Vision-based Navigation with Language-based Assistance via Imitation Learning with Indirect Intervention

Khanh Nguyen
University of Maryland, College Park
kxnguyen@cs.umd.edu

Debadeepta Dey, Chris Brockett, Bill Dolan
Microsoft Research, Redmond
{dedey,Chris.Brockett,billdol}@microsoft.com

Abstract

We present Vision-based Navigation with Language-based Assistance (VNLA), a grounded vision-language task where an agent with visual perception is guided via language to find objects in photorealistic indoor environments. The task emulates a real-world scenario in that (a) the requester may not know how to navigate to the target objects and thus makes requests by only specifying high-level end-goals, and (b) the agent is capable of sensing when it is lost and querying an advisor, who is more qualified at the task, to obtain language subgoals to make progress. To model language-based assistance, we develop a general framework termed Imitation Learning with Indirect Intervention (I3L), and propose a solution that is effective on the VNLA task. Empirical results show that this approach significantly improves the success rate of the learning agent over other baselines in both seen and unseen environments.

Our code and data are publicly available at https://github.com/debadeepta/vnla.

1. Introduction

Rich photorealistic simulators are finding increasing use as research testbeds and precursors to real-world embodied agents such as self-driving cars and drones [53, 22, 33]. Recently, growing interest in grounded visual navigation from natural language is facilitated by the development of more realistic and complex simulation environments [2, 35, 9, 46, 59, 58, 51] in place of simple toy environments [5, 12, 39, 8, 7]. Several variants of this task have been proposed. In [2], agents learn to execute natural language instructions crowd-sourced from humans. [18] train agents to navigate to answer questions about objects in the environment. [21] present a scenario where a guide and a tourist chat to direct the tourist to a destination.

In this paper, we present Vision-based Navigation with Language-based Assistance (VNLA), a grounded vision-language task that models a practical scenario: a mobile agent, equipped with monocular visual perception, is requested via language to assist humans with finding objects in indoor environments. A realistic setup of this scenario must (a) not assume the requester knows how to accomplish a task before requesting it, and (b) provide additional assistance to the agent when it is completely lost and can no longer make progress on a task. To accomplish (a), instead of using detailed step-by-step instructions, we request tasks through high-level instructions that only describe end-goals (e.g. "Find a pillow in one of the bedrooms"). To fulfill (b), we introduce into the environment an advisor who is present at all times to assist the agent upon request with low-level language subgoals such as "Go forward three steps, turn left". In VNLA, therefore, the agent must (a) ground the object referred with the initial end-goal in raw visual inputs, (b) sense when it is lost and use an assigned budget for requesting help, and (c) execute language subgoals to make progress.

VNLA motivates a novel imitation learning setting that we term Imitation Learning with Indirect Intervention (I3L). In conventional Imitation Learning (IL), a learning agent learns to mimic a teacher, who is only available at training time, by querying the teacher's demonstrations on situations the agent encounters. I3L extends this framework in two ways. First, an advisor is present in the environment to assist the agent not only during training time but also at test time. Second, the advisor assists the agent not by directly making decisions on the agent's behalf, but by modifying the environment to influence its decisions. I3L models assistance via language subgoals by treating the subgoals as extra information added to the environment. We devise an algorithm for the I3L setting that yields significant improvements over baselines on the VNLA task on both seen and unseen environments.

The contributions of this paper are: (a) a new task VNLA that represents a step closer to real-world applications of mobile agents accomplishing indoor tasks (b) a novel IL framework that extends the conventional framework to modeling indirect intervention, and (c) a general solution to I3L that is shown to be effective on the VNLA

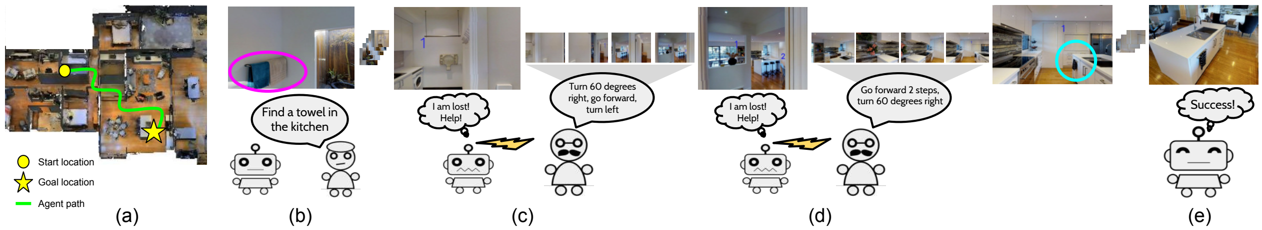

Figure 1: An example run in an unseen environment. (a) A bird-eye view of the environment annotated with the agent's path. The agent observes the environment only through a first-person view. (b) A requester (wearing a hat) asks the agent to "find a towel in the kitchen". Two towels (pink circle) are in front of the agent but the room is labeled as a "bathroom". The agent ignores them without being given the room label. (c) The agent escapes the bathroom but runs into an unfamiliar region. Sensing that it is lost, the agent signals the advisor (with mustache) for help. The advisor responds with an "easier" low-level subgoal "turn 60 degrees right, go forward, turn left". (d) After executing the subgoal, the agent is closer to the kitchen but is still confused. It thus requests help one more time. After making this request, the agent has exhausted its request budget and can only rely on its own. (e) Executing the second subgoal helps the agent see the target towel (cyan circle). It successfully walks to the goal without further assistance. A video demo is at https://youtu.be/Vp6C29qTKQ0.

task. The task is accompanied by a large-scale dataset based on the photorealistic Matterport3D environment [10, 2].

2. Related Work

Language and robots. Learning to translate natural language commands to physical actions is well-studied at the intersection of language and robotics. Proposals include a variety of grounded parsing models that are trained from data [41, 42, 36] and models that interact with robots via natural language queries against a knowledge base [52] Most relevant to the present work are [44] who ground natural language to robotic manipulator instructions using Learning-from-Demonstration (LfD) and [23] who employ imitation learning of natural language instructions using humans following directions as demonstration data. In [30], verbal constraints are used for safe robot navigation in complex real-world environments.

Simulated environments. Simple simulators as Puddle World Navigation [31] and Rich 3D Blocks World [5, 4] have facilitated understanding of fundamental representational and grounding issues by allowing for fast experimentation in easily-managed environments. Game-based and synthetic environments offer more complex visual contexts and interaction dynamics [33, 51, 9, 46, 22, 58, 14, 17]. Simulators that are more photo-realistic, realistic-physics enabled simulators are beginning to be utilized to train real-world embodied agents [53, 10, 13, 59, 21, 40].

End-to-end learning in rich simulators. [18] present the "Embodied Question Answering" (EmbodiedQA) task where an agent explores and answers questions about the environment. [19] propose a hierarchical solution for this task where each level of the hierarchy is independently warmed up with imitation learning and further improved with reinforcement learning. [27] and [58] similarly use reinforcement learning in simulated 3D environments for successful execution of written instructions. On the vision-language navigation task [2], cross-modal matching and self-learning significantly improve generalizability to unseen environments [24, 57].

Imitation learning. Imitation learning [20, 47, 49, 48, 11, 54] provides an effective learning framework when a teacher for a task is available or can be simulated. There has been rich work that focuses on relaxing unrealistic assumptions on the teacher. [25, 11, 45, 28, 55] study cases where teachers provide imperfect demonstrations. [60, 34, 32, 37] construct policies to minimize the number of queries to the teacher. [16] provide language instructions at every time step to guide meta-policy learning. To the best of our knowledge, however, no previous work on imitation learning has explored the case where the agent actively requests changes to the environment to facilitate its learning process.

3. Vision-based Navigation with Language-based Assistance

Setup. Our goal is to train an agent, with vision as the perception modality, that can navigate indoors to find objects by requesting and executing language instructions from humans. The agent is able to "see" the environment via a monocular camera capturing its first-person view as an RGB image. It is also capable of executing language instructions and requesting additional help when in need. The camera image stream and language instructions are the only external input signals provided; the agent is not given a map of the environments or its own location (e.g. via GPS or indoor localization techniques).

The agent starts at a random location in an indoor environment. A *requester* assigns it an object-finding task by sending a high-level *end-goal*, namely to locate an object in a particular room (e.g., "Find a cup in one of the bathrooms."). The task is always feasible: there is always an object instance in the environment that satisfies the end-goal. The agent is considered to have fulfilled the end-goal if it stops at a location within d meters along the shortest path to an instance of the desired object. Here d is the *success-radius*, a task-specific hyperparameter. During execution, the agent may get lost and become unable to progress. We enable the agent to automatically sense when this happens and signal an *advisor* for help.[1] The advisor then responds with language providing a *subgoal*. The subgoal is a short-term task that is significantly easier to accomplish than the end-goal. In this work, we consider subgoals that *describe the next k optimal actions* (e.g. "Go forward two steps, look right."). We assume that strictly following the subgoal helps the agent make progress.

By specifying the agent's task with a high-level end-goal, our setup does *not* assume the requester knows how to accomplish the task before requesting it. This aspect, along with the agent-advisor interaction, distinguishes our setup from instruction-following setups [2, 44, 43, 6, 12, 13], in which the requester provides the agent with detailed sequential steps to execute a task only at the beginning.

Constraint formulation. The agent faces a multi-objective problem: maximizing success rate while minimizing help requests to the advisor. Since these objectives are in conflict, as requesting help more often only helps increase success rate, we instead use a hard-constrained formulation: *maximizing success rate without exceeding a budgeted number of help requests*. The hard constraint indirectly specifies a trade-off ratio between success rate and help requests. The problem is reduced to single-objective once the constraint is specified by users based on their preferences.

4. Imitation Learning with Indirect Intervention

Motivated by the VNLA problem, we introduce Imitation Learning with Indirect Intervention (I3L), which models (realistic) scenarios where a learning agent is monitored by a more qualified expert (e.g., a human) and receives help through an imperfect communication channel (e.g., language).

Advisor. Conventional Imitation Learning (IL) settings [20, 47, 49, 48, 11, 54, 55] involve interaction between a learning agent and a teacher: the agent learns by querying and imitating demonstrations of the teacher. In I3L, in addition to interacting with a teacher, the agent also receives guidance from an *advisor*. Unlike the teacher, who only interacts with the agent at training time, the advisor assists the agent during both training and test time.

[1]For simplicity, we assume the advisor has perfect knowledge of the environment, the agent, and the task. In general, as the advisor's main task is to help the agent, perfect knowledge is not necessary. The advisor needs to only possess advantages over the agent (e.g., human-level common sense or reasoning ability, greater experience at indoor navigation, etc.).

Intervention. The advisor directs the agent to take a sequence of actions through an *intervention*, which can be direct or indirect. Interventions are direct when the advisor overwrites the agent's decisions with its own. By definition, direct interventions are always executed perfectly, i.e. the agent always takes actions the advisor wants it to take. In the case of indirect interventions, the advisor does not "take over" the agent but instead modifies the environment to influence its decisions.[2] To utilize indirect interventions, the agent must learn to *interpret* them, by mapping them from signals in the environment to sequences of actions in its action space. This introduces a new type of error into the learning process: intervention interpretation error, which measures how much the interpretations of the interventions diverge from the advisor's original intents.

Formulation. We assume the environment is a Markov decision process with state transition function $\mathcal{T}$. The agent maintains two policies: a main policy π_{main} for making decisions on the main task, and a help-requesting policy π_{help} for deciding when the advisor should intervene. We also assume the existence of teacher policies π^*_{main} and π^*_{help}, and an advisor Φ. Teacher policies are only available during training, while the advisor is always present. Having a policy π_{help} that decides when to ask for help reduces efforts of the advisor to monitor the agent. However, it does not prevent the advisor from actively intervening when necessary, because the advisor is able to control π_{help}'s decisions by modifying the environment appropriately. At a state s_t, if π_{help} decides that the advisor should intervene, the advisor outputs an indirect intervention that directs the agent to take a sequence of actions. In this work, we consider the case when the intervention instructs the agent to take the next k actions $(a_t, a_{t+1}, \cdots, a_{t+k-1})$ suggested by the teacher

$$\begin{aligned} a_{t+i} &= \pi^*_{\text{main}}(s_{t+i}), \\ s_{t+i+1} &= \mathcal{T}\left(s_{t+i}, a_{t+i}\right) \quad 0 \le i < k \end{aligned} \tag{1}$$

The state distribution induced by the agent, $\mathbf{p}_{\text{agent}}$, depends on both π_{main} and π_{help}. As in standard imitation learning, in I3L, the agent's objective is to minimize expected loss on the agent-induced state distribution:

$$\hat{\pi}_{\text{main}}, \hat{\pi}_{\text{help}} = \arg\min_{\pi_{\text{main}}, \pi_{\text{help}}} \mathbb{E}_{s\sim\mathbf{p}_{\text{agent}}}\left[\mathcal{L}\left(s, \pi_{\text{main}}, \pi_{\text{help}}\right)\right] \tag{2}$$

where $\mathcal{L}(., ., .)$ is a loss function.

Learning to Interpret Indirect Interventions. I3L can be viewed as an imitation learning problem in a *dynamic* environment, where the environment is altered due to indirect interventions. Provided that teacher policies are well-defined in the altered environments, an I3L problem can be

[2]The direct/indirect distinction is illustrated more tangibly in a physical agent such as a self-driving car. Turning off automatic driving mode and taking control of the steering wheel constitutes a direct intervention, while issuing a verbal command to stop the car represents an indirect intervention (the command is treated as new information added to the environment).

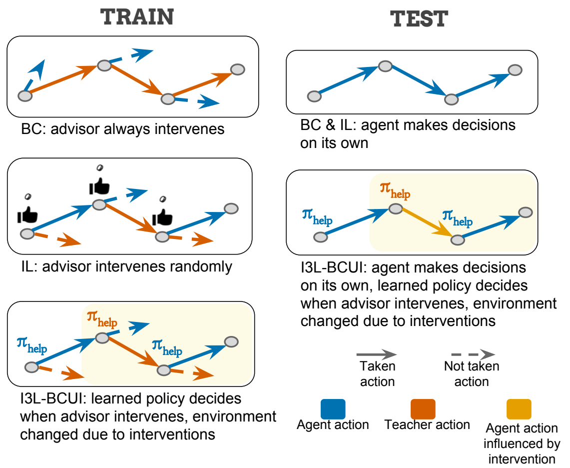

Figure 2: Comparison between I3L trained with behavior cloning under interventions (I3L-BCUI), imitation learning (IL), and behavior cloning (BC) at training time (left) and test time (right). Gray dots represent states and arrows represent actions. Bounding boxes of different colors represent different environments.

decomposed into a series of IL problems, each of which can be solved with standard IL algorithms. It turns out, however, that defining such policies in VNLA is non-trivial. Even though in VNLA, we use an optimal shortest-path teacher navigation policy, introducing a subgoal to the environment may invalidate this policy. Suppose when an agent is executing a subgoal, it makes a mistake and deviates from the trajectory suggested by the subgoal (e.g., first turning right for a subgoal "turn left, go forward"). Then, continuing to follow the subgoal is no longer optimal. Always following the teacher is also not a good choice because the agent may learn to ignore the advisor and not be able to utilize subgoals effectively at test time.

Our solution, which we term as BCUI (**B**ehavior **C**loning **U**nder **I**nterventions), mixes IL with behavior cloning[3]. In this approach, the agent uses the teacher policy as the acting policy (behavior cloning) when executing an intervention (k steps since the intervention is issued). Thus, the agent never deviates from the trajectory suggested by the intervention and thus never encounters conflicts between the teacher and the advisor.[4] When no intervention is being executed, the agent uses the learned policy as the acting policy.

Connection to imitation learning and behavior cloning. Figure 2 illustrates why I3L trained under BCUI (I3L-BCUI) is a general framework that subsumes both IL and behavior cloning as special cases. The advisor in I3L-BCUI intervenes both directly (through behavior cloning) and indirectly (by modifying the environment) at training time, but intervenes only indirectly at test time. The teacher in IL or behavior cloning can be seen as an advisor who is only available during training and intervenes only directly. IL and behavior cloning employ simple help-requesting policies. In behavior cloning, the help-requesting policy is to always have the teacher intervene, since the agent always lets the teacher make decisions during training. Most IL algorithms employ a mixed policy as the acting policy during training, which is equivalent to using a Bernoulli-distribution sampler as the help-requesting policy. I3L-BCUI imposes no restrictions on the help-requesting policy, which can even be learned from data.

5. Environment and Data

Matterport3D simulator. The Matterport3D dataset [10] is a large RGB-D dataset for scene understanding in indoor environments. It contains 10,800 panoramic views inside 90 real building-scale scenes, constructed from 194,400 RGB-D images. Each scene is a residential building consisting of multiple rooms and floor levels, and is annotated with surface construction, camera poses, and semantic segmentation. Using this dataset, [2] implemented a simulator that emulates an agent walking in indoor environments. The pose of the agent is specified by its viewpoint and orientation (heading angle and elevation angle). Navigation is accomplished by traversing edges in a pre-defined environment graph in which edges connect reachable panoramic viewpoints that are less than 5m apart.

Visual input. The agent's pose is not provided as input to the agent. Given a pose, the simulator generates an RGB image representing the current first-person view. The image is fed into a ResNet-152 [26] pretrained on Imagenet [50] to extract a mean-pooled feature vector, which serves as the input to the agent. We use the precomputed image feature vectors publicly released by [2].

Action space. Following [2], we use a state-independent action space which consists of six actions: `left`, `right`, `up`, `down`, `forward` and `stop`. The `left`, `right`, `up`, `down` actions rotate the camera by 30 degrees. The `forward` action is defined as follows[5]: executing this action takes the agent to the next viewpoint on the shortest path from the current location to the goals if the viewpoint lies within 30 degrees of the center of the current view, or if it lies horizontally within 30 degrees of the center and the agent cannot bring the viewpoint closer to the center by looking up or down further; otherwise, executing this action takes the agent to the viewpoint closest to the center of the

[3] Behavior cloning in IL is equivalent to standard supervised learning in sequence-to-sequence learning, where during training ground-truth tokens (instead of predicted tokens) are always used to transition to the next steps.

[4] A known disadvantage of behavior cloning is that it creates a gap between training and testing conditions, because at test time the agent acts on the learned policy. Addressing this problem is left for future work.

[5] Our definition of the `forward` action, which is different from the one defined in [2], ensures the navigation teacher never suggests the agent actions that cause it to deviate from the shortest path to the goals.

Split	Number of data points	Number of goals
Train	94,798	139,757
Dev seen	4,874	7,768
Dev unseen	5,005	8,245
Test seen	4,917	7,470
Test unseen	5,001	7,537

Table 1: ASKNAV splits. A data point contains a single starting viewpoint but multiple goal viewpoints.

current view. We also define a help-requesting action space comprising two actions: `request` and `do_nothing`.

Data Generation. Using annotations provided in the Matterport3D dataset, we construct a dataset for the VNLA task, called ASKNAV. We use the same environment splits as [2]: 61 training, 11 development, and 18 test. After filtering out labels that occur less than five times, are difficult to recognize (e.g., "door frame"), low relevance (e.g., "window") or unknown, we obtain 289 object labels and 26 room labels. We define each data point as a tuple *(environment, start pose, goal viewpoints, end-goal)*. An end-goal is constructed as "Find [O] in [R]", where [O] is replaced with "a/an [object label]" (if singular) or "[object label]" (if plural), and [R] is replaced with "the [room label]" (if there is one room of the requested label) or "one of the `pluralize`([room label])" (if there are multiple rooms of the requested label). Table 1 summarizes the ASKNAV dataset. The development and test sets are further divided into an *unseen* set and a *seen* set. The seen set comprises data points that are generated in the training environments but do not appear in the training set. The unseen set contains data points generated in the development or test environments. The detailed data generation process is described in the Appendix.

6. Implementation

Notation. The agent maintains two policies: a navigation policy π_{nav} and a help-requesting policy π_{ask}. Each policy is stochastic, outputting a distribution $\mathbf{p}$ over its action space. An action a is chosen by selecting the maximum probability action of or sampling from the output distribution. The agent is supervised by a navigation teacher π^*_{nav} and a help-requesting teacher π^*_{ask} (both are deterministic policies), and is assisted by an advisor Φ. A dataset D is provided where the d-th data point consists of a start viewpoint $\mathbf{x}^{\text{start}}_d$, a start orientation ψ^{start}_d, a set of goal viewpoints $\{\mathbf{x}^{\text{end}}_{d,i}\}$, an end-goal $\mathbf{e}_d$, and the full map $\mathbf{M}_d$ of the corresponding environment. At any time, the teachers and the advisor have access to the agent's current pose and information provided by the current data point.

Algorithm. Algorithm 1 describes the overall procedure for training a VNLA agent. We train the navigation policy under the I3L-BCUI algorithm (Section 4) and train the help-requesting policy under behavior cloning. At time step t, the agent first receives a view of the environment from the current pose (Line 10). It computes a *tentative* navigation distribution $\mathbf{p}^{\text{nav}}_{t,1}$ (Line 11), which is used as an input to compute a help-requesting distribution $\mathbf{p}^{\text{ask}}_t$ (Line 12). Since the help-requesting policy is trained under behavior cloning, the agent invokes the help-requesting teacher π^*_{ask} (not the learned policy π_{ask}) to decide if it should request help (Line 13). If the help-requesting teacher decides that the agent should request help and the help-requesting budget has not been exhausted, the advisor Φ is invoked to provide help via a language subgoal $\mathbf{g}^{\text{sub}}_t$ (Lines 14-15). The subgoal is then *prepended* to the original end-goal $\mathbf{g}^{\text{main}}_{0,d}$ to form a new end-goal $\mathbf{g}^{\text{main}}_t$ (Line 16). If the condition for requesting help is not met, the end-goal is kept unchanged (Line 19). After the help-requesting decision has been made, the agent computes a *final* navigation distribution $\mathbf{p}^{\text{nav}}_{t,2}$ by invoking the learned policy π_{nav} the second time. Note that when computing this distribution, the last help-requesting action is no longer a^{ask}_{t-1} but has become a^{ask}_t. The agent selects the acting navigation policy based on the principle of the I3L-BCUI algorithm. Specifically, if the agent has requested help within the last k steps, i.e. it is still executing a subgoal, it uses the teacher policy to act (Line 24). Otherwise, it samples an action from the final navigation distribution (Line 26). In Lines 28-29, the learned policies are updated using an online learning algorithm. Finally, the agent transitions to the next pose according to the taken navigation action (Line 33).

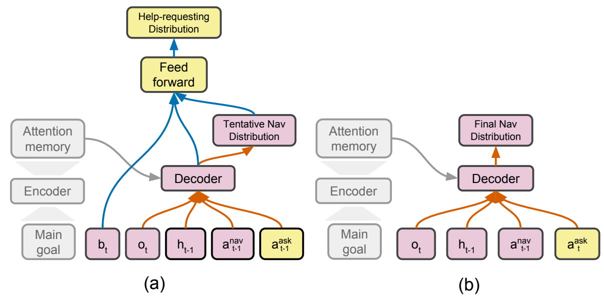

Figure 3: Two decoding passes of the navigation module. (a) The first decoding pass computes the tentative navigation distribution, which is used as a feature for computing the help-requesting distribution. (b) The second pass computes the final navigation distribution.

6.1. Agent

We model the navigation policy π_{nav} and the help-requesting policy π_{ask} as two separate neural network modules. The **navigation module** is an encoder-decoder model [3] with a multiplicative attention mechanism [38] and coverage modeling [56], which encodes an end-goal (a se-

Algorithm 1 VLNA training procedure

Initialize $\pi_{\text{nav}}, \pi_{\text{ask}}$ randomly.
k is the number of next actions a subgoal describes.
for $d = 1 \dots D$ **do**
  Reset environment to $(\mathbf{x}_d^{\text{start}}, \psi_d^{\text{start}})$.
  Compute time budget $\hat{T}$ and help-request budget $\hat{B}$.
  Initialize current help-request budget $b = \hat{B}$.
  Initialize $a_0^{\text{nav}}, a_0^{\text{ask}}$ to special action `<start>`.
  Initialize $\mathbf{g}_0^{\text{main}} = \mathbf{e}_d$, $\mathbf{x}^{\text{curr}} = \mathbf{x}_d^{\text{start}}$, $\psi^{\text{curr}} = \psi_d^{\text{start}}$.
  for $t = 1 \dots \hat{T}$ **do**
    Receive an image $\mathbf{o}_t$ of the current view.
    $\mathbf{p}_{t,1}^{\text{nav}} = \pi_{\text{nav}}(\mathbf{o}_t, \mathbf{g}_{t-1}^{\text{main}}, a_{t-1}^{\text{nav}}, a_{t-1}^{\text{ask}})$
    $\mathbf{p}_t^{\text{ask}} = \pi_{\text{ask}}(\mathbf{o}_t, \mathbf{g}_{t-1}^{\text{main}}, \mathbf{p}_{t,1}^{\text{nav}}, b)$
    $a_t^{\text{ask}} = a_t^{\text{ask}*} = \pi_{\text{ask}}^*(\mathbf{p}_{t,1}^{\text{nav}}, b, \mathbf{x}^{\text{curr}}, \psi^{\text{curr}}, \{\mathbf{x}_{d,i}^{\text{end}}\}, \mathbf{M}_d)$
    if $b > 0$ and $a_t^{\text{ask}} ==$ `request` **then**
      $\mathbf{g}_t^{\text{sub}} = \Phi(\mathbf{x}^{\text{curr}}, \psi^{\text{curr}}, \{\mathbf{x}_{d,i}^{\text{end}}\}, \mathbf{M}_d, k)$
      $\mathbf{g}_t^{\text{main}} = \mathbf{g}_t^{\text{sub}} \odot \mathbf{e}_d$
      $b \leftarrow b - 1$
    else
      $\mathbf{g}_t^{\text{main}} = \mathbf{g}_{t-1}^{\text{main}}$
    end if
    $\mathbf{p}_{t,2}^{\text{nav}} = \pi_{\text{nav}}(\mathbf{o}_t, \mathbf{g}_t^{\text{main}}, a_{t-1}^{\text{nav}}, a_t^{\text{ask}})$
    $a_t^{\text{nav}*} = \pi_{\text{nav}}^*(\mathbf{x}^{\text{curr}}, \psi^{\text{curr}}, \{\mathbf{x}_{d,i}^{\text{end}}\}, \mathbf{M}_d)$
    if requested help within last k steps **then**
      $a_t^{\text{nav}} = a_t^{\text{nav}*}$
    else
      $a_t^{\text{nav}} \sim \mathbf{p}_{t,2}^{\text{nav}}$
    end if
    $\pi_{\text{ask}} \leftarrow \text{UpdatePolicy}(\pi_{\text{ask}}, \mathbf{p}_t^{\text{ask}}, a_t^{\text{ask}*})$
    $\pi_{\text{nav}} \leftarrow \text{UpdatePolicy}(\pi_{\text{nav}}, \mathbf{p}_{t,2}^{\text{nav}}, a_t^{\text{nav}*})$
    if $a_t^{\text{nav}} ==$ `stop` **then**
      break
    end if
    $\mathbf{x}^{\text{curr}}, \psi^{\text{curr}} \leftarrow \mathcal{T}(\mathbf{x}^{\text{curr}}, \psi^{\text{curr}}, a_t^{\text{nav}})$
  end for
end for

quence of words) and decodes a sequence of actions. Both the encoder and decoder are LSTM-based recurrent neural networks [29]. During time step t, if the end-goal is updated, the encoder generates an attention memory $\mathcal{M}_t = \left\{\mathbf{m}_1^{\text{enc}}, \cdots, \mathbf{m}_{|\mathbf{g}_t^{\text{main}}|}^{\text{enc}}\right\}$ by recurrently computing

$$\mathbf{m}_i^{\text{enc}} = \text{LSTM}_{\text{enc}}\left(\mathbf{m}_{i-1}^{\text{enc}}, \mathbf{g}_{t,i}^{\text{main}}\right), \quad 1 \leq i \leq |\mathbf{g}_t^{\text{main}}| \tag{3}$$

where LSTM_{enc} is the encoding LSTM, $\mathbf{g}_{t,i}^{\text{main}}$ is the embedding of the i-th word of the end-goal. Otherwise, $\mathcal{M}_t = \mathcal{M}_{t-1}$. The decoder runs *two* forward passes to compute the tentative and the final navigation distributions (Figure 3). The i-th decoding pass proceeds as:

$$\mathbf{h}_{t,i}^{\text{dec}} = \text{LSTM}_{\text{dec}}\left(\mathbf{h}_{t-1,2}^{\text{dec}}, \left[\mathbf{o}_t; \mathbf{a}_{t-1}^{\text{nav}}; \bar{\mathbf{a}}_t^{\text{ask}}\right]\right) \tag{4}$$
$$\mathbf{h}_{t,i}^{\text{att}} = \text{ATTEND}\left(\mathbf{h}_{t,i}^{\text{dec}}, \mathcal{M}_t\right) \tag{5}$$
$$\mathbf{p}_{t,i}^{\text{nav}} = \text{SOFTMAX}\left(\mathbf{W}_s^{\text{nav}}\, \mathbf{h}_{t,i}^{\text{att}}\right) \tag{6}$$

where $i \in \{1, 2\}$, $\mathbf{W}_s^{\text{nav}}$ is learned parameters, $\text{ATTEND}(.,.)$ is the multiplicative attention function, $\mathbf{o}_t$ is the visual feature vector of the current view, $\mathbf{a}_{t-1}^{\text{nav}}$ is the embedding of the last navigation action, and

$$\bar{\mathbf{a}}_t^{\text{ask}} = \begin{cases} \mathbf{a}_{t-1}^{\text{ask}} & \text{if } i = 1, \\ \mathbf{a}_t^{\text{ask}} & \text{if } i = 2 \end{cases} \tag{7}$$

is the embedding of the last help-requesting action.

The **help-requesting module** is a multi-layer feed-forward neural network with RELU activation functions and a softmax final layer. Its input features are:

- The visual $\mathbf{o}_t$ and the embedding of the current help-request budget $\mathbf{b}_t$.
- The tentative navigation distribution, $\mathbf{p}_{t,1}^{\text{nav}}$.
- The tentative navigation decoder states, $\mathbf{h}_{t,1}^{\text{dec}}$ and $\mathbf{h}_{t,1}^{\text{att}}$.

These features are concatenated and fed into the network to compute the help-requesting distribution

$$\mathbf{h}_t^{\text{ask}} = \text{FEED-FORWARD}_l\left(\left[\mathbf{o}_t; \mathbf{b}_t; \mathbf{p}_{t,1}^{\text{nav}}; \mathbf{h}_{t,1}^{\text{dec}}; \mathbf{h}_{t,1}^{\text{att}}\right]\right) \tag{8}$$
$$\mathbf{p}_t^{\text{ask}} = \text{SOFTMAX}\left(\mathbf{W}_s^{\text{ask}}\, \mathbf{h}_t^{\text{ask}}\right) \tag{9}$$

where $\mathbf{W}_s^{\text{ask}}$ is a learned parameter and FEED-FORWARD_l is a feed-forward network with l hidden layers. During training, we do not backpropagate errors of the help-requesting module through its input features. Preliminary experiments showed that doing so resulted in lower performance.

6.2. Teachers

Navigation teacher. The navigation teacher always chooses actions to traverse along the shortest path from the current viewpoint to the goal viewpoints. This path is optimal with respect to minimizing the walking distance to the goals, but is not necessarily optimal in the number of navigation actions. Given an agent's pose, the navigation teacher first adjusts the orientation using the camera-adjusting actions (`left`, `right`, `up`, `down`) until selecting the `forward` action advances the agent to the next viewpoint on the shortest path to the goals. The teacher issues the `stop` action when one of the goal viewpoints is reached.

Help-requesting teacher. Even with perfect information about the environment and the agent, computing an optimal help-requesting teacher policy is expensive because this policy depends on (a) the agent's internal state, which lies in a high-dimensional space and (b) the current learned navigation policy, which changes constantly during training. We design a heuristic-driven teacher, which decides to request help when:

(a) The agent deviates from the shortest path by more than δ meters. The distance from the agent to a path is defined as the distance from its current viewpoint to the nearest viewpoint on the path.

(b) The agent is "confused", defined as when the difference between the entropy of the uniform distribution and the entropy of the agent's tentative navigation distribution $\mathbf{p}^{\text{nav}}_{t,1}$ is smaller than a threshold ϵ.
(c) The agent has remained at the same viewpoint for the last μ steps.
(d) The help-request budget is greater than or equal to the number of remaining steps.
(e) The agent is at a goal viewpoint but the highest-probability action of the tentative navigation distribution is `forward`.

Although this heuristic-based teacher may not be optimal, our empirical results show that not only is it effective but it is also easy to imitate. Moreover, imitating a clairvoyant teacher is more sample-efficient (theoretically proven [48, 55]) and results in safer, more robust policies compared to maximizing a reward function with reinforcement learning (empirically shown [15]). The latter approach imposes weaker constraints on the regularity of the solution and may produce exploitative but unintuitive policies [1].

6.3. Advisor

Upon receiving a request from the agent, the advisor queries the navigation teacher for k consecutive steps to obtain a sequence of k actions (Equation 1). Next, actions $\{$`left`, `right`, `up`, `down`, `forward`, `stop`$\}$ are mapped to phrases {"turn left", "turn right", "look up", "look down", "go forward", "stop"}, respectively. Then, repeated actions are aggregated to make the language more challenging to interpret. For example, to describe a turn-right action that is repeated X times, the advisor says "turn Y degrees right" where $Y = X \times 30$ is the total degrees the agent needs to turn after repeating the turn-right action X times. Similarly, Z repeated `forward` actions are phrased as "go forward Z steps". The `up`, `down`, `stop` actions are not aggregated because they are rarely or never repeated. Finally, action phrases are joined by commas to form the final subgoal (e.g., "turn 60 degrees left, go forward 2 steps").

6.4. Help-request Budget

Let $\hat{T}$ be the time budget and B be the help-request budget. Suppose the advisor describes the next k optimal actions in response to each request. We define a hyperparameter $\tau \in [0, 1]$, which is the ratio between the total number of steps where the agent receives assistance and the time budget, i.e. $\tau \equiv \frac{B \cdot k}{\hat{T}}$. Given τ, $\hat{T}$ and k, we approximate B by an integral random variable $\hat{B}$

$$\begin{aligned} \hat{B} &= \lfloor B \rfloor + r \\ r &\sim \textsc{Bernoulli}\left(\{B\}\right) \\ B &= \frac{\hat{T} \cdot \tau}{k} \end{aligned} \tag{10}$$

where $\{B\} = B - \lfloor B \rfloor$ is the fractional part of B. The random variable r guarantees that $\mathbb{E}_r\left[\frac{\hat{B} \cdot k}{\hat{T}}\right] = \tau$ for a fixed $\hat{T}$ and any positive value of k, ensuring fairness when comparing agents interacting with advisors of different ks. Due to the randomness introduced by r, we evaluate an agent with multiple samples of $\hat{B}$. Detail on how we determine $\hat{T}$ for each data point is provided in the appendix.

7. Experimental Setup

Baselines. We compare our learned help-requesting policy (LEARNED) with the following baseline policies:

- NONE: never requests help.
- FIRST: requests help continuously from the beginning, up to $\hat{B}$.
- RANDOM: uniformly randomly chooses $\hat{B}$ steps to request help.
- TEACHER: follows the help-requesting teacher (π^*_{ask}).

In each experiment, the same help-requesting policy is used during training and evaluation.

Evaluation metrics. Our primary metrics are *success rate*, *room-finding success rate*, and *navigation error*. Success rate is the fraction of the test set on which the agent successfully fulfills the task. Room-finding success rate is the fraction of the test set on which the agent's final location is in the right room type. Navigation error measures the length of the shortest path from the agent's end viewpoint to the goal viewpoints. We evaluate each agent with five different random seeds and report means with 95% confidence intervals.

Hyperparameters. See the Appendix for details.

8. Results

Main results. Our main results are presented in Table 2. Overall, empowering the agent with the ability to ask for help and assisting it via subgoals greatly boost its performance. Requesting help is more useful in unseen environments, improvements over NONE of all other policies being higher on TEST UNSEEN than on TEST SEEN. Even a simple policy like FIRST yields success rate improvements of 12% and 14% over NONE on TEST SEEN and TEST UNSEEN respectively. The LEARNED policy outperforms all agent-agnostic polices (NONE, FIRST, RANDOM), achieving 9-10% improvement in success rate over RANDOM and 24-28% over NONE. An example run of the LEARNED agent is shown in Figure 1. The insignificant performance gaps between LEARNED and TEACHER indicates that the latter is not only effective but also easy to imitate[6]. RANDOM is largely more successful than FIRST, hinting that it

[6]There is a tradeoff between performance and learnability of the help-requesting teacher. By varying hyperparameters, we can obtain a teacher that achieves higher success rate but is harder to imitate.

π_{ask}	Success rate (%) ↑	Room-finding success rate (%) ↑	Mean navigation error (m) ↓
	Test seen		
NONE	28.39 ± 0.00	48.97 ± 0.00	6.29 ± 0.00
FIRST	40.33 ± 0.35	59.64 ± 0.22	4.36 ± 0.03
RANDOM	42.98 ± 0.44	54.61 ± 0.28	4.53 ± 0.03
LEARNED	52.09 ± 0.13	64.84 ± 0.23	3.48 ± 0.01
TEACHER	**52.26 ± 0.16**	**65.42 ± 0.25**	**3.42 ± 0.01**
	Test unseen		
NONE	6.36 ± 0.00	14.34 ± 0.00	11.30 ± 0.00
FIRST	20.00 ± 0.10	30.23 ± 0.40	7.56 ± 0.02
RANDOM	25.05 ± 0.31	33.72 ± 0.37	7.09 ± 0.05
LEARNED	34.50 ± 0.23	44.50 ± 0.36	5.66 ± 0.02
TEACHER	**34.95 ± 0.33**	**44.85 ± 0.39**	**5.61 ± 0.02**

Table 2: Performance of help-requesting policies on ASKNAV test sets.

Advisor	Subgoals	Train iterations	Test seen	Test unseen
Direct	✗	70k	51.07 ± 0.17	32.19 ± 0.28
Direct	✓	100k	52.09 ± 0.13	34.56 ± 0.21
Indirect	✓	100k	52.09 ± 0.13	34.50 ± 0.23

Table 3: Success rates (%) on ASKNAV test sets of agents interacting with different advisors. We compare agents that achieve comparable success rates on the DEV SEEN split.

may be ineffective to request help early too often. Nevertheless, FIRST is better than RANDOM at finding rooms on TEST SEEN. This may be because on TEST SEEN, although the complete tasks are previously unseen, the room-finding subtasks might have been assigned to the agent during training. For example, the agent might have never been requested to "find an armchair in the living room" during training, but it might have been taught to go to the living room to find other objects. When the agent is asked to find objects in a room it has visited, once the agent recognizes a familiar action history, it can reach the room by memory without much additional assistance. As the first few actions are crucial, requesting help early is closer to an optimal strategy than randomly requesting in this case.

Effects of subgoals. Subgoals not only serve to direct the agent, but also act as extra, informative input features. We hypothesize that the agent still benefits from receiving subgoals even when it interacts with an advisor who intervenes directly (in which case subgoals seem unneeded). To test this hypothesis, we train agents interacting with a *direct* advisor, who overwrites the agents' decisions by its decisions during interventions. We consider two variants of this advisor: one responds with a subgoal in response to each help request and the other does not. Table 3 compares these with

π_{ask}	Training set	Test seen	Test unseen
RANDOM	NOROOM	43.69 ± 0.37	33.41 ± 0.39
LEARNED	NOROOM	53.71 ± 0.19	44.77 ± 0.27
LEARNED	ASKNAV	53.85 ± 0.45	41.63 ± 0.24

Table 4: Success rates (%) on NOROOM test sets.

our standard indirect advisor, who at test time sends subgoals but does not overwrite the agent's decisions. Since success rates on TEST SEEN tend to take a long time to converge, we compare the success rates on TEST UNSEEN of agents that have comparable success rates on DEV SEEN (the success rates differ by no more than 0.5%). The two agents interpreting subgoals face a harder learning problem, and thus require more iterations to attain success rates on DEV SEEN comparable to that of the agent not interpreting subgoals. Receiving subgoals boosts sucess rate by more than 2% on TEST UNSEEN regardless of whether intervention is direct or indirect.

Does the agent learn to identify objects? The agent might have only learned to find the requested rooms and have "luckily" stood close to the target objects because there are only few viewpoints in a room. To verify if the agent has learned to identify objects after being trained with room type information, we setup a *transfer learning* experiment where an agent trained to fulfill end-goals with room types is evaluated with end-goals *without* room types. Following a procedure similar the one used to generate the ASKNAV in Section 5, we generate a NOROOM dataset, which contains end-goals without room type information. Each end-goal in the dataset has the form "Find [O]", where [O] is an object type. Finding any instance of the requested object in any room satisfies the end-goal. The number of goals in the training split of this dataset is comparable to that of the ASKNAV dataset (around 140k). More detail is provided in the Appendix. The results in Table 4 indicate that our agent, equipped with a learned help-requesting policy and trained with room types, learns to recognize objects, as it can find objects without room types significantly better than an agent equipped with a random help-requesting policy and trained specifically to find objects without room types (+10% on TEST SEEN and +8% on TEST UNSEEN in success rate). Unsurprisingly, directly training to find objects without room types yields best results in this setup because training and test input distributions are not mismatched.

9. Future Work

We are exploring ways to provide more natural, fully-linguistic question and answer interactions between advisor and agent, and better theoretical understanding of the I3L setting and resulting algorithms. We will also be investigating how to transfer from simulators to real-world robots.

References

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. 7

[2] 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*, volume 2, 2018. 1, 2, 3, 4, 5

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, 2015. 5

[4] Yonatan Bisk, Kevin J Shih, Yejin Choi, and Daniel Marcu. Learning interpretable spatial operations in a rich 3d blocks world. In *Association for the Advancement of Artificial Intelligence*, 2018. 2

[5] Yonatan Bisk, Deniz Yuret, and Daniel Marcu. Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761, 2016. 1, 2

[6] Valts Blukis, Dipendra Misra, Ross A Knepper, and Yoav Artzi. Mapping navigation instructions to continuous control actions with position-visitation prediction. In *Conference on Robot Learning*, pages 505–518, 2018. 3

[7] SRK Branavan, David Silver, and Regina Barzilay. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012. 1

[8] Satchuthananthavale RK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 82–90. Association for Computational Linguistics, 2009. 1

[9] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville. Home: A household multimodal environment. In *Proceedings of Advances in Neural Information Processing Systems*, 2017. 1, 2

[10] 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. 2, 4

[11] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume III, and John Langford. Learning to search better than your teacher. In *Proceedings of the International Conference of Machine Learning*, 2015. 2, 3

[12] David L Chen and Raymond J Mooney. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence*, volume 2, pages 1–2, 2011. 1, 3

[13] Howard Chen, Alane Shur, Dipendra Misra, Noah Snavely, Ian Artzi, Yoav, Stephen Gould, and Anton van den Hengel. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3

[14] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *Proceedings of the International Conference on Learning Representations*, 2019. 2

[15] Sanjiban Choudhury, Mohak Bhardwaj, Sankalp Arora, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Data-driven planning via imitation learning. *The International Journal of Robotics Research*, page 0278364918781001, 2018. 7

[16] John D Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, John DeNero, Pieter Abbeel, and Sergey Levine. Guiding policies with language via meta-learning. In *Proceedings of the International Conference on Learning Representations*, 2019. 2

[17] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532, 2018. 2

[18] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2

[19] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. *Proceedings of the Conference on Robot Learning*, 2018. 2

[20] Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009. 2, 3

[21] Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*, 2018. 1, 2

[22] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1, 2

[23] Felix Duvallet, Thomas Kollar, and Anthony Stentz. Imitation learning for natural language direction following through unknown environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1047–1053. IEEE, 2013. 2

[24] 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 *Neural Information Processing Systems (NeurIPS)*, 2018. 2

[25] Yang Gao, Ji Lin, Fisher Yu, Sergey Levine, Trevor Darrell, et al. Reinforcement learning from imperfect demonstrations. *Proceedings of the 35th International Conference on Machine Learning*, 2018. 2

[26] 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. 4

[27] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017. 2

[28] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, et al. Deep q-learning from demonstrations. *Association for the Advancement of Artificial Intelligence*, 2018. 2

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 6

[30] Zhe Hu, Jia Pan, Tingxiang Fan, Ruigang Yang, and Dinesh Manocha. Safe navigation with human instructions in complex scenes. *IEEE Robotics and Automation Letters*, 2019. 2

[31] Michael Janner, Karthik Narasimhan, and Regina Barzilay. Representation learning for grounded spatial reasoning. *Transactions of the Association of Computational Linguistics*, 6:49–61, 2018. 2

[32] Kshitij Judah, Alan P Fern, Thomas G Dietterich, et al. Active lmitation learning: formal and practical reductions to iid learning. *The Journal of Machine Learning Research*, 15(1):3925–3963, 2014. 2

[33] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016. 1, 2

[34] Beomjoon Kim and Joelle Pineau. Maximum mean discrepancy imitation learning. In *Robotics: Science and systems*, 2013. 2

[35] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 1

[36] Jayant Krishnamurthy and Tom M Mitchell. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765. Association for Computational Linguistics, 2012. 2

[37] Michael Laskey, Sam Staszak, Wesley Yu-Shu Hsieh, Jeffrey Mahler, Florian T Pokorny, Anca D Dragan, and Ken Goldberg. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 462–469. IEEE, 2016. 2

[38] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of Emperical Methods in Natural Language Processing*, 2015. 5

[39] Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Association for the Advancement of Artificial Intelligence*, 2006. 1

[40] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. *arXiv preprint arXiv:*, 2019. 2

[41] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference of Machine Learning*, 2012. 2

[42] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*, pages 403–415. Springer, 2013. 2

[43] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678. Association for Computational Linguistics, 2018. 3

[44] Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell me dave: Contextsensitive grounding of natural language to mobile manipulation instructions. In *in RSS*, 2014. 2, 3

[45] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018. 2

[46] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2

[47] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010. 2, 3

[48] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014. 2, 3, 7

[49] 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. 2, 3

[50] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large

scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 4

[51] Manolis Savva, Angel X Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017. 1, 2

[52] Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra K Misra, and Hema S Koppula. Robobrain: Large-scale knowledge engine for robots. *arXiv preprint arXiv:1412.0691*, 2014. 2

[53] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018. 1, 2

[54] Amr Sharaf and Hal Daumé III. Structured prediction via learning to search under bandit feedback. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 17–26, 2017. 2, 3

[55] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *Proceedings of the International Conference of Machine Learning*, 2017. 2, 3, 7

[56] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the Association for Computational Linguistics*, 2016. 5

[57] 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*, 2019. 2

[58] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. In *Workshop Track of the International Conference on Learning Representation*, 2018. 1, 2

[59] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018. 1, 2

[60] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end simulated driving. In *Association for the Advancement of Artificial Intelligence*, pages 2891–2897, 2017. 2