Supplementary Materials for "Learning Vision-and-Language Navigation from YouTube Videos"

Kunyang Lin[1 2*] Peihao Chen[1*] Diwei Huang[1] Thomas H. Li[6] Mingkui Tan[1 5†] Chuang Gan[3 4]

[1]South China University of Technology, [2]Information Technology R&D Innovation Center of Peking University,

[3]UMass Amherst, [4]MIT-IBM Watson AI Lab, [5]Key Laboratory of Big Data and Intelligent Robot, Ministry of Education,

[6]Peking University Shenzhen Graduate School

In the supplementary, we provide more details of our method. We organize the supplementary as follows.

- In Section A, we present more details on video collection.
- In Section B, we present more details on frame filtering.
- In Section C, we present more details on trajectory generation.
- In Section D, we present more details on instruction generation.
- In Section E, we present statistics and visualization examples of our YouTube-VLN dataset.
- In Section F, we provide more implementation details of experiments.
- In Section G, we provide more details of the effectiveness of trajectory judgment task on layout reasoning ability.
- In Section H, we provide the transferability results of our method.
- In Section I, we present qualitative results of our method.
- In Section J, we discuss potential future research and social impact.

A. More Details on YouTube Video Collection

We collect real estate tour videos from YouTube[1]. Specifically, we restrict the videos to real estate tour videos. It is unrealistic to play all categories of videos one by one to check whether they meet our requirements. On the contrary, we look for several well-known YouTubers whose playlists have been well-categorized for house tour videos, and each list has a consistent style. To find such YouTubers, we only spend less than an hour of mannual search time. In YouTube, each video has its corresponding video id (*e.g.*, $C99YjG\_JBsg$[2]), we obtain all the house tour videos according to the video ids and regard the video ids as house ids.

B. More Details on Frame Filtering

Before generating a navigation trajectory, we first pre-process these videos by sparse sampling and using off-the-shelf image classifiers [6, 7] to filter out redundant frames and noisy frames (*i.e.*, frames with persons or outdoor scenes). A video can be sampled at up to 60 frames per second. Generally, there is almost no obvious change between screens within 2s intervals in a video. Therefore, we sparsely sample the videos with 0.5 frames per second. Considering that in the downstream indoor VLN task, persons and outdoor images are not allowed to occur in the observations, we discard such noisy frames in the house tour videos. Specifically, we employ a Resnet [7] model pre-trained on Place 365 [17] and Mask RCNN [6] model pre-trained on COCO [13] to detect the outdoor images (as shown in Figure Ia) and images with persons (as shown in Figure Ib), respectively. In addition, some images are filtered since they do not contain any objects and can not be extracted to region features (as shown in Figure Ic).

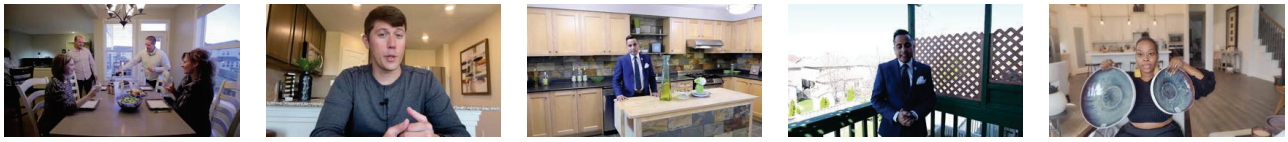
*Equal contribution. Email: {imkunyanglin, phchencs}@gmail.com

†Corresponding author. Email: mingkuitan@scut.edu.cn

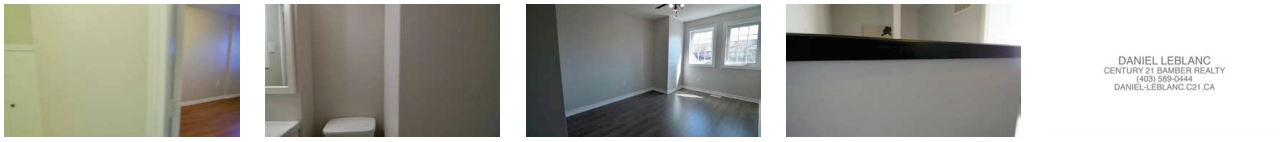
[1]https://www.youtube.com

[2]https://www.youtube.com/watch?v=C99YjG_JBsg

(a) Frames with outdoor scenes.

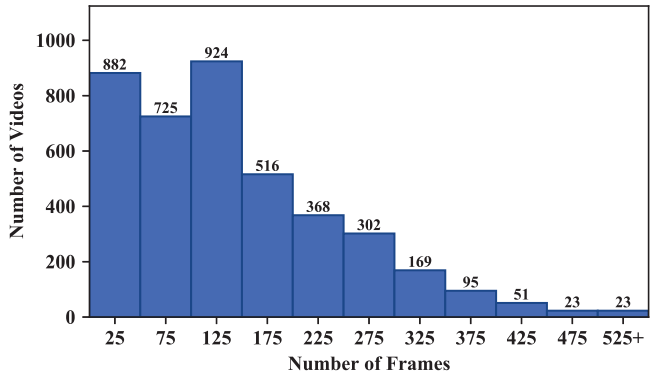


(b) Frames with persons.

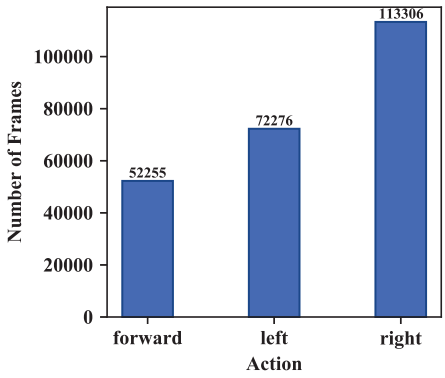


(c) Frames without region features.

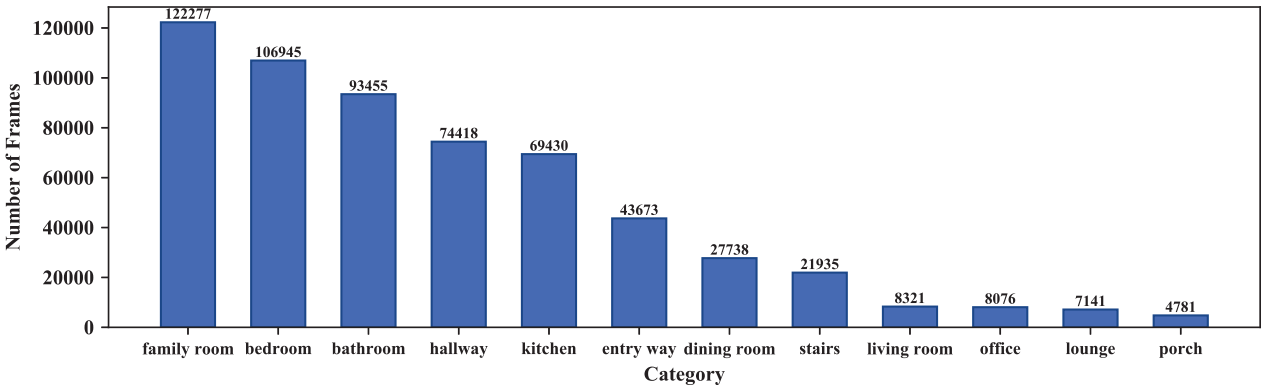
Figure I: Examples of filtered frames, including those with outdoor scenes (a), persons (b) or no region features (c).



(a) Distribution of the number of frames per video.



(b) Distribution of actions.



(c) Distribution of predicted scene categories on YouTube frames.

Figure II: Statistics of YouTube-VLN Dataset.

C. More Details on Trajectory Generation

To mimic the sequential navigation path in R2R, we typically choose $K \in [4, 7]$ as the length of a trajectory. As mentioned in Section 3.1, a trajectory consists of room nodes and transition nodes. We randomly sample $R \in [2, 7]$ room nodes in temporal order for a trajectory. Considering that 1) navigation is a continuous problem in both temporal dimension and spatial dimension, and 2) instruction does not necessarily describe all observations on a trajectory, the remaining $(K - R)$ nodes are filled with transition nodes. Each image is encoded into region features by Faster R-CNN bottom-up top-down attention [1] model pre-trained on Visual Genome [9]. In order to approximate the panoramic visual context, we merge the region features from similar room types per image. The images we merged are from consecutive frames and in the same group, usually taken by the real estate agent around similar locations in order to better introduce the room.

D. More Details on Instruction Generation

As for instruction generation, we first harvest 14,031 fill-in-the-blank templates from the R2R training set. Specifically, we extract the noun phrases and verb phrases for each human-annotated navigation instruction in the R2R training set. We then randomly select a template that has R noun phrase blanks and $(R - 1)$ verb phrase blanks for a trajectory with R room nodes. We use CLIP [16] model to caption the room nodes with a template "[room] with [object]" following [12], where the "[room]" and "[object]" represent the room category and object category of a room node, respectively. To better include the details of the rooms, we fill a noun blank with "[room] with [object]" or "[room]" or "[object]". A CNN inverse action model [3] infers the transition action from one room node to another. Finally, we obtain R captions and $(R-1)$ action words for a template. The captions are used to fill the noun phrase blanks in the template sequentially. For each noun phrase blank filled with the captions of one room node, we find its closest verb phrase blank and fill it with the action which is executed to reach the next room node. Our instruction generation strategy fills the verb phrase blanks with pseudo-labeled actions, providing a natural transition between two nodes to the created instruction.

E. Statistics and Visualizations of YouTube-VLN Dataset

In Figure II, we show some key statistics about our YouTube-VLN dataset. We construct the YouTube-VLN from the collected 4078 videos. After filtering the noisy frames, we harvest 568K images in total. In Figure IIa, we present the number of frames per video via a histogram. It shows that most of the videos contain more than 25 effective frames, indicating that each video can provide sufficient image samples for an agent to learn and reason about this house. Figure IIc presents the predicted room types of the frames. We used CLIP [16] to categorize each frame into one of the 12 labeled room types in Matterport dataset [2]. It can be observed that most of the images in the proposed YouTube-VLN dataset cover the core part of a house (*e.g.*, family room and bedroom). This enables the agent to learn the layout prior knowledge more efficiently. Moreover, these labels are further used for instruction generation and image merging. In addition, we also show the distribution of the pseudo-labeled actions in Figure IIb. Each action is the predicted native action from one room node to another, representing the direction that the agent should follow. As shown in the histogram, the pseudo-labeled actions are evenly distributed into three types of actions, endowing the agent to understand the actions efficiently. We also show some visualization examples of the generated path-instruction pairs in YouTube-VLN as in Figure III, Figure IV, and Figure V.

F. More Implementation Details

The model architecture details are shown in Figure VI. The meanings of each layer are as follows:

- **Embed-Lang**: language token embeddings, which consist of word embeddings, position embeddings, and token type embeddings.
- **Embed-Vis**: vision token embeddings, which consist of visual feature embeddings and position embeddings of the current node.
- **Embed-Node**: node embeddings, which consist of visual feature embeddings, position embeddings, and navigation step embeddings of all nodes in the navigation graph.
- **Self-Att-Lang**: self-attention layers for language input.
- **Self-Att-Vis**: self-attention layers for vision input.

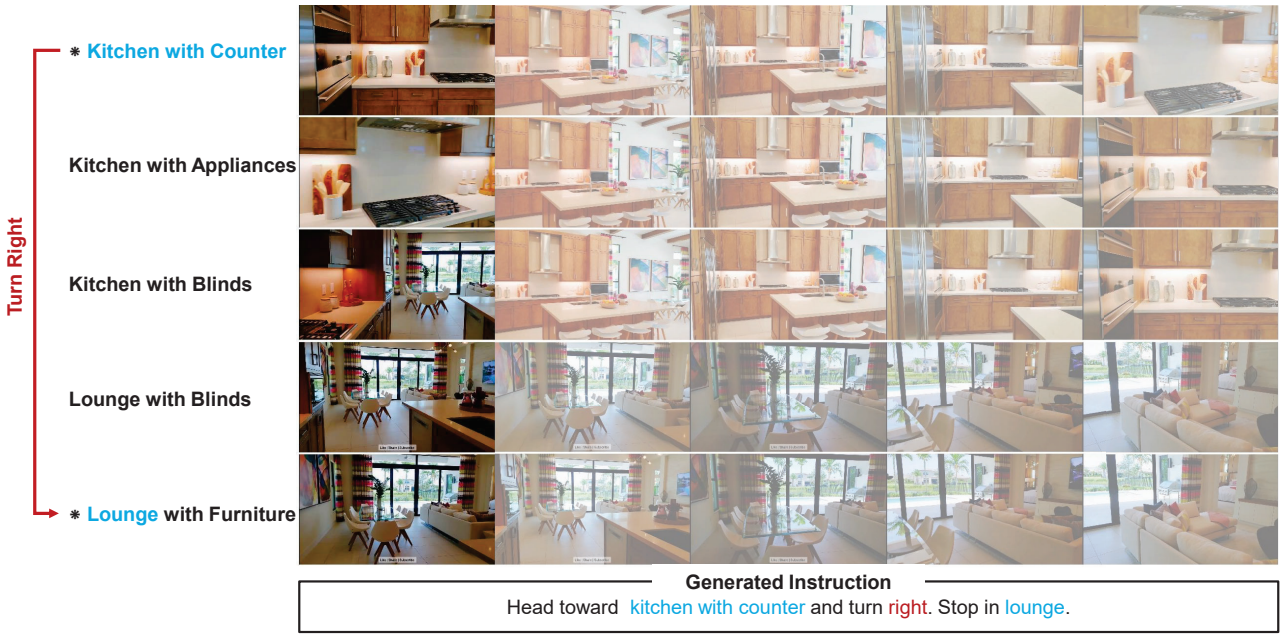


Figure III: Examples of path-instruction pairs in the proposed YouTube-VLN dataset. The generated trajectory contains 2 room nodes (marked as *) and 3 transition nodes. The translucent images are the merged images.

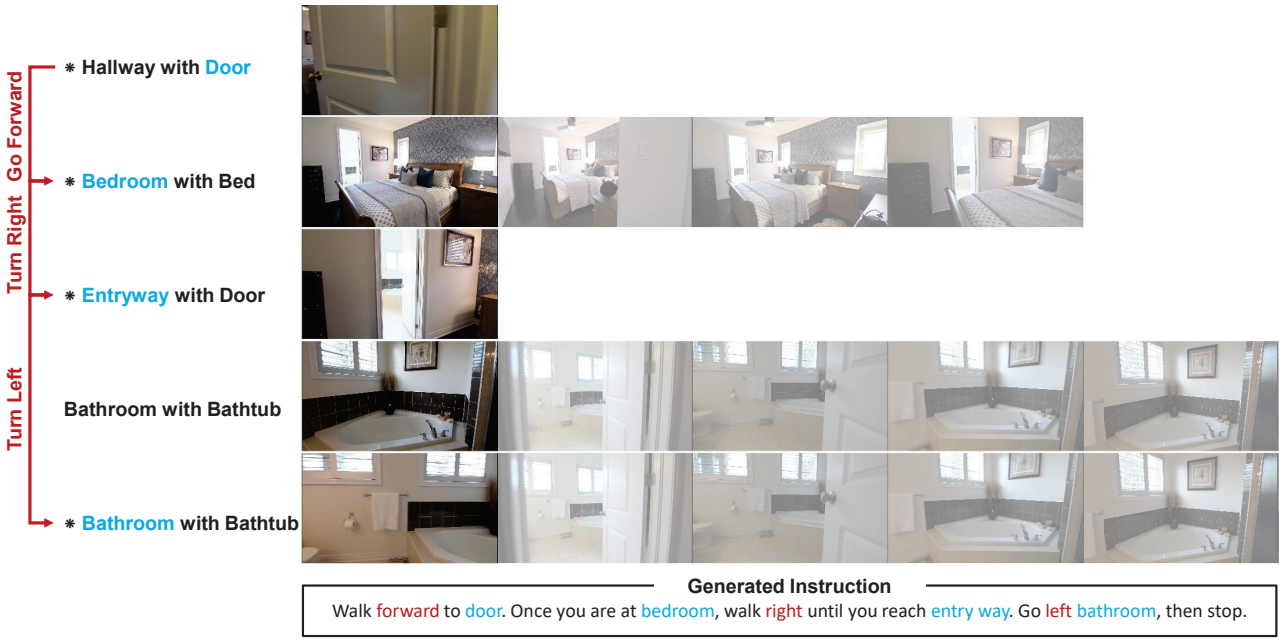


Figure IV: Examples of path-instruction pairs in the proposed YouTube-VLN dataset. The generated trajectory contains 4 room nodes (marked as *) and 1 transition nodes. The translucent images are the merged images.

- **Cross-Att-Vis**: cross-modal attention layers for vision branch.
- **Cross-Att-Lang**: cross-modal attention layers for language branch.
- **FFN**: feed-forward network, which consists of two linear layers.

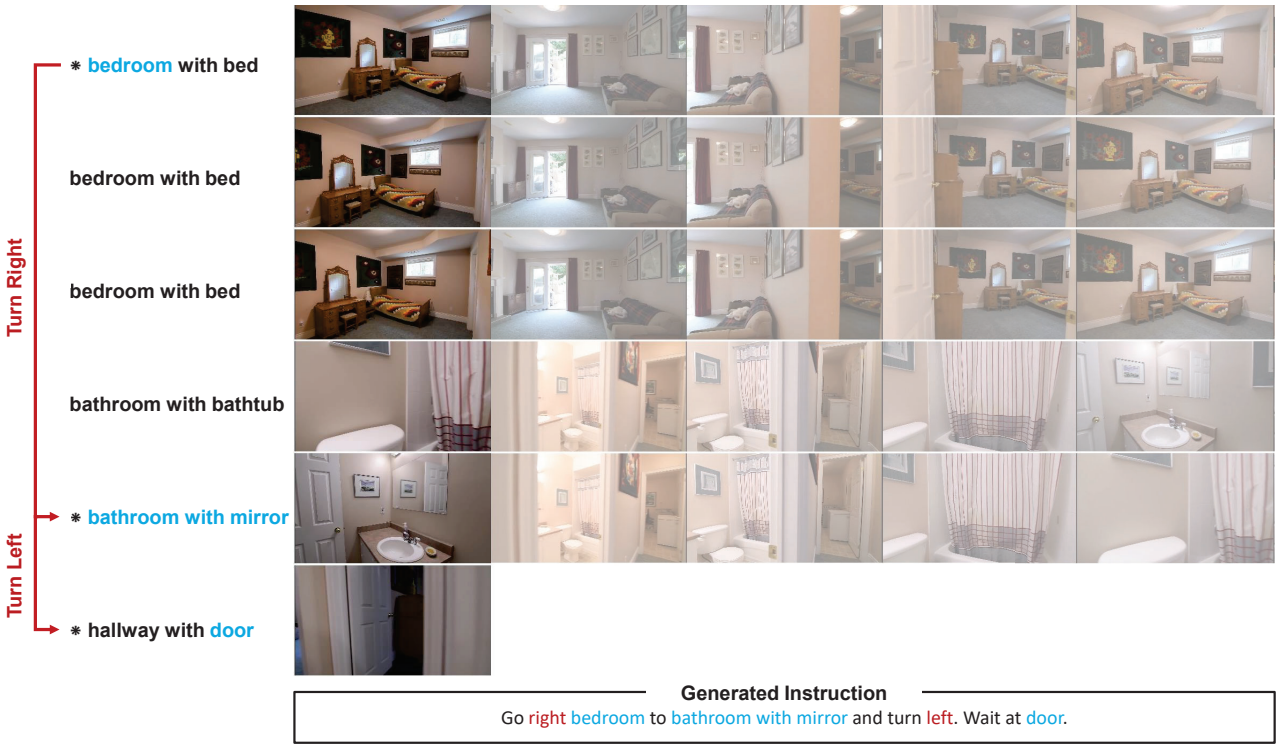


Figure V: Examples of path-instruction pairs in the proposed YouTube-VLN dataset. The generated trajectory contains 3 room nodes (marked as *) and 3 transition nodes. The translucent images are the merged images.

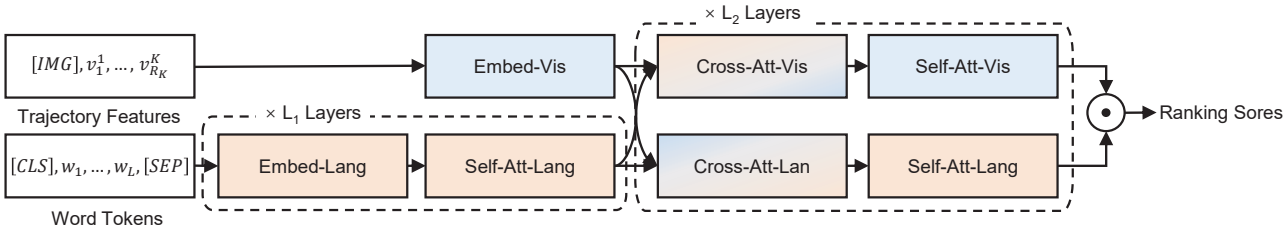
F.1. Pre-training Details

As described in Section 4.1, we adopt a ViLBERT-like architecture as the same as Airbert [5]. The model architecture of pre-training is shown in Figure VIa. For a fair comparison, we set both L_1 and L_2 to 6, consistent with Airbert in the discriminative setting. For generative adaption, the number of layers L_1 is equal to 9 and L_2 is equal to 5 for a fair comparison with DUET [4]. For both settings, we distribute training over 4 NVIDIA 3090 GPUs (24GB each) for 500 epochs to convergence. The batch size is 8 (2 for each GPU) and the learning rate is 2×10^{-5}. We randomly selected 95% videos per epoch as the training set and 5% videos as the test set.

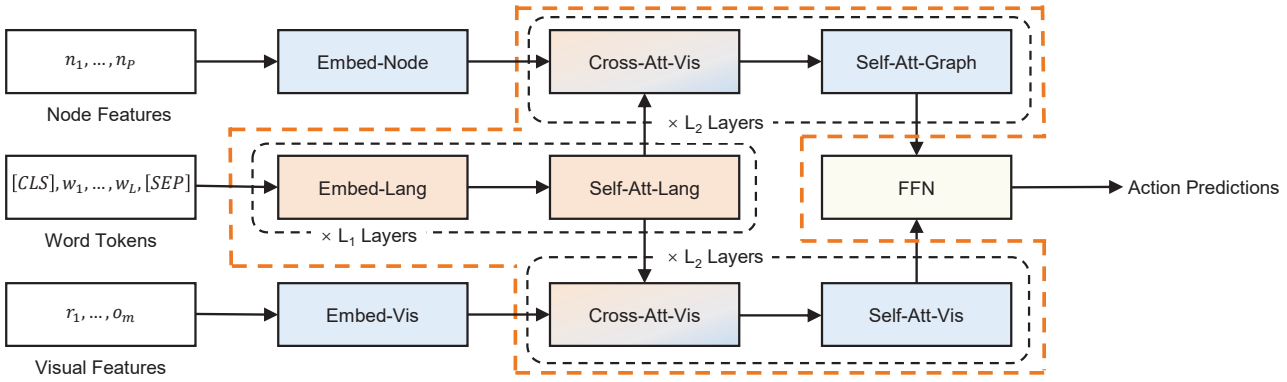
F.2. Fine-tuning Details

Discriminative Setting. In our experiments, discriminative evaluation is conducted on R2R dataset. In this setting, VLN is formulated as a path selection problem. As shown in Figure VIa, the model architecture is the same as the pre-trained model whose classifier used in the path ranking pretext task can be directly for path selection. We follow a two-stage fine-tuning as Airbert, which fine-tunes the agent with MLM task and MVM task in stage one and PR task in stage two. In stage one, the batch size is 12 on 4 VIDIA 3090 GPUs (24GB each) and the learning rate is 4×10^{-5}. In stage two, we set the batch size as 16 on 8 NVIDIA 3090 GPUs (24GB each) and the learning rate as 1×10^{-5}. The agent is fine-tuned for 30 epochs in both stages. The visual features are also encoded by a bottom-up top-down attention [1] model. We select the model checkpoint with the highest success rate on the val unseen validation split for the test set evaluation and leaderboard submission.

Generative Setting. In the generative setting, the agent needs to predict actions sequentially in order to reach the goal (R2R) or find the object (REVERIE). We adopt DUET [4] as the architecture for fine-tuning as it is the state-of-the-art model. As illustrated in Figure VIb, DUET is a three-stream architecture which is fed with P node features $\{n_i\}_{i=1}^{P}$, word tokens $\{[\texttt{CLS}], \{w_i\}_{i=1}^{L}, [\texttt{SEP}]\}$ and the current panorama encoding with image features $\{r_i\}_{i=1}^{n}$ together with object features $\{o_i\}_{i=1}^{m}$. The BERT-like architecture is used to determine which node should the agent go to or what the object goal id is. We initialize the language stream and the cross-modal streams using the corresponding modules in our pre-trained



(a) Adapting Lily agent to the discriminative setting (same as Airbert [5]).



(b) Adapting Lily agent to the generative setting (same as DUET [4]).

Figure VI: The adapted model in both discriminative and generative settings for downstream VLN tasks.

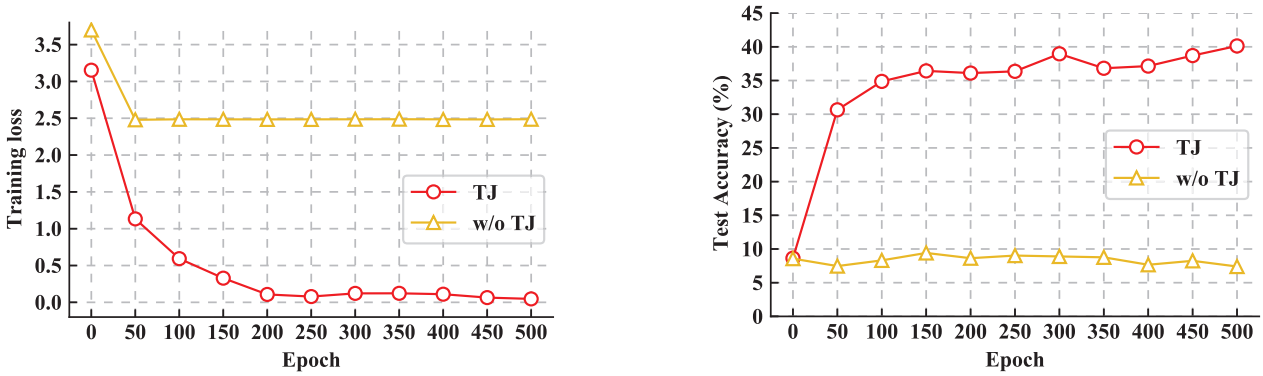


Figure VII: Evolution of training cross-entropy (CE) loss and test accuracy *w.r.t.* training epochs on R2R dataset.

model, highlighted as the orange dotted line in Figure VIb. The other settings remain the same as DUET for a fair comparison.

G. Effectiveness of Trajectory Judgment Task on Layout Reasoning Ability

To evaluate the effectiveness of the proposed trajectory judgment task, we conduct an experiment to evaluate whether the agent trained with this task is able to figure out the direction of an unexplored room based on current observation. Since the common room layout knowledge is required for figuring out this question, the accuracy of this question reveals the layout reasoning ability of the agent.

We conduct this experiment on the R2R [2] dataset, which provides a navigation graph for each environment. Specifically, we randomly initialize an agent on a navigation node. Then, we sample another node from the candidate node list of the current node and use the CLIP [16] model to identify the room type of this node. Given this room type and the panorama visual

feature of the current node as text input and visual input, respectively, the agent is asked to predict the relative orientation of the node for that room type. We define the angle between the matching node of the room type and the current orientation as $s \in [-180°, 180°]$. We then divide $[-180°, 180°]$ uniformly into twelve intervals and compute the interval that s belongs to. This task thus becomes a twelve-category problem and is optimized by minimizing the cross-entropy loss:

$$L_{CE} = -\sum_{i=1}^{12} y_i \ln \hat{y}_i \tag{I}$$

where $\hat{y}_i = \frac{e^{-x_i}}{\sum_{k=1}^{12} e^{-x_k}}$ represents the predicted probability of s belonging to i^{th} interval, x_i represents i^{th} output logit of the model and $y_i \in \{0, 1\}$ indicates whether s belongs to i^{th} interval. We train two agents for 500 epochs, one with the proposed trajectory judgment task and one without it.

As shown in Figure VII, the training cross-entropy loss almost does not decrease without being pre-trained with the trajectory judgment task. This indicates that this variant has not learned the layout reasoning ability at all. Pre-trained with the trajectory judgment task, the agent model drops the training cross-entropy loss rapidly and the test accuracy increases stably. Finally, the highest accuracy of the variant pre-trained with the trajectory judgment task reaches around 40%, while the other variant is 10% (nearly equal to $\frac{1}{12}$). These results verify that the trajectory judgment task facilitates learning layout reasoning ability, achieving substantial improvement.

H. Transferability Results to Other VLN Benchmarks

To better confirm the effectiveness of the our method, we evaluate the transferability of Lily on the other three VLN benchmarks, *i.e.*, RxR [10], R4R [8] and SOON [18]. Specifically, we transfer the model trained on R2R to RxR and R4R and the model trained on REVERIE to SOON without fine-tuning. In Table I, most of the results share the same trend as the results on the R2R and REVERIE dataset, *i.e.*, consistently surpassing the SOTA with large margins on val unseen split. These results demonstrate that our method can generalize well to different domains with varying complexity.

Methods	RxR		R4R		SOON	
	SR	SPL	SR	SPL	SR	SPL
DUET [4]	23.05	18.05	16.01	13.20	2.83	2.09
Lily (ours)	**27.20**	**20.51**	**20.76**	**17.34**	**5.72**	**4.10**

Table I: Transfer results on RxR, R4R and SOON under val unseen split.

I. Qualitative Results

We also present some visualization examples of our Lily agent and the state-of-the-art agent Airbert on the R2R dataset. As shown in Figure VIII, given the instruction that asks the agent to go to a dining room, our Lily agent is able to arrive at the office more quickly than Airbert. We speculate that our Lily agent can be aware of the layout knowledge that an office is usually located in a room on either side of a hallway. Hence, our Lily agent goes straight to the hallway (the red arrow in step 4), while Airbert goes to an entryway connecting the door to the outside(the red arrow in step 4). Our method also improves the understanding of the actions in an instruction. In Figure IX, our Lily agent can easily understand the instruction which asks it to turn left (the red arrow in step 4). However, when pre-trained with incorrect actions, Airbert feels confused about the action words and does not execute the "Turn left" in the instruction, and keeps going forward (the red arrow in step 4), eventually not finding the kitchen.

J. Potential Future Research and Social Impact

Our method is still restricted to graph-based environments. In a real-world application, we may expect the agent actuates the action continuously. This requires us to build more continuous navigation trajectories for the pre-training dataset. Besides, with more powerful vision and language foundation models [11, 15, 14], the models used to construct the proposed dataset can be further improved as more precise and open-world. We can also increase the data diversity by adding richer video and instruction templates.

In the future, agents can be able to actively learn some helpful skills by watching videos like us humans and then assist people with their jobs (*e.g.*, delivering and cleaning), thereby reducing high training costs. However, data security can be an important issue. For some videos that humans keep secret or do not want agents to see, countermeasures should be taken to prevent agents from accessing such videos, otherwise, it may affect human survival one day.

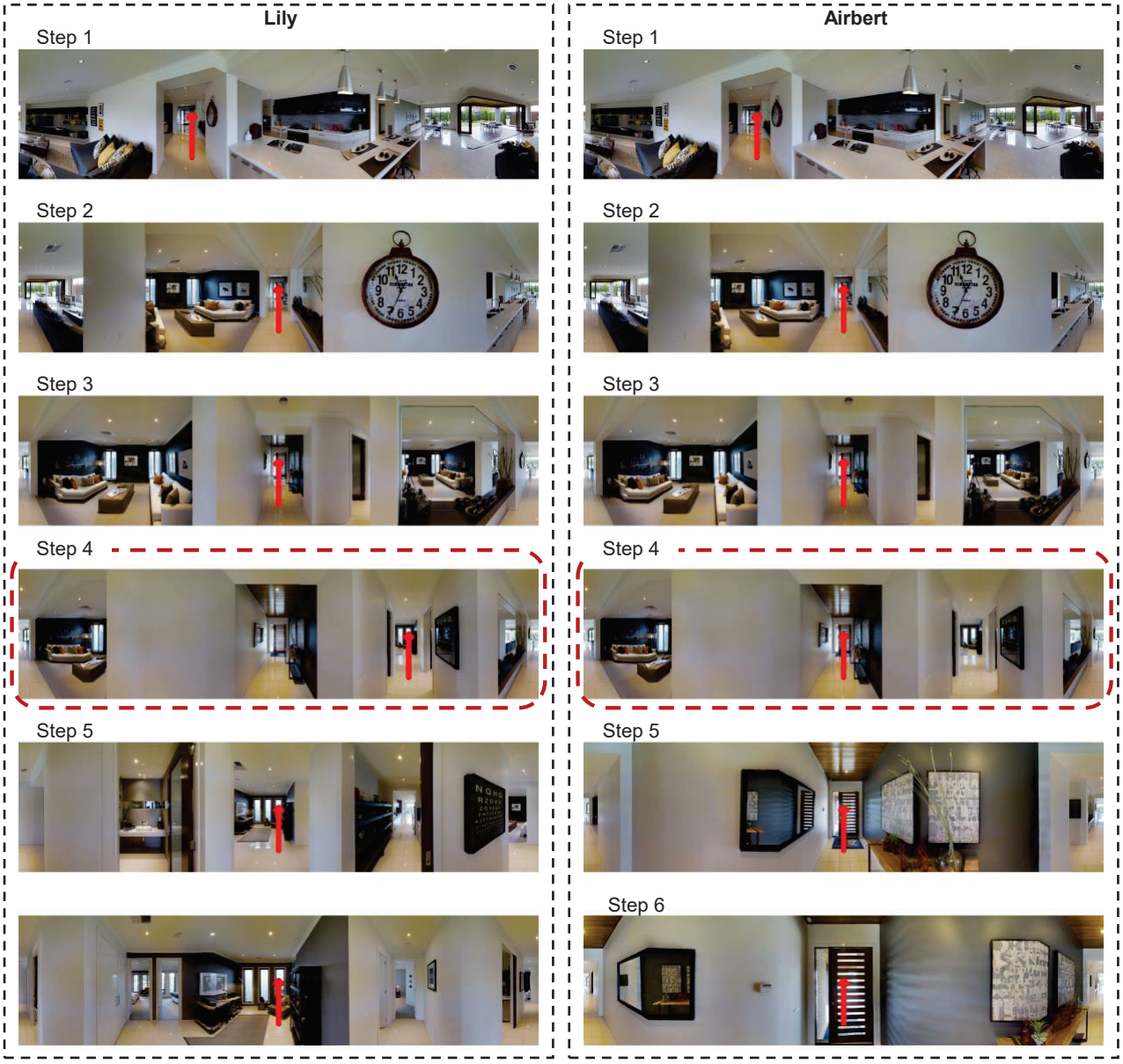


Figure VIII: Visualisation of a trajectory where we compare the performance of our Lily agent with Airbert. The centre of each panorama is the heading direction of the agent at the corresponding time step. Red arrows indicate the predicted actions in each time step. Our Lily agent successfully leverages the layout prior knowledge to find the office.

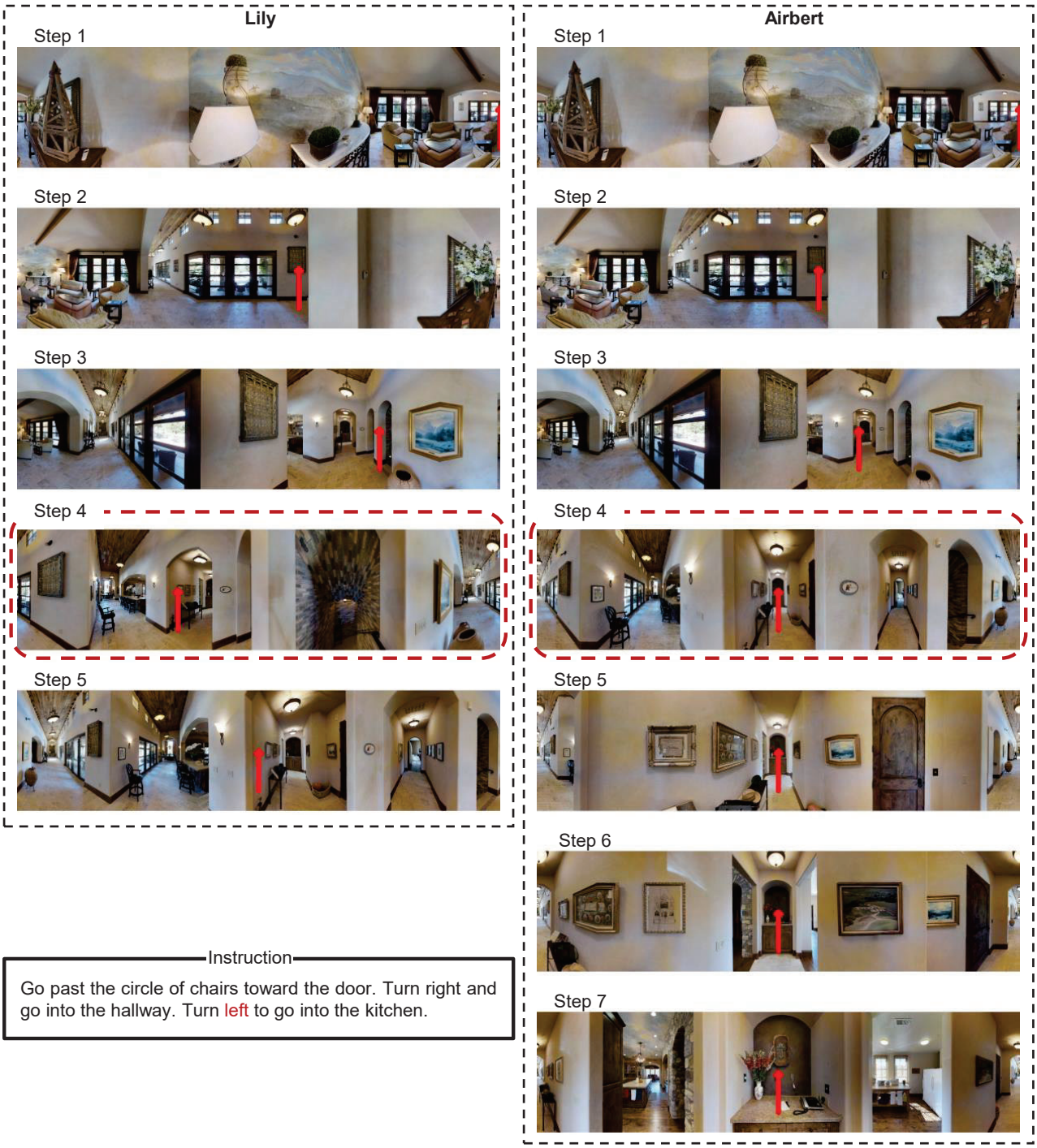


Figure IX: Visualisation of a trajectory where we compare the performance of our Lily agent with Airbert. The centre of each panorama is the heading direction of the agent at the corresponding time step. Red arrows indicate the predicted actions in each time step. Our Lily agent correctly understands the action in the instruction and executes it.

References

[1] 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*, pages 6077–6086, 2018. 3, 5

[2] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, pages 3674–3683, 2018. 3, 6

[3] M. Chang, A. Gupta, and S. Gupta. Semantic visual navigation by watching youtube videos. In *NeurIPS*, pages 4283–4294, 2020. 3

[4] S. Chen, P. Guhur, M. Tapaswi, C. Schmid, and I. Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *CVPR*, pages 16516–16526, 2022. 5, 6, 7

[5] P. Guhur, M. Tapaswi, S. Chen, I. Laptev, and C. Schmid. Airbert: In-domain pretraining for vision-and-language navigation. In *ICCV*, pages 1614–1623, 2021. 5, 6

[6] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017. 1

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[8] V. Jain, G. Magalhães, A. Ku, A. Vaswani, E. Ie, and J. Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. In *ACL*, pages 1862–1872, 2019. 7

[9] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123:32–73, 2017. 3

[10] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *EMNLP*, pages 4392–4412, 2020. 7

[11] J. Li, D. Li, C. Xiong, and S. C. H. Hoi. BLIP: bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, pages 12888–12900. 7

[12] X. Liang, F. Zhu, L. Li, H. Xu, and X. Liang. Visual-language navigation pretraining via prompt-based environmental self-exploration. In *ACL*, pages 4837–4851, 2022. 3

[13] T. 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*, pages 740–755, 2014. 1

[14] S. Ma, Y. Wang, Y. Wei, J. Fan, T. H. Li, H. Liu, and F. Lv. Cat: Localization and identification cascade detection transformer for open-world object detection. In *CVPR*, pages 19681–19690, 2023. 7

[15] OpenAI. GPT-4 technical report. 2023. 7

[16] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021. 3, 6

[17] B. Zhou, À. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *TIPAMI*, 40:1452–1464, 2018. 1

[18] F. Zhu, X. Liang, Y. Zhu, Q. Yu, X. Chang, and X. Liang. SOON: scenario oriented object navigation with graph-based exploration. In *CVPR*, pages 12689–12699, 2021. 7