin, Florian Strub, and Olivier Pietquin. Self-educated language agent with hindsight experience replay for instruction following. *arXiv preprint arXiv:1910.09451*, 2019.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[20] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[21] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[22] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[23] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2610–2621, 2019.

[24] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019.

[25] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3314–3325, 2018.

[26] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10012–10022, 2020.

[27] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 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*, pages 627–635, 2011.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[29] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. *arXiv preprint arXiv:1705.06366*, 2017.

[30] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal

reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[31] James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael Littman, Smaranda Muresan, Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. Grounding english commands to reward functions. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[32] Edward C Williams, Nakul Gopalan, Mine Rhee, and Stefanie Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In *2018 IEEE International Conference on Robotics and Automation*, pages 1–7, 2018.

[33] Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. Using natural language for reward shaping in reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 2385–2391, 2019.

[34] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 9414–9426, 2019.

[35] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

[36] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] William S Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, 28(1):47–65, 1991.

[39] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[41] Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019.

Appendix

A Dataset and Environment Details

The Room-2-Room (R2R) dataset [1] [22] is a dataset for visually-grounded natural language navigation task in realistic 3D indoor environments. The dataset is built on top of the Matterport3D simulator [22]. Matterport3D simulator provides APIs to interact with 90 3D indoor environments, including homes, offices, churches and hotels. An agent can navigation between viewpoints in the 3D environment. At each viewpoint, the agent can turn around and perceive a 360-degree panoramic image. Images are all real rather than synthetic [36].

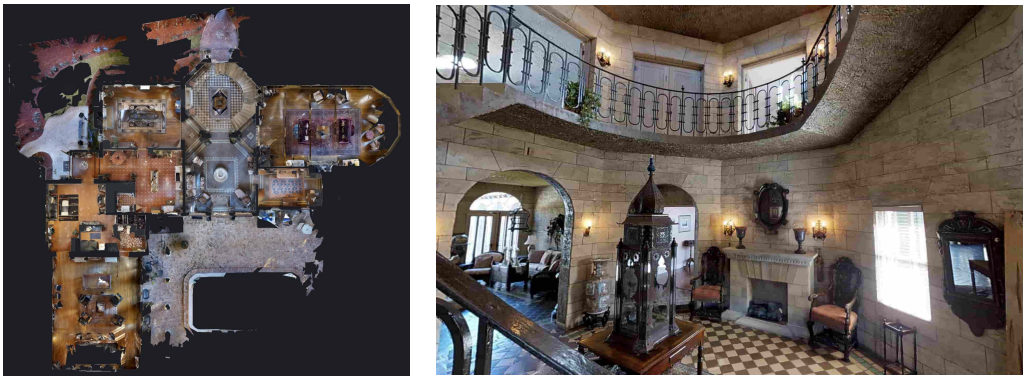


Figure 2: The left-side figure shows a bird view of the indoor environment. This view is for demonstration purpose and is not available to the embodied agent. The right-side figure is the camera view of the embodied agent at a viewpoint. The agent can look around or move to the next viewpoint.

The routes in the R2R dataset was sampled from the simulator by first sampling two viewpoints in the environment and then calculating the shortest-path from one viewpoint to the other. In total, there are $7,189$ visually diverse routes sampled. Then each routes was annotated by 3 Amazon Mechanical Turk workers. The workers are asked to write directions to provide navigation instructions for sampled routes so that a robot can follow the instructions to reach to the end viewpoints. In total there are $21,567$ instructions, and the average length of each instruction is 29 words. The size of the vocabulary for training is 991. The dataset is split into train (61 environments and $14,025$ instructions), seen validation (61 environments same as train set, and $1,020$ instructions), unseen validation (11 new environments and $2,349$ instructions), and test (18 new environments and $4,173$ instructions).

The following are 3 navigation instructions in the dataset describing the same route:

1. *Head indoors and take the hallway to the living room. Stop and wait on the right hand side of the purple wall.*
2. *Turn slightly left and walk across the hallway. Turn slightly right and walk towards the green sofas. Walk to the right entrance where the window panes are at and wait there.*
3. *Go inside and walk straight down the hallway. Keep going until you see a green couch on the right and then walk towards it. Go to the right side of the TV and stop in the doorway leading to an interior room with a table in it.*

For more details about the dataset, please see Anderson et al. [22].

B Implementation Details

Observation Space: The Matterport3D simulator outputs a RGB image corresponding to the embodied agent's first-person camera view. The image resolution is 640×480. Following Fried et al. [25] and Tan et al. [23], at the each viewpoint, the agent first look around and get a 360 degree panoramic

[1] https://bringmeaspoon.org/

image, which consists of 36 first-person camera view images (12 heading and 3 elevation with 30 degree increments). Image features are 2048-dimensional vecotors extracted from a pre-trained ResNet-152 model [2] [37].

State Space: The observation of the embodied agent only captures views from the current viewpoint. We define the state of the embodied agent as the set of all previous observations in the current trajectory. We use a LSTM model to encode previous and current observations into a n-dimensional vector. If we update the LSTM model parameters during training, it will make our problem a POMDP problem [38]. For the purpose of simplicity, we pre-train the LSTM model using the base model below and fix its parameters during training.

Action Space: The action space at any given state is the collection of nearby viewpoints. The maximum number of nearby viewpoints is 13. The feature vector of each action (viewpoint) is a m-dimensional vector that will be described in the base model below. We add a m-dimensional vector of all zeros to represent the stop action, so the maximum number of actions given a state is 14.

Base Model: we use a base model architecture that is similar to many prior works [25, 23, 24]. The base model is a sequence-to-sequence model. The encoder takes natural language goals as input and the decoder outputs a sequence of actions. Let $\{o_{t,i}\}_{i=1}^{36}$ be the observations (36 first-person camera view images) at time step t. Let $\theta_{t,i}$ and $\phi_{t,i}$ be the heading and elevation of the camera view $o_{t,i}$. The feature vector of $o_{t,i}$ is $f_{t,i} = [f_{\text{ResNet}}(o_{t,i}); \cos\theta_{t,i}; \sin\theta_{t,i}; \cos\phi_{t,i}; \sin\phi_{t,i}]$, where f_{ResNet} is a pre-trained ResNet-152 model on ImageNet dataset [3]. Let the natural language goal be $G = \{w_1, w_2, ..., w_N\}$. The natural language goal is encoded by a LSTM model:

$$h_1^w, h_2^w, ..., h_N^w = \text{LSTM}_{enc}(e_1^w, e_2^w, ..., e_N^w)$$

where e_n^w is the n-th word embedding. We use GloVe [39] to initialize the weights of word embeddings. The decoder is also a LSTM model. Let the hidden state of the decoder at step t be h_t^s. We use h_{t-1}^s to attend over the observations at t

$$\alpha_i^s = \text{Softmax}(\text{Linear}(h_{t-1}^s) \cdot f_{t,i})$$
$$f_t = \sum_{i=1}^{36} \alpha_i^s f_{t,i}$$

Then we update the decoder model:

$$e_t^{angle} = \text{Linear}([\cos\theta_t; \sin\theta_t; \cos\phi_t; \sin\phi_t])$$
$$h_t^s = \text{LSTM}_{dec}(h_{t-1}^s, [f_t; e_t^{angle}])$$

where θ_t and ϕ_t are the current heading and elevation of the embodied agent. To predict the next action, we first use h_t^s to attend over the natural language goal: $\alpha_n^w = \text{Softmax}(\text{Linear}(h_t^s) \cdot h_n^w)$. Then the summarized goal embedding is $g_t = \sum_{n=1}^{N} \alpha_n^w h_n^w$. Then the probability of selecting an action is given by

$$p_i^a = \text{Softmax}(\text{BiLinear}(\text{Linear}([h_t^s; g_t]), e_{t,i}^a)) \tag{6}$$

where $\text{BiLinear}(v, u) = v^\top W u$, $e_{t,i}^a = [f_{\text{ResNet}}(o_{t,a}); \cos\theta_{t,a}; \sin\theta_{t,a}; \cos\phi_{t,a}; \sin\phi_{t,a}]$ is the action embedding, $o_{t,a}$ is the camera view of the action (viewpoint) from the current viewpoint, and θ and ϕ are heading and elevation of the action from the current viewpoint. The base model is trained with cross entropy loss $L_{base} = \text{CrossEntropy}(p^a, y^a)$ where y^a is the ground-truth action. The trained LSTM_{dec} is used as the pre-trained state encoding model for all algorithms in our experiments.

Policy Model: We use soft actor-critic (SAC) [20] as our policy optimization algorithm. SAC includes a Q network and a policy network. The Q network $Q(s, a, G)$ is defined as

$$Q_\psi(s_t, a_{t,i}, G) = \text{MLP}([h_t^s; g_t; e_{t,i}^a])$$

where MLP is a multilayer feedforward neural network, h_t^s is the state of the embodied agent, which is given by the pre-trained LSTM_{dec} model in the base model, g_t is the summarized goal embedding

[2] https://github.com/peteanderson80/Matterport3DSimulator

[3] http://image-net.org/index

calculated in the same way as the base model, and $e^a_{t,i}$ is the action embedding. The policy network is defined as

$$\pi_w(s_t, a_{t,i}, G) = \text{Softmax}(e^a_{t,i} \cdot \text{MLP}([h^s_t; g_t]))$$

By default the action space in SAC is continuous, however, in our setting the action space is discrete and state-conditioned. We modify the SAC algorithm slightly so that it can support discrete action space. The Q network parameters in SAC are trained to minimize the following soft Bellman residual:

$$J_Q(\psi) = \mathbb{E}_{(s_t,a_{t,i},G)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\psi(s_t, a_{t,i}, G) - \left(r_t + \gamma\mathbb{E}_{s_{t+1}}\left[V_{\bar{\psi}}(s_{t+1}, G)\right]\right)\right)^2\right]$$

where $\bar{\psi}$ is the target Q network that is obtained as an exponentially moving average of the Q network. In discrete action space setting, the value function is given by

$$V_{\bar{\psi}}(s_t, G) = \sum_{i=1}^{K_t} \pi_w(a_{t,i}|s_t, G)[Q_{\bar{\psi}}(s_t, a_{t,i}, G) - \alpha \log \pi_w(a_{t,i}|s_t, G)]$$

where K_t is the number of actions at state s_t. Similarly, in discrete setting, the policy network is learned by minimizing the KL divergence between the policy and the exponential of the Q network

$$J_\pi(w) = \mathbb{E}_{(s_t,G)\sim\mathcal{D}}\left[\sum_{i=1}^{K_t} \pi_w(a_{t,i}|s_t, G)[\alpha \log \pi_w(a_{t,i}|s_t, G) - Q_\psi(s_t, a_{t,i}, G)]\right]$$

The temperature α is updated by minimizing

$$J(\alpha) = \sum_{i=1}^{K_t} \pi_w(a_{t,i}|s_t, G)[-\alpha_t \log \pi_w(a_{t,i}|s_t, G) - \alpha_t\bar{\mathcal{H}}]$$

where $\bar{\mathcal{H}}$ is the desired minimum expected entropy of $\pi_w(a_t|s_t, G)$.

Discriminator: Our model is based on the adversarial inverse reinforcement learning (AIRL) framework [3]. The generator in the AIRL framework is the SAC policy model we just described, and the discriminator is defined as follows:

$$D_{\theta,\phi}(s_t, a_{t,i}, s_{t+1}, G) = \frac{\exp\{f_{\theta,\phi}(s_t, a_{t,i}, s_{t+1}, G)\}}{\exp\{f_{\theta,\phi}(s_t, a_{t,i}, s_{t+1}, G)\} + \pi(a_{t,i}|s_t, G)}$$

where $f_{\theta,\phi}(s_t, a_{t,i}, s_{t+1}, G) = g_\theta(s_t, a_{t,i}, G) + \gamma h_\phi(s_{t+1}, G) - h_\phi(s_t, G)$. g_θ and h_ϕ are multilayer feedforward neural network

$$g(s_t, a_{t,i}, G) = \text{MLP}([h^s_t; g_t; e^a_{t,i}])$$
$$h(s_t, G) = \text{MLP}([h^s_t; g_t])$$

The parameter θ and ϕ are optimized by minimizing the cross entropy loss with label smoothing [40]

$$\begin{aligned} L(\theta, \phi) =& \mathbb{E}_{(s_t,a_{t,i},s_{t+1},G)\sim\mathcal{D}}\left[(1-\ell)\log D(s_t, a_{t,i}, s_{t+1}, G) + \ell\log(1 - D(s_t, a_{t,i}, s_{t+1}, G))\right] \\ &+ \mathbb{E}_{(s_t,a_{t,i},s_{t+1},G)\sim\mathcal{D}'}\left[\ell\log D(s_t, a_{t,i}, s_{t+1}, G) + (1-\ell)\log(1 - D(s_t, a_{t,i}, s_{t+1}, G))\right] \end{aligned}$$

where ℓ is the label smoothing hyper-parameter, and $\mathcal{D}$ and $\mathcal{D}'$ are the positive and negative examples discussed in Section 4.

Variational Goal Generator: The variational goal generator has a encoder and a decoder. The encoder encodes the sequence of observations in a trajectory τ, and the decoder samples a natural language goal G for that trajectory. Let e^a_t be the embedding of action selected at step t. We first use a bi-directional LSTM to encode actions in the trajectory $h^a_1, h^a_2, ..., h^a_T = \text{biLSTM}(e^a_1, e^a_2, ..., e^a_T)$, then we use h^a_{t-1} to attend over observations at t: $\alpha^a_i = \text{Softmax}(\text{Linear}(h^a_{t-1}) \cdot f_{t,i})$. Then the summarized observation feature vector at t is $f_t = \sum_{i=1}^{36} \alpha^a_i f_{t,i}$. Then the generative process of the

n-th word in the natural language goal is as follows:

$$
\begin{aligned}
h_1^o, h_2^o, ..., h_T^o &= \text{biLSTM}(f_1, f_2, ..., f_T) \\
h^o &= \text{ElementWiseMax}(h_1^o, h_2^o, ..., h_T^o) \\
\mu_{prior}, \sigma_{prior}^2 &= \text{MLP}(h^o), \text{Softplus}(\text{MLP}(h^o)) \\
p(z) &= \mathcal{N}(z|\mu_{prior}, \sigma_{prior}^2 \mathbf{I}) \\
z &\sim \mathcal{N}(\mu_{prior}, \sigma_{prior}^2 \mathbf{I}) \\
h_n^g &= \text{LSTMCell}\left(h_{n-1}^g, e_n^w\right) \\
\alpha_t^o &= \text{Softmax}\left(\text{Linear}([h_n^g; z]) \cdot h_t^o\right) \\
p_{n+1}(w) &= \text{Softmax}\left(\text{MLP}\left(\left[\sum_{t=1}^{T} \alpha_t^o h_t^o; h_n^g; z\right]\right)\right) \\
w &= \arg\max_w p_{n+1}(w)
\end{aligned}
$$

where Softplus is a function $f(x) = \ln(1 + e^x)$ to ensure that σ^2 is positive, e_n^w is the word embedding of w_n, and $p_{n+1}(w)$ is the probability of selecting a word w at position $n + 1$. The posterior distribution of the latent variable z is approximated by

$$
\begin{aligned}
h^g &= \text{ElementWiseMax}(h_1^g, h_2^g, ..., h_N^g) \\
\mu_{posterior}, \sigma_{posterior}^2 &= \text{MLP}(h^o, h^g), \text{Softplus}(\text{MLP}(h^o, h^g)) \\
q(z|\tau, G) &= \mathcal{N}(\mu_{posterior}, \sigma_{posterior}^2 \mathbf{I})
\end{aligned}
$$

We maximize the variational lower bound $-\lambda D_{KL}(q(z|\tau, G)||p(z)) + \mathbb{E}_{q(z|\tau,G)}[\log p(G|z, \tau)]$. λ is KL multiplier that gradually increases from 0 to 1 during training.

Goal-agnostic Policy π_p. In Section 3.4, we propose to learn a goal-agnostic policy π_p that can sample trajectories in new environments to construct expert demonstrations. We learn this goal-agnostic policy π_p using the same network architecture as the base model mentioned above, except that now that the action selection is not conditioned on goals. That is, the probability of selecting an action which originally in Equation 6 now becomes

$$
p_i^a = \text{Softmax}\left(\text{BiLinear}\left(h_t^s, e_{t,i}^a\right)\right)
$$

Hyper-parameters: By default, the hidden size of MLP in all the models in this paper is 512, and the number of layers is 2. The hidden size of LSTM is 512 (256 for biLSTM) and the number of layer is 1. The size of word embedding is 300. The dropout rate is 0.5 for all the models. The batch sizes for training SAC and the discriminator are both 100. All the models are optimized by Adam and the learning rate is $1e-4$. The target entropy in SAC is $-\log(1/\text{MAX\_NUM\_ACTIONS}) * 0.1$ where MAX_NUM_ACTIONS $= 14$. The size of replay buffer in SAC is 10^6. The reward discount factor is 0.99. Our implementation of SAC is based on the rlpyt code base [41]. We apply label smoothing when optimizing the discriminator. The positive label is 0.95 and the negative label is 0.05. The size of the latent variable in variational goal generator is 128. When training the variational goal generator, we gradually increase the KL multiplier λ from 0 to 1 during the first $50,000$ gradient updates, and we also apply a word dropout with probability 0.25. All the experiments are run 8 times and the average and standard deviation of the success rate are reported. One run includes 1 million interactions with the environments and takes about 30 hours to train on a NVIDIA V100 GPU.

C Examples of Generated Goals

The following are examples of human-annotated vs. GoalGenerator-generated instructions for a trajectory in the Room-2-Room dataset.

Example 1:

> **Human**: *Exit room through the doorway near the fireplace. Keep right and walk through the hallway, turn right, enter the bedroom and wait near the bed.*

GoalGenerator: *Walk through the doorway to the right of the fireplace. Walk past the fireplace and turn right. Walk down the hallway and turn right into the bedroom . Stop in front of the bed.*

Example 2:

Human: *Turn right and exit the room through the door on the left. Turn left and walk out into the hallway. Turn right and enter bedroom. Walk through the bedroom and into the bathroom. Stop once you are in front of the sink.*

GoalGenerator: *Turn around and walk through the doorway. Turn left and walk down the hallway. Turn right at the end of the hall and enter the bathroom. Stop in front of the sink.*

Example 3:

Human: *Turn around and walk outside through the door behind you. Once outside, turn right and walk to the other end of the pool, At the end of the pool turn right and enter the door back into the house and stop behind the 2 white chairs.*

GoalGenerator: *Turn around and walk out of the room. Once out, turn right and walk towards the pool. Once you reach the pool, turn right and walk towards the large glass doors. Once you reach the 2 chairs, turn right and enter the large room. Stop once you reach the couch.*

The following are goals generated by the GoalGenerator for trajectories sampled by the goal-agnostic policy in new environments (Section 3.4).

1. *Walk straight across the room and past the couch. Walk straight until you get to a room with a large green vase. Wait there.*
2. *Exit the bathroom and turn right. Walk down the hallway and turn right. Walk straight until you get to a kitchen area. Wait near the stove.*
3. *Walk straight down the hallway and into the room with the large mirror. Walk through the door and stop in front of the table with the plant on it.*

D Additional Experiments

Expert Goal Relabeling	Hindsight Goal Relabeling	Hindsight Goal Sampling	Hindsight Experience Replay	seen validation success rate
–	–	–	–	0.449 ± 0.0130
✓	–	–	–	0.450 ± 0.0122
–	✓	–	–	0.427 ± 0.0048
–	–	✓	–	0.399 ± 0.0124
–	–	–	✓	0.464 ± 0.0126
✓	✓	–	–	0.504 ± 0.0185
✓	–	✓	–	0.444 ± 0.0134
✓	–	–	✓	0.459 ± 0.0050
–	✓	✓	–	0.503 ± 0.0117
–	✓	–	✓	0.101 ± 0.0790
–	–	✓	✓	0.378 ± 0.0192
✓	✓	✓	–	$\mathbf{0.530 \pm 0.0138}$
✓	✓	–	✓	0.491 ± 0.0146
✓	–	✓	✓	0.452 ± 0.0088
–	✓	✓	✓	0.476 ± 0.0166
✓	✓	✓	✓	0.521 ± 0.0093

Table 3: A complete ablation study of EGR, HGR, HGS and HER on seen validation set.

	5,000	10,000	15,000	20,000
Random Explore	0.274 ± 0.0086	0.303 ± 0.0092	0.347 ± 0.0098	0.348 ± 0.0115
LangGoalIRL_SS_G2	0.430 ± 0.0080	0.459 ± 0.0084	0.483 ± 0.0124	0.490 ± 0.0068
LangGoalIRL_SS	0.452 ± 0.0067	0.475 ± 0.0062	0.491 ± 0.0065	$\mathbf{0.496 \pm 0.0039}$

Table 4: The success rate in unseen validation with different self-supervised learning setting. Algorithms sample $5,000$ to $20,000$ demonstrations in unseen environments. Random Explore samples demonstrations by randomly select an unvisited nearby viewpoint, while LangGoalIRL_SS and LangGoalIRL_SS_G2 sample demonstrations using goal-agnostic policy π_p. During fine-tuning, LangGoalIRL_SS_G2 samples goals only from $\{(\boldsymbol{E}, \boldsymbol{G}^v) | v \leq 2\}$, while the other two algorithms sample goals from $\{(\boldsymbol{E}, \boldsymbol{G}^v) | v \geq 1\}$.

	0	5000	10,000	15,000	20,000
LangGoalIRL_BASE	0.449 ± 0.0130	0.469 ± 0.0111	0.472 ± 0.0105	0.478 ± 0.0108	0.481 ± 0.0156
LangGoalIRL	0.530 ± 0.0138	0.532 ± 0.0126	0.535 ± 0.0113	0.538 ± 0.0155	$\mathbf{0.543 \pm 0.0125}$

Table 5: Self-supervised learning can also be applied to seen validation set. This table shows the success rate in seen validation when self-supervised learning is used to augment the expert demonstrations in seen validation. We use the goal-agnostic policy π_p to sample $5,000$ to $20,000$ trajectories, relabel them with goal generator, and add them to the seen validation for training.

#relabeled expert goals.	0	1	2	3
LangGoalIRL	0.503 ± 0.0117	0.525 ± 0.0090	$\mathbf{0.530 \pm 0.0138}$	0.528 ± 0.0105
LangGoal_ONLY_EGR	0.449 ± 0.0130	0.446 ± 0.0169	0.450 ± 0.0122	0.450 ± 0.0155

Table 6: Success rate in seen validation given different number of relabeled expert goals. This corresponds to setting N in Section 3.3.1 to 0, 1, 2 and 3.

Table 3 shows the full ablation study results which also include hindsight experience replay. Notably, when apply HER together with HGR, the success rate drops to 0.101. The reason is that without EGR and HGS, the discriminator over-estimate the rewards under relabeled goals (as relabeled goals only appear in positive examples of the discriminator), and hence cannot give a good estimate of rewards to trajectories relabeled by HER.

Table 4 shows the policy performance in unseen validation with three different self-supervised learning settings. Random Explore randomly select actions instead of using π_p when generating demonstrations in unseen environments (but avoid visiting a viewpoint twice). After collecting demonstrations, Random Explore fine-tune the policy in the same way as LangGoalIRL_SS. LangGoalIRL_SS_G2 is different from LangGoalIRL_SS in that it does not iteratively sample goals from $\{(\boldsymbol{E}, \boldsymbol{G}^v) | v > 2\}$ as described in Section 3.3.3, so the policy experience a less diverse set of natural language goals compared with LangGoalIRL_SS. The number of generated demonstrations in unseen environments is set to $5,000$, $10,000$, $15,000$ or $20,000$. We can see that LangGoalIRL_SS performs much better than Random Explore, showing the importance of collecting demonstrations by π_p instead of a random policy. LangGoalIRL_SS also performs better than LangGoalIRL_SS_G2, however, the gap between LangGoalIRL_SS and LangGoalIRL_SS_G2 decreases as the number of generated demonstrations increases.

Table 5 shows the success rate of LangGoalIRL and LangGoalIRL_BASE when self-supervised learning is applied to existing environments. More specifically, in Section 3.4 we propose a self-supervised learning algorithm in new environments (unseen validation set), we can also apply the same idea to existing environments (seen validation set). To do so, we use the goal-agnostic policy π_p to sample trajectories in existing environments and relabel them with the variational goal generator. Then we augment the seen validation set with these generated demonstrations. During the training, EGR is only applied to the original expert demonstrations in the seen validation set. From the results we can see that self-supervised learning improves the performance of LangGoalIRL and LangGoalIRL_BASE in seen validation. Self-supervised learning has a larger impact on LangGoalIRL_BASE than on LangGaolIRL. This is reasonable since LangGoalIRL can already generalize the policy and reward function by our proposed three strategies (EGR, HGR and HGS).

Table 6 shows the success rate of LangGoalIRL in seen validation when we vary the number of relabeled expert goals (N in Section 3.3.1). We can see that the policy performances are comparable when the number of relabeled expert goals $N \geq 1$.