Vision-Language Navigation with Self-Supervised Auxiliary Reasoning Tasks

Fengda Zhu\(^1\)  Yi Zhu\(^2\)  Xiaojun Chang\(^1\)  Xiaodan Liang\(^{3,4}\)
\(^1\)Monash University  \(^2\)University of Chinese Academy of Sciences
\(^3\)Sun Yat-sen University  \(^4\)Dark Matter AI Inc.

zhufengda@yahoo.com  zhu.yee@outlook.com
cxj273@gmail.com  xdliang328@gmail.com

Abstract

Vision-Language Navigation (VLN) is a task where agents learn to navigate following natural language instructions. The key to this task is to perceive both the visual scene and natural language sequentially. Conventional approaches exploit the vision and language features in cross-modal grounding. However, the VLN task remains challenging, since previous works have neglected the rich semantic information contained in the environment (such as implicit navigation graphs or sub-trajectory semantics). In this paper, we introduce Auxiliary Reasoning Navigation (AuxRN), a framework with four self-supervised auxiliary reasoning tasks to take advantage of the additional training signals derived from the semantic information. The auxiliary tasks have four reasoning objectives: explaining the previous actions, estimating the navigation progress, predicting the next orientation, and evaluating the trajectory consistency. As a result, these additional training signals help the agent to acquire knowledge of semantic representations in order to reason about its activity and build a thorough perception of the environment. Our experiments indicate that auxiliary reasoning tasks improve both the performance of the main task and the model generalizability by a large margin. Empirically, we demonstrate that an agent trained with self-supervised auxiliary reasoning tasks substantially outperforms the previous state-of-the-art method, being the best existing approach on the standard benchmark\(^1\).

1. Introduction

Increasing interest rises in Vision-Language Navigation (VLN)\(^5\) tasks, where an agent navigates in 3D indoor environments following a natural language instruction, such as Walk between the columns and make a sharp turn right. Walk down the steps and stop on the landing. The agent begins at a random point and goes toward a goal by means of active exploration. A vision image is given at each step and a global step-by-step instruction is provided at the beginning of the trajectory.

Recent research in feature extraction\(^{14, 4, 24, 31, 46}\), attention\(^{4, 9, 22}\) and multi-modal grounding\(^{6, 21, 36}\) have helped the agent to understand the environment. Previous works in Vision-Language Navigation have focused on improving the ability of perceiving the vision and language inputs\(^{13, 10, 42}\) and cross-modal matching\(^{41}\). With these approaches, an agent is able to perceive the vision-language inputs and encode historical information for navigation.

However, the VLN task remains challenging since rich semantic information contained in the environments is neglected: 1) Past actions affect the actions to be taken in the future. To make a correct action requires the agent to have a thorough understanding of its activity in the past. 2) The agent is not able to explicitly align the trajectory with

\(^1\)VLN leaderboard: https://evalai.cloudcv.org/web/challenges/challenge-page/97/leaderboard/270
the instruction. Thus, it is uncertain whether the vision-language encoding can fully represent the current status of the agent. 3) The agent is not able to accurately assess the progress it has made. Even though Ma et al. [23] proposed a progress monitor to estimate the normalized distance toward the goal, labels in this method are biased and noisy. 4) The action space of the agent is implicitly limited since only neighbour nodes in the navigation graph are reachable. Therefore, if the agent gains knowledge of the navigation map and understands the consequence of its next action, the navigation process will be more accurate and efficient.

We introduce auxiliary reasoning tasks to solve these problems. There are three key advantages to this solution. First of all, auxiliary tasks produce additional training signals, which improves the data efficiency in training and makes the model more robust. Secondly, using reasoning tasks to determine the actions makes the actions easier to explain. It is easier to interpret the policy of an agent if we understand why the agent takes a particular action. An explainable mechanism benefits human understanding of how the agent works. Thirdly, the auxiliary tasks have been proven to help reduce the domain gap between seen and unseen environments. It has been demonstrated [34, 35] that self-supervised auxiliary tasks facilitate domain adaptation. Besides, it has been proven that finetuning the agent in an unseen environment effectively reducing the domain gap [41, 37]. We use auxiliary tasks to align the representations in the unseen domain alongside those in the seen domain during finetuning.

In this paper, we introduce Auxiliary Reasoning Navigation (AuxRN), a framework facilitates navigation learning. AuxRN consists of four auxiliary reasoning tasks: 1) A trajectory retelling task, which makes the agent explain its previous actions via natural language generation; 2) A progress estimation task, to evaluate the percentage of the trajectory that the model has completed; 3) An angle prediction task, to predict the angle by which the agent will turn next. 4) A cross-modal matching task which allows the agent to align the vision and language encoding. Unlike “proxy tasks” [21, 36, 33] which only consider the cross-modal alignment at one time, our tasks handle the temporal context from history in addition to the input of a single step. The knowledge learning of these four tasks are presumably reciprocal. As shown in Fig. 1, the agent learns to reason about the previous actions and predict future information with the help of auxiliary reasoning tasks.

Our experiment demonstrates that AuxRN dramatically improves the navigation performance on both seen and unseen environments. Each of the auxiliary tasks exploits useful reasoning knowledge respectively to indicate how an agent understands an environment. We adopt Success weighted by Path Length (SPL) [3] as the primary metric for evaluating our model. AuxRN pretrained in seen environments with our auxiliary reasoning tasks outperforms our baseline [37] by 3.45% on validation set. Our final model, finetuned on unseen environments with auxiliary reasoning tasks obtains 65%, 4% higher than the previous state-of-the-art result, thereby becoming the first-ranked result in the VLN Challenge in terms of SPL.

2. Related Work

Vision-Language Reasoning Bridging vision and language is attracting attention from both the computer vision and the natural language processing communities. Various associated tasks have been proposed, including Visual Question Answering (VQA) [1], Visual Dialog Answering [38], Vision-Language Navigation (VLN) [5] and Visual Commonsense Reasoning (VCR) [44]. Vision-Language Reasoning [29] plays an important role in solving these problems. Anderson et al. [4] apply an attention mechanism on detection results to reason visual entities. More recent works, such as LXMERT [36], ViLBERT [21], and B2T2 [2] obtain high-level semantics by pretraining a model on a large-scale dataset with vision-language reasoning tasks.

Learning with Auxiliary Tasks Self-supervised auxiliary tasks have been widely applied in the field of machine learning. Moreover, the concept of learning from auxiliary tasks to improve data efficiency and robustness [16, 28, 39, 23] has been extensively investigated in reinforcement learning. Mirowski et al. [25] propose a robot which obtains additional training signals by recovering a depth image with colored image input and predicting whether or not it reaches a new point. Furthermore, self-supervised auxiliary tasks have been widely applied in the fields of computer vision [45, 12, 27], natural language processing [9, 19] and meta learning [40, 20]. Gidaris et al. [11] unsupervisedly learn image features with a 2D rotate auxiliary loss, while Sun et al. [35] indicate that self-supervised auxiliary tasks are effective in reducing domain shift.

Vision Language Navigation A number of simulated 3D environments have been proposed to study navigation, such as Doom [17], AI2-THOR [18] and House3D [43]. However, the lack of photorealism and natural language instruction limits the application of these environments. Anderson et al. [5] propose Room-to-Room (R2R) dataset, the first Vision-Language Navigation (VLN) benchmark based on real imagery [8].

The Vision-Language Navigation task has attracted widespread attention since it is both widely applicable and challenging. Earlier work [42] combined model-free [26] and model-based [30] reinforcement learning to solve VLN. Fried et al. propose a speaker-follow framework for data augmentation and reasoning in supervised learning. In addition, a concept named “panoramic action space” is proposed to facilitate optimization. Later work [41] has found it ben-
of to combine imitation learning [7, 15] and reinforcement learning [26, 32]. The self-monitoring method [23] is proposed to estimate progress made towards the goal. Researchers have identified the existence of the domain gap between training and testing data. Unsupervised pre-exploration [41] and Environmental dropout [37] are proposed to improve the ability of generalization.

3. Method

3.1. Problem Setup

The Vision-and-Language Navigation (VLN) task gives a global natural sentence \( I = \{w_0, ..., w_l\} \) as an instruction, where each \( w_i \) is a token while the \( l \) is the length of the sentence. The instruction consists of step-by-step guidance toward the goal. At step \( t \), the agent observes a panoramic view \( O_t = \{o_{t,i}\}_{i=1}^{36} \) as the vision input. The panoramic view is divided into 36 RGB image views, while each of these views consists of image feature \( v_i \) and an orientation description \((\sin \theta_{t,i}, \cos \theta_{t,i}, \sin \phi_{t,i}, \cos \phi_{t,i})\). For each step, the agent chooses a direction to navigate over all candidates in the panoramic action space [10]. Candidates in the panoramic action space consist of \( k \) neighbours of the current node in the navigation graph and a stop action. Candidates for the current step are defined as \( \{c_{t,1}, ..., c_{t,k+1}\} \), where \( c_{t,k+1} \) stands for the stop action. Note that for each step, the number of neighbours \( k \) is not fixed.

3.2. Vision-Language Forward

We first define the attention module, which is widely applied in our pipeline. Then we illustrate vision embedding and vision-language embedding mechanisms. At last, we demonstrate the approach of action prediction.

Attention Module At first we define the attention module, an important part of our pipeline. Suppose we have a sequence of feature vectors noted as \( \{f_0, ..., f_n\} \) to fuse and a query vector \( q \). We implement an attention layer \( \hat{f} = \text{Attn}(\{f_0, ..., f_n\}, q) \) as:

\[
\alpha_i = \text{softmax}(f_i W_{\text{Attn}} q) \\
\hat{f} = \sum \alpha_i f_i.
\] (1)

\( W_{\text{Attn}} \) represents the fully connected layer of the attention mechanism. \( \alpha_i \) is the weight for the \( i \)th feature for fusing.

Vision Embedding As mentioned above, the panoramic observation \( O_t \) denotes the 36 features consisting of vision and orientation information. We then fuse \( \{o_{t,1}, ..., o_{t,36}\} \) with cross-modal context of the last step \( \hat{f}_{t-1} \) and introduce an LSTM to maintain a vision history context \( \hat{f}_t^o \) for each step:

\[
\hat{f}_t^o = \text{Attn}_o(\{o_{t,1}, ..., o_{t,36}\}, \hat{f}_{t-1}) \\
\hat{f}_t^o = \text{LSTM}_v(\hat{f}_t^o, h_{t-1})
\] (2)

where \( \hat{f}_t^o = h_t \) is the output of the LSTM. Note that unlike the other two LSTM layers in our pipeline (as shown in Fig. 2) which are computed within a step. LSTM is computed over a whole trajectory.

Vision-Language Embedding Similar to [10, 37], we embed each word token \( w_i \) to word feature \( f_i^w \), where \( i \) stands for the index. Then we encode the feature sequence by a Bi-LSTM layer to produce language features and a global language context \( \tilde{f}^w \):

\[
\{\tilde{f}_0^w, ..., \tilde{f}_l^w\} = \text{Bi-LSTM}_w(\{f_0^w, ..., f_l^w\}) \\
\tilde{f}^w = \frac{1}{l} \sum_{i=1}^{l} \tilde{f}_i^w.
\] (3)
The global language context participates $\bar{f}_l^w$ the auxiliary task learning described in Sec. 3.4. Finally, we fuse the language features $\{f_0^w, ..., f_t^w\}$ with the vision history context $\bar{f}_t^v$ to produce the cross-modal context $\hat{f}_t$ for the current step:

$$\hat{f}_t = \text{Attn}_w(\{f_0^w, ..., f_t^w\}, \bar{f}_t^v). \quad (4)$$

**Action Prediction** In the VLN setting, the adjacent navigable node is visible. Thus, we can obtain the reachable candidates $C = \{c_{t,1}, ..., c_{t,k+1}\}$ from the navigation graph. Similar to observation $O$, candidates in $C$ are concatenated features of vision features and orientation descriptions. We obtain the probability function $p_t(a_t)$ for action $a_t$ by:

$$p_t(a_t) = \text{softmax}(\bar{f}_t^v). \quad (5)$$

Three ways for action prediction are applied to different scenarios: 1) imitation learning: following the labeled teacher action $a_t^*$ regardless of $p_t$; 2) reinforcement learning: sample action following the probability distribution $a_t \sim p_t(a_t)$; 3) testing: choose the candidate which has the greatest probability $a_t = \text{argmax}(p_t(a_t))$.

**3.3. Objectives for Navigation**

In this section, we introduce two learning objectives for the main task: imitation learning (IL) and reinforcement learning (RL). The main task is jointly optimized by these two objectives.

**Imitation Learning** forces the agent to mimic the behavior of its teacher. IL has been proven [10] to achieve good performance in VLN tasks. Our agent learns from the teacher action $a_t^*$ for each step:

$$L_{IL} = \sum_t -a_t^* \log(p_t), \quad (6)$$

where $a_t^*$ is a one-hot vector indicating the teacher choice.

**Reinforcement Learning** is introduced for generalization since adopting IL alone could result in overfitting. We implement the A2C algorithm, the parallel version of A3C [26], and our loss function is calculated as:

$$L_{RL} = -\sum_t a_t \log(p_t) A_t. \quad (7)$$

$A_t$ is a scalar representing the advantage defined in A3C.

**3.4. Auxiliary Reasoning Learning**

The vision-language navigation task remains challenging, since the rich semantics contained in the environments are neglected. In this section, we introduce auxiliary reasoning learning to exploit additional training signals from environments.

In Sec. 3.2, we obtain the vision context $\bar{f}_l^v$ from Eq. 2, the global language context $\bar{f}_l^o$ from Eq. 3 and the cross-modal context $\hat{f}_t$ from Eq. 4. In addition to action prediction, we give the contexts to the reasoning modules in Fig. 2 to perform auxiliary tasks. We discuss four auxiliary objectives use the contexts for reasoning below.

**Trajectory Retelling Task** Trajectory reasoning is critical for an agent to decide what to do next. Previous works train a speaker to translate a trajectory to a language instruction. The methods are not end-to-end optimized, which limit the performances.

As shown in Fig. 2, we adopt a teacher forcing method to train an end-to-end speaker. The teacher is defined as $\{f_0^w, ..., f_t^w\}$, the same word embeddings as in Eq. 4. We use LSTM to encode these word embeddings. We then introduce a cycle reconstruction objective named trajectory retelling task:

$$\{\hat{f}_0^w, ..., \hat{f}_t^w\} = \text{LSTM}_w(\{f_0^w, ..., f_t^w\}),$$

$$\hat{f}_t = \text{Attn}_s(\{\hat{f}_0^w, ..., \hat{f}_T^w\}, \bar{f}_t^v),$$

$$L_{Speaker} = -\frac{1}{l} \sum_{i=1}^{l} \log p(w_i|\hat{f}_t^v). \quad (8)$$

Our trajectory retelling objective is jointly optimized with the main task. It helps the agent to obtain better feature representations since the agent come to know the semantic meanings of the actions. Moreover, trajectory retelling makes the activity of the agent explainable.

**Progress Estimation Task** We propose a progress estimation task to learn the navigation progress. Earlier research [23] uses normalized distances as labels and optimizes the prediction module with Mean Square Error (MSE) loss. However, we use the percentage of steps $r_t$, noted as a soft binary label $\{\frac{1}{T}, 1-\frac{1}{T}\}$ to represent the progress:

$$L_{progress} = -\frac{1}{T} \sum_{i=1}^{T} r_i \log \sigma(W_r \hat{f}_t). \quad (9)$$

Here $W_r$ is the weight of the fully connected layer and $\sigma$ is the sigmoid activation layer. Our ablation study reveals that the method that learning from percentage of steps $r_t$ with BCE loss achieves higher performance than previous method. Normalized distance labels introduce noise, which limits performance. Moreover, we also find that Binary Cross Entropy (BCE) loss performs better than MSE loss with our step-percentage label since logits learned from BCE loss are unbiased. The progress estimation task requires the agent to align the current view with corresponding words in the instruction. Thus, it is beneficial to vision language grounding.

**Cross-modal Matching Task** We propose a binary classification task, motivated by LXMERT [36], to predict whether
or not the trajectory matches the instruction. We shuffle \(f^w\) from Eq. 3 with feature vector in the same batch with the probability of 0.5. The shuffled operation is marked as “S” in the white circle in Fig. 2 and the shuffled feature is noted as \(\hat{f}^w\). We concatenate the shuffled feature with the attended vision-language feature \(f_t\). We then supervise the prediction result with \(m_t\), a binary label indicating whether the feature has been shuffled or remains unchanged.

\[
L_{\text{Matching}} = -\frac{1}{T} \sum_{t=1}^{T} m_t \log \sigma(W_m[f_t, \hat{f}^w]),
\]

where \(W_m\) stands for the fully connected layer. This task requires the agent to align historical vision-language features in order to distinguish if the overall trajectory matches the instruction. Therefore, it facilitates the agent to encode historical vision and language features.

**Angle Prediction Task** The agent make the choice among the candidates to decide which step it will take next. Compared with the noisy vision feature, the orientation is much cleaner. Thus we consider learning from orientation information in addition to learning from candidate classification. We thus propose a simple regression task to predict the orientation that the agent will turn to:

\[
L_{\text{angle}} = -\frac{1}{T} \sum_{t=1}^{T} \| e_t - W_e \hat{f}_t \|,
\]

where \(e_t\) is the angle of the teacher action in the imitation learning, while \(W_e\) stands for the fully connected layer. Since this objective requires a teacher angle for supervision, we do not forward this objective in RL.

Above all, we jointly train all the four auxiliary reasoning tasks in an end-to-end manner:

\[
L_{\text{total}} = L_{\text{Speaker}} + L_{\text{Progress}} + L_{\text{Angle}} + L_{\text{Matching}},
\]

4. Experiment

4.1. Setup

**Dataset and Environments** We evaluate the proposed AuxRN method on the Room-to-Room (R2R) dataset [5] based on Matterport3D simulator [8]. The dataset, consisting of 90 different housing environments, is split into a training set, a seen validation set, an unseen validation set, and a test set. The training set consists of 61 environments and 14,025 instructions, while the unseen validation set consists of the remaining 18 environments with 4,173 instructions.

**Evaluation Metrics** A large number of metrics are used to evaluate models in VLN, such as Trajectory Length (TL), the trajectory length in meters, Navigation Error (NE), the navigation error in meters, Oracle Success Rate (OR), the rate if the agent successfully stops at the closest point, Success Rate (SR), the success rate of reaching the goal, and Success rate weighted by (normalized inverse) Path Length (SPL) [3]. In our experiment, we take all of these into consideration and regard SPL as the primary metric.

**Implementation Details** We introduce self-supervised data to augment our dataset. We sample the augmented data from training and testing environments and use the speaker trained in Sec. 3.2 to generate self-supervised instructions. Our training process consists of three steps: 1) we pre-train our model on the training set; 2) we pick the best model (the model with the highest SPL) at step 1 and fine-tune the model on the augmented data sampled from training set [37]; 3) we finetune the best model at step 2 on the augmented data sampled from testing environments for pre-exploration, which is similar to [41, 37]. We pick the last model at step 3 to test. The training iterations for each steps are 80K. We train each model with auxiliary tasks. At steps 2 and 3, since augmented data contains more noise than labeled training data, we reduce the loss weights for all auxiliary tasks by half.
4.2. Test Set Results

In this section, we compare our model with previous state-of-the-art methods. We compare the proposed AuxRN with two baselines and five other methods. A brief description of previous models as followed. 1) Random: randomly take actions for 5 steps. 2) Seq-to-Seq: A sequence to sequence model reported in [5]. 3) Look Before You Leap: a method combining model-free and model-based reinforcement learning. 4) Speaker-Follower: a method introduces a data augmentation approach and panoramic action space. 5) Self-Monitoring: a method regularized by a self-monitoring agent. 6) Reinforced Cross-Modal: a method with cross-modal attention and combining imitation learning with reinforcement learning. 7) Environmental Dropout: a method augment data with environmental dropout. Additionally, we evaluate our models on three different training settings: 1) Single Run: without seeing the unseen environments and 2) Pre-explore: finetuning a model in the unseen environments with self-supervised approach. 3) Beam Search: predicting the trajectories with the highest rate to success.

As shown in Tab. 1, AuxRN outperforms previous models in a large margin on all three settings. In Single Run, we achieve 3% improvement on oracle success, 4% improvement on success rate and 4% improvement on SPL. In Pre-explore setting, our model greatly reduces the error to 3.69, which shows that AuxRN navigates further toward the goal. AuxRN significantly boost oracle success by 5%, success rate 4% and SPL to 4%. AuxRN achieves similar improvements on two domains, which indicates that the auxiliary reasoning tasks we proposed is immune from domain gap.

We also achieve the state-of-the-art in Beam Search setup. Our final model with Beam Search algorithm achieves 71% success rate, which is 2% higher than Environmental Dropout, the previous state-of-the-art. Our beam search result has the least trajectory length, which reveals that AuxRN is time efficient in navigation.

4.3. Ablation Experiment

**Auxiliary Reasoning Tasks Comparison** In this section, we compare performances between different auxiliary reasoning tasks. We evaluate our models on two validation splits: validation for the seen and unseen environments. Four metrics are compared, including NE, OR, SR and SPL.

<table>
<thead>
<tr>
<th>Models</th>
<th>Val Seen</th>
<th>Val Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NE (m)</td>
<td>OR (%)</td>
</tr>
<tr>
<td>baseline</td>
<td>4.51</td>
<td>65.62</td>
</tr>
<tr>
<td>baseline+L_{\text{progress}}</td>
<td>4.13</td>
<td>69.05</td>
</tr>
<tr>
<td>baseline+L_{\text{Angle}}</td>
<td>4.70</td>
<td>65.33</td>
</tr>
<tr>
<td>baseline+L_{\text{Total}}</td>
<td>4.25</td>
<td>70.03</td>
</tr>
<tr>
<td>baseline+BT [5]</td>
<td>4.04</td>
<td>72.28</td>
</tr>
<tr>
<td>baseline+BT+L_{\text{Total}}</td>
<td>3.33</td>
<td>77.77</td>
</tr>
</tbody>
</table>

Table 2. Ablation study for different auxiliary reasoning tasks. We evaluate our models on two validation splits: validation for the seen and unseen environments. Four metrics are compared, including NE, OR, SR and SPL.

<table>
<thead>
<tr>
<th>Models</th>
<th>Val Seen</th>
<th>Val Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OR(%)</td>
<td>SR(%)</td>
</tr>
<tr>
<td>baseline+BT+L_{\text{Total}}</td>
<td>66.90</td>
<td>60.33</td>
</tr>
<tr>
<td>baseline+BT+L_{\text{Total}}</td>
<td>65.62</td>
<td>59.55</td>
</tr>
</tbody>
</table>

Table 3. Ablation study for Trajectory Retelling Task. Four metrics are compared, including OR, SR, SPL and Acc (sentence prediction accuracy).
to attend visual features. 3) Matching Critic: regards opposite number of the speaker loss as a reward to encourage the agent. 4) Student Forcing: a seq-to-seq approach translating visual contexts to word tokens without ground truth sentence input. In addition to OR, SR, and SPL, we add a new metric, named sentence prediction accuracy (Acc). This metric calculates the precision model predict the correct word.

The result of ablation study for Trajectory Retelling Task is shown as Tab. 3. Firstly, teacher forcing outperforms Matching Critic [41] by 1.8% and 4.22% respectively. Teacher forcing performs 7.07% and 6.76% more than Matching Critic in terms of accuracy. Secondly, teacher forcing outperforms student forcing by 1.46% and 2.04% in terms of SPL in two validation sets. The results also indicate that teacher forcing is better in sentence prediction compared with student forcing. Thirdly, in terms of SPL, standard teacher forcing outperforms the teacher forcing with shared context on the unseen validation set by 0.77%. Besides, we notice that the teacher forcing with shared context outperforms standard teacher forcing about 12% in word prediction accuracy (Acc). We infer that the teacher forcing with shared context overfits on the trajectory retelling task.

**Progress Estimation Task** To validate the progress estimation task, we investigate two variants in addition to our standard progress estimator. 1) Progress Monitor: We implement Progress Monitor [23] based on our baseline method. 2) we train our model use Mean Square Error (MSE) rather than BCE Loss with the same step-wise label \( \frac{1}{T} \). We compare these models with four metrics: OR, SR, Error and SPL. The Error is calculated by the mean absolute error between the progress estimation prediction and the label. To give a fair comparison between the error for step-wise label and distance label, we normalized all label distributions and the corresponding predictions.

The result is shown as Tab. 4. Our standard model outperforms other two variants and baseline on most of the metrics. Our Step-wise MSE model performs 2.62% higher on the seen validation set 2.53% higher on the unseen validation set than Progress Monitor [23], indicating that label measured by normalized distances is noisier than label measured by steps. In addition, we find that the Progress Monitor we implement performs even worse than baseline. Since the original method [23] trains the agent with imitation learning rather than the mixed training policy combining imitation learning and reinforcement learning. We infer that the performance drop is due to reinforcement learning. When the agent begins to deviate from the labeled path, the progress label become even more noisy, making the performance even worse than baseline.

We compare different loss functions with step-wise labels. Our model with BCE loss is 6.34% higher on the seen validation set and 1.58% higher on the unseen validation set. Furthermore, the prediction error of the model trained by MSE loss is higher than which trained by BCE loss. The Error of the Step-wise+MSE model is 0.14 higher on the seen validation set and 0.16 higher on the unseen validation set than Step-wise+BCE model. It shows that step-wise label and BCE loss are suitable for progress estimation task.

### 4.4. Visualization

**Regularized Language Attention** We visualize the attention map for Attn\(_w\) after Bi-LSTM\(_w\). The dark region in the map stands for where the language features receive high attention. We observe from Fig. 4 that the attention regions on both two maps go left when the navigation step increasing (marked as 1). It means that both models learns to pay an increasing attention to the latter words. At the last few steps, models pay high attention to the first feature and the last feature (marked as 2 and 3), since the Bi-LSTM encodes sentence information at the first and the last feature.

At early steps, our model pays a more focused attention to a specific region which makes the left bottom region (marked as 1) in the left image looks darker than the right one. In addition, it indicates that our model has a higher success rate than the baseline, since the attention map tends to be an uniform distribution when the agent gets lost. We infer from our experiments that auxiliary reasoning losses help regularize the language attention map, which turns out to be beneficial.

**Navigation Visualization** We visualize two sample trajec-
Figure 4. Visualization process of two trajectories in testing. Two complex language instructions are shown in top boxes. Each image is a panoramic view, which is the vision input for AuxRN, where the direction the agent is going to navigate is represented as a red arrow. For each step, the results progress estimator and the matching function are shown as left.

5. Conclusion

In this paper, we presented a novel framework, auxiliary Reasoning Navigation (AuxRN). Our experiments confirm that AuxRN improves the performance of the VLN task quantitatively and qualitatively. We plan to build a general framework for auxiliary reasoning tasks to exploit the common sense information in the future.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No.U19A2073 and in part by the National Natural Science Foundation of China (NSFC) under Grant No.61976233, and by the Air Force Research Laboratory and DARPA under agreement number FA8750-19-2-0501, Australian Research Council Discovery Early Career Researcher Award (DE190100626).
References


