Active Visual Information Gathering for Vision-Language Navigation

Hanqing Wang[1], ✉Wenguan Wang[2], Tianmin Shu[3], Wei Liang[1], and Jianbing Shen[4]

[1]School of Computer Science, Beijing Institute of Technology [2]ETH Zurich
[3]Massachusetts Institute of Technology [4]Inception Institute of Artificial Intelligence
https://github.com/HanqingWangAI/Active_VLN

Abstract. Vision-language navigation (VLN) is the task of entailing an agent to carry out navigational instructions inside photo-realistic environments. One of the key challenges in VLN is how to conduct a robust navigation by mitigating the uncertainty caused by ambiguous instructions and insufficient observation of the environment. Agents trained by current approaches typically suffer from this and would consequently struggle to avoid random and inefficient actions at every step. In contrast, when humans face such a challenge, they can still maintain robust navigation by actively exploring the surroundings to gather more information and thus make more confident navigation decisions. This work draws inspiration from human navigation behavior and endows an agent with an active information gathering ability for a more intelligent vision-language navigation policy. To achieve this, we propose an end-to-end framework for learning an exploration policy that decides **i)** when and where to explore, **ii)** what information is worth gathering during exploration, and **iii)** how to adjust the navigation decision after the exploration. The experimental results show promising exploration strategies emerged from training, which leads to significant boost in navigation performance. On the R2R challenge leaderboard, our agent gets promising results all three VLN settings, *i.e.*, single run, pre-exploration, and beam search.

Keywords: Vision-Language Navigation · Active Exploration

1 Introduction

Vision-language navigation (VLN) [1] aims to build an agent that can navigate a complex environment following human instructions. Existing methods have made amazing progress via **i)** efficient learning paradigms (*e.g.*, using an ensemble of imitation learning and reinforcement learning [23,24], auxiliary task learning [10,12,23,28], or instruction augmentation based semi-supervised learning [7,20]), **ii)** multi-modal information association [9], and **iii)** self-correction [11,13]. However, these approaches have not addressed one of the core challenges in VLN – the uncertainty caused by ambiguous instructions and partial observability.

✉ Corresponding author: *Wenguan Wang* (wenguanwang.ai@gmail.com).

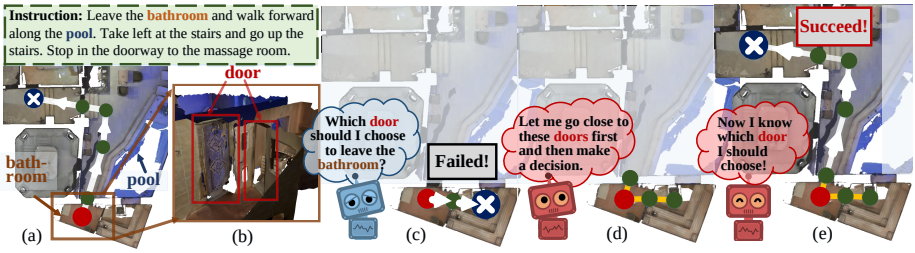

Fig. 1. (a) A top-down view of the environment with the groundtruth navigation path, based on the instructions. The start and end points are noted as red and blue circles, respectively. The navigation paths are labeled in white. (b) A side view of the bathroom in (a). (c) Previous agents face difficulties as there are two doors in the bathroom, hence causing the navigation fail. (d) Our agent is able to actively explore the environment for more efficient information collection. The exploration paths are labeled in yellow. (e) After exploring the two doors, our agent executes the instructions successfully.

Consider the example in Fig. 1, where an agent is required to navigate across rooms following human instructions: "*Leave the bathroom and walk forward along the pool.* ⋯". The agent might be confused because the bathroom has two doors, and it consequently fails to navigate to the correct location (Fig. 1(c)). In contrast, when faced with the same situation, our humans may perform better as we would first explore the two doors, instead of directly making a *risky* navigation decision. After collecting enough information, *i.e.*, confirming which one allows us to "*walk forward along the pool*", we can take a more confident navigation action. This insight from human navigation behavior motivates us to develop an agent that has a similar active exploration and information gathering capability. When facing ambiguous instructions or low confidence on his navigation choices, our agent can actively explore his surroundings and gather information to better support navigation-decision making (Fig. 1(d-e)). However, previous agents are expected to conduct navigation at all times and only collect information from a limited scope. Compared with these, which perceive a scene *passively*, our agent gains a larger *visual field* and improved robustness against complex environments and ambiguous instructions by actively exploring the surrounding.

To achieve this, we develop an active exploration module, which learns to 1) decide when the exploration is necessary, 2) identify which part of the surroundings is worth exploring, and 3) gather useful knowledge from the environment to support more robust navigation. During training, we encourage the agent to collect relevant information to help itself make better decisions. We empirically show that our exploration module successfully learns a good information gathering policy and, as a result, the navigation performance is significantly improved.

With above designs, our agent gets promising results on R2R [1] benchmark leaderboard, over all three VLN settings, *i.e.*, single run, pre-exploration, and beam search. In addition, the experiments show that our agent performs well in both seen and unseen environments.

2 Related Work

Vision and Language. Over the last few years, unprecedented advances in the design and optimization of deep neural network architectures have led to tremendous progress in computer vision and natural language processing. This progress, in turn, has enabled a multitude of multi-modal applications spanning both disciplines, including image captioning [25], visual question answering [3], visual grounding [26], visual dialog [6,27], and vision-language navigation [1]. The formulation of these tasks requires a comprehensive understanding of both visual and linguistic content. A typical solution is to learn a joint multi-modal embedding space, *i.e.*, CNN-based visual features and RNN-based linguistic representations are mapped to a common space by several non-linear operations. Recently, neural attention [25], which is good at mining cross-modal knowledge, has shown to be a pivotal technique for multi-modal representation learning.

Vision-Language Navigation (VLN). In contrast to previous vision-language tasks (*e.g.*, image captioning, visual dialog) only involving *static* visual content, VLN entails an agent to *actively* interact with the environment to fulfill navigational instructions. Although VLN is relatively new in computer vision (dating back to [1]), many of its core units/technologies (such as instruction following [2] and instruction-action mapping [15]) were introduced much earlier. Specifically, these were originally studied in natural language processing and robotics communities, for the focus of either language-based navigation in a controlled environmental context [2,5,14,15,17,21], or vision-based navigation in visually-rich real-world scenes [16,29]. The VLN simulator described in [1] unites these two lines of research, providing photo-realistic environments and human-annotated instructions (as opposed to many prior efforts using virtual scenes or formulaic instructions). Since its release, increased research has been conducted in this direction. Sequence-to-sequence [1] and reinforcement learning [24] based solutions were first adopted. Then, [7,20] strengthened the navigator by synthesizing new instructions. Later, combining imitation learning and reinforcement learning became a popular choice [23]. Some recent studies explored auxiliary tasks as self-supervised signals [10,12,23,28], while some others addressed self-correction for intelligent path planning [11,13]. In addition, Thomason *et al.* [22] identified unimodal biases in VLN, and Hu *et al.* [9] then achieved multi-modal grounding using a mixture-of-experts framework.

3 Methodology

Problem Description. Navigation in the Room-to-Room task [1] demands an agent to perform a sequence of navigation actions in real indoor environments and reach a target location by following natural language instructions.

Problem Formulation and Basic Agent. Formally, a language instruction is represented via textual embeddings as $\boldsymbol{X}$. At each navigation step t, the agent has a panoramic view [7], which is discretized into 36 single views (*i.e.*, RGB images). The agent makes a navigation decision in the panoramic action

space, which consists of K navigable views (reachable and visible), represented as $\boldsymbol{V}_t = \{\boldsymbol{v}_{t,1}, \boldsymbol{v}_{t,2}, \cdots, \boldsymbol{v}_{t,K}\}$. The agent needs to make a decision on which navigable view to go to (*i.e.*, choose an action $a_t^{\text{nv}} \in \{1, \cdots, K\}$ with the embedding $\boldsymbol{a}_t^{\text{nv}} = \boldsymbol{v}_{t,a_t^{\text{nv}}}$), according to the given instruction $\boldsymbol{X}$, history panoramic views $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \cdots, \boldsymbol{V}_{t-1}\}$ and previous actions $\{\boldsymbol{a}_1^{\text{nv}}, \boldsymbol{a}_2^{\text{nv}}, \cdots, \boldsymbol{a}_{t-1}^{\text{nv}}\}$. Conventionally, this dynamic navigation process is formulated in a recurrent form [1, 20]:

$$\boldsymbol{h}_t^{\text{nv}} = \text{LSTM}([\boldsymbol{X}, \boldsymbol{V}_{t-1}, \boldsymbol{a}_{t-1}^{\text{nv}}], \boldsymbol{h}_{t-1}^{\text{nv}}). \tag{1}$$

With current navigation state $\boldsymbol{h}_t^{\text{nv}}$, the probability of k^{th} navigation action is:

$$p_{t,k}^{\text{nv}} = \text{softmax}_k(\boldsymbol{v}_{t,k}^{\top} \boldsymbol{W}^{\text{nv}} \boldsymbol{h}_t^{\text{nv}}). \tag{2}$$

Here, $\boldsymbol{W}^{\text{nv}}$ indicates a learnable parameter matrix. The navigation action a_t^{nv} is chosen according to the probability distribution $\{p_{t,k}^{\text{nv}}\}_{k=1}^{K}$.

Basic Agent Implementation. So far, we have given a brief description of our basic navigation agent from a high-level view, also commonly shared with prior art. In practice, we choose [20] to implement our agent (but not limited to).

Core Idea. When following instructions, humans do not expect every step to be a "perfect" navigation decision, due to current limited visual perception, the inevitable ambiguity in instructions, and the complexity of environments. Instead, when we are uncertain about the future steps, we tend to explore the surrounding first and gather more information to mitigate the ambiguity, and then make a more informed decision. Our core idea is thus to equip an agent with such active exploration/learning ability. To ease understanding, we start with a naïve model which is equipped with a simplest exploration function (§3.1). We then complete the naïve model in §3.2 and §3.3 and showcase how a learned active exploration policy can greatly improve the navigation performance.

3.1 A Naïve Model with A Simple Exploration Ability

Here, we consider the most straightforward way of achieving our idea. At each navigation step, the agent simply explores all the navigable views and only one exploration step is allowed for each. This means that the agent explores the first direction, gathers surrounding information and then returns to the original navigation position. Next, it goes one step towards the second navigable direction and turns back. Such one-step exploration process is repeated until all the possible directions have been visited. The information gathered during exploration will be used to support current navigation-decision making.

Formally, at t^{th} navigation step, the agent has K navigable views, *i.e.*, $\boldsymbol{V}_t = \{\boldsymbol{v}_{t,1}, \boldsymbol{v}_{t,2}, \cdots, \boldsymbol{v}_{t,K}\}$. For k^{th} view, we further denote its K' navigable views as $\boldsymbol{O}_{t,k} = \{\boldsymbol{o}_{t,k,1}, \boldsymbol{o}_{t,k,2}, \cdots, \boldsymbol{o}_{t,k,K'}\}$ (see Fig. 2(a)). The subscript (t, k) will be omitted for notation simplicity. If the agent makes a one-step exploration in k^{th} direction, he is desired to collect surrounding information from $\boldsymbol{O}$. Specifically, keeping current navigation state $\boldsymbol{h}_t^{\text{nv}}$ in mind, the agent assembles the visual information from $\boldsymbol{O}$ by an attention operation (Fig. 2(b)):

$$\hat{\boldsymbol{o}}_{t,k} = \text{att}(\boldsymbol{O}, \boldsymbol{h}_t^{\text{nv}}) = \sum\nolimits_{k'=1}^{K'} \alpha_{k'} \boldsymbol{o}_{k'}, \text{ where } \alpha_{k'} = \text{softmax}_{k'}(\boldsymbol{o}_{k'}^{\top} \boldsymbol{W}^{\text{att}} \boldsymbol{h}_t^{\text{nv}}). \tag{3}$$

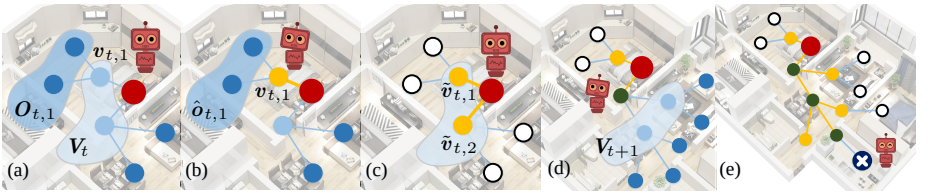

Fig. 2. Illustration of our naïve model (§3.1). (a) At t^{th} navigation step, the agent has a panoramic view $\boldsymbol{V}_t$. For k^{th} subview, we further denote its panoramic view as $\boldsymbol{O}_{t,k}$. (b) After making a one-step exploration in the first direction $\boldsymbol{v}_{t,1}$, the agent collects information $\hat{\boldsymbol{o}}_{t,1}$ from $\boldsymbol{O}_{t,1}$ via Eq. 3. (c) After exploring all the directions, the agent updates his knowledge, *i.e.*, $\{\tilde{\boldsymbol{v}}_{t,1}, \tilde{\boldsymbol{v}}_{t,2}\}$, via Eq. 4. (d) With the updated knowledge, the agent computes the navigation probability distribution $\{p^{\text{nv}}_{t,k}\}_k$ (Eq. 5) and makes a more reliable navigation decision (*i.e.*, $a^{\text{nv}}_t = 2$). (e) Visualization of navigation route, where yellow lines are the exploration routes and green circles are navigation landmarks.

Then, the collected information $\hat{\boldsymbol{o}}_{t,k}$ is used to update the current visual knowledge $\boldsymbol{v}_{t,k}$ about k^{th} view, computed in a residual form (Fig. 2(c)):

$$\tilde{\boldsymbol{v}}_{t,k} = \boldsymbol{v}_{t,k} + \boldsymbol{W}^{\text{o}}\hat{\boldsymbol{o}}_{t,k}. \tag{4}$$

In this way, the agent successively makes one-step explorations of all K navigable views and enriches his corresponding knowledge. Later, with the updated knowledge $\{\tilde{\boldsymbol{v}}_{t,1}, \tilde{\boldsymbol{v}}_{t,2}, \cdots, \tilde{\boldsymbol{v}}_{t,K}\}$, the probability of making k^{th} navigable action (originated in Eq. 2) can be formulated as (Fig. 2(d)):

$$p^{\text{nv}}_{t,k} = \text{softmax}_k(\tilde{\boldsymbol{v}}^{\top}_{t,k}\boldsymbol{W}^{\text{nv}}\boldsymbol{h}^{\text{nv}}_t). \tag{5}$$

Through this exploration, the agent should be able to gather more information from its surroundings, and then make a more reasonable navigation decision. In §4.3, we empirically demonstrate that, by equipping the basic agent with such a naïve exploration module, we achieve 4∼6% performance improvement in terms of Successful Rate (SR). This is impressive, as we only allow the agent to make one-step exploration. Another notable issue is that the agent simply explores all the possible directions, resulting in long Trajectory Length (TL)[1]. Next we will improve the naïve model, by tackling two key issues: "**how to decide where to explore**" (§3.2) and "**how to make deeper exploration**" (§3.3).

3.2 Where to Explore

In the naïve model (§3.1), the agent conducts exploration of all navigable views at every navigation step. Such a strategy is unwise and brings longer trajectories, and goes against the intuition that exploration is only needed at a few navigation steps, in a few directions. To address this, the agent should learn an *exploration-decision making* strategy, *i.e.*, more actively deciding which direction to explore.

[1] Here the routes for navigation and exploration are both involved in TL computation.

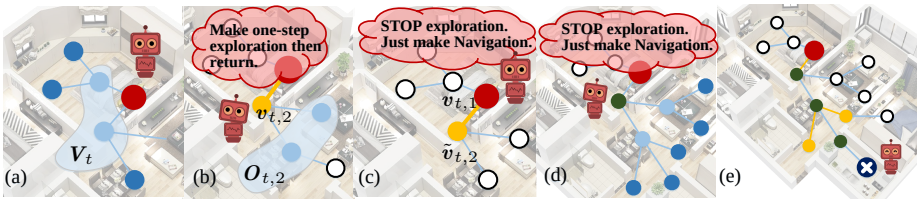

Fig. 3. Equip our agent with an exploration-decision making ability (§3.2). (a) The agent predicts a probability distribution $\{p^{\text{ep}}_{t,k}\}_{k=1}^{K+1}$ over exploration action candidates (*i.e.*, Eq. 6). (b) According to $\{p^{\text{ep}}_{t,k}\}_{k=1}^{K+1}$, the most "valuable" view is selected to make a one-step exploration. (c) The agent updates his knowledge $\tilde{\boldsymbol{v}}_{t,2}$ and makes a second-round exploration decision (Eq. 7). If `STOP` action is selected, the agent will make a navigation decision (Eq. 5) and start $(t+1)^{th}$ navigation step.

To achieve this, at each navigation step t, we let the agent make an exploration decision $a^{\text{ep}}_t \in \{1, \cdots, K+1\}$ from current K navigable views as well as a `STOP` action. Thus, the exploration action embedding $\boldsymbol{a}^{\text{ep}}_t$ is a vector selected from the visual features of the K navigable views (*i.e.*, $\boldsymbol{V}_t = \{\boldsymbol{v}_{t,1}, \boldsymbol{v}_{t,2}, \cdots, \boldsymbol{v}_{t,K}\}$), and the `STOP` action embedding (*i.e.*, $\boldsymbol{v}_{t,K+1} = \boldsymbol{0}$). To learn the exploration-decision making strategy, with current navigation state $\boldsymbol{h}^{\text{nv}}_t$ and current visual surrounding knowledge $\boldsymbol{V}_t$, the agent predicts a probability distribution $\{p^{\text{ep}}_{t,k}\}_{k=1}^{K+1}$ for the $K+1$ exploration action candidates (Fig. 3(a)):

$$\hat{\boldsymbol{v}}_t = \text{att}(\boldsymbol{V}_t, \boldsymbol{h}^{\text{nv}}_t), \qquad p^{\text{ep}}_{t,k} = \text{softmax}_k(\boldsymbol{v}^{\top}_{t,k} \boldsymbol{W}^{\text{ep}} [\hat{\boldsymbol{v}}_t, \boldsymbol{h}^{\text{nv}}_t]). \tag{6}$$

Then, an exploration action k^* is made according to $\arg\max_k p^{\text{ep}}_{t,k}$. If the `STOP` action is selected (*i.e.*, $k^* = K+1$), the agent directly turns to making a navigation decision by Eq. 2, without exploration. Otherwise, the agent will make a one-step exploration in a most "valuable" direction $k^* \in \{1, \cdots, K\}$ (Fig. 3(b)). Then, the agent uses the collected information $\hat{\boldsymbol{o}}_{t,k^*}$ (Eq. 3) to enrich his knowledge $\boldsymbol{v}_{t,k^*}$ about k^{*th} viewpoint (Eq. 4). With the updated knowledge, the agent makes a second-round exploration decision (Fig. 3(c)):

$$\begin{gathered} \tilde{\boldsymbol{V}}_t \leftarrow \{\boldsymbol{v}_{t,1}, \cdots, \tilde{\boldsymbol{v}}_{t,k^*}, \cdots, \boldsymbol{v}_{t,K}\}, \qquad \hat{\boldsymbol{v}}_t \leftarrow \text{att}(\tilde{\boldsymbol{V}}_t, \boldsymbol{h}^{\text{nv}}_t), \\ p^{\text{ep}}_{t,k^u} \leftarrow \text{softmax}_{k^u}(\boldsymbol{v}^{\top}_{t,k^u} \boldsymbol{W}^{\text{ep}} [\hat{\boldsymbol{v}}_t, \boldsymbol{h}^{\text{nv}}_t]). \end{gathered} \tag{7}$$

Note that the views that have been already explored are removed from the exploration action candidate set, and k^u indicates an exploration action that has not been selected yet. Based on the new exploration probability distribution $\{p^{\text{ep}}_{t,k^u}\}_{k^u}^{K+1}$, if the `STOP` action is still not selected, the agent will make a second-round exploration in a new direction. The above multi-round exploration process will be repeated until either the agent is satisfied with his current knowledge about the surroundings (*i.e.*, choosing the `STOP` decision), or all the K navigable directions are explored. Finally, with the newest knowledge about the surroundings $\tilde{\boldsymbol{V}}_t$, the agent makes a more reasonable navigation decision (Eq. 5, Fig. 3(d)). Our experiments in §4.3 show that, when allowing the agent to actively select

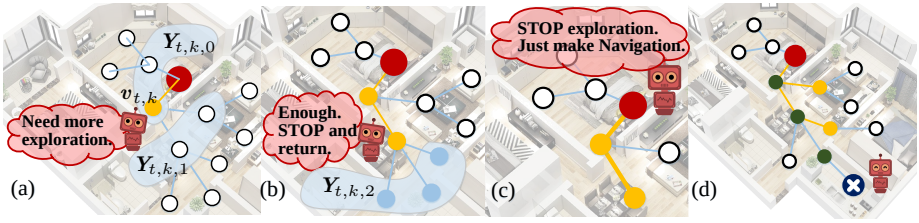

Fig. 4. Our full model can actively make multi-direction, multi-step exploration. (a) The agent is in 1^{st} exploration step ($s=1$), starting from k^{th} view at t^{th} navigation step. According to the exploration probability $\{p^{\text{ep}}_{s,k'}\}_{k'}$ (Eq. 11), the agent decides to make a further step exploration. (b) At 2^{nd} exploration step, the agent decides to finish the exploration of k^{th} view. (c) The agent thinks there is no other direction worth exploring, then makes a navigation decision based on the updated knowledge.

navigation directions, compared with the naïve model, TL is greatly decreased and even SR is improved (as the agent focuses on the most valuable directions).

3.3 Deeper Exploration

So far, our agent is able to make explorations only when necessary. Now we focus on how to let him conduct multi-step exploration, instead of simply constraining the maximum exploration length as one. Ideally, during the exploration of a certain direction, the agent should be able to go ahead a few steps until sufficient information is collected. To model such a sequential exploration decision-making process, we design a recurrent network based exploration module, which also well generalizes to the cases discussed in §3.1 and §3.2. Specifically, let us assume that the agent starts an exploration episode from k^{th} view $\boldsymbol{v}_{t,k}$ at t^{th} navigation step (Fig. 4(a)). At an exploration step s, the agent perceives the surroundings with a panoramic view and collects information from K' navigable views $\boldsymbol{Y}_{t,k,s} = \{\boldsymbol{y}_{t,k,s,1}, \boldsymbol{y}_{t,k,s,2}, \cdots, \boldsymbol{y}_{t,k,s,K'}\}$. With such a definition, we have $\boldsymbol{Y}_{t,k,0} = \boldsymbol{V}_t$. In §3.1 and §3.2, for k^{th} view at t^{th} navigation step, its panoramic view $\boldsymbol{O}_{t,k}$ is also $\boldsymbol{Y}_{t,k,1}$. The subscript (t,k) will be omitted for notation simplicity.

Knowledge Collection During Exploration: As the exploration module is in a recurrent form, the agent has a specific state $\boldsymbol{h}^{\text{ep}}_s$ at s^{th} exploration step. With $\boldsymbol{h}^{\text{ep}}_s$, the agent actively collects knowledge by assembling the surrounding information $\boldsymbol{Y}_s$ using an attention operation (similar to Eq. 3):

$$\hat{\boldsymbol{y}}_s = \text{att}(\boldsymbol{Y}_s, \boldsymbol{h}^{\text{ep}}_s). \tag{8}$$

Knowledge Storage During Exploration: As the agent performs multi-step exploration, the learned knowledge $\hat{\boldsymbol{y}}_s$ is stored in a memory network:

$$\boldsymbol{h}^{\text{kw}}_s = \text{LSTM}^{\text{kw}}(\hat{\boldsymbol{y}}_s, \boldsymbol{h}^{\text{kw}}_{s-1}), \tag{9}$$

which will eventually be used for supporting navigation-decision making.

Sequential Exploration-Decision Making for Multi-Step Exploration: Next, the agent needs to decide whether or not to choose a new direction for further exploration. In the exploration action space, the agent either selects one direction from the current K' reachable views to explore or stops the current exploration episode and returns to the original position at t^{th} navigation step. The exploration action a_s^{ep} is represented as a vector $\boldsymbol{a}_s^{\text{ep}}$ from the visual features of the K' navigable views (*i.e.*, $\boldsymbol{Y}_s = \{\boldsymbol{y}_{s,1}, \boldsymbol{y}_{s,2}, \cdots, \boldsymbol{y}_{s,K'}\}$), as well as the `STOP` action embedding (*i.e.*, $\boldsymbol{y}_{s,K'+1} = \boldsymbol{0}$). a_s^{ep} is predicted according to the current exploration state $\boldsymbol{h}_s^{\text{ep}}$ and collected information $\boldsymbol{h}_s^{\text{kw}}$. Hence, the computation of $\boldsymbol{h}_s^{\text{ep}}$ is conditioned on the current navigation state $\boldsymbol{h}_t^{\text{nv}}$, history exploration views $\{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \cdots, \boldsymbol{Y}_{s-1}\}$, and previous exploration actions $\{\boldsymbol{a}_1^{\text{ep}}, \boldsymbol{a}_2^{\text{ep}}, \cdots, \boldsymbol{a}_{s-1}^{\text{ep}}\}$:

$$\boldsymbol{h}_s^{\text{ep}} = \text{LSTM}^{\text{ep}}([\boldsymbol{h}_t^{\text{nv}}, \boldsymbol{Y}_{s-1}, \boldsymbol{a}_{s-1}^{\text{ep}}], \boldsymbol{h}_{s-1}^{\text{ep}}), \text{ where } \boldsymbol{h}_0^{\text{ep}} = \boldsymbol{h}_t^{\text{nv}}. \tag{10}$$

For k'^{th} exploration action candidate (reachable view), its probability is:

$$p_{s,k'}^{\text{ep}} = \text{softmax}_{k'}(\boldsymbol{y}_{s,k'}^{\top} \boldsymbol{W}^{\text{ep}} [\boldsymbol{h}_s^{\text{kw}}, \boldsymbol{h}_s^{\text{ep}}]). \tag{11}$$

The exploration action a_s^{ep} is chosen according to $\{p_{s,k'}^{\text{ep}}\}_{k'=1}^{K'+1}$.

Multi-Round Exploration-Decision Making for Multi-Direction Exploration: After S-step exploration, the agent chooses the `STOP` action when he thinks sufficient information along a certain direction k has been gathered (Fig. 4 (b)). He goes back to the start point at t^{th} navigation step and updates his knowledge about k^{th} direction, *i.e.*, $\boldsymbol{v}_{t,k}$, with the gathered information $\boldsymbol{h}_S^{\text{kw}}$. Thus, Eq. 4 is improved as:

$$\tilde{\boldsymbol{v}}_{t,k} = \boldsymbol{v}_{t,k} + \boldsymbol{W}^{\text{o}} \boldsymbol{h}_S^{\text{kw}}. \tag{12}$$

With the updated knowledge regarding the surroundings, the agent makes a second-round exploration decision:

$$\begin{aligned} \tilde{\boldsymbol{V}}_t \leftarrow \{\boldsymbol{v}_{t,1}, \cdots, \tilde{\boldsymbol{v}}_{t,k}, \cdots, \boldsymbol{v}_{t,K}\}, \qquad \hat{\boldsymbol{v}}_t \leftarrow \text{att}(\tilde{\boldsymbol{V}}_t, \boldsymbol{h}_t^{\text{nv}}), \\ p_{t,k^u}^{\text{ep}} \leftarrow \text{softmax}_{k^u}(\boldsymbol{v}_{t,k^u}^{\top} \boldsymbol{W}^{\text{ep}} [\hat{\boldsymbol{v}}_t, \boldsymbol{h}_t^{\text{nv}}]). \end{aligned} \tag{13}$$

Again, k^u indicates an exploration action that has not been selected yet. Then the agent can make another round of exploration in a new direction, until he chooses the `STOP` action (*i.e.*, the collected information is enough to help make a confident navigation decision), or has explored all K directions (Fig. 4(c)).

Exploration-Assisted Navigation-Decision Making: After multi-round multi-step exploration, with the newest knowledge $\tilde{\boldsymbol{V}}_t$ about the surroundings, the agent makes a more reliable navigation decision (Eq. 5): $p_{t,k}^{\text{nv}} = \text{softmax}_k(\tilde{\boldsymbol{v}}_{t,k}^{\top} \boldsymbol{W}^{\text{nv}} \boldsymbol{h}_t^{\text{nv}})$. Then, at $(t+1)^{th}$ navigation step, the agent makes multi-step explorations in several directions (or can even omit exploration) and then chooses a new navigation action. In §4.3, we will empirically demonstrate that our full model gains the highest SR score with only slightly increased TL.

Memory based Late Action-Taking Strategy: After finishing exploration towards a certain direction, if directly "going back" the start position and making next-round exploration/navigation, it may cause a lot of revisits. To alleviate

this, we let the agent store the visited views during exploration in an outside memory. The agent then follows a late action-taking strategy, *i.e.*, moving only when it is necessary. When the agent decides to stop his exploration at a direction, he stays at his current position and "images" the execution of his following actions without really going back. When he needs to visit a new point that is not stored in the memory, he will go to that point directly and updates the memory accordingly. Then, again, holding the position until he needs to visit a new point that is not met before. Please refer to the supplementary for more details.

3.4 Training

Our entire agent model is trained with two distinct learning paradigms, *i.e.*, 1) imitation learning, and 2) reinforcement learning.

Imitation Learning (IL). In IL, an agent is forced to mimic the behavior of its teacher. Such a strategy has been proved effective in VLN [1,7,13,15,20,23,24]. Specifically, at navigation step t, the teacher provides the teacher action $a_t^* \in \{1, \cdots, K\}$, which selects the next navigable viewpoint on the shortest route from the current viewpoint to the target viewpoint. The negative log-likelihood of the demonstrated action is computed as the IL loss:

$$\mathcal{L}_{\text{IL}}^{\text{nv}} = \sum\nolimits_t -\log\ p_{t,a_t^*}^{\text{nv}}. \tag{14}$$

The IL loss for the exploration is defined as:

$$\mathcal{L}_{\text{IL}}^{\text{ep}} = \sum\nolimits_t \sum\nolimits_{s=0}^{S} -\log\ p_{s,a_{t+s}^*}^{\text{ep}}, \tag{15}$$

where S is the maximum number of steps allowed for exploration. At t^{th} navigation step, the agent performs S-step exploration, simply imitating the teacher's navigation actions from t to $t+S$ steps. Though the goals of navigation and exploration are different, here we simply use the teacher navigation actions to guide the learning of exploration, which helps the exploration module learn better representations, and quickly obtain an initial exploration policy.

Reinforcement Learning (RL). Through IL, the agent can learn an off-policy that works relatively well on seen scenes, but it is biased towards copying the route introduced by the teacher, rather than learning how to recover from its erroneous behavior in an unseen environment [24]. Recent methods [20,23,24,29] demonstrate that the on-policy RL method Advantage Actor-Critic (A2C) [18] can help the agent explore the state-action space outside the demonstration path.

For RL based navigation learning, our agent samples a navigation action from the distribution $\{p_{t,k}^{\text{nv}}\}_{k=1}^{K}$ (see Eq. 2) and learns from rewards. Let us denote the reward after taking a navigation action a_t^{nv} at current view v_t as $r(v_t, a_t^{\text{nv}})$. As in [20,23], at each non-stop step t, $r^{\text{nv}}(v_t, a_t^{\text{nv}})$ is the change in the distance to the target navigation location. At the final step T, if the agent stops within 3 meters of the target location, we set $r^{\text{nv}}(v_T, a_T^{\text{nv}}) = +3$; otherwise $r^{\text{nv}}(v_T, a_T^{\text{nv}}) = -3$. Then, to incorporate the influence of the action a_t^{nv} on the future and account for the local greedy search, the total accumulated return with a discount factor

is adopted: $R_t^{\text{nv}} = \sum_{t'=t}^{T} \gamma^{t'-t} r^{\text{nv}}(v_{t'}, a_{t'}^{\text{nv}})$, where the discounted factor γ is set as 0.9. In A2C, our agent can be viewed as an actor and a state-value function $b^{\text{nv}}(\boldsymbol{h}_t)$, viewed as critic, is evaluated. For training, the actor aims to minimize the negative log-probability of action a_t^{nv} scaled by $R_t^{\text{nv}} - b^{\text{nv}}(\boldsymbol{h}_t^{\text{nv}})$ (known as the *advantage* of action a_t^{nv}), and the critic $b^{\text{nv}}(\boldsymbol{h}_t)$ aims to minimize the Mean-Square-Error between R_t^{nv} and the estimated value:

$$\mathcal{L}_{\text{RL}}^{\text{nv}} = -\sum\nolimits_t (R_t^{\text{nv}} - b^{\text{nv}}(\boldsymbol{h}_t^{\text{nv}}))\log\, p_{t,a_t^{\text{nv}}}^{\text{nv}} + \sum\nolimits_t (R_t^{\text{nv}} - b^{\text{nv}}(\boldsymbol{h}_t^{\text{nv}}))^2. \tag{16}$$

For RL-based exploration learning, we also adopt on-policy A2C for training. Specifically, let us assume a set of explorations $\{a_{t,k,s}^{\text{ep}}\}_{s=1}^{S_{t,k}}$ are made in a certain direction k at navigation step t, and the original navigation action (before exploration) is $a_t'^{\text{nv}}$. Also assume that the exploration-assisted navigation action (after exploration) is a_t^{nv}. The basic reward $r^{\text{ep}}(v_t, \{a_{t,k,s}^{\text{ep}}\}_s)$ for the exploration actions $\{a_{t,k,s}^{\text{ep}}\}_s$ is defined as:

$$r^{\text{ep}}(v_t, \{a_{t,k,s}^{\text{ep}}\}_s) = r^{\text{nv}}(v_t, a_t^{\text{nv}}) - r^{\text{nv}}(v_t, a_t'^{\text{nv}}). \tag{17}$$

This means that, after making explorations $\{a_{t,k,s}^{\text{ep}}\}_s$ at t^{th} navigation step in k^{th} direction, if the new navigation decision a_t^{nv} is better than the original one $a_t'^{\text{nv}}$, *i.e.*, helps the agent make a better navigation decision, a positive exploration reward will be assigned. More intuitively, such an exploration reward represents the benefit that this set of explorations $\{a_{t,k,s}^{\text{ep}}\}_s$ could bring for the navigation. We average $r^{\text{ep}}(v_t, \{a_{t,k,s}^{\text{ep}}\}_s)$ to each exploration action $a_{t,k,s}^{\text{ep}}$ as the immediate reward, *i.e.*, $r^{\text{ep}}(v_t, a_{t,k,s}^{\text{ep}}) = \frac{1}{S_{t,k}} r^{\text{ep}}(v_t, \{a_{t,k,s}^{\text{ep}}\}_s)$. In addition, to limit the length of exploration, we add a negative term β ($=-0.1$) to the reward of each exploration step. Then, the total accumulated discount return for an exploration action $a_{t,k,s}^{\text{ep}}$ is defined as: $R_{t,k,s}^{\text{ep}} = \sum_{s'=s}^{S_{t,k}} \gamma^{s'-s}(r^{\text{ep}}(v_t, a_{t,k,s'}^{\text{ep}}) + \beta)$. The RL loss for the exploration action $a_{t,k,s}^{\text{ep}}$ is defined as:

$$\mathcal{L}(a_{t,k,s}^{\text{ep}}) = -(R_{t,k,s}^{\text{ep}} - b^{\text{ep}}(\boldsymbol{h}_{t,k,s}^{\text{ep}}))\log\, p_{t,k,a_{t,k,s}^{\text{ep}}}^{\text{ep}} + (R_{t,k,s}^{\text{ep}} - b^{\text{ep}}(\boldsymbol{h}_{t,k,s}^{\text{ep}}))^2, \tag{18}$$

where b^{ep} is the critic. Then, similar to Eq. 16, the RL loss for all the exploration actions is defined as:

$$\mathcal{L}_{\text{RL}}^{\text{exp}} = -\sum\nolimits_t \sum\nolimits_k \sum\nolimits_s \mathcal{L}(a_{t,k,s}^{\text{ep}}). \tag{19}$$

Curriculum Learning for Multi-Step Exploration. During training, we find that, once the exploration policy is updated, the model easily suffers from extreme variations in gathered information, particularly for long-term exploration, making the training jitter. To avoid this, we adopt curriculum learning [4] to train our agent with incrementally improved exploration length. Specifically, in the beginning, the maximum exploration length is set to 1. After the training loss converges, we use current parameters to initialize the training of the agent with at most 2-step exploration. In this way, we train an agent with at most 6-step exploration (due to the limited GPU memory and time). This strategy

greatly improves the convergence speed (about $\times 8$ faster) with no noticeable diminishment in performance. Experiments related to the influence of the maximum exploration length can be found in §4.3.

Back Translation Based Training Data Augmentation. Following [7, 20], we use back translation to augment training data. The basic idea is that, in addition to training a *navigator* that finds the correct route in an environment according to the given instructions, an auxiliary *speaker* is trained for generating an instruction given a route inside an environment. In this way, we generate extra instructions for 176k unlabeled routes in Room-to-Room [1] training environments. After training the agent on the labeled samples from the Room-to-Room training set, we use the back translation augmented data for fine-tuning.

4 Experiment

4.1 Experimental Setup

Dataset. We conduct experiments on the Room-to-Room (R2R) dataset [1], which has 10,800 panoramic views in 90 housing environments, and 7,189 paths sampled from its navigation graphs. Each path is associated with three ground-truth navigation instructions. R2R is split into four sets: `training`, `validation seen`, `validation unseen`, and `test unseen`. There are no overlapping environments between the unseen and training sets.

Evaluation Metric. As in conventions [1, 7], five metrics are used for evaluation: *Success Rate* (SR), *Navigation Error* (NE), *Trajectory Length* (TL), *Oracle success Rate* (OR), and *Success rate weighted by Path Length* (SPL).

Implementation Detail. As in [1, 7, 23, 24], the viewpoint embedding $\boldsymbol{v}_{t,k}$ is a concatenation of image feature (from an ImageNet [19] pre-trained ResNet-152 [8]) and a 4-d orientation descriptor. A bottleneck layer is applied to reduce the dimension of $\boldsymbol{v}_{t,k}$ to 512. Instruction embeddings $\boldsymbol{X}$ are obtained from an LSTM with a 512 hidden size. For each LSTM in our exploration module, the hidden size is 512. For back translation, the speaker is implemented as described in [7].

4.2 Comparison Results

Performance Comparisons Under Different VLN Settings. We extensively evaluate our performance under three different VLN setups in R2R.

(1) Single Run Setting: This is the basic setup in R2R, where the agent conducts navigation by selecting the actions in a step-by-step, greedy manner. The agent is not allowed to: 1) run multiple trials, 2) explore or map the test environments before starting. Table 1 reports the comparison results under such a setting. The following are some essential observations. **i)** Our agent outperforms other competitors on the main metric SR, and some other criteria, *e.g.*, NE and OR. For example, in terms of SR, our model improves AuxRN [28] 3% and 5%, on `validation unseen` and `test unseen` sets, respectively, demonstrating our strong generalizability. **ii)** Our agent without data augmentation already outperforms many existing methods on SR and NE. **iii)** Our TL and SPL scores

Table 1. Comparison results on `validation seen`, `validation unseen`, and `test unseen` sets of R2R [1] under **Single Run** setting (§4.2). For compliance with the evaluation server, we report SR as fractions. *: back translation augmentation.

Models	Single Run Setting														
	`validation seen`					`validation unseen`					`test unseen`				
	SR↑	NE↓	TL↓	OR↑	SPL↑	**SR**↑	NE↓	TL↓	OR↑	SPL↑	**SR**↑	NE↓	TL↓	OR↑	SPL↑
Random	0.16	9.45	9.58	0.21	-	0.16	9.23	9.77	0.22	-	0.13	9.77	9.93	0.18	0.12
Student-Forcing [1]	0.39	6.01	11.3	0.53	-	0.22	7.81	8.39	0.28	-	0.20	7.85	8.13	0.27	0.18
RPA [24]	0.43	5.56	8.46	0.53	-	0.25	7.65	7.22	0.32	-	0.25	7.53	9.15	0.33	0.23
E-Dropout [20]	0.55	4.71	10.1	-	0.53	0.47	5.49	9.37	-	0.43	-	-	-	-	-
Regretful [13]	0.65	3.69	-	0.72	0.59	0.48	5.36	-	0.61	0.37	-	-	-	-	-
Ours	0.66	3.35	19.8	0.79	0.49	0.55	4.40	19.9	0.70	0.38	-	-	-	-	-
Speaker-Follower [7]*	0.66	3.36	-	0.74	-	0.36	6.62	-	0.45	-	0.35	6.62	14.8	0.44	0.28
RCM [23]*	0.67	3.53	10.7	0.75	-	0.43	6.09	11.5	0.50	-	0.43	6.12	12.0	0.50	0.38
Self-Monitoring [12]*	0.67	3.22	-	0.78	0.58	0.45	5.52	-	0.56	0.32	0.43	5.99	18.0	0.55	0.32
Regretful [13]*	0.69	3.23	-	0.77	0.63	0.50	5.32	-	0.59	0.41	0.48	5.69	13.7	0.56	0.40
E-Dropout [20]*	0.62	3.99	11.0	-	0.59	0.52	5.22	10.7	-	0.48	0.51	5.23	11.7	0.59	0.47
Tactical Rewind [11]*	-	-	-	-	-	0.56	4.97	21.2	-	0.43	0.54	5.14	22.1	0.64	0.41
AuxRN [28]*	0.70	3.33	-	0.78	0.67	0.55	5.28	-	0.62	0.50	0.55	5.15	-	0.62	0.51
Ours*	0.70	3.20	19.7	0.80	0.52	0.58	4.36	20.6	0.70	0.40	0.60	4.33	21.6	0.71	0.41

Table 2. Comparison results on `test unseen` set of R2R [1], under **Pre-Explore** and **Beam Search** settings (§4.2). To comply with the evaluation server, we report SR as fractions. *: back translation augmentation. −: unavailable statistics. †: a different beam search strategy is used, making the scores uncomparable.

Models	Pre-Explore Setting					Beam Search Setting		
	`test unseen`							
	SR↑	NE↓	TL↓	OR↑	SPL↑	**SR**↑	TL↓	SPL↑
Speaker-Follower [7]*	-	-	-	-	-	0.53	1257.4	0.01
RCM [23]*	0.60	4.21	9.48	0.67	0.59	0.63	357.6	0.02
Self-Monitoring [12]*	-	-	-	-	-	0.61	373.1	0.02
E-Dropout [20]*	0.64	3.97	9.79	0.70	0.61	0.69	686.8	0.01
AuxRN [28]*	0.68	3.69	-	0.75	0.65	0.70	†	†
Ours*	0.67	3.66	9.78	0.73	0.64	0.70	204.4	0.05

are on par with current art, with considering exploration routes into the metric computation. **iv)** If considering the routes for pure navigation, on `validation unseen` set, our TL is only about 9.4.

(2) Pre-Explore Setting: This setup, first introduced by [23], allows the agent to pre-explore the unseen environment before conducting navigation. In [23], the agent learns to adapt to the unseen environment through semi-supervised methods, using only pre-given instructions [23], without paired routes. Here, we follow a more strict setting, as in [20], where only the unseen environments can be accessed. Specifically, we use back translation to synthesize instructions for routes *sampled* from the unseen environments and fine-tune the agent on the synthetic data. As can be seen from Table 2, the performance of our method is significantly better than the existing methods [20, 23], improving the SR score from 0.64 to 0.67, and is on par with AuxRN [28].

Table 3. Ablation study on the `validation seen` and `validation unseen` sets of R2R [1] under the **Single Run** setting. See §4.3 for details.

Aspect	Model	Single Run Setting									
		`validation seen`					`validation unseen`				
		SR↑	NE↓	TL↓	OR↑	SPL↑	**SR**↑	NE↓	TL↓	OR↑	SPL↑
Basic agent	*w/o.* any exploration	0.62	3.99	11.0	0.71	0.59	0.52	5.22	10.7	0.58	0.48
Component	Our naïve model (§3.1) *1-step exploration+all directions*	0.66	3.55	40.9	0.81	0.19	0.54	4.76	35.7	0.71	0.16
	w. exploration decision (§3.2) *1-step exploration+parts of directions*	0.66	3.72	12.2	0.76	0.53	0.55	4.82	13.7	0.66	0.42
	w. further exploration *at most 4-step exploration+all directions*	0.70	3.15	69.6	0.95	0.13	0.60	4.27	58.8	0.89	0.12
Full model (§3.3) *parts of directions*	*as most* 1-step exploration	0.66	3.72	12.2	0.76	0.53	0.55	4.82	13.7	0.66	0.42
	at most 3-step exploration	0.68	3.21	17.3	0.79	0.52	0.57	4.50	18.6	0.69	0.40
	at most **4-step exploration**	0.70	3.20	19.7	0.80	0.52	0.58	4.36	20.6	0.70	0.40
	at most 6-step exploration	0.70	3.13	22.7	0.83	0.49	0.58	4.21	23.6	0.73	0.38

(3) Beam Search Setting: Beam search was originally used in [7] to optimize SR metric. Given an instruction, the agent is allowed to collect multiple candidate routes to score and pick the best one [11]. Following [7,20], we use the speaker to estimate the candidate routes and pick the best one as the final result. As shown in Table 2, our performance is on par with or better than previous methods.

4.3 Diagnostic Experiments

Effectiveness of Our Basic Idea. We first examine the performance of the naïve model (§3.1). As shown in Table 3, even with a simple exploration ability, the agent gains significant improvements over SR, NE and OR. It is no surprise to see drops in TL and SPL, as the agent simply explores all directions.

Exploration Decision Making. In §3.2, the agent learns to select some valuable directions to explore. As seen, the improved agent is indeed able to collect useful surrounding information by only conducting necessary exploration, as TL and SPL are improved without sacrificing improvements in SR, NE and OR.

Allowing Multi-Step Exploration. In §3.3, instead of only allowing one-step exploration, the agent learns to conduct multi-step exploration. To investigate the efficacy of such a strategy individually, we allow our naïve model to make at most 4-step exploration (*w/o.* exploration decision making). In Table 3, we can observe further improvements over SR, NE and OR scores, with larger TL.

Importance of All Components. Next we study the efficacy of our full model from §3.3, which is able to make multi-direction, multi-step exploration. We find that, by integrating all the components together, our agent with at most 4-step exploration achieves the best performance in most metrics.

Influence of Maximum Allowable Exploration Step. From Table 3, we find that, with more maximum allowable exploration steps (1→4), the agent attains better performance. However, allowing further exploration steps (4→6) will hurt the performance. For at most 4-step exploration, the average exploration rate is 15.3%. During exploration, the percentage of wrong navigation actions being

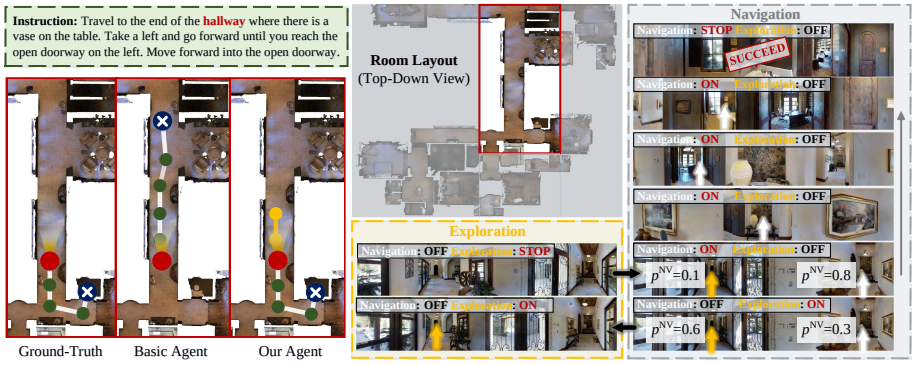

Fig. 5. Left: The basic agent is confused by the ambiguous instruction "*Travel to the end of the hallway*$\cdots$", causing failed navigation. Our agent can actively collect information (the yellow part) and then make a better navigation decision. **Middle Bottom**: First view during exploration. **Right:** First view during navigation. We can find that, before exploration, the wrong direction gains a high navigation probability (*i.e.*, 0.6). However, after exploration, the score for the correct direction is improved.

corrected is ∼65.2%, while right navigation action being changed wrongly is ∼10.7%. The percentages of maximum exploration steps, from 1 to 4, are 53.6%, 12.5%, 8.7%, and 25.3%, respectively. We find that, in most cases, one-step exploration is enough. Sometimes the agent may choose long exploration, which maybe because he needs to collect more information for hard examples.

Qualitative Results. Fig. 5 depicts a challenge example, with the ambiguous instruction "*Travel to the end of the hallway*$\cdots$". The basic agent chooses the wrong direction and ultimately fails. However, our agent is able to actively explore the environment and collect useful information, to support the navigation-decision making. We observe that, after exploration, the correct direction gains a significant score and our agent reaches the goal location successfully.

5 Conclusion

This work proposes an end-to-end trainable agent for the VLN task, with an active exploration ability. The agent is able to intelligently interact with the environment and actively gather information when faced with ambiguous instructions or unconfident navigation decisions. The elaborately designed exploration module successfully learns its own policy with the purpose of supporting better navigation-decision making. Our agent shows promising results on R2R dataset.

Acknowledgements This work was partially supported by Natural Science Foundation of China (NSFC) grant (No.61472038), Zhejiang Lab's Open Fund (No. 2020AA3AB14), Zhejiang Lab's International Talent Fund for Young Professionals, and Key Laboratory of Electronic Information Technology in Satellite Navigation (Beijing Institute of Technology), Ministry of Education, China.

References

1. Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., van den Hengel, A.: Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In: CVPR (2018) 1, 2, 3, 4, 9, 11, 12, 13
2. Andreas, J., Klein, D.: Alignment-based compositional semantics for instruction following. In: EMNLP (2015) 3
3. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., Parikh, D.: VQA: Visual question answering. In: ICCV (2015) 3
4. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: ICML (2009) 10
5. Chen, D.L., Mooney, R.J.: Learning to interpret natural language navigation instructions from observations. In: AAAI (2011) 3
6. Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J.M., Parikh, D., Batra, D.: Visual dialog. In: CVPR (2017) 3
7. Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., Darrell, T.: Speaker-follower models for vision-and-language navigation. In: NeurIPS (2018) 1, 3, 9, 11, 12, 13
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 11
9. Hu, R., Fried, D., Rohrbach, A., Klein, D., Darrell, T., Saenko, K.: Are you looking? grounding to multiple modalities in vision-and-language navigation. In: ACL (2019) 1, 3
10. Huang, H., Jain, V., Mehta, H., Ku, A., Magalhaes, G., Baldridge, J., Ie, E.: Transferable representation learning in vision-and-language navigation. In: ICCV (2019) 1, 3
11. Ke, L., Li, X., Bisk, Y., Holtzman, A., Gan, Z., Liu, J., Gao, J., Choi, Y., Srinivasa, S.: Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In: CVPR (2019) 1, 3, 12
12. Ma, C.Y., Lu, J., Wu, Z., AlRegib, G., Kira, Z., Socher, R., Xiong, C.: Self-monitoring navigation agent via auxiliary progress estimation. In: ICLR (2019) 1, 3, 12, 13
13. Ma, C.Y., Wu, Z., AlRegib, G., Xiong, C., Kira, Z.: The regretful agent: Heuristic-aided navigation through progress estimation. In: CVPR (2019) 1, 3, 9, 12
14. MacMahon, M., Stankiewicz, B., Kuipers, B.: Walk the talk: connecting language, knowledge, and action in route instructions. In: AAAI (2006) 3
15. Mei, H., Bansal, M., Walter, M.R.: Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In: AAAI (2016) 3, 9
16. Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A.J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., et al.: Learning to navigate in complex environments. In: ICLR (2017) 3
17. Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., Artzi, Y.: Mapping instructions to actions in 3d environments with visual goal prediction. In: EMNLP (2018) 3
18. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: ICML (2016) 9
19. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115**(3), 211–252 (2015) 11

20. Tan, H., Yu, L., Bansal, M.: Learning to navigate unseen environments: Back translation with environmental dropout. In: NAACL (2019) 1, 3, 4, 9, 11, 12, 13
21. Tellex, S., Kollar, T., Dickerson, S., Walter, M.R., Banerjee, A.G., Teller, S., Roy, N.: Understanding natural language commands for robotic navigation and mobile manipulation. In: AAAI (2011) 3
22. Thomason, J., Gordon, D., Bisk, Y.: Shifting the baseline: Single modality performance on visual navigation & qa. In: NAACL (2019) 3
23. Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y.F., Wang, W.Y., Zhang, L.: Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In: CVPR (2019) 1, 3, 9, 11, 12, 13
24. Wang, X., Xiong, W., Wang, H., Yang Wang, W.: Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In: ECCV (2018) 1, 3, 9, 11, 12
25. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: ICML (2015) 3
26. Yu, L., Poirson, P., Yang, S., Berg, A.C., Berg, T.L.: Modeling context in referring expressions. In: ECCV (2016) 3
27. Zheng, Z., Wang, W., Qi, S., Zhu, S.C.: Reasoning visual dialogs with structural and partial observations. In: CVPR (2019) 3
28. Zhu, F., Zhu, Y., Chang, X., Liang, X.: Vision-language navigation with self-supervised auxiliary reasoning tasks. In: CVPR (2020) 1, 3, 11, 12, 13
29. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: ICRA (2017) 3, 9