Vision Language Navigation with Multi-granularity Observation and Auxiliary Reasoning Tasks

Fengda Zhu, Yi Zhu, Yanxin Long, Xiaojun Chang, and Xiaodan Liang

Abstract—Vision Language Navigation (VLN) is a task where agents learn to navigate by following natural language instructions. The key to this task involves perceiving 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 Multi-granularity Auxiliary Reasoning Navigation (MG-AuxRN), a navigation framework which employs four auxiliary reasoning tasks to reason over global image features and detected object features. The auxiliary tasks have four reasoning objectives: explaining the previous actions, estimating the navigation progress, predicting the next observation, and evaluating the trajectory consistency. These auxiliary tasks take the advantage of the additional training signals derived from the various semantic information. As a result, these additional training signals help the agent to acquire knowledge of semantic representations in order to reason about the navigation policy and build a thorough perception of the environment. Our experiments indicate that the use of auxiliary reasoning tasks improves both the performance of the main task and the model generalizability by a large margin. We further empirically demonstrate that an agent trained with self-supervised auxiliary reasoning tasks substantially outperforms the previous state-of-the-art methods.

Index Terms—Vision Language Navigation, Vision Language Reasoning, Self-supervised Learning, Auxiliary Task.

1 Introduction

Increasing interest rises in Vision Language Navigation (VLN) [1] tasks, in which an agent navigates in 3D indoor environments by 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 travels towards 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 [2], [3], [4], attention [3] and multi-modal grounding [5] have advanced the navigation agent in a better understanding the environment. Previous works in the field of Vision Language Navigation (VLN) have focused on improving the ability of perceiving the vision and language inputs [6] and cross-modal matching [7], [8]. These approaches have been widely applied in vision language navigation. However, the VLN task remains challenging, since existing approaches neglect the rich semantic information contained in the environments: 1) Past actions affect the actions to be taken in the future. Making a correct action requires the agent to have a thorough understanding of its past activity. 2) The agent is not able to explicitly align the trajectory with the instruction. Thus, it is unclear whether the vision-language encoding can fully represent the current status of the agent. 3) The agent is unable to accurately assess the progress it has already made. Even though Ma *et al.* [9] proposed a progress monitor to estimate the normalized distance toward the goal, the progress labels used in this method are biased and noisy. 4) The simulator limits the action space. Only neighbour nodes in the navigation graph are reachable. Accordingly, if the agent gains knowledge of the navigation map and understands the consequence of its next action, the navigation process will become more accurate and efficient.

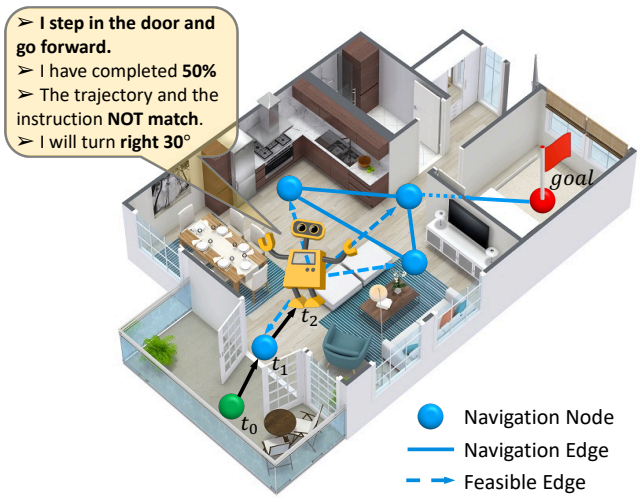


Fig. 1. A simple demonstration of an agent learning to navigate with auxiliary reasoning tasks. The green circle is the start position and the red circle is the goal. Four nodes are reachable by the agent in the navigation graph. Auxiliary reasoning tasks (in the yellow box) help the agent to infer its current status.

In this paper, we introduce auxiliary reasoning tasks to solve these problems. There are three key advantages to this solution.

- *Fengda Zhu and Xiaojun Chang are with Faculty of Information Technology, Monash University.*
  E-mail: {fengda.zhu, xiaojun.chang}@monash.edu
- *Yi Zhu is from University of Chinese Academy of Science (UCAS).*
  E-mail: zhu.yee@outlook.com
- *Yanxin Long and Xiaodan Liang are from Sun Yat-sen University.*
  E-mail: {yestinl1129, xdliang328}@gmail.com

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 these navigation policy explainable. 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 [10], [11] that self-supervised auxiliary tasks facilitate domain adaptation. Moreover, it has also been proven that finetuning the agent in an unseen environment effectively reduces the domain gap [7], [12]. We use auxiliary tasks to align the representations in the unseen domain with those in the seen domain during finetuning.

Specifically, we propose Multi-granularity Auxiliary Reasoning Navigation (MG-AuxRN) to facilitate navigation learning. MG-AuxRN perceives multi-granularity input which combining dense object features and global image features. MG-AuxRN comprises 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 **temporal difference task**, to predict its next observation; 4) a **cross-modal matching task** that allows the agent to align the vision and language encoding; Unlike "proxy tasks" [5], [13], which only consider the cross-modal alignment at a time, our tasks handle the temporal context from history in addition to the input of a single step. As shown in Fig. 1, the agent learns to reason about its previous actions and predict future information with the help of auxiliary reasoning tasks.

We would address several novelties in this paper compared with our previous work AuxRN [8]: Firstly, we find that the data to train auxiliary tasks is often noisy; it is because the sampled trajectories in a batch have unequal length, which is different from image batch input with uniform width and height. Thus, to efficiently generate high quality data is critical for auxiliary task learning. Different from AuxRN, which neglects the negative impact of the trajectory noise, we propose a $O(n)$ algorithm to effectively generate training data and mask out the data noise for auxiliary tasks. Moreover, we find the performance of the auxiliary tasks in our previous work is limited by the capability of the navigation backbone. We consider two key points to improve the navigation backbone. Firstly, we enrich vision input. Different from most of VLN works, which use global visual features only, we incorporate dense visual feature to give the agent more detailed information about the current state. Secondly, we find that the attention mechanism substantially affects the navigation performance. We compare different attention variants and find that multi-head attention is the best for navigation in training from scratch setting. Compared with soft dot-product attention [14], the multi-head attention has greater representation ability.

Our experiment demonstrates that MG-AuxRN dramatically improves the navigation performance in both seen and unseen environments. We adopt Success weighted by Path Length (SPL) [15] as the primary metric for evaluating our model. Firstly, we show that our final model obtains a score of 65%, 4% higher than the previous state-of-the-art result. Secondly, in an ablation study, we quantitatively demonstrate that each auxiliary task exploits useful reasoning knowledge to indicate how an agent understands an environment. We investigate how each task boosts navigation learning by comparing our approach with other variants. Moreover, we design several types of visualizing experiment to qualitatively validate our method. We use ablations to demonstrate that a proper attention mechanism is critical in vision language navigation. To provide an intuitive perspective on the performance of our model, we visualize the testing trajectories along with the actions and outputs of the auxiliary tasks. In addition to evaluating navigation trajectories from a panoramic view, we compare the testing trajectory between baseline and MG-AuxRN using top-down views to obtain a better understanding of the improvement we achieve. Finally, we visualize the dense inputs and the distribution of attention values to demonstrate how dense features and multi-head attention boosts navigation.

2 Related Work

In this section, we briefly review related works on the fields relevant to our study: vision-language reasoning, self-supervised learning, and vision-language navigation.

Vision-Language Reasoning Bridging vision and language has attracted attention from both the computer vision and the natural language processing communities. Various related tasks have been proposed to investigate the problem of integrating the vision and language modalities. Image caption generation is an important field of vision language research for which a large number of datasets [16], [17], [18] have been created. Similar to image description generation, several datasets [19], [20], [21] were created to address the video description generation task. Agrawal *et al.* propose the visual question answering (VQA) task [22], where an agent tries to answer a 'common sense' question using visual elements of an image. Das *et al.* propose a task named visual dialog answering [23] to investigate how an agent tries to find the correct answer by observing an image and a dialog history. Anderson *et al.* propose a well-posed problem named vision-language navigation (VLN) [1], where an agent is required to reason over vision language information and predict action in unstructured, previously unseen environments. Other vision language reasoning tasks have recently been proposed, such as CLEVR [24], an image reasoning task, CLEVRER [25], a video reasoning task, and VCR [26], a common-sense reasoning task. Several key components have been extensively studied in an attempt to solve the vision-language reasoning problem. A number of work have contribute to jointly embedding both vision and text modalities. Donahue *et al.* develop a novel recurrent convolutional architecture to encode variable-length inputs such as video frames or language text [27]. Karpathy *et al.* build a model to embed fragments of images and sentences [28]. Mao *et al.* present a multimodal Recurrent Neural Network (m-RNN) to generate image captions which consists of two sub-networks for embedding image and text respectively [29]. Embedding image and text as global features introduces noise and bias to the prediction stage. A more fine-grained approach to tackle this problem involve localizing entities features in image or text and then using an attention mechanism to assign different importance (to features from different regions. The attention Model (AM), which was first introduced for Machine Translation by Bahdanau *et al.* [30], has achieved great success in various vision language tasks. Many research works [31], [32], [33] simply apply attention layers to the output of a CNN network in order to weight each output of each spatial location of the visual features. Anderson *et al.* [3] propose a 'bottom-up' attention mechanism applied to visual entities to extract semantic vision features. Attention over vision and

language entities has achieved significant success in several cross-modal alignment works [34], [35]. Self-attention is an attention mechanism that weights entities from different positions in a single sequence to compute a representation [36], [37]. Transformer, a multi-head self-attention mechanism proposed by Vaswani *et al.* achieved state-of-the-art results in sequence modeling [38]. In our work, we embed vision information using a ResNet-101 model [2] pretrained on ImageNet [39] and embed language information using a word embedding [40] layer and an LSTM layer [41]. In our paper, we find that each attention has unique function in vision language navigation task and we use ablation studies to validate the proper attention for each part of our navigation backbone.

Self-supervised learning In contrast to supervised learning, where a model is trained using human labels, self-supervised learning optimizes a model using unlabeled data. A popular method of facilitating learning using unlabeled data is to propose various auxiliary tasks for networks to solve. The networks obtain training signals by learning the objective functions of these auxiliary tasks, and the features are learned through this process [42]. Previous computer vision works have proposed several auxiliary tasks, such as colorizing grayscale images [43], image inpainting [44] and image jigsaw puzzle [45], etc. Recently, Devlin *et al.* proposed the Bert model [46], which uses several proxy tasks to pretrain a bidirectional transformer network [38]. Bert-like models, such as ViLBERT [5], VL-BERT [47], LXMERT [13] and UNITER [48], can be easily generalized to easily generalize to several vision language tasks, such as VQA v2.0 [49], GQA [50], NLVR2 [51] and VCR [26]. Moreover, the concept of learning from auxiliary tasks to improve data efficiency and robustness [52], [53] has been extensively investigated in the reinforcement learning context. Mirowski *et al.* [54] proposed a robot that obtains additional training signals by recovering a depth image with a colored image input and predicting whether or not it will reach a new point. Silver *et al.* [55] design a self-play framework to train a Go agent from scratch without human knowledge input. Furthermore, self-supervised learning has also been widely applied in more general forms such as meta learning [56]. Veeriah *et al.* [57] introduce a self-supervised value function to discover questions that are formulated as general value functions. Gidaris *et al.* [58] devised Few-Shot Visual Learning with Self-Supervision. Lee *et al.* [59] propose a new denoising approach by adapting their network parameters to the given input through self-supervision without altering the networks architectures. Gidaris *et al.* [11] learn image features with a 2D rotating auxiliary loss, and Sun *et al.* [10] determined that self-supervised auxiliary tasks are effective in reducing domain shift. In our work, we present several auxiliary tasks to exploit information in the navigation environment from several different aspects.

Embodied Navigation It is challenging to teach an agent to navigate in a 3D embodied environment. Sampling training data from a simulated rather than a real-world environment is time-efficient. A number of simulated 3D environments have been proposed to facilitate the study of 3D embodied navigation, such as Doom [60], AI2-THOR [61] and House3D [62]. A number of such works focus on building an agent that can navigate in a simulated environment both efficiently and accurately. Gupta *et al.* [6] propose combining deep learning approach with a SLAM-based approach [63], [64]. Zhu *et al.* [65] uses a neural network to forward vision information from different modalities. Wijmans *et al.* [66] present the Decentralized Distributed Proximal Policy Optimization (DD-PPO) method to train a navigation agent efficiently and achieve state-of-the-art results on the pointGoal task. However, the lack of photorealism and natural language instruction limits the application of these environments. Anderson *et al.* [1] propose the Room-to-Room (R2R) dataset, the first Vision-Language Navigation (VLN) benchmark based on real imagery [67]. The Vision-Language Navigation task has attracted widespread attention, since it is both widely applicable and challenging. Earlier work [68] combined model-free [69] and model-based [70] reinforcement learning to solve VLN. Fried *et al.* propose a speaker-follower framework for data augmentation and reasoning in the supervised learning context. In addition, a concept named "panoramic action space" was further proposed to facilitate optimization. Later work [7] has found it beneficial to combine imitation learning [71], [72] and reinforcement learning [69], [73]. The self-monitoring method [9] was 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 [7] and environmental dropout [12] has also been proposed to improve the generalization ability. Our work proposes that self-supervised vision-language regularization benefits vision-language navigation, and our model achieves outperforms other methods on standard vision language navigation (VLN) benchmark.

3 METHOD

3.1 Problem Setup

The vision-language navigation (VLN) task is given a global natural sentence $I = \{w_0, ..., w_l\}$ as an instruction. Each w_i is a token, while 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. This panoramic view is divided into 36 RGB image views, each of which consists of an 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 [74]. Candidates in the panoramic action space consist of the k neighbors 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 $c_{t,k+1}$ denotes the stop action. Note that for each step, the number of neighbors k is not fixed.

3.2 Vision-language Navigation

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

3.2.1 *Attention Module*

Attention mechanisms have become an integral part of compelling sequence modeling in various tasks. We here introduce the three variants of attention module, soft-dot attention, multi-head attention and gated attention used in our pipeline.

Soft-dot Attention We first define the soft-dot attention module, a lightweight attention module that is accordingly not prone to overfitting in navigation tasks. Suppose we have a sequence of feature vectors denoted as $\{f_0, ..., f_n\}$ to fuse and a query vector

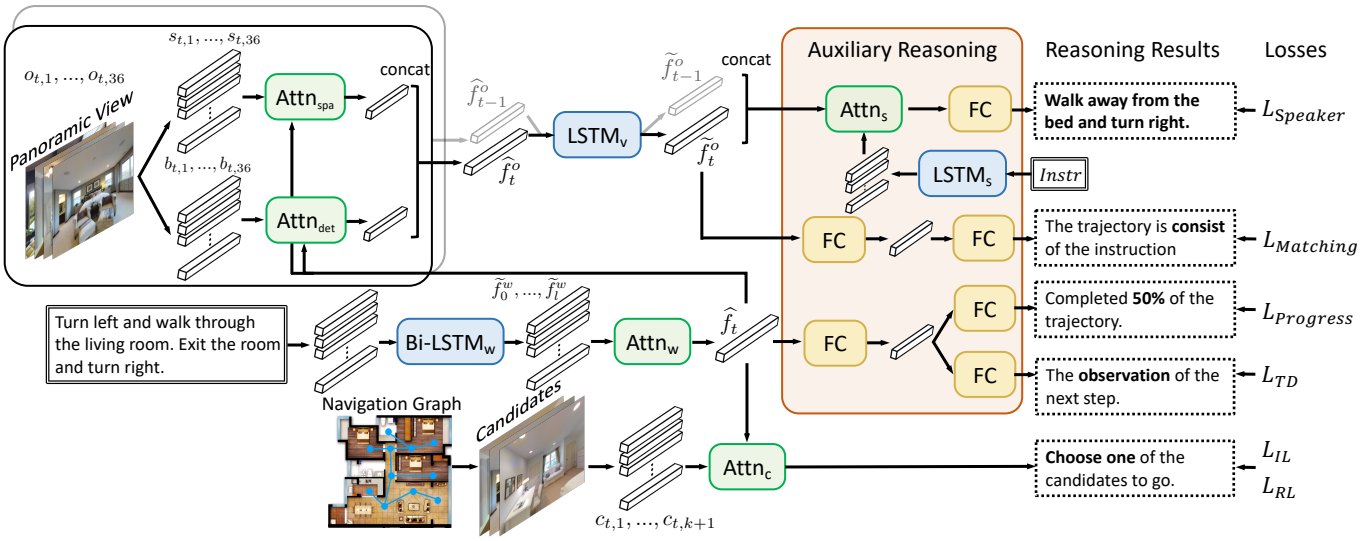


Fig. 2. An overview of MG-AuxRN. The agent embeds vision and language features respectively and performs co-attention between them. The embedded features are passed to reasoning modules and supervised by auxiliary losses. The feature produced by vision-language attention is fused with the candidate features to predict an action. The "P", "S", and "C" in the white circles stand for the mean pooling, random shuffle and concatenate operations respectively.

q. We implement an attention layer $\widehat{f} = \text{Attn}(\{f_0, ..., f_n\}, q)$ as follows:

$$\begin{aligned} \alpha_i &= \text{softmax}(f_i W_S \cdot q) \\ \widehat{f} &= \sum \alpha_i f_i. \end{aligned} \tag{1}$$

The W_S represents the fully connected layer of the soft-dot attention. α_i is the weight for the ith feature for fusing. However, one disadvantage of the soft-dot attention is that the distribution of the attention values is single-peak. Thus it is hard to learn from complex scenes where lots of views have to focus.

Multi-head Attention The multi-head attention module is used to tackle this problem. We calculate attention value from different attention head mean the attention score as follows:

$$\begin{aligned} \alpha_{i,j} &= \text{softmax}(f_i W_{H,j} \cdot q) \\ \widehat{f}_j &= \sum_j \alpha_{i,j} f_i. \\ \widehat{f} &= Concat([\widehat{f}_1, ..., \widehat{f}_{N_H}]). \end{aligned} \tag{2}$$

The W_H is the fully connected layer of the multi-head attention j is the index of the attention head. We concatenate the features from different attention heads as the output. At last we introduce the gated attention module.

Gated Attention One disadvantage of using the above attention modules is that not all features are useful in all situations. In attending detection features, for example, the detection results such as walls, windows or ceilings provide little help with navigation and sometimes introduce noise. Therefore, we have to set the attention weight of these features as 0 to get rid of this difficulty. Otherwise, the useless features can introduce noise and harm the navigation performance. Our gated attention module is formed as:

$$\begin{aligned} \alpha'_i &= \sigma(f_i W_G q) \\ \alpha_i &= \begin{cases} \alpha'_i & \text{if } \alpha'_i \leq \tau \\ 0 & \text{if } \alpha'_i \leq \tau \end{cases} \\ \widehat{f} &= \sum \alpha_i f_i. \end{aligned} \tag{3}$$

The W_G is the weight of the gated attention. The σ indicates the sigmoid activation layer and τ is the threshold to set the features with minor weights as 0.

3.2.2 Multi-granularity Vision Language Forward

In this section, we introduce how the language features and the multi-granularity vision features are attended with eath other and how the navigation action is predicted.

Vision Embedding As mentioned above, the panoramic observation O_t denotes the 36 features consisting of vision and orientation information. We extract global image feature using ResNet-101 [2] similar to previous works [7], [74]. The corresponding global image features are denoted as $S_t = \{s_{t,i}\}_{i=1}^{n}$, where n is the number of images in a panoramic view. In addition to most of the VLN works which only uses global image features as the input, we incorporate $B_t = \{b_{t,i}\}_{i=1}^{m}$ as object features to give a detailed information about the environment, where m is the number of detected objects. We then fuse S_t and B_t with cross-modal context of the last step $\widehat{f}_{t-1}$ and introduce an LSTM to maintain a vision history context $\widetilde{f}_t^o$ for each step:

$$\begin{aligned} \widehat{f}_t^o &= \text{Attn}_o(\{o_{t,1}, ..., o_{t,36}\}, \widetilde{f}_{t-1}) \\ \widehat{f}_t^b &= \text{Attn}_b(\{b_{t,1}, ..., b_{t,n}\}, \widetilde{f}_{t-1}) \\ \widetilde{f}_t^o &= \text{LSTM}_v(\widehat{f}_t^o, h_{t-1}), \end{aligned} \tag{4}$$

where $\widetilde{f}_t^o = h_t$ is the output of the LSTM_v. Note that unlike the other two LSTM layers in our pipeline (as shown in Fig. 2) which are computed within a step. LSTM_v is computed over a whole trajectory. Similarly, we uses Attn_b to attend dense features $\{b_{t,1}, ..., b_{t,n}\}$. The attended global image feature and dense feature are concatenated and encoded by an LSTM layer to generate the trajectory vision encoding.

Vision-Language Embedding Similar to [12], [74], we embed each word token w_i to word feature f_i^w, where i stands for the index. We then encode the feature sequence by a Bi-LSTM layer

to produce language features and a global language context $\overline{f}^w$:

$$\begin{aligned}\{\widetilde{f}_0^w, ..., \widetilde{f}_l^w\} &= \text{Bi-LSTM}_w(\{f_0^w, ..., f_l^w\}) \\ \overline{f}^w &= \frac{1}{l}\sum_{i=1}^{l} \widetilde{f}_i^w.\end{aligned} \tag{5}$$

The global language context participates $\overline{f}^w$ the auxiliary task learning descripted in Sec. 3.3. Finally, we fuse the language features $\{\widetilde{f}_0^w, ..., \widetilde{f}_l^w\}$ with the vision history context $\widetilde{f}_t^o$ to produce the cross-modal context $\widehat{f}_t$:

$$\widehat{f}_t = \text{Attn}_w(\{\widetilde{f}_0^w, ..., \widetilde{f}_l^w\}, \widetilde{f}_t^o). \tag{6}$$

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:

$$\begin{aligned}\widehat{f}_t^c &= \text{Attn}_c(\{c_{t,1}, ..., c_{t,k+1}\}, \widehat{f}_t) \\ p_t(a_t) &= \text{softmax}(\widehat{f}_t^c).\end{aligned} \tag{7}$$

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.

3.2.3 Objectives for Navigation

In this section, we introduce two learning objectives for our navigation task: imitation learning (IL) and reinforcement learning (RL). We use the joint optimization policy to train our model.

Imitation Learning forces the agent to mimic the behavior of its teacher. The teacher actions are the trajectory actions which navigates from the source to the target by the shortest distance. Imitation learning has been proven [74] to achieve good performance in VLN tasks. Our agent learns from the teacher action a_t^* for each step:

$$L_{IL} = \sum_t -a_t^* \log(p_t), \tag{8}$$

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

Reinforcement Learning is introduced for generalization since adopting imitation learning alone could result in overfitting. The imitation learning learns to navigate from the source to the target in the shortest distance following an instruction. However, the sub-optimal trajectories are punished even though they also achieve good performance. In reinforcement learning, we choose the distance towards the target as the reward; therefore, the sub-optimal trajectories are encouraged accompany with the optimal trajectory. We implement the A2C algorithm, the parallel version of A3C [69], and our loss function is calculated as follows:

$$L_{RL} = -\sum_t a_t \log(p_t) A_t. \tag{9}$$

Here, A_t is a scalar representing the advantage defined in A3C.

Joint Optimization Firstly, the model samples the trajectory using the teacher forcing approach and calculates gradients with imitation learning. Secondly, the model samples the trajectory under the same instruction using the student forcing approach and calculates gradients with reinforcement learning. Finally, we add the gradients together and use the added gradients to update the model. Unlike AuxRN [8], we mask out the position the position where the agent comes from to achieve a better performance. It is based on a prior understanding that the best trajectory solution in VLN does not have a loop.

3.3 Auxiliary Reasoning Learning

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

In Sec. 3.2, we obtain the vision context $\widetilde{f}_t^o$ from Eq. 4, the global language context $\overline{f}^w$ from Eq. 5 and the cross-modal context $\widehat{f}_t$ from Eq. 6. In addition to action prediction, we give the contexts to the reasoning modules in Fig. 2 in order to perform auxiliary tasks. We discuss the four auxiliary objectives and 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 $\{f_0^w, ..., f_l^w\}$, the same word embeddings as in Eq. 6. We use LSTM_s to encode these word embeddings. We then introduce a cycle reconstruction objective named trajectory retelling task:

$$\begin{aligned}\{\widetilde{f}_0^w, ..., \widetilde{f}_l^w\} &= \text{LSTM}_s(\{f_0^w, ..., f_l^w\}), \\ \widehat{f}_i^s &= \text{Attn}_s(\{\widetilde{f}_0^o, ..., \widetilde{f}_T^o\}, \widetilde{f}_i^w), \\ L_{Speaker} &= -\frac{1}{l}\sum_{i=1}^{l} \log p(w_i|\widehat{f}_i^s).\end{aligned} \tag{10}$$

Our trajectory retelling objective is jointly optimized with the main task. Note that the trajectory retelling part uses the same architecture in the speaker model. It helps the agent to obtain better feature representations since the agent comes to know the semantic meanings of the actions. Moreover, trajectory retelling makes the activity of the agent explainable. Since the model could deviate a lot in student forcing, we does not train the trajectory retelling task in RL scenarios.

Progress Estimation Task We propose a progress estimation task to learn the navigation progress. Earlier research [9] uses normalized distances as labels and optimizes the prediction module with Mean Square Error (MSE) loss. In our work, however, we use the percentage of steps r_t, denoted as a soft label $\{\frac{t}{T}, 1-\frac{t}{T}\}$ to represent the progress as follows:

$$L_{progress} = -\frac{1}{T}\sum_{t=1}^{T} r_t \log \sigma(W_r \widehat{f}_t). \tag{11}$$

Here, W_r is the weight of the fully connected layer and σ is the sigmoid activation layer. Our ablation study reveals that the method that involves learning from a percentage of steps r_t with BCE loss achieves higher performance than previous methods. 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 labels, since the 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.

Algorithm 1: Cross-modal Matching Task with Mask

input : f_θ: the matching prediction branch
input : $\widetilde{f}_0^o, ..., \widetilde{f}_T^o$: trajectory encoding vector in batch
input : $m_0, ..., m_T$: mask vector in batch
$h = \widetilde{f}_0^o$
for $i = 1, \ldots, T$ **do**
  $h = (1 - m) \cdot \widetilde{f}_i^o + m \cdot h$ // Get the last trajectory encoding before the agent has stopped
$O = \{1, 2, ..., B\} \leftarrow \text{Order}(B)$ // Generate the ordered index list
$P = \{p_1, ..., p_B\} \leftarrow \text{Permute}(B)$ // Randomly permute the index list
$L = \{l_1, ..., l_B\} \leftarrow \text{Random}(B)$ // Matching task label where 1 is matching and 0 is not matching
for $i = 1, \ldots, B$ **do**
  if $p_i = o_i$ **then**
    $l_i \leftarrow 1$ // some indexes are the same after permutation
$L_{Matching} = -\sum_{i=1}^{B} l_i \log(f_\theta(h_{p_i}))$
output: $L_{Matching}$

Cross-modal Matching Task We propose a binary classification task, motivated by LXMERT [13], designed to predict whether or not the trajectory matches the instruction. We shuffle $\overline{f}^w$ from Eq. 5 with a feature vector in the same batch with probability of 0.5. This shuffled operation is marked as "S" in the white circle in Fig. 2 and the shuffled feature is denoted as $\overline{f'}^w$. We concatenate the shuffled feature with the attended vision-language feature $\widehat{f}_t$. We then supervise the prediction result with m_t, a binary label that indicates whether the feature has been shuffled or remains unchanged.

$$L_{Matching} = -\frac{1}{T}\sum_{t=1}^{T} m_t \log \sigma(W_m[\widehat{f}_t, \overline{f'}^w]), \quad (12)$$

where W_m represents the fully connected layer. This task requires the agent to align the temporal vision-language features in order to determine whether the overall trajectory matches the instruction.

After we have forwarded an agent batch from different starting points with different instructions, there are three kinds of trajectories: 1) the unfinished trajectory, which has not output a 'stop' action; 2) the trajectory just completed, which has just output a 'stop' action; 3) the trajectory completed before, which has output a 'stop' action a few steps ago. However, in implementation, since we use temporally forward batches to update our LSTM state, the vision encoding representing a trajectory is updated each time. Even though the trajectory is completed, its representation is updated by the current view image. On the other hand, we find that a trajectory that is incomplete does not align well with the corresponding instruction since some semantics are missing. Previous work that neglects the above two situations could introduce noise into the auxiliary task. Therefore, we propose an O(n) algorithm to generate data and mask out the noise, as shown in Algo. 1.

Temporal Difference Task It is understood that the navigation is highly dependent on the temporal information, we propose to learn the hidden temporal transition function of the embodied environment using temporal difference tasks. The environmental transition function contains the relationship of the vision texture between different viewpoints and the semantics of the room structure. Accordingly, we suggest two subtasks that exploit that the temporal information between two frames in the VLN environment. The first is to predict the angle direction to the next step. The agent makes the choice among the candidates to decide which step it will take next. Thus we consider learning from the 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:

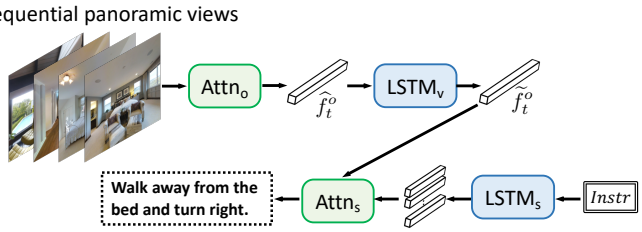


Fig. 3. The pipeline of the speaker model. The speaker sequentially encode the panoramic visual input by means of an attention layer and an LSTM layer. Then it predict the instruction using a vision language attention mechanism.

$$L_{angle} = -\frac{1}{T}\sum_{t=1}^{T} \| e_t - W_a \widehat{f}_t \|, \quad (13)$$

where e_t is the angle of the teacher action in the imitation learning and reinforcement learning, while W_a represents the fully connected layer. Similarly, we have:

$$L_{feature} = -\frac{1}{T}\sum_{t=1}^{T} \| e_t - W_f \widehat{f}_t \|, \quad (14)$$

to help estimate the visual feature of the next step. Finally, we regard the temporal difference task as jointly learning both angle prediction task and the feature prediction task:

$$L_{TD} = L_{angle} + \lambda L_{feature}. \quad (15)$$

End-to-end Learning Finally, we jointly train all the four auxiliary reasoning tasks in an end-to-end manner as follows:

$$\begin{aligned} L_{total} =& \lambda_1 L_{Speaker} + \lambda_2 L_{Progress} \\ &+ \lambda_3 L_{TD} + \lambda_4 L_{Matching}. \end{aligned} \quad (16)$$

3.4 Data Augmentation and Adaptation

Data Bias in Vision Language Navigation Semi-supervised data augmentation has been widely applied in computer vision tasks. Since the data in vision language navigation tasks introduce bias from several aspects: 1) diverse language instructions are able to describe the trajectory, but only three of them are provided; 2) the trajectories are not diverse enough and the navigation model can easily overfit to a certain visual scene; 3) the room structure and visual scene are largely different between the training split and testing split, resulting in a significant performance gap. Thus, we propose to use two methods to boost our model: 1) Back-translation for data augmentation, and 2) Pre-explore adaptation.

Back-Translation Tan *et al.* [12] propose a data augmentation approach known as back-translation. In contrast to the dropout layer which randomly dropouts a percentage of features for each forward, this approach proposes to dropout the same input channels across a trajectory, which maintains the consistency of the environmental information. In generating augmented data, back-translation first samples source and target viewpoint pairs,

TABLE 1
Leaderboard results comparing MG-AuxRN with the previous state-of-the-art on test split in unseen environments. We compare three training settings: Single Run (without seeing unseen environments), Pre-explore (finetuning in unseen environments), and Beam Search (comparing success rate regardless of TL and SPL). The primary metric for Single Run and Pre-explore is SPL, while the primary metric for Beam Search is the success rate (SR). We only report two decimal places due to the precision limit of the leaderboard.

Leader-Board (Test Unseen)	Single Run				Pre-explore				Beam Search		
Models	NE	OR	SR	*SPL*	NE	OR	SR	*SPL*	TL	SR	*SPL*
Random [1]	9.79	0.18	0.17	0.12	-	-	-	-	-	-	-
Seq-to-Seq [1]	20.4	0.27	0.20	0.18	-	-	-	-	-	-	-
Look Before You Leap [68]	7.5	0.32	0.25	0.23	-	-	-	-	-	-	-
Speaker-Follower [74]	6.62	0.44	0.35	0.28	-	-	-	-	1257	0.54	0.01
Self-Monitoring [9]	5.67	0.59	0.48	0.35	-	-	-	-	373	0.61	0.02
The Regretful Agent [75]	5.69	0.48	0.56	0.40	-	-	-	-	13.69	0.48	0.40
FAST [76]	5.14	-	0.54	0.41	-	-	-	-	196.53	0.61	0.03
Reinforced Cross-Modal [7]	6.12	0.50	0.43	0.38	4.21	0.67	0.61	0.59	358	0.63	0.02
ALTR [77]	5.49	-	0.48	0.45	-	-	-	-	-	-	-
Environmental Dropout [12]	5.23	0.59	0.51	0.47	3.97	0.70	0.64	0.61	687	0.69	0.01
MG-AuxRN(Ours)	**5.15**	**0.62**	**0.55**	**0.51**	**3.69**	**0.75**	**0.68**	**0.65**	41	**0.71**	**0.21**

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.

	Val Seen				Val Unseen			
Models	NE (m)	OR (%)	SR (%)	*SPL* (%)	NE (m)	OR (%)	SR (%)	*SPL* (%)
baseline	4.51	65.62	58.57	55.87	5.77	53.47	46.40	42.89
baseline+$L_{Speaker}$	4.13	69.05	60.92	57.71	5.64	57.05	49.34	45.24
baseline+$L_{Progress}$	4.35	68.27	60.43	57.15	5.80	56.75	48.57	44.74
baseline+$L_{Matching}$	4.70	65.33	56.51	53.55	5.74	55.85	47.98	44.10
baseline+L_{TD}	4.25	70.03	60.63	57.68	5.87	55.00	47.94	43.77
baseline+L_{Total}	4.22	72.28	62.88	**58.89**	5.63	59.60	50.62	**45.67**
baseline+BT [12]	4.04	70.13	63.96	61.37	5.39	56.62	50.28	46.84
baseline+BT+L_{Total}	**3.33**	**77.77**	**70.23**	**67.17**	**5.28**	**62.32**	**54.83**	**50.29**

then calculates the shortest distance trajectory using the Dijkstra algorithm. It then uses a speaker [74] model, as shown in Fig. 3, to translate the sequential panoramic views into language instruction.

Pre-exploration Adaptation Since the R2R dataset [1] provides limited house scenes, the navigation model can easily overfit to the training scenes, especially to the low-level visual appearance [78]. To make the trained agent achieve a good performance in testing scenes, previous works [7], [12] have adopted an adaptation stage, named 'pre-exploration', where the agent learns to adapt to the testing environment. The pre-exploration approach randomly samples trajectories from the testing environment. Then it uses a speaker model, as shown in Fig. 3, to translate the trajectories in to instructions. The model is trained by the sampled instruction-trajectory pairs. This method does not require additional human labeling and the testing data is not used in adaptation. We find that auxiliary tasks can regularize the pre-exploration process and help the navigation agent achieve better performance.

4 Experiment

4.1 Setup

Dataset and Environments We evaluate the proposed MG-AuxRN method on the Room-to-Room (R2R) dataset [1] based on the Matterport3D simulator [67], that comprising 90 different housing environments, is split into a training set, a seen validation set, an unseen validation set and a testing set. The training set consists of 61 environments and 14,025 instructions, while the seen validation set has 1,020 instructions using the same environments as the training set. The unseen validation set consists of another 11 environments with 2,349 instructions, while the testing 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: 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) [15], the primary metric in our experiment.

Implementation Details We introduce self-supervised data to augment our dataset. We sample the augmented data from the training and testing environments and use the speaker trained in Sec. 3.2 to generate self-supervised instructions. Our training process comprises three steps: 1) we pretrain our navigation agent and the speaker model on the training set; 2) we pick the best model (the model with the highest SPL on the unseen validation split) at step 1, then finetune this model on the augmented data sampled from the training set [12]; 3) we finetune the best model at step 2 on the augmented data sampled from the testing environments for pre-exploration, using a similar method to [7], [12]. We pick the last model at step 3 for testing. The number of training iterations for each step is 80K. We train each auxiliary task with different loss weights and pick the best weight for each task. We show in our experiments that all losses converge steadily to the same scale, even though we did not carefully tune the weights. At steps

TABLE 3
Ablation study for Trajectory Retelling Task. Four metrics are compared, including OR, SR, SPL and Acc (sentence prediction accuracy).

	Models	OR(%)	SR(%)	Acc(%)	SPL(%)
Val Seen	Baseline	65.62	58.57	-	55.87
	Matching Critic [7]	63.76	55.73	19.58	52.77
	Student Forcing [1]	65.72	57.59	25.37	54.95
	Teacher Forcing(share)	66.90	60.33	**34.85**	**57.23**
	Teacher Forcing(ours)	65.62	59.55	26.34	56.99
Val Unseen	Baseline	53.47	46.40	-	42.89
	Matching Critic	55.26	46.74	18.88	43.44
	Student Forcing	54.92	47.42	25.04	43.78
	Teacher Forcing(share)	56.41	48.19	**38.49**	44.47
	Teacher Forcing	57.05	49.34	25.95	**45.24**

2 and 3, we reduce the loss weights for all auxiliary tasks by half, since augmented data contains more noise than labeled training data. In both training and testing, we mask out the position the position where the agent comes. It is based on prior knowledge that the best solution could not has loop. This trick is able to significantly improve the performance. We have released our code at: https://github.com/ZhuFengdaaa/MG-AuxRN.

4.2 Test Set Results

In this section, we compare our model with previous state-of-the-art methods. We compare the proposed MG-AuxRN with two baselines and five other methods. These previous models can be briefly described as follows: 1) Random: randomly take actions for 5 steps. 2) Seq-to-Seq: A sequence-to-sequence model reported in [1]. 3) Look Before You Leap: a method combining model-free and model-based reinforcement learning. 4) Speaker-Follower: a method that introduces a data augmentation approach and panoramic action space. 5) Self-Monitoring: a method regularized by a self-monitoring agent. 6) The Regretful Agent: a method based on learnable heuristic search. 7) FAST: a search-based method that enables backtracking. 8) Reinforced Cross-Modal: a method with cross-modal attention and that combines imitation learning with reinforcement learning. 9) ALTR: a method focused on adapting vision and language representations. 10) Environmental Dropout: a method that augments data with environmental dropout. Additionally, we evaluate our models on three different training settings: 1) Single Run: navigating in an unseen environment for a single trial; 2) Pre-explore: finetuning a model in the unseen environments without human labeling; and 3) Beam Search: performing heuristic search for multi-routes on an unseen environment and choosing the trajectories with the highest score.

As shown in Tab. 2, MG-AuxRN outperforms previous models by a large margin on all three settings. In Single Run, we achieve a 3% improvement on oracle success, 4% improvement on success rate and 4% improvement on SPL. In the Pre-explore setting, our model greatly reduces the error to 3.69, which shows that MG-AuxRN navigates further toward the goal. MG-AuxRN significantly boosts the oracle success by 5%, success rate 4% and SPL to 4%. In addition, MG-AuxRN achieves similar improvements on the other two domains, indicating that the auxiliary reasoning tasks are immune from the domain gap. Our final model with Beam Search algorithm achieves a 71% success rate, which is 2% higher than Environmental Dropout, the previous state-of-the-art.

TABLE 4
Ablation study for Progress Estimation Task. Four metrics are compared, including OR, SR, SPL and Error (normalized absolute error).

	Models	OR(%)	SR(%)	Error	SPL(%)
Val Seen	Baseline	65.62	58.57	-	55.87
	Progress Monitor [9]	66.01	57.1	0.72	53.43
	Step-wise+MSE(ours)	64.15	53.97	0.27	50.81
	Step-wise+BCE(ours)	**68.27**	**60.43**	**0.13**	**57.15**
Val Unseen	Baseline	53.47	46.40	-	42.89
	Progress Monitor	**57.09**	46.57	0.80	42.21
	Step-wise+MSE(ours)	55.90	46.74	0.32	43.16
	Step-wise+BCE(ours)	56.75	**48.57**	**0.16**	**44.74**

4.3 Ablation Experiment

Auxiliary Reasoning Tasks Comparison In this section, we compare the performances of different auxiliary reasoning tasks. We use the previous state-of-the-art [12] as our baseline, then train the models with each single task based on our baseline. We evaluate our models on both the seen and unseen validation set. The results are presented in Tab. 2. As can bee seen from the table, each task promotes the performance based on our baseline independently. Moreover, training all tasks together is able to further boost the performance, achieving 58.89% on the seen validation set and 45.67% on the unseen validation set. It outperforms the baseline of 3.02% on the seen set and 2.78% on the unseen set. This indicates that the auxiliary reasoning tasks are presumably reciprocal. Moreover, our experiments further reveal that our auxiliary losses and back-translation method have a mutual promotion effect. On the seen validation set, baseline with back-tranlation achieves a 5.50% improvement while combining it with back-translation promotes SPL by 11.30%, which is greater than the sum of the performance improvement of baseline with auxiliary losses and with back-translation independently. Similar results can be observed on the unseen validation set: baseline with back-translation achieves a 3.95% improvement while combining it with back-translation boosts SPL by 7.40%.

Ablation for Trajectory Retelling Task We evaluate four implementation variants for the trajectory retelling task: 1) Teacher Forcing: The standard Trajectory Retelling approach as described in Sec. 3.3. 2) Teacher Forcing (share): teacher forcing that uses $\tilde{f}_w$ to attend visual features. 3) Matching Critic: regards the 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 further add a new metric, named sentence prediction accuracy (Acc). This metric calculates the precision with which the model predicts the correct word.

The result of the ablation study for the Trajectory Retelling Task as presented in Tab. 3. Firstly, teacher forcing method outperforms Matching Critic [7] by 1.8% and 4.22% respectively. Teacher forcing also performs 7.07% and 6.76% better than Matching Critic in terms of accuracy. Secondly, teacher forcing outperforms student forcing by 1.46% and 2.04% in terms of SPL on two validation sets. These results also indicate that teacher forcing is better at 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%. Moreover, we notice that the teacher forcing with shared context outperforms standard teacher forcing by about 12% in

TABLE 5
Ablation study for vision language matching task.

Model	Trajectory	Seen Val		Unseen Val	
		SR(%)	SPL(%)	SR(%)	SPL(%)
baseline	-	**58.57**	**55.87**	46.40	42.89
Mean features	IL only	58.28	55.53	46.23	42.42
Attn features	IL only	56.90	53.84	47.00	43.46
First feature	IL only	54.65	51.32	46.23	42.63
First+Last features	IL only	56.81	54.06	**47.25**	**44.01**
Mean features	IL + RL	55.14	52.07	47.17	43.54
Attn features	IL + RL	56.22	53.35	46.45	42.75
First States	IL + RL	54.46	51.29	47.08	43.49
First+Last features	IL + RL	54.55	51.75	46.62	42.84

TABLE 6
Ablation study for Temporal Different Task.

Model	Seen Val		Unseen Val	
	SR(%)	SPL(%)	SR(%)	SPL(%)
baseline	**58.57**	**55.87**	46.40	42.89
Angle Prediction	55.83	52.96	**47.77**	**44.17**
Feature Prediction	56.71	53.53	47.21	43.56

TABLE 7
Ablation Study for dense visual feature (DV) and dense label feature (DL) with 2D or 3D location information.

Model	Seen Val		Unseen Val	
	SR(%)	SPL(%)	SR(%)	SPL(%)
baseline	58.57	55.87	46.40	42.89
DV	58.28	54.79	47.85	43.57
DV + 2D location	**60.14**	**56.79**	**49.51**	**45.89**
DV + pano orientation	54.65	51.32	47.55	43.32
DL	56.22	53.57	48.32	44.70
DL + 2D location	53.48	50.88	47.77	44.44
DL + pano orientation	57.30	54.41	47.30	43.60

TABLE 8
Ablation study for visual attention for dense features.

Model	Seen Val		Unseen Val	
	SR(%)	SPL(%)	SR(%)	SPL(%)
Softmax (baseline)	57.30	54.12	46.15	42.64
Tanh	50.93	48.30	43.17	40.09
Sigmoid	**58.28**	**54.79**	**47.85**	**43.57**

terms of 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 [9] based on our baseline method. 2) We train our model using Mean Square Error (MSE) rather than BCE Loss with the same step-wise label $\frac{t}{T}$. We compare these models with four metrics: OR, SR, Error and SPL. The Error is calculated by determining the mean absolute error between the progress estimation prediction and the label.

The results are listed in Tab. 4. Our standard model outperforms the other two variants and the baseline on most of the metrics. The step-wise MSE model performs 2.62% higher on the seen validation set and 2.53% higher on the unseen validation set than Progress Monitor [9], indicating that labels measured by normalized distances are noisier than labels measured by steps. In addition, we find that the Progress Monitor performs even worse than the baseline. When the agent begins to deviate from the labeled path, the progress label become even noisier. We compare different loss functions with step-wise labels. Our model with BCE loss scores 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 using MSE loss is higher than that trained using 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.

Vision-language Matching Task In this part, we investigate how the network structures influence the language matching task. There are a lot of structure variants to explore. Our experiments are motivated by two major aspects: 1) determining which sentence encoding is helpful in matching tasks 2) determining whether trajectories sampled from reinforcement learning are helpful. We find that exploiting the language feature significantly effects the navigation performance, which is also by [79]. We research four variants of network structures to exploit the language feature: 1) taking the mean of the language features; 2) using attention module to fuse language features, 3) using the first language feature and 4) concatenating the first feature and the last feature. We also determining via ablation whether using trajectories sampled from reinforcement learning is helpful in navigation learning. The ablation results for the vision-language matching task are shown in Tab. 5. We find that both the network structure and whether the trajectories are sampled from imitation learning (IL) or imitation learning and reinforcement learning (IL+RL) can significantly affect the performance. The motivation of using trajectories sampled from RL is to use sub-optimal trajectories to help with generation. However, incorporating sub-optimal trajectories in learning could introduce data noise. We find that the network structure and the trajectories used in learning has a mutual beneficial effect. There are two models of interest: the model that weights language features by attention and the model that uses the first and last features and benefits from IL-only trajectories The Attn model outperforms baseline by 0.57% and the First+Last model outperforms the baseline by 1.12% and is also the best model. In the other two models, the model using mean language features and the first feature benefits from the additional RL trajectories. The Mean model outperforms the baseline by 0.65%, while the First+Last model outperforms the baseline by 0.60%.

Temporal Difference Task In Tab. 6, we evaluate how different forms of temporal difference tasks help with the navigation training. We implement two kinds of temporal difference tasks, an angle prediction task and a feature prediction task, as demonstrated in Section 3. We find that the models with angle and feature prediction tasks outperforms baseline in unseen validation split. The angle prediction task boosts the success rate by 1.37% and boosts SPL by 1.28% in the unseen split. The feature prediction task improves the success rate by 0.81% and boosts SPL 0.67% the unseen split. However, the models with the angle prediction and the feature prediction task perform worse in the seen validation split. This indicates that both the angle prediction and the feature prediction tasks are able to regularize the features of the model and improve the generalization ability.

4.4 Navigation Backbone Improvement

In this section, we experiment on potential improvement to the navigation backbone. The navigation backbone greatly influences the navigation performance. We improve the navigation backbone from two different aspects: 1) enrich the vision input information, and 2) improve the attention mechanism.

Adding Dense Information Adding object features for visual input could provide richer information that would help with

TABLE 9
Ablation study for Multi-head attention of different parts.

Model	Seen Val		Unseen Val	
	SR(%)	SPL(%)	SR(%)	SPL(%)
baseline	58.57	55.87	46.40	42.89
Attn_{spa}, head=2	**61.12**	**57.28**	48.11	43.63
Attn_{spa}, head=3	59.94	56.79	**49.51**	**45.89**
Attn_{spa}, head=4	57.49	54.55	47.34	43.41
Attn_{w}, head=2	55.14	52.13	46.23	42.37
Attn_{w}, head=3	52.89	49.93	44.66	41.15
Attn_{w}, head=4	51.71	48.74	43.72	40.04
Attn_{c}, head=2	52.50	49.70	44.80	40.90
Attn_{c}, head=3	55.93	52.67	47.30	43.17
Attn_{c}, head=4	52.99	50.04	45.68	41.86

navigation. Firstly, we use the detection method outlined in [3] to extract the labels, dense features and bounding box positions of the visual objects. We then use these features as the dense visual input and employ an attention layer (see Fig. 5) to embed this dense visual input. There are two variants of dense visual input: 1) the visual feature extracted by ROI pooling [4], and 2) the glove feature [80] of the detection label. In addition, we need to provide the position of the object. We discuss two types of implementation of the location information: 1) a 2D xy-axis of the bounding boxes with the panoramic image index, and 2) a 3D panoramic orientation pointing at the object. In Tab. 7, we use ablation to determine the effects of different kinds of object features and location information on VLN. We discover that the dense feature is able to significantly improve the navigation performance: specifically, it improves the success rate by 1.45% and SPL by 0.68% in the unseen validation split, and also improves success rate by 1.92% and SPL by 1.54% in the unseen validation split. We further find that different kinds of location information have different effects on dense features. The best model is the application of 2D location on dense visual features. This approach outperforms the baseline on success rate by 1.57% and SPL by 0.92% on the seen split and on success rate by 3.11% and SPL by 3.00% on the unseen split. This proves that the object location is important to the dense visual features but is unimportant to the dense label features.

Dense Feature Weighting We find that the weighting mechanism of dense features significantly impacts the navigation performance. We use ablation to evaluate three kinds of weighting approaches: 1) using the softmax function to generate the probability distribution; 2) using the Tanh function to generate weights ranging from $[-1, 1]$; 3) using the sigmoid function with a threshold to generate weights ranging from $[0, 1]$. As Tab. 8 shows, the attention layer with softmax outperforms the other two models on both seen and unseen splits. Compared with the softmax function, the most popular approach in attention methods is to use the sigmoid function as the weighting function improves the success rate by 0.98% and SPL by 0.67% on the seen split and improves the success rate by 1.70% and SPL by 0.90% on the unseen split. We infer that the threshold is able to filter out the features with low weights. The detection result contains a lot of noise due to the domain gap between the training set of the detection model, which is MSCOCO [16] and the testing set, which is our environment. Thus the gated sigmoid mechanism is more robust to data noise.

Multi-head Attention We compare the impact of different number of attention heads on the navigation performance. We experiment on three attention layers one at a time: 1) Attn_{spa}, the global image vision attention; 2) Attn_{w} for vision-language alignment; 3) Attn_{c}, the candidate attention layer used to select which candidate to move. We compare each module by attention head number (1,2 and 3 respectively) with the baseline, that uses the soft-dot attention. As the results in Tab. 9 show, the multi-head attention only works on the global image visual input. The model with head number 2 performs achieves the best performance, improving the success rate by 2.55% and SPL by 1.41% on the seen split and also improving the success rate by 3.11% and SPL by 3.00% on the unseen split. Among all Attn_{spa} ablation experiments, we find that the attention head=2 reaches the highest SPL (57.3%) on the seen validation split while head=3 reaches the highest SPL (45.9%) on the unseen validation split. However, multi-head attention fails in multi-modal attention and candidate attention. We infer that multi-head attention contributes to visual attention, because the visual scene is complex and multi-peak attention distribution is beneficial.

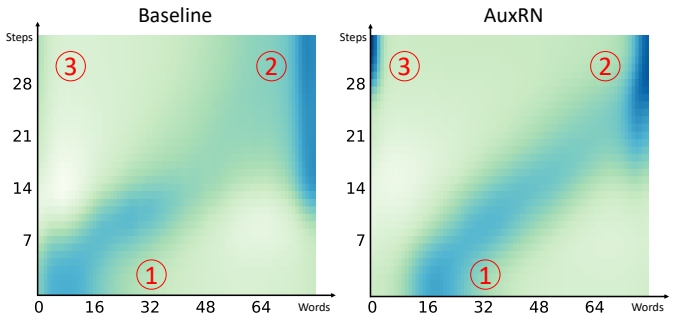


Fig. 4. The language attention map for the baseline model and our final model.

4.5 Visualization

Regularized Language Attention We visualize the attention map for Attn_{w} after Bi-LSTM$_{w}$. The dark region in the map indicates where the language features receive more attention. The x-axis of the map represents the position of the words. And the y-axis indicates the navigation steps. Compared with the baseline model (the left figure), the attention region of our model (the right figure) moves more regular with the navigation steps. In the early steps, our model pays 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. This indicates that our model has a higher success rate than the baseline, since the attention map tends to develop a uniform distribution when the agent gets lost. By comparing region ② and region ③ for both maps, we can conclude that our model learns a more definite attention on the beginning features and the end features. We therefore infer from our experiments that auxiliary reasoning losses help to regularize the language attention map, which turns out to be beneficial.

Navigation Visualization In Fig. 5, we visualize two testing trajectories in a panoramic view to illustrate the process of vision-language navigation. To demonstrate how MG-AuxRN understands the navigation process, we present the result of the progress estimator and the matching tasks. The estimated progress continues to grow during navigation while the matching result increases exponentially. When MG-AuxRN reaches the goal, the progress and matching results jump to almost 1. It turns out that the MG-AuxRN is able to accurately estimate the current progress and the instruction trajectory consistency. Thus the agent is able to understand the navigation process.

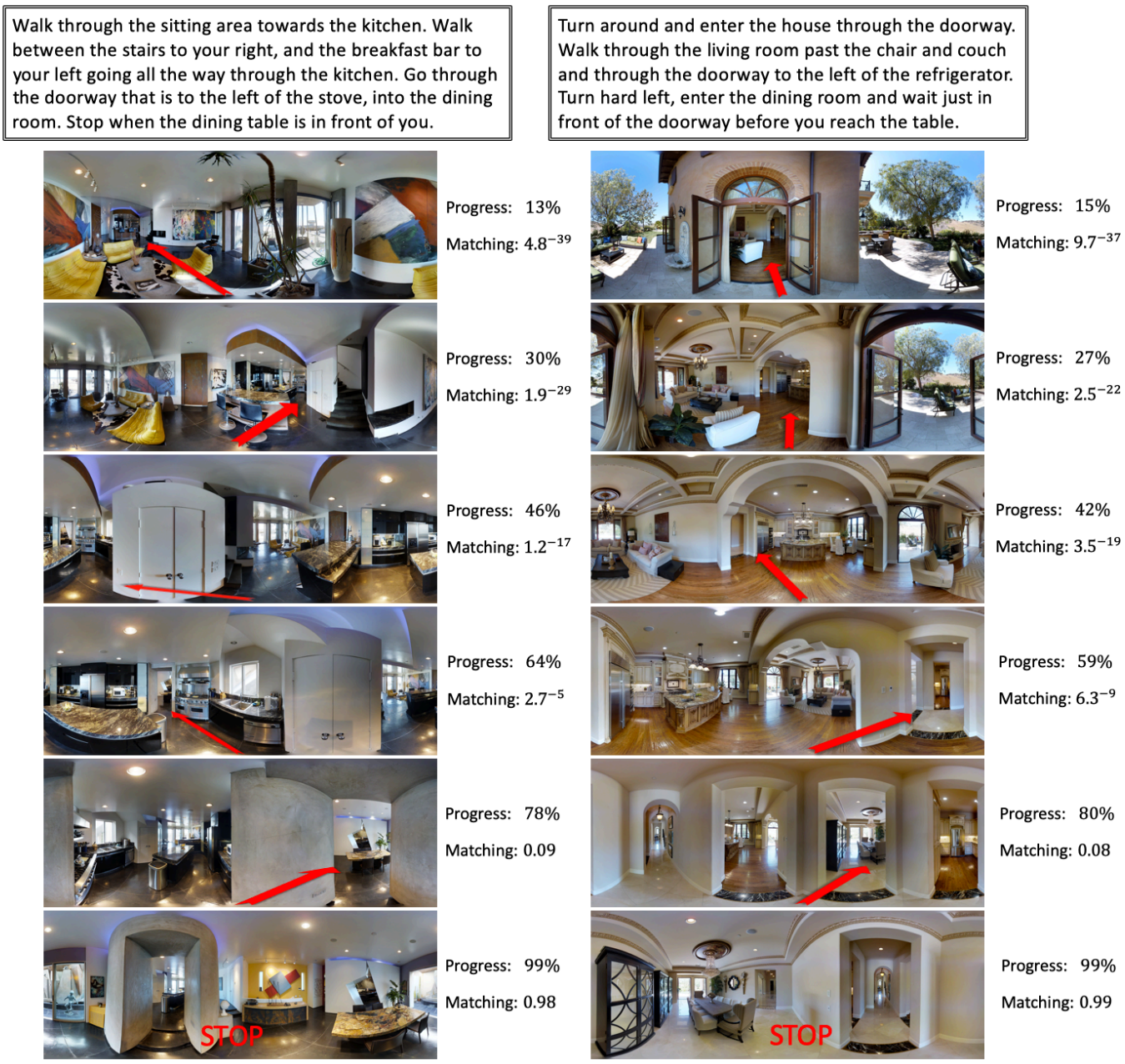


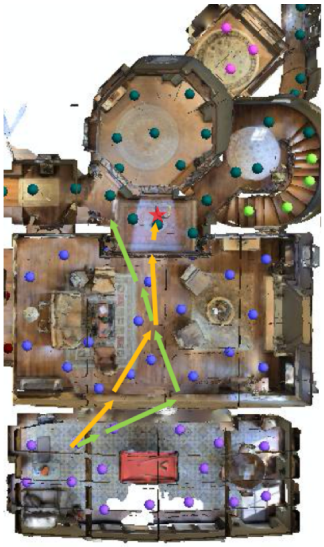
Fig. 5. Visualization process of two trajectories in testing. Two complex language instructions are shown in top boxes. Each image is a panoramic view, which is the vision input for MG-AuxRN. Each red arrow represents the direction to the next step. For each step, presented on the right.

Top-down Trajectory Visualization In Fig. 6, we provide a top-down view of four testing trajectories for the baseline (green lines) and our final model (yellow lines). The points represents the navigable viewpoints. Viewpoints that are marked with the same color belong to the same room. The instruction of each trajectory are shown on top of the figures. We can infer that the navigation environment is challenging, with diverse scenes and complex room structures. Two agents starts from the same point and are instructed to go to the point marked with a red star. We find that both model are able to move toward the goal over a significant distance. Our model is able to navigate toward the goal with few hops and less trajectory distance. Thus, our model is able to navigate more accurately and more efficiently.

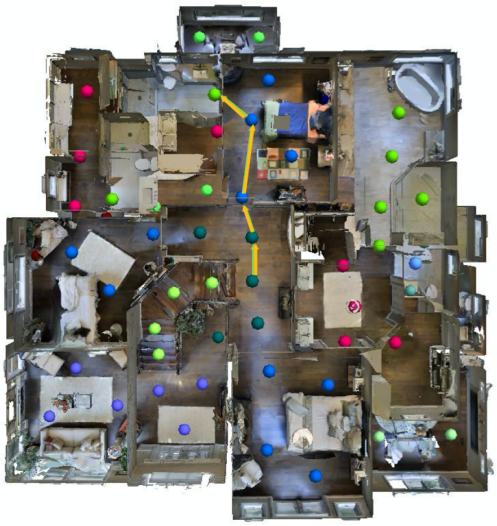
Visual Attention Visualization In Fig. 7, we visualize the detected objects and the attention results in testing. In the panoramic images, we visualize the detected objects (the red bounding boxes) whose detection confidence is larger than 0.85. We find that the detection is able to accurately detect various indoor objects despite the domain gap between its training domain (MSCOCO) and testing domain (Matterport), which facilitates the navigation. Also we visualize the sigmoid weighting values beside the detected object class. We find that lots of the weights are set as 0. It shows that the sigmoid weighting function efficiently filters out the unrelated object features. We also compare the attention distributions of the baseline (the green curve) with the multi-head attention model (the red curve and the blue curve) for each view. The x-axis represents the heading angle of the camera, indicating different panoramic views at the current navigation step. The y-axis represents the attention values. Compared with the baseline, multi-head attention usually outputs an attention distribution that is multimodal (rather than unimodal, as in the baseline, enabling it to attend on more views and result in comprehensive understanding of the current

Turn around and walk through the kitchen. Continue through the hallway to the dining room. Stop between the dining table and the pink sofa.

Walk past the pool table and walk into the room with the fireplace. With the fireplace on your right walk down the walkway and stop at the end before you enter the next room.



Walk down the hallway past the vases, enter the bedroom, walk to the foot of the bed, walk toward the bathroom on the left, wait in the doorway to the bathroom.



Leave the bathroom. Go through the door straight across the hall walk straight through the kitchen. Turn left at the end of the bar turn left again and go in the room to the right stand in between the closet and the sink.

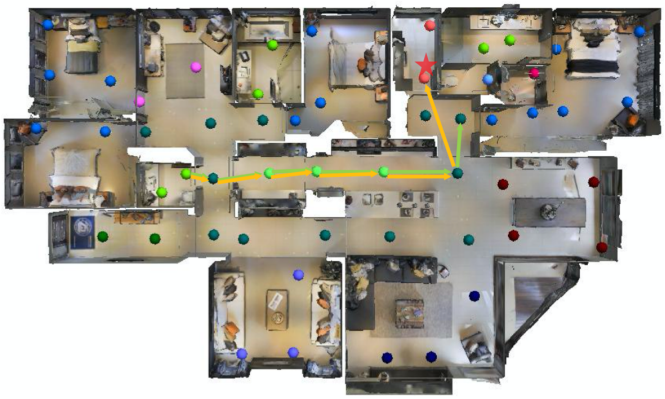


Fig. 6. Top-down view of testing trajectories of the baseline (green) and our final model (yellow). The points stands for the navigable viewpoints.

scene. However, as shown in Tab. 9, too many attention heads will harm the performance. Compared with the model with two attention heads, the model with four attention heads exibits an 0.22% performance drop. By comparing the orange curves with the blue curves in Fig. 7, we can see that the blue curves are usually flat, giving the position views similar attention, which harms the attention representation ability. The action of the agent with multi-head attention (head=2) is shown in Fig. 7 (the orange arrow). The action direction is correlated with the views and objects with high attention values, indicating the multi-head attention is effective. Thus, we conclude that the multi-head attention helps navigation.

5 Conclusion

In this paper, we presented a novel framework, Multi-granularity Auxiliary Reasoning Navigation (MG-AuxRN), that facilitates navigation learning with four auxiliary reasoning tasks and multi-granularity inputs. The auxiliary reasoning tasks help the agent to acquire knowledge about semantic representations in order to reason about its activity and build a thorough perception of the environment. The experimental results confirm that the proposed framework improves the performance of the vision-language navigation task both quantitatively and qualitatively.

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 No. 61702415, as well as by Australian Research Council Discovery Early Career Researcher Award (DE190100626).

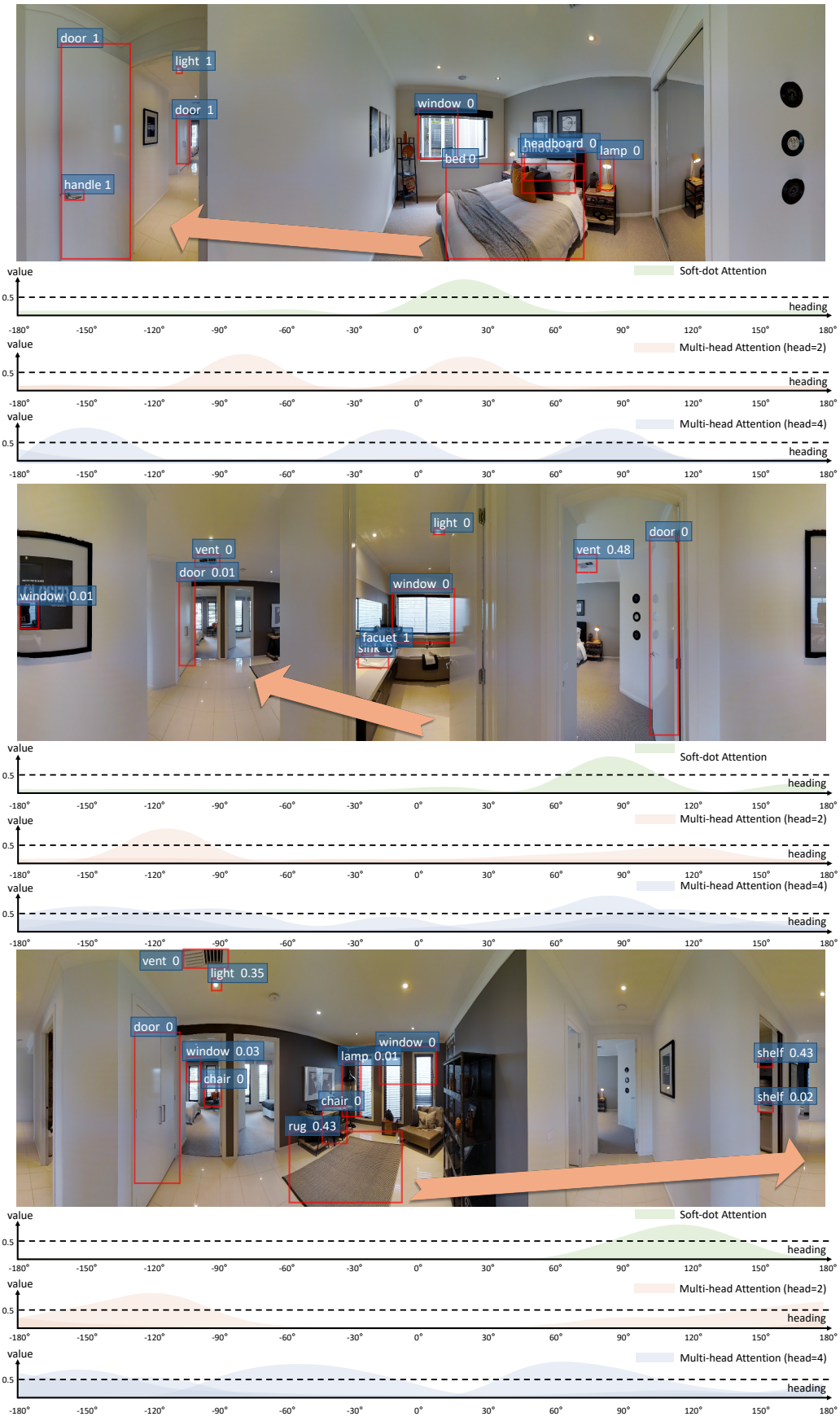


Fig. 7. Visualization of the input information and the attention results of global image features in testing. In the panoramic images, we visualize the detected bounding boxes. Besides, we visualize the global image attention for different orientation views of the panoramic image.

REFERENCES

[1] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sunderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *CVPR 2018*.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR 2016*.

[3] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *CVPR 2018*, pp. 6077–6086.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *TPAMI 2017*.

[5] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *NeurIPS 2019*, pp. 13–23.

[6] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *IJCV 2017*.

[7] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *CVPR 2018*, pp. 6629–6638.

[8] F. Zhu, Y. Zhu, X. Chang, and X. Liang, "Vision-language navigation with self-supervised auxiliary reasoning tasks," in *CVPR 2020*, pp. 10 012–10 022.

[9] C.-Y. Ma, J. lu, Z. Wu, G. AlRegib, Z. Kira, R. socher, and C. Xiong, "Self-monitoring navigation agent via auxiliary progress estimation," in *ICLR 2019*.

[10] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt, "Test-time training for out-of-distribution generalization," in *ICML 2020*.

[11] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR 2018*.

[12] H. Tan, L. Yu, and M. Bansal, "Learning to navigate unseen environments: Back translation with environmental dropout," in *NAACL 2019*.

[13] H. Tan and M. Bansal, "Lxmert: Learning cross-modality encoder representations from transformers," in *EMNLP 2019*, pp. 5099–5110.

[14] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *EMNLP 2015*, pp. 1412–1421.

[15] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," in *arXiv preprint arXiv:1807.06757*, 2018.

[16] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV 2014*, pp. 740–755.

[17] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," in *TACL 2014*.

[18] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," in *IJCV 2017*.

[19] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1194–1201.

[20] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele, "A dataset for movie description," in *CVPR 2015*, pp. 3202–3212.

[21] D. Chen and W. Dolan, "Collecting highly parallel data for paraphrase evaluation," in *ACL 2011*, pp. 190–200.

[22] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh, "Vqa: Visual question answering," in *ICCV 2015*.

[23] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. F. Moura, D. Parikh, and D. Batra, "Visual dialog," in *CVPR 2017*.

[24] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning," in *CVPR 2017*, pp. 1988–1997.

[25] K. Yi, C. Gan, Y. Li, P. Kohli, J. Wu, A. Torralba, and J. B. Tenenbaum, "Clevrer: Collision events for video representation and reasoning," in *ICLR 2020*.

[26] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From recognition to cognition: Visual commonsense reasoning," in *CVPR 2019*, pp. 6720–6731.

[27] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR 2015*, pp. 2625–2634.

[28] A. Karpathy, A. Joulin, and F. F. F. Li, "Deep fragment embeddings for bidirectional image sentence mapping," in *NeurIPS 2014*, 2014, pp. 1889–1897.

[29] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," in *ICLR 2015*.

[30] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR 2014*.

[31] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," in *EMNLP 2016*.

[32] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," in *NeurIPS 2016*, pp. 289–297.

[33] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei, "Visual7w: Grounded question answering in images," in *CVPR 2016*, 2016, pp. 4995–5004.

[34] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI 2018*, pp. 3942–3951.

[35] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, "Stacked cross attention for image-text matching," in *ECCV 2018*, pp. 212–228.

[36] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *ICLR 2017*.

[37] A. P. Parikh, O. Tackstrom, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *EMNLP 2016*, pp. 2249–2255.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS 2017*, pp. 5998–6008.

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," in *IJCV 2015*.

[40] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR (Workshop Poster) 2013*.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," in *Neural Computation*, 1997, pp. 1735–1780.

[42] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," in *TPAMI 2020*, pp. 1–1.

[43] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV 2016*, pp. 649–666.

[44] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR 2016*, pp. 2536–2544.

[45] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *ECCV 2016*, pp. 69–84.

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT 2019*.

[47] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," in *ICLR 2020*.

[48] Y.-C. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "Uniter: Universal image-text representation learning," in *CVPR 2019*.

[49] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering," in *CVPR 2017*, pp. 6325–6334.

[50] D. A. Hudson and C. D. Manning, "Gqa: a new dataset for compositional question answering over real-world images," in *CVPR 2019*.

[51] A. Suhr, S. Zhou, A. Zhang, I. Zhang, H. Bai, and Y. Artzi, "A corpus for reasoning about natural language grounded in photographs," in *ACL 2019*, pp. 6418–6428.

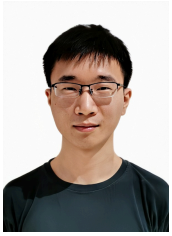
[52] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *ICLR 2017*.

[53] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *ICML 2017*, pp. 2778–2787.

[54] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," in *ICLR 2017*.

[55] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," in *Nature 2017*, pp. 354–359.

[56] J. Vanschoren, "Meta-learning: A survey," in *arXiv preprint arXiv:1810.03548*, 2018.
[57] V. Veeriah, R. L. Lewis, J. Rajendran, D. Silver, and S. Singh, "The discovery of useful questions as auxiliary tasks," in *NeurIPS 2019*.
[58] S. Gidaris, A. Bursuc, N. Komodakis, P. P. Perez, and M. Cord, "Boosting few-shot visual learning with self-supervision," in *ICCV 2019*, pp. 8058–8067.
[59] S. Lee, D. Cho, J. Kim, and T. H. Kim, "Self-supervised fast adaptation for denoising via meta-learning," in *arXiv preprint arXiv:2001.02899*, 2020.
[60] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *CIG: Computational Intelligence and Games*, 2016.
[61] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," in *CVPR 2017*.
[62] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3d environment," in *ICLR 2018*.
[63] S. Thrun, *Probabilistic Robotics*, 2005.
[64] A. J. Davison and D. W. Murray, "Mobile robot localisation using active vision," in *ECCV 1998*, pp. 809–825.
[65] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *ICRA 2017*, pp. 3357–3364.
[66] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," in *arXiv*, 2019, pp. arXiv–1911.
[67] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," in *3DV: International Conference on 3D Vision*, 2017, pp. 667–676.
[68] X. Wang, W. Xiong, H. Wang, and W. Y. Wang, "Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation," in *ECCV 2018*.
[69] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML 2016*, pp. 1928–1937.
[70] S. Racanière, T. Weber, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. W. Battaglia, D. Hassabis, D. Silver, and D. Wierstra, "Imagination-augmented agents for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 5690–5701.
[71] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," in *CVPR 2016*.
[72] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NeurIPS 2016*.
[73] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," in *arXiv preprint arXiv:1707.06347*, 2017.
[74] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-follower models for vision-and-language navigation," in *NeurIPS 2018*.
[75] C.-Y. Ma, Z. Wu, G. AlRegib, C. Xiong, and Z. Kira, "The regretful agent: Heuristic-aided navigation through progress estimation," in *CVPR 2019*.
[76] L. Ke, X. Li, Y. Bisk, A. Holtzman, Z. Gan, J. Liu, J. Gao, Y. Choi, and S. Srinivasa, "Tactical rewind: Self-correction via backtracking in vision-and-language navigation," in *CVPR 2019*.
[77] H. Huang, V. Jain, H. Mehta, A. Ku, G. Magalhaes, J. Baldridge, and E. Ie, "Transferable representation learning in vision-and-language navigation," in *ICCV 2019*.
[78] Y. Zhang, H. Tan, and M. Bansal, "Diagnosing the environment bias in vision-and-language navigation," in *IJCAI 2020*.
[79] X. Li, C. Li, Q. Xia, Y. Bisk, A. Celikyilmaz, J. Gao, N. Smith, and Y. Choi, "Robust navigation with language pretraining and stochastic sampling," in *arXiv preprint arXiv:1909.02244*, 2019.
[80] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *EMNLP 2014*, pp. 1532–1543.



Fengda Zhu received the bachelor's degree in School of Software Engineering from Beihang University, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the Faculty of Information Technology, Monash University under the supervision of Prof. Xiaojun Chang. His research interests include machine learning, deep learning and reinforcement learning.

Yi Zhu received the B.S. degree in software engineering from Sun Yat-sen University, Guangzhou, China, in 2013. Since 2015, she has been a Ph.D student in computer science with the School of Electronic, Electrical, and Communication Engineering, University of Chinese Academy of Sciences, Beijing, China. Her current research interests include object recognition, scene understanding, weakly supervised learning and visual reasoning.

Yanxin Long is a first-year master in the School of Intelligent Systems Engineering, Sun Yat-Sen University. He works at the Human Cyber Physical Intelligence Integration Lab under the supervision of Prof. Xiaodan Liang. Before that, he received his Bachelor Degree from the Comunication College, Xidian University in 2020.

Xiaojun Chang is currently a Senior Lecturer with the Faculty of Information Technology, Monash University Clayton Campus, Clayton, VIC, Australia. He is also with the Monash University Centre for Data Science. Before joining Monash University, he was a Post-Doctoral Research Associate with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, working with Prof. A. Hauptmann. He has spent most of the time working on exploring multiple signals (visual, acoustic, and textual) for automatic content analysis in unconstrained or surveillance videos.

Dr. Chang is an ARC Discovery Early Career Researcher Award (DECRA) Fellow from 2019 to 2021. He has achieved top performance in various international competitions, such as TRECVID MED, TRECVID SIN, and TRECVID AVS.

Xiaodan Liang is currently an Associate Professor at Sun Yat-sen University. She was a postdoc researcher in the machine learning department at Carnegie Mellon University, working with Prof. Eric Xing, from 2016 to 2018. She received her PhD degree from Sun Yat-sen University in 2016, advised by Liang Lin. She has published several cutting-edge projects on human-related analysis, including human parsing, pedestrian detection and instance segmentation, 2D/3D human pose estimation and activity recognition.