Vision-Language Navigation with Random Environmental Mixup

Chong Liu [1,2*] Fengda Zhu [3*] Xiaojun Chang [4] Xiaodan Liang[5] Zongyuan Ge[3] Yi-Dong Shen [1†]
[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Monash University, Melbourne, Australia
[4] RMIT University, Melbourne, Australia
[5] Sun Yat-sen University, Guangzhou, China
liuchong@ios.ac.cn fengda.zhu@monash.edu xiaojun.chang@rmit.edu.au
xdliang328@gmail.com zongyuan.ge@monash.edu ydshen@ios.ac.cn

Abstract

Vision-language Navigation (VLN) tasks require an agent to navigate step-by-step while perceiving the visual observations and comprehending a natural language instruction. Large data bias, which is caused by the disparity ratio between the small data scale and large navigation space, makes the VLN task challenging. Previous works have proposed various data augmentation methods to reduce data bias. However, these works do not explicitly reduce the data bias across different house scenes. Therefore, the agent would overfit to the seen scenes and achieve poor navigation performance in the unseen scenes. To tackle this problem, we propose the Random Environmental Mixup (REM) method, which generates cross-connected house scenes as augmented data via mixuping environment. Specifically, we first select key viewpoints according to the room connection graph for each scene. Then, we cross-connect the key views of different scenes to construct augmented scenes. Finally, we generate augmented instruction-path pairs in the cross-connected scenes. The experimental results on benchmark datasets demonstrate that our augmentation data via REM help the agent reduce its performance gap between the seen and unseen environment and improve the overall performance, making our model the best existing approach on the standard VLN benchmark.

1. Introduction

Recently, there is a surge of research interests in Vision-Language Navigation (VLN) [4] tasks, in which an agent learns to navigate by following a natural language instruction. The agent begins at a random point and goes towards

*Equal Contribution
†Corresponding author

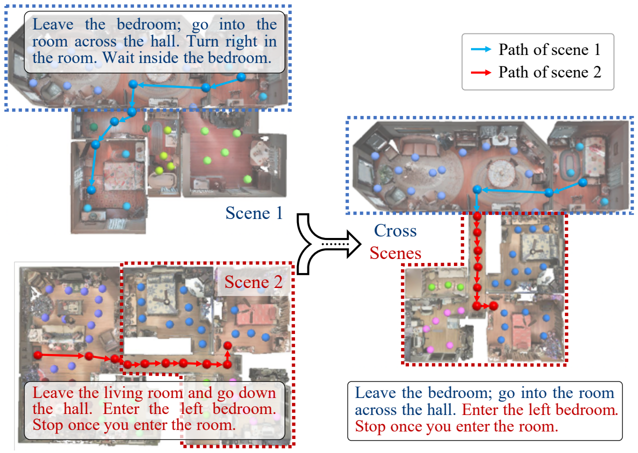
Leave the bedroom; go into the room across the hall. Turn right in the room. Wait inside the bedroom.
Path of scene 1
Path of scene 2
Scene 1
Cross Scenes
Scene 2
Leave the living room and go down the hall. Enter the left bedroom. Stop once you enter the room.
Leave the bedroom; go into the room across the hall. Enter the left bedroom. Stop once you enter the room.

Figure 1. **REM** mixes up two scenes and generates data triplets (environment, path, instruction). We divide the two scenes and recombine them to construct a new cross-connected scene, and reconstruct the corresponding paths and instructions.

a goal via actively exploring the environments. Before the navigation starts, the agent receives a language instruction. At every step, the agent can get the surrounding visual information. The key to this task is to perceive the visual scene and comprehend natural language instructions sequentially and make actions step-by-step.

Recent advances made by deep learning works in the domains of feature extraction [19, 3, 40, 43], attention [3, 15, 35] and multi-modal grounding [5, 34, 50] facilitate the agent to understand the environment. Moreover, many reinforcement learning works [41, 26, 48] help the agent to obtain a robust navigation policy. Benefited from these works, previous attempts in the field of Vision-Language Navigation have made great progress in improving the ability to perceive the vision and language inputs [17, 16, 56, 55], and learning a robust navigation policy [62, 18, 23]. However, the VLN task still contains large bias due to the disparity ratio between small data scale and large navigation space, which impacts the generalization ability of navigation. Although the mostly widely used dataset, Room-to-room dataset [4], contains only 22K instruction-path pairs, the actual possible navigation path space increases exponentially along with the path length. Thus, the learned navigation policy can easily overfit to the seen scenes and is hard to generalize to the unseen scenes.

Previous works have proposed various data augmentation methods in an attempt to reduce data bias. Fried *et al.* propose a speaker-follower framework [16] to generate more data pairs in order to reduce the data bias of the data samples. Tan *et al.* [51] propose an environmental dropout method to augment the vision features in environments; thereby reducing the vision bias inside a house scene. However, these methods focus on intra-scene data augmentation and fail to explicitly reduce the data bias across different house scenes.

Accordingly, in this paper, we propose to reduce the domain gap across different house scenes by means of scene-wise data augmentation. If an agent sees different house scenes during a navigation process, it will be less likely to overfit to a part of the scene textures or room structures. Inspired by this motivation, we propose our method, named Random Environmental Mixup (REM), to improve the generalization ability of a navigation agent. REM breaks up two scenes and the corresponding paths, followed by recombining them to obtain a cross-connected scene between the two scenes. The REM method provides more generalized data, which helps reduce the generalization error, so that the agent's navigation ability in the seen and unseen scenes can be improved.

The REM method comprises three steps. First, REM selects the key vertexes in the room connection graph according to the betweenness centrality [8]. Second, REM splits the scenes by the key vertexes and cross-connect them to generate new augmented scenes. We propose an orientation alignment approach to solve the feature mismatch problem. Third, REM splits trajectories and instructions into sub-trajectories and sub-instructions by their context, then cross-connects them to generate augmented training data. An overview of the REM method is presented in Fig. 1.

The experimental results on benchmark datasets demonstrate that REM can significantly reduce the performance gap between seen and unseen environments, which dramatically improves the overall navigation performance. Our ablation study shows that the proposed augmentation method outperforms other augmentation methods at the same augmentation data scales. Our final model obtains 59.1% in Success weighted by Path Length (SPL) [2], which is 2.4% higher than the previous state-of-the-art result; accordingly, our method becomes the new state-of-the-art method on the standard VLN benchmark.

2. Related Work

Embodied Navigation Environments are attracting rising attention in artificial intelligence. House3D [58] is a manually created large-scale environment. AI2-THOR [30] is an interactable indoor environment. Agents can interact with certain interactable objects, such as opening a drawer or picking up a statue. Recent works have tended to focus on simulated environments based on real imagery. The Active Vision dataset [1] consists of dense scans of 16 different houses. Moreover, Matterport3D [4], Gibson [59] and Habitat [47] propose high-resolution photo-realistic panoramic view to simulate more realistic environment.

Vision-Language Navigation has attracted widespread attention, since it is both widely applicable and a challenging task. Anderson *et al.* [4] propose the Room-to-Room (R2R) dataset, which is the first Vision-Language Navigation (VLN) benchmark to combine real imagery [9] and natural language navigation instructions. In addition, the TOUCHDOWN dataset [11] with natural language instructions is proposed for street navigation. To address the VLN task, Fried *et al.* propose a speaker-follower framework [16] for data augmentation and reasoning in a supervised learning context, along with a concept named "panoramic action space" that is proposed to facilitate optimization. Wang *et al.* [55] demonstrate the benefit of combining imitation learning [6, 20] and reinforcement learning [41, 48]. Other methods [56, 36, 38, 51, 28, 24] have been proposed to solve the VLN tasks from various perspectives. Due to the success of BERT [15], researchers have extended it to learn vision-language representations in VLN. PRESS [33] applies the pre-trained BERT to process instructions. PREVALENT [18] pre-trains an encoder with image-text-action triplets to align the language and visual states, while VLN-BERT [39] fine-tunes ViLBERT [34] with trajectory-instruction pairs. Hong *et al.* [23] implements a recurrent function to leverage the history-dependent state representations based on previous models.

Data Augmentation is widely adopted in diverse deep learning methods. Early data augmentation methods in the field of computer vision were manually designed; these include distortions, scaling, translation, rotation and color shifting [12, 54, 49, 46]. The traditional approach to text augmentation tends to involve a primary focus on word-level cases [61, 29, 57]. Some works have achieved success in using GAN to generate augmentation data [7, 45]. Zhang *et al.* [60] propose mixup, which is a linear interpolation augmentation method used to regularize the training of neural networks. Data augmentation has also been investigated in RL, including domain randomization [52, 42, 44], cutout [13] and random convolution [32]. In the vision-language navigation context, Fried *et al.* propose to use a generation method [16] to generate data pairs, while Tan *et al.* [51] propose an environmental dropout method to augment the vision features in various environments. In a departure from these augmentation methods, our REM method cross-connects the scenes and the instruction-trajectory pairs, thereby improving the model's generalization ability among different scenes.

3. Preliminaries

3.1. Vision-language Navigation

Given a series of triples (environment E, path P, instruction I), the VLN task requires the agent to understand the instruction in order to find a matching path in the corresponding environment. The environment E contains a large number of seen and unseen scenes, the path $P = \{p_0, \ldots, p_n\}$ is composed of a list of viewpoints with a length of n; morever, instruction $I = \{w_0, \ldots, w_m\}$ consists of m words, and a certain correspondence exists between path P and instruction I. At time step t, the agent observes panoramic views $O_t = \{o_{t,i}\}_{i=1}^{36}$ and navigable viewpoints (at most k). The picture O_t is divided into 12 views in the horizontal direction and 3 views in the vertical direction, for a total of 36 views. At the t-th step, the agent predicts an action $a \sim \pi_\theta(I, O_t)$, where π_θ is the policy function defined by the parameter θ. The actions include 'turn left', 'turn right' and 'move forward' as defined in the Matterpot environment [4]. In the Matterport dataset [9], each scene is discretisized by a navigation graph consists of viewpoints. We model each scene as a graph $G = (V, E)$, where the vertexes set V is a set of scene viewpoints, while E is the connection between viewpoints.

3.2. Reduce Generalization Error in VLN

The learning problem can be formulated as the search of the function $f \in \mathcal{F}$, which minimizes the expectation of a given loss $\ell(f(x), y)$. However, the distribution $\mathcal{P}$ of sample (x, y) is generally unknown. We can often obtain a set $\mathcal{T} \sim \mathcal{P}$ and use it as a training set. The approximate function $\hat{f}$ can then be implemented by Empirical Risk Minimization (ERM) [53]. However, a gap still exists between $\hat{f}$ and f. This error describes the generalization ability of $\hat{f}$. The generalization error can be expressed as follows:

$$\begin{aligned} R_{ge}(\hat{f}) &= R(f) - R(\hat{f}) \\ &= \int (\ell(f(x), y) - \ell(\hat{f}(x), y)) d\mathcal{P}(x, y). \end{aligned} \tag{1}$$

In order to enhance the generalization ability of $\hat{f}$, it is necessary to reduce R_{ge}. According to the Vicinal Risk Minimization (VRM) [10]:

$$R_v(f) = \frac{1}{m} \sum_{i=1}^{n} \ell(f(\tilde{x}), \tilde{y}), \tag{2}$$

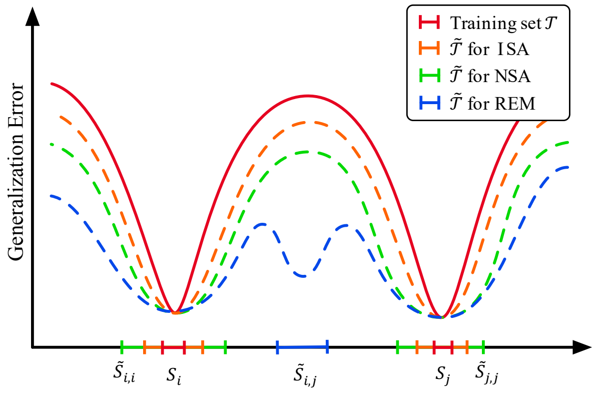

Figure 2. The generalization error of the original training set and different augmentation training sets. With Original (**Red**) → ISA (**Orange**) → NSA (**Green**) → REM (**Blue**), the distance between $\mathcal{T}$ and $\tilde{\mathcal{T}}$ is getting farther and farther, and the generalization error decreases accordingly.

where $(\tilde{x}, \tilde{y}) \in (\mathcal{T} \cup \tilde{\mathcal{T}})$, $\tilde{\mathcal{T}} \sim \mathcal{P}$, $\tilde{\mathcal{T}} \nsubseteq \mathcal{T}$. This means that more samples are needed to lower R_{ge}. When the number of samples is certain, the farther the distance $d((\tilde{x}, \tilde{y}), \mathcal{T})$ from the sample $(\tilde{x}, \tilde{y})$ to the training set $\mathcal{T}$, the better the generalization ability.

In the VLN task, the training set consists of N scenes:

$$\mathcal{T} = (S_1 \cup S_2 \cup \cdots \cup S_N). \tag{3}$$

We define a data augmentation function $\text{aug}(S_i, S_j)$. The generated augmentation data follows the distribution $\mathcal{P}$:

$$\begin{aligned} \tilde{S}_{i,i} &= \text{aug}(S_i, S_i) \sim \mathcal{P}, \\ \tilde{S}_{i,j} &= \text{aug}(S_i, S_j) \sim \mathcal{P}, \end{aligned} \tag{4}$$

where $\tilde{S}_{i,i}$ is the intra-scene augmentation data set, and $\tilde{S}_{i,j}$ is the inter-scene augmentation data set. According to Eqs. 1 and 2, we have the following assumption: compared with $\tilde{S}_{i,i}$, the distance from $\tilde{S}_{i,j}$ to S_i is farther, denoted as $d(\tilde{S}_{i,i}, S_i) < d(\tilde{S}_{i,j}, S_i)$. Therefore, the model learned on the inter-scene augmentation data has a smaller generalization error than which learned on the intra-scene augmentation data.

Previous methods have proposed two kinds of data augmentation methods in VLN: the intra-scene augmentation (ISA) method, as in [16], only constructs new paths and instructions in the scene; the near-scene augmentation (NSA) method, as in [51], breaks through the limitations of the scene to a certain extent by adding Gaussian noise to the scene, but only expands the scene to a small neighborhood. For our part, we propose a inter-scene data augmentation method: Random Environmental Mixup (REM). REM method mixes up two scenes constructs a cross-connected scene between the two scenes. In contrast to the other meth-

Algorithm 1: Selecting key vertexes

```
   Input: Scene graph $G$; Paths list $P = \{p_1, ..., p_{|P|}\}$
   Output: Key vertexes $v_s^{key}, v_t^{key}$
1  Get vertexes set $V$ of $G$;
2  Get edges set $E$ of $G$;
3  $\tilde{V} = \{v \mid \text{top } 10\ v \text{ in } V \text{ ordered by } VC_B(v)\}$;
4  $\tilde{E} = \{e \mid \text{top } 10\ e \text{ in } E \text{ ordered by } EC_B(e)\}$;
5  $m = 0$;
6  // Select the vertex passed by the most paths.
7  for $e$ in $E$ do
8      Get vertexes $v_s, v_t$ of the edge $e$;
9      $n_e = \sum_{i=1}^{|P|}(1_{\{e \in p_i\}} + 0_{\{e \notin p_i\}})$;
10     if $v_s, v_t \in V$ then
11         if $m < n_e$ then
12             $m = n_e$;
13             $v_s^{key} = v_s$;
14             $v_t^{key} = v_t$;
15         end
16     end
17 end
18 return $v_s^{key}, v_t^{key}$
```

ods, it exceeds the limitation of the scene itself and constructs augmentation data under a broader data distribution.

Fig. 2 illustrates the difference between three methods. The inter-scene method provides more generalized data; this helps to reduce the generalization error, meaning that the agent's navigation ability in the seen scene and the unseen scene can be improved. Subsequent experiments have verified this assumption.

4. Random Environmental Mixup

We propose an inter-scene data augmentation method to construct new environments, paths, and instructions with the aid of training sets. In the training set of the VLN task, there are a large number of different scenes. We randomly select two scenes from the set of training scenes and mix them up to generate cross-connected scenes. Adopting this approach enables us to construct the corresponding paths and instructions. When mixing of scenes, we have the following problems: 1) how to choose key vertexes in the scene for mixup? 2) how to mix up two scenes to obtain cross-connected scenes? 3) how to construct new paths and instructions in cross-connected scenes? Solutions to these problems are presented below construct a large number of cross-connected scenes, that are unseen relative to the original training set.

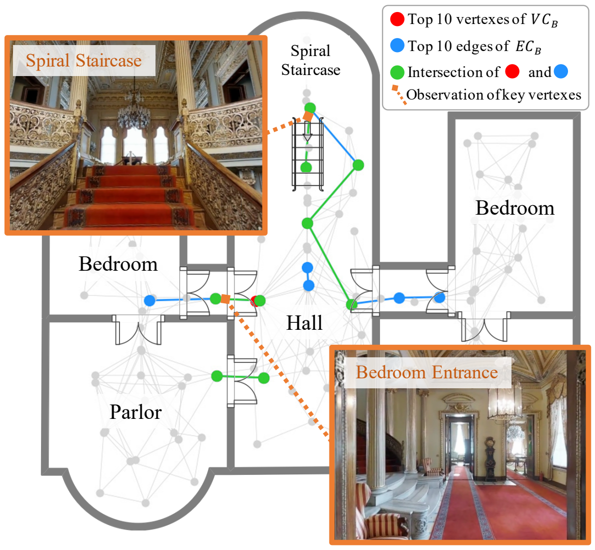

Figure 3. Selecting key vertexes through betweenness centrality. The green edges are often the entrances and exits of rooms or corridors, we choose the **two vertexes** of the green edge that contains the most paths as the key vertexes.

4.1. Select Key Vertexes

Key vertexes are crucial to mixing between scenes. Their characteristics can be summarized as follows: 1) the entrance or corridor that connects two rooms; 2) the vertex has many paths through it. In order to match the above characteristics, key vertexes can be selected with reference to the betweenness centrality [8] of the graph:

$$\begin{aligned} VC_B(v) &= \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}, \\ EC_B(e) &= \sum_{s \neq t \in V; e \in E} \frac{\sigma_{st}(e)}{\sigma_{st}}, \end{aligned} \tag{5}$$

where $VC_B(v)$ is the betweenness centrality of the vertex v, $EC_B(e)$ is the betweenness centrality of the edge e; $\sigma_{st}(v)$ is the number of shortest paths from s to t, passes through the vertex v; $\sigma_{st}(e)$ is the number of shortest paths from s to t through edge e; σ_{st} is the number of all shortest paths from s to t. Betweenness centrality describes the importance of vertex by the number of shortest paths passing through vertexes or edges. Once the vertex is removed from the graph, the points on both sides will be disconnected.

As shown in Fig. 3, we select the top 10 vertexes and edges of betweenness centrality to obtain the corresponding sets V^{VC_B} and E^{EC_B}; subsequently, by excluding edges in E^{EC_B} whose vertexes are not in V^{VC_B}, we obtain the final key subgraph G^{C_B}. In order to ensure that more paths are subsequently generated, we select the edge e^{key} that contains the most supervised paths from G^{C_B}, along with its corresponding vertexes v_s^{key}, v_t^{key}. We observe from Fig. 3

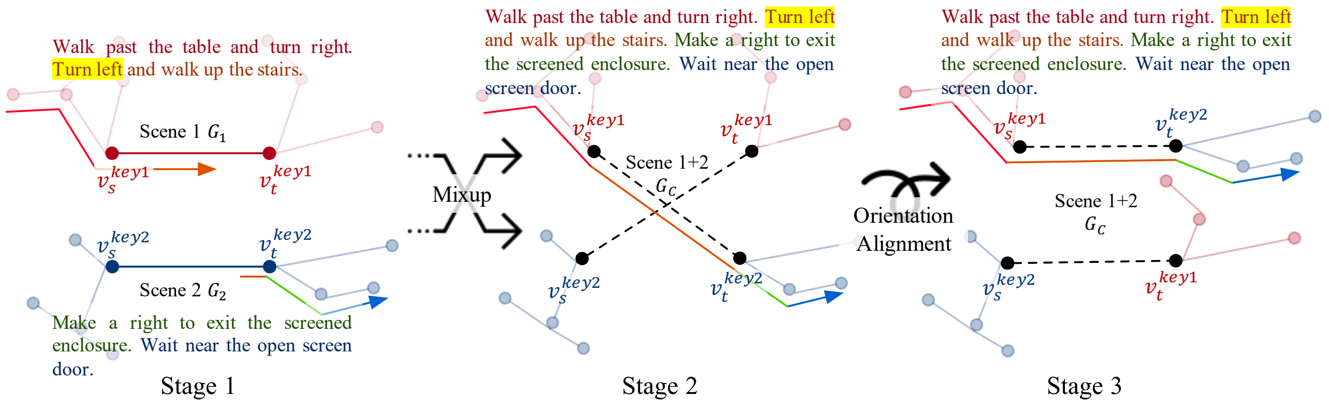
Walk past the table and turn right. Turn left and walk up the stairs.
Scene 1 G_1
v_s^{key1}
v_t^{key1}
v_s^{key2}
v_t^{key2}
Scene 2 G_2
Make a right to exit the screened enclosure. Wait near the open screen door.
Stage 1
Mixup
Walk past the table and turn right. Turn left and walk up the stairs. Make a right to exit the screened enclosure. Wait near the open screen door.
Scene 1+2 G_C
Stage 2
Orientation Alignment
Scene 1+2 G_C
Stage 3

Figure 4. Three stages of mixup scenes. **Stage 1:** select key vertexes (v_s^{key1}, v_t^{key1}) and (v_s^{key2}, v_t^{key2}) for scene 1 and scene 2. **Stage 2:** mixup scene 1 and scene 2, relink (v_s^{key1}, v_t^{key2}) and (v_s^{key2}, v_t^{key1}). **Stage 3:** fix the position of the vertexes, align the orientation of (v_s^{key1}, v_t^{key2}) and (v_s^{key2}, v_t^{key1}). The instructions are fine-grained, and the sub-paths of different colors are matched with the sub-instructions of the corresponding colors. As the scene is mixed up, paths and instructions are also broken up and reconstructed. The constructed scenes, paths and instructions are combined into triples, which become augmentation data for VLN tasks.

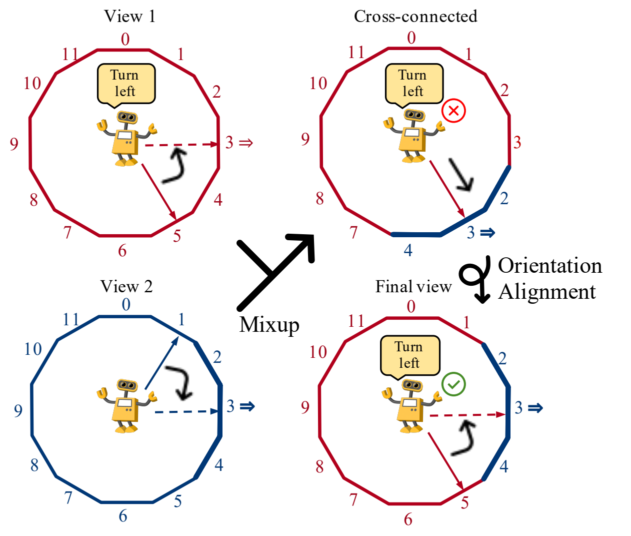

Figure 5. The process of mixing up viewpoints. **View 1** is the visual observation of v_s^{key1} in Scene 1, while **View 2** is the visual observation of v_s^{key2} in Scene 2. The solid arrow is the current direction of the agent. The dotted arrow is the direction of the agent after taking the action. '⇒' represents the direction to the next viewpoint. 'turn left' is the instruction received by the agent. '↰,→' are the 'turn left' and 'forward' actions taken by the agent in order to arrive at the next viewpoint.

that the entrances and exits of rooms or corridors often have the highest betweenness. The process of selecting key vertexes is summarized in Algo. 1.

4.2. Construct Augmented Triplets

Construct Cross-Connected Scenes We randomly select two scenes (Scene1 G_1 and Scene2 G_2) in the training set. We construct the cross-connected scene G_C for G_1 and G_2 in three stages (Fig. 4). In stage 1: according to Algo.1, we obtain the key vertexes (v_s^{key1}, v_t^{key1}) for G_1 and (v_s^{key2}, v_t^{key2}) for G_2. In stage 2: we mix up G_1,G_2 into graph G_C, disconnect the two key edges e^{key1},e^{key2}, and link (v_s^{key1}, v_t^{key2}),(v_s^{key2}, v_t^{key1}). In this way, we obtain a cross-connected scene G_C. In stage 3: we align the orientation of G_C; ensure the matching of the cross path and the instruction by adjusting the vertex position in G_C.

Construct Cross Viewpoints G_C is a graph containing the connection relationship information without visual observation. Therefore, we build a cross viewpoint on the basis of G_C to obtain a new cross-connected environment. The process of building a new cross-connected environment is illustrated in Fig. 5. Taking the v_s^{key1} in Scene 1+2 as an example, as described in Sec. 3.1, each viewpoint panoramic view is divided into 12 views in the horizontal direction (indicated by the numbers 0–11). By mixing the views of View 1 and View 2, we can obtain a panoramic view of View 1+2. More specifically, the view is based on the direction of the next viewpoint. We replace three views around the original angle of View 2 with View 1 to get Cross-connected view(red 0–3 7–11 from View 1, blue 2-4 from View 2). The hyperparameter settings for replacing 3 views will be discussed in the experimental part.

Construct Cross Paths and Instructions Cross-connecting the instructions and paths requires the instructions and paths to be fine-grained. In order to obtain the fine-grained data, we use Fine-Grained R2R [22] to split the instructions and paths, as well as to align the sub-instructions and the sub-paths. As shown in Fig. 4 (stage 3), we obtain the path and instructions in the cross-connected scene by combining the sub-paths before and after the key vertex, along with the corresponding sub-instructions.

Orientation Alignment Following Fig. 4 (stages 1 and 2), we construct the cross-connected scenes and corresponding cross viewpoints. Simply connecting v_s^{key1} and v_t^{key2} leads to a mismatch of the related orientations of the ver-

Method	R2R Validation Seen				R2R Validation Unseen				R2R Test Unseen			
	TL	NE↓	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑
Random	9.58	9.45	16	-	9.77	9.23	16	-	9.89	9.79	13	12
Human	-	-	-	-	-	-	-	-	11.85	1.61	86	76
Seq2Seq-SF [4]	11.33	6.01	39	-	8.39	7.81	22	-	8.13	7.85	20	18
Speaker-Follower [16]	-	3.36	66	-	-	6.62	35	-	14.82	6.62	35	28
SMNA [37]	-	3.22	67	58	-	5.52	45	32	18.04	5.67	48	35
RCM+SIL [55]	10.65	3.53	67	-	11.46	6.09	43	-	11.97	6.12	43	38
PRESS [33]	10.57	4.39	58	55	10.36	5.28	49	45	10.77	5.49	49	45
FAST-Short [28]	-	-	-	-	21.17	4.97	56	43	22.08	5.14	54	41
EnvDrop [51]	11.00	3.99	62	59	10.70	5.22	52	48	11.66	5.23	51	47
AuxRN [62]	-	3.33	70	67	-	5.28	55	50	-	5.15	55	51
PREVALENT [18]	10.32	3.67	69	65	10.19	4.71	58	53	10.51	5.30	54	51
RelGraph [21]	10.13	3.47	67	65	9.99	4.73	57	53	10.29	4.75	55	52
VLN↻Bert [23]	11.13	**2.90**	**72**	**68**	12.01	**3.93**	**63**	**57**	12.35	**4.09**	**63**	**57**
IL+RL* [51]	10.25	4.91	53.8	50.7	9.38	5.89	46.2	42.5	9.58	5.88	46.4	43.3
IL+RL+REM	10.18	4.61	58.2	55.3	9.40	5.59	48.6	44.8	9.81	5.67	48.7	45.1
EnvDrop* [51]	10.46	3.78	64.4	62.0	9.50	5.52	51.1	47.3	11.32	5.84	50.5	46.5
EnvDrop+REM	11.13	3.14	70.1	66.7	14.84	4.99	53.8	48.8	10.73	5.40	54.1	50.4
VLN↻Bert* [23]	12.09	2.99	70.7	65.9	12.58	4.02	61.4	55.6	11.68	4.35	61.4	56.7
VLN↻Bert+REM	10.88	**2.48**	**75.4**	**71.8**	12.44	**3.89**	**63.6**	**57.9**	13.11	**3.87**	**65.2**	**59.1**

Table 1. Comparison of agent performance on **R2R** in single-run setting. * reproduced results in my environment.

Method	R4R Validation Seen						R4R Validation Unseen					
	NE↓	SR↑	SPL↑	CLS↑	nDTW↑	SDTW↑	NE↓	SR↑	SPL↑	CLS↑	nDTW↑	SDTW↑
Speaker-Follower [27]	5.35	51.9	37.3	46.4	-	-	8.47	23.8	12.2	29.6	-	-
RCM [27]	5.37	52.6	30.6	55.3	-	-	8.08	26.1	7.7	34.6	-	-
PTA [31]	**4.53**	**58.0**	**39.0**	**60.0**	**58.0**	**41.0**	8.25	24.0	10.0	37.0	32.0	10.0
EGP [14]	-	-	-	-	-	-	**8.00**	**30.2**	-	44.4	**37.4**	**17.5**
BabyWalk [63]	-	-	-	-	-	-	8.2	27.3	**14.7**	**49.4**	39.6	17.3
IL+RL* [51]	5.94	35.3	32.5	37.1	38.7	26.5	8.88	31.9	18.7	32.3	31.7	12.2
IL+RL+REM	6.72	39.9	36.5	42.4	47.3	31.2	8.83	33.1	20.1	38.6	37.6	15.7
EnvDrop* [51]	5.94	42.7	39.5	40.2	41.8	29.6	9.18	34.7	21.0	37.3	34.7	12.1
EnvDrop+REM	5.83	46.3	43.5	45.1	49.7	33.4	8.21	37.9	25.0	42.3	39.7	18.5
VLN↻Bert* [23]	4.84	55.7	46.0	47.8	55.8	37.9	6.48	42.5	32.4	41.4	41.8	20.9
VLN↻Bert+REM	**3.77**	**66.8**	**57.4**	56.8	**61.5**	**41.5**	**6.21**	**46.0**	**38.1**	44.9	**46.3**	**22.7**

Table 2. Comparison of agent performance on **R4R** in single-run setting. * reproduced results in my environment.

texes in the cross-connected scenes.; it is therefore necessary to align the orientation of the vertexes. More specifically, after the scenes and views are mixed, the direction of ‘⇒’ changes (Fig. 5 from 90° to 150°). Correspondingly, to enable it to go to the next viewpoint, the agent’s action also changes (from ’↰’ to ’→’). However the instruction is still ‘turn left’. To solve this problem of mismatch between action and instruction, we need to fix the position on the cross-connected scenes. To achieve this, as shown in Fig. 4 (stage 3), we move the vertexes v_t^{key1}, v_t^{key2} and their associated vertexes, exchanging the position of the two vertexes, meaning that that the relative positions of the key vertexes remain unchanged. Through fixing the vertexes’ position, the orientation of ‘⇒’ is aligned (see Fig. 5 final view). The agent’s action and instructions accordingly match again.

4.3. Augmentation in Vision Language Navigation

At this point, we have constructed augmented triplets for training: (environment, path, instruction). Our method is able to mix up any two scenes into a new cross-connected scene. We can accordingly generate a large number of new scenes and their corresponding paths and instructions.

For VLN tasks, we need to export cross-connected scenes for training, including the viewpoints, connection relations and vertex positions. The augmented triplets will be merged directly with the original training set, namely $\mathcal{T}_{aug} = \tilde{\mathcal{T}} \cup \mathcal{T}$; we use $\mathcal{T}_{aug}$ in place of $\mathcal{T}$ in training. The observation features in different directions for the cross viewpoint are derived from different scenes.

	Method	NE↓	OR↑	SR↑	SPL↑
Val Seen	Baseline	4.91	62.3	53.8	50.7
	Before OA	4.83	63.1	54.6	53.2
	Before OA + CCV	4.72	64.3	56.8	54.1
	After OA	4.78	63.7	55.6	53.8
	After OA + CCV	**4.61**	**65.6**	**58.2**	**55.3**
Val Unseen	Baseline	5.89	54.5	46.2	42.5
	Before OA	5.92	53.0	46.0	42.4
	Before OA + CCV	5.88	54.8	46.9	42.8
	After OA	5.73	55.2	47.2	43.2
	After OA + CCV	**5.59**	**56.0**	**48.6**	**44.8**

Table 3. Model performance before and after orientation alignment. **Before OA** means before orientation alignment; **After OA** means after orientation alignment; **CCV** means replacement visual observation in construct cross viewpoints.

	Method	NE↓	OR↑	SR↑	SPL↑
Val Seen	0 View	4.78	63.7	55.6	53.8
	1 View	4.70	64.5	56.8	54.6
	2 Views	4.64	65.1	57.2	54.9
	3 Views	**4.61**	**65.6**	**58.2**	**55.3**
	4 Views	4.67	64.0	57.6	55.0
Val Unseen	0 View	5.73	55.2	47.2	43.2
	1 View	5.68	55.9	47.5	43.4
	2 Views	5.63	**56.2**	48.1	44.1
	3 Views	**5.59**	56.0	**48.6**	**44.8**
	4 Views	5.66	55.3	47.9	44.3

Table 4. The impact of replacing the number of different views on the model performance. "0 View" means that visual observation is not replaced

5. Experiment

5.1. Dataset and Evaluation Setup

Dataset and Simulator We evaluate our agent on the Room-to-Room (R2R) [4] and R4R [27] based on Matterport3D simulator [9]. This is a powerful navigation simulator. R4R builds upon R2R and aims to provide an even more challenging setting for embodied navigation agents. In a scene, the agent will jump between pre-defined viewpoints on the connectivity graph of the environment.

Evaluation Metrics There are already many recognized indicators used to evaluate models in VLN: Trajectory Length (TL), trajectory length in meters; Navigation Error (NE), error from the target point in meters; Success Rate (SR), the proportion of times that the agent successfully arrived within 3 meters of the target; and the success rate weighted by the path length (SPL) [2]. In R4R, CLS [27], nDTW and SDTW [25] take into account the agent's steps and are sensitive to intermediate errors in the navigation path.

Implementation Details We use EnvDrop [51] and VLN↻Bert [23] as the baselines to evaluate our method. In the interests of fairness, we use the same experimental settings as the original method. On the basis of not changing the hyperparameter settings, augmented triplets are added for training. We randomly paired and mixed up the 61 scenes in the training set, finally obtaining 116 cross-connected scenes, 5916 paths and 7022 instructions.

5.2. Results on VLN Standard Benchmark

In this section, our method is compared with several other representative methods. We apply the proposed REM to three baseline methods and compare them with other methods. Tab. 1 shows the results on R2R. REM achieves excellent performance on the three baseline methods. In the state-of-the-art method, REM can further improve performance. Tab. 2 shows the results on R4R. Through REM, all three baseline methods have been significantly improved. In addition to the success rate and SPL, REM can also significantly improve CLS and nDTW, which shows that the proposed method can make the agent follow the instructions and make the navigation path more matched.

5.3. Method Analysis

Orientation Alignment In Sec. 4.2, we propose the orientation alignment operation. Tab. 3 shows the performance difference between no orientation alignment (Before OA) and orientation alignment (After OA). Orientation alignment increases the success rate of the baseline by 1%. If the orientation is not aligned, the performance of the model will decrease instead. This is because the agent's actions and instructions do not match, and the agent cannot correctly learn the relationship between instructions and action. In addition, we tested the effect of replacing visual observation (CCV) on the results, and After OA achieved the highest improvement.

Replace Visual Observation In the process of constructing the cross viewpoint, we performed the operation of replacing the visual observation in the specified direction. There are a total of 12 view directions in the horizontal direction. We experimented to determine how many views should be replaced to achieve the best results for the model. Tab.4 outlines the effect of replacing different numbers of views on model performance. As the table shows, three views is the optimal choice. Excessive replacement of visual observation information is thus suboptimal. Through experiments, we choose each cross viewpoint in REM to replace views in three horizontal directions. Fig. 6 shows a schematic diagram of cross viewpoints.

5.4. Ablation Analysis

In order to compare the impact of the number of mixup environments on REM performance, we limited the amount of training data (a data ratio of 1 means 7022 instructions),

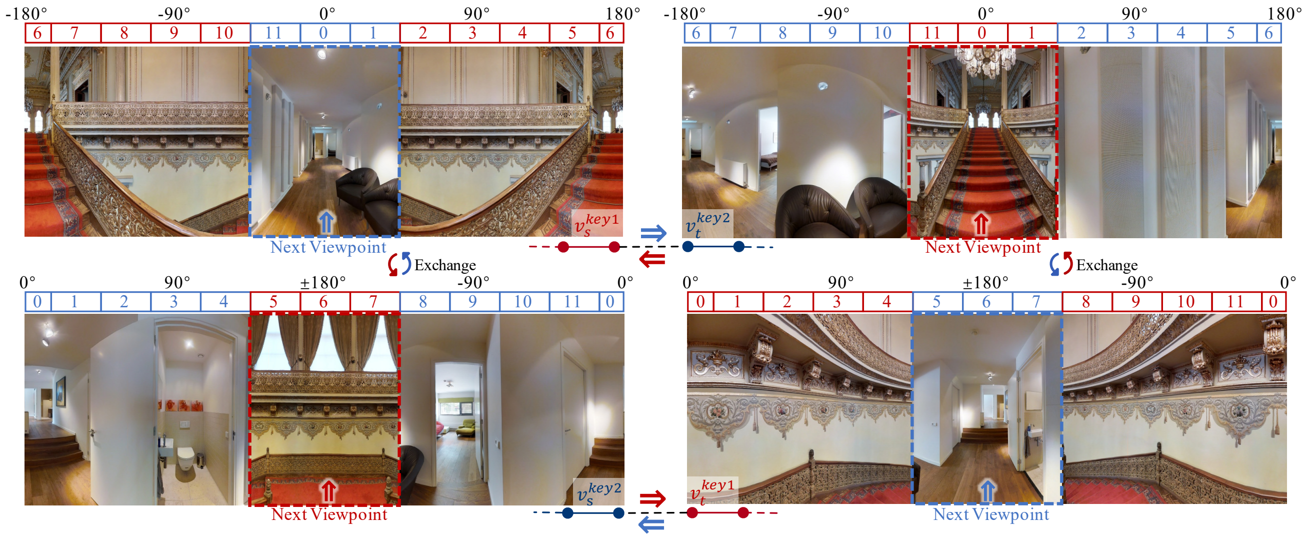

Figure 6. Schematic diagram of cross viewpoints. After the key viewpoints of the two scenes are mixed; the upper and lower two viewpoints' divided views exchange each other; the two viewpoints on the left and right are connected to each other, and the agent can go to each other through '⇒'.

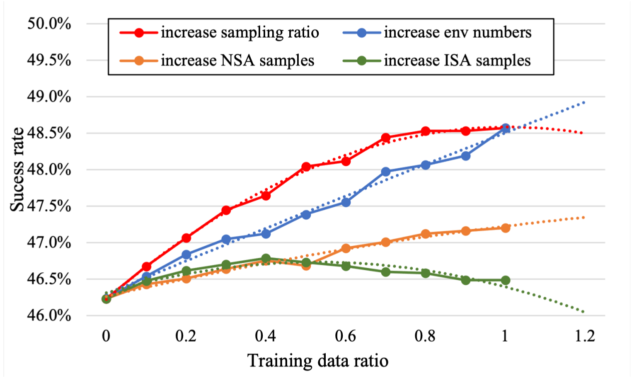

Figure 7. Success rates of agents trained with different amounts of data. The same data ratio in the figure indicates that the same amount of data is used. The **blue** line indicates that the results are increased by gradually adding new environments to the supervised training method. The **red** line only gradually increases the amount of data and randomly selects data from all training environments.

and compared four different settings: 1) as the amount of data increases, the environments number available for mixup also increases in the same proportion; 2) mixup is always used for all environments, but the instructions number generated; 3) NSA is used to generate the same instructions number; 4) ISA is used to generate the same instructions number. The success rate can indicate the generalization ability of different methods. The higher the success rate of the method, the stronger its generalization ability.

The results are shown in Fig. 7. With the increase of sampled data, all methods achieve improvements in model performance to a certain extent. When the data ratio is 1, the red and blue dots have the same setting, the red dot reaches the peak of performance; this means that when the number of mixed scenes is fixed, continuing to increase the sample data cannot further reduce the generalization error. For the blue line, there is no performance degradation trend observed when the data ratio is 1, which shows that increasing the number of mixed scenes can continue to reduce the generalization error. The difference between the red-blue and orange-green lines indicates that, when the sample number is the same, the inter-scene data augmentation is significantly better than the intra-scene data augmentation. This verifies the assumption presented in Sec. 3.2.

6. Conclusion

In this paper, we analyze the factors that affect generalization ability and put forward the assumption that inter-scene data augmentation can more effectively reduce generalization errors. We accordingly propose the Random Environmental Mixup (REM) method, which generates cross-connected house scenes as augmented data via mixuping environment. The experimental results on benchmark datasets demonstrate that REM can significantly reduce the performance gap between seen and unseen environments. Moreover, REM dramatically improves the overall navigation performance. Finally, the ablation analysis verifies our assumption pertaining to the reduction of generalization errors.

Acknowledgments

This work is supported in part by China National 973 program (Grant No. 2014CB340301), Guangdong Outstanding Youth Fund (Grant No. 2021B1515020061), Zhejiang Lab's Open Fund (No. 2020AA3AB14), CSIG Young Fellow Support Fund, and Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA) under DE190100626 and Zhejiang Lab's Open Fund (No. 2020AA3AB14).

References

[1] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Kosecka, and Alexander C. Berg. A dataset for developing and benchmarking active vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1378–1385, 2017. 2

[2] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 2, 7

[3] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2018. 1

[4] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1, 2, 3, 6, 7

[5] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, pages 2425–2433, 2015. 1

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 2

[7] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger N. Gunn, Alexander Hammers, David Alexander Dickie, Maria del C. Valdés Hernández, Joanna M. Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv: Computer Vision and Pattern Recognition*, 2018. 2

[8] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001. 2, 4

[9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017. 2, 3, 7

[10] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. *Advances in neural information processing systems*, pages 416–422, 2001. 3

[11] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*, 2019. 2

[12] Dan Cireşan, Ueli Meier, and Juergen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012. 2

[13] Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289, 2019. 2

[14] Zhiwei Deng, Karthik Narasimhan, and Olga Russakovsky. Evolving graphical planner: Contextual global planning for vision-and-language navigation. *arXiv preprint arXiv:2007.05655*, 2020. 6

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1, 2

[16] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, 2018. 1, 2, 3, 6

[17] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 2017. 1

[18] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *CVPR*, 2020. 1, 2, 6

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[20] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016. 2

[21] Yicong Hong, Cristian Rodriguez, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation. In *NeurIPS*, 2020. 6

[22] Yicong Hong, Cristian Rodriguez-Opazo, Qi Wu, and Stephen Gould. Sub-instruction aware vision-and-language navigation. *arXiv preprint arXiv:2004.02707*, 2020. 5

[23] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. *arXiv preprint arXiv:2011.13922*, 2020. 1, 2, 6, 7

[24] Haoshuo Huang, Vihan Jain, Harsh Mehta, Alexander Ku, Gabriel Magalhaes, Jason Baldridge, and Eugene Ie. Transferable representation learning in vision-and-language navigation. In *ICCV*, 2019. 2

[25] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *ViGIL@NeurIPS*, 2019. 7

[26] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *ICLR 2017 : International Conference on Learning Representations 2017*, 2017. 1

[27] Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1862–1872, 2019. 6, 7

[28] Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *CVPR*, 2019. 2, 6

[29] Sosuke Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 452–457, 2018. 2

[30] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 2

[31] Federico Landi, Lorenzo Baraldi, Marcella Cornia, Massimiliano Corsini, and Rita Cucchiara. Perceive, transform, and act: Multi-modal attention networks for vision-and-language navigation. *arXiv preprint arXiv:1911.12377*, 2019. 6

[32] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *Eighth International Conference on Learning Representations*, 2020. 2

[33] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A Smith, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. In *EMNLP-IJCNLP*, 2019. 2, 6

[34] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019. 1, 2

[35] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances in Neural Information Processing Systems*, pages 289–297, 2016. 1

[36] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019. 2

[37] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Selfmonitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019. 6

[38] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *CVPR*, 2019. 2

[39] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. *arXiv preprint arXiv:2004.14973*, 2020. 2

[40] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013. 1

[41] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016. 1, 2

[42] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems XIV*, volume 14, 2018. 2

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 1

[44] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. In *Robotics: Science and Systems 2017*, volume 13, 2017. 2

[45] Veit Sandfort, Ke Yan, Perry J Pickhardt, and Ronald M Summers. Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Scientific reports*, 9(1):1–9, 2019. 2

[46] Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. Apac: Augmented pattern classification with neural networks. *arXiv preprint arXiv:1505.03229*, 2015. 2

[47] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *CVPR*, 2019. 2

[48] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 1, 2

[49] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, volume 3, pages 958–963, 2003. 2

[50] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *2019 Conference on Empirical Methods in Natural Language Processing*, 2019. 1

[51] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *ACL*, 2019. 2, 3, 6, 7

[52] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. 2

[53] V Vapnik and V Vapnik. Statistical learning theory 156–160, 1998. 3

[54] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1058–1066, 2013. 2

[55] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019. 1, 2, 6

[56] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. *arXiv preprint arXiv:1803.07729*, 2018. 1, 2

[57] Jason W. Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6381–6387, 2019. 2

[58] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. In *ICLR*, 2018. 2

[59] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. 2

[60] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2

[61] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*, 2015. 2

[62] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR*, 2020. 1, 6

[63] Wang Zhu, Hexiang Hu, Jiacheng Chen, Zhiwei Deng, Vihan Jain, Eugene Ie, and Fei Sha. Babywalk: Going farther in vision-and-language navigation by taking baby steps. *arXiv preprint arXiv:2005.04625*, 2020. 6