Appendix:
VLN↻BERT: A Recurrent Vision-and-Language BERT for Navigation

Yicong Hong[1] Qi Wu[2] Yuankai Qi[2] Cristian Rodriguez-Opazo[1,2] Stephen Gould[1]
[1]The Australian National University [2]The University of Adelaide
Australian Centre for Robotic Vision
{yicong.hong, cristian.rodriguez, stephen.gould}@anu.edu.au
qi.wu01@adelaide.edu.au, qykshr@gmail.com

Project URL: https://github.com/YicongHong/Recurrent-VLN-BERT

Appendix A. Datasets

We evaluate the performance of our proposed model on two distinct datasets for VLN:

- Room-to-Room (R2R) [2]: The agent is required to navigate in photo-realistic environments (Matterport3D [3]) to reach a target following low-level natural language instructions. Most of the previous works apply a panoramic action space for navigation [4], where the agent jumps among viewpoints pre-defined on the connectivity graph of the environment. The dataset contains 61 scenes for training; 11 and 18 scenes for validation and testing, respectively, in unseen environments.
- REVERIE [13]: The agent needs to first navigate to a point where the target object is visible, then, it needs to identify the target object from a list of given candidates. In REVERIE, the navigational instructions are high-level while the instructions for object grounding are very specific. The dataset has in total 4,140 target objects in 489 categories, and each target viewpoint has 7 objects with 50 bounding boxes in average.

Appendix B. Implementation Details

We provide the implementation details of preparing visual features (§3.3[1]), decision making (§3.3), adaptation to PREVALENT [5] (§3.3 & 3.5), adaptation to REVERIE [13] (§3.3 & 3.5), critic function and reward shaping in reinforcement learning (§3.4).

B.1. Visual Features (§3.3[1])

Navigation in R2R [2] and REVERIE [13] are conducted in the Matterport3D Simulator [3]. At each navigable viewpoint in the environment, the agent observes a 360° panorama, consisting of 36 single-view images at 12 headings (30° separation) and 3 elevation angles ($\pm 30^\circ$).

Scene Features In our experiments, we only consider the scene features (grid features of the single-view images provided by ResNet-152 [6] pre-trained on Places365 [18]) at the navigable directions as visual features to VLN↻BERT. Each visual feature is direction-aware, which is formulated by the concatenation of the convolutional image feature $\boldsymbol{f}_t^{v,i} \in \mathbb{R}^{2048}$ and the directional encoding $\boldsymbol{d}_t^i \in \mathbb{R}^{128}$ as:

$$\boldsymbol{I}_t^{v,i} = \left[\boldsymbol{f}_t^{v,i}; \boldsymbol{d}_t^i\right] \tag{1}$$

The directional encoding is formed by replicating vector $(\cos\theta_t^i, \sin\theta_t^i, \cos\phi_t^i, \sin\phi_t^i)$ by 32 times, where θ_t^i and ϕ_t^i represents the heading and elevation angles of the image with respect to the agent's orientation [4, 16]. Moreover, the action features $\boldsymbol{a}_t$ which is fed to the state representation is exactly the directional encoding at the selected direction $\boldsymbol{d}_t^c$, where $c = a_t^s$.

Object Features In REVERIE [13], the target objects can appear at any single-view of the panorama, we extract the object features (regional features encoded by Faster-RCNN [14] pre-trained on Visual Genome [10] by Anderson *et al*. [1]) according to the positions of objects provided in REVERIE [13]. The object features are position- and direction-aware, formulated as

$$\boldsymbol{I}_t^{o,k} = \left[\boldsymbol{f}_t^{o,k}; \boldsymbol{p}_t^k \boldsymbol{W}^p; \boldsymbol{d}_t^k\right] \tag{2}$$

where $\boldsymbol{f}_t^{o,k} \in \mathbb{R}^{2048}$ is the convolutional object features, $\boldsymbol{d}_t^k \in \mathbb{R}^{128}$ is the directional encoding of the single-view

[1]Link to Section 3.3 in Main Paper.

which contains the object. $\boldsymbol{p}_t^k$ represents the spatial position of the object within the image, as in MATTN [17], we apply $\boldsymbol{p}_t^k = \left[\frac{x_{tl}}{W}, \frac{y_{tl}}{H}, \frac{x_{br}}{W}, \frac{y_{br}}{H}, \frac{w \cdot h}{W \cdot H}\right]$, where (x_{tl}, y_{tl}) and (x_{br}, y_{br}) are the top-left and bottom-right coordinates of the object, (w, h) and (W, H) are the width and height of the object and the image, respectively. The matrix $\boldsymbol{W}^p \in \mathbb{R}^{5\times128}$ is a learnable linear projection.

B.2. Decision Making (§3.3)

In R2R [2], there are two types of decisions that an agent infers during navigation; it either selects a navigable direction to move, or it decides to `stop` at the current viewpoint. As in most of the previous work, stopping in VLN↻BERT is implemented by adding a zero vector $\boldsymbol{v}_t^{\text{stop}}$ to the list of visual features at navigable directions [4, 7, 11, 16], as

$$\boldsymbol{V}_t = \langle \boldsymbol{v}_t^1, \boldsymbol{v}_t^2, ..., \boldsymbol{v}_t^n, \boldsymbol{v}_t^{\text{stop}} \mid \boldsymbol{v}_t^i \in \mathbb{R}^{2176} \rangle \tag{3}$$

Our VLN↻BERT determines to stop by predicting the largest attention score to the `stop` representation at the final transformer layer. Otherwise, the agent will move to a navigable direction with the largest score.

However, in REVERIE [13], we directly apply the attention scores over the candidate objects for stopping. To be specific, the visual tokens in REVERIE consists of sequences of scene features and object features:

$$\langle \boldsymbol{v}_t^1, \boldsymbol{v}_t^2, ..., \boldsymbol{v}_t^n, \boldsymbol{o}_t^1, \boldsymbol{o}_t^2, ..., \boldsymbol{o}_t^m \mid \boldsymbol{v}_t^i \in \boldsymbol{V}_t, \boldsymbol{o}_t^k \in \boldsymbol{O}_t \rangle \tag{4}$$

When the model predicts larger attention scores for at least one of the object token than all of the scene tokens, the agent will stop and will select the object with the largest score as the grounded object for REF. Such formulation has two advantages; First, it relates the object searching process to navigation, *i.e.*, the agent should not stop navigation if it has low confidence of localising the target object. Second, it allows the reinforcement learning to benefit the object grounding, since the action logits for `stop` is the greatest attention scores over objects.

B.3. Adaptation to PREVALENT (§3.3 & §3.5)

As shown in Fig. 1, we adapt our VLN↻BERT to the LXMERT-like [15] architecture in PREVALENT [5]. At initialisation, the transformer *TRM-Lang1* encodes the instruction $\boldsymbol{U}$ and uses the output features of the `[CLS]` token to represent agent's initial state. During navigation, the concatenated sequence of the previous state $\boldsymbol{s}_{t-1}$, the encoded language from initialisation $\boldsymbol{X}$ and the new visual observation $\boldsymbol{V}_t$ will be fed to the *cross-modality encoder* to obtain the language-aware state feature $\boldsymbol{s}_{t-1}^L$ and the language-aware visual features $\boldsymbol{V}_t^L$. Finally, *TRM-Vis2* will process $\boldsymbol{s}_{t-1}^L$ and $\boldsymbol{V}_t^L$ to produce a new state $\boldsymbol{s}_t$ and a decision $\boldsymbol{p}_t$.

Pre-training of PREVALENT [5] applies the outputs from *TRM-Lang3* for attended masked language modelling and action prediction, but fine-tuning on R2R [2] only use the output language features from *TRM-Lang1* and rely on a downstream network EnvDrop [16] for navigation. In contrast, our method does not require any downstream network, we leverage the visual transformers *TRM-Vis1* and *TRM-Vis2* to learn state-language-vision relationship for better decision marking.

Cross-Modal Matching The *cross-modal matching* for *State Refinement* (see Eq. 12 in §3.3) is applied to enhance the state representation of PREVALENT-based VLN↻BERT. Since the final transformer *TRM-Vis2* only process the state and visual features, we apply the averaged attention scores over the visual features with respect to $\boldsymbol{s}_t$ to weight the input visual tokens $\boldsymbol{V}_t$, but apply the averaged attention scores over the textual features with respect to $\boldsymbol{s}_{t-1}^L$ to weight the input language tokens $\boldsymbol{X}$. Results in Table 1 show that cross-modal matching for state improves the agent's performance in unseen environments.

Models	R2R Validation Seen				R2R Validation Unseen			
	TL	NE↓	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑
w/o Matching	10.87	**2.44**	**76.79**	**73.13**	11.72	4.08	62.07	56.15
with Matching	11.13	2.90	72.18	67.72	12.01	**3.93**	**62.75**	**56.84**

Table 1. Performance of PREVALENT-based VLN↻BERT with and without cross-modal matching for agent's state.

B.4. Critic Function (§3.4)

We apply the A2C [12] for reinforcement learning. At each time step, the critic, a multi-layer perceptron, predicts an expected value from the updated state representation as:

$$e_t = (\text{ReLU}(\boldsymbol{s}_t \boldsymbol{W}^{E1}))\boldsymbol{W}^{E2} \tag{5}$$

where ReLU is the Rectified Linear Unit function, $\boldsymbol{W}^{E1}$ and $\boldsymbol{W}^{E2}$ are learnable linear projections.

B.5. Reward Shaping (§3.4)

In addition to the progress rewards defined in EnvDrop [16], we apply the normalised dynamic time warping [9] as a part of the reward to encourage the agent to follow the instruction to navigate. Moreover, we introduce a negative reward to penalise the agent if it misses the target.

Progress Reward As formulated in the EnvDrop [16], we apply the progress rewards as strong supervision signals for directing the agent to approach the target. To be specific, let D_t to be the distance from agent to target at time step t, and $\Delta D_t = D_t - D_{t-1}$ to be the change of distance by action a_t, the reward at each step ($a_t \neq$ `stop`) is defined as:

$$r_t^{D,step} = \begin{cases} +1.0, & \Delta D_t > 0.0 \\ -1.0, & \text{otherwise} \end{cases} \tag{6}$$

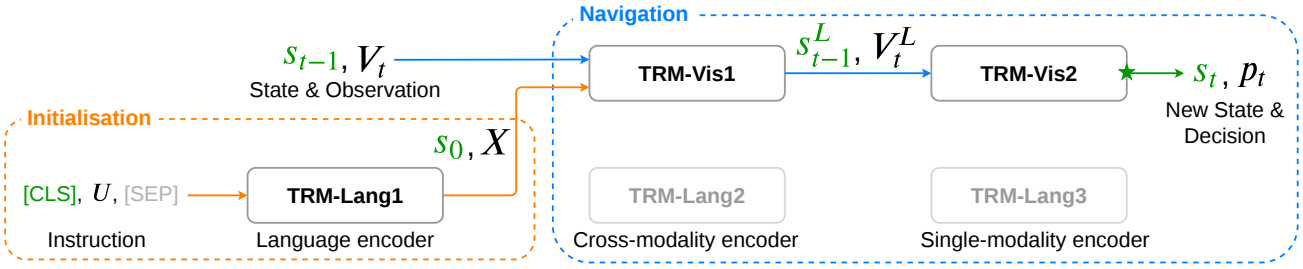

Figure 1. Adaptation to recurrent PREVALENT. At initialisation, the entire instruction is encoded by a language transformer (*TRM-Lang1*), where the output feature of the `[CLS]` token servers as the initial state representation of the agent. During navigation, the concatenated sequence of state, encoded language and new visual observation are fed to the cross-modality and the single-modality encoders to obtain the updated state and decision probabilities. The updated state and the language encoding from initialisation will be fused and applied as input at the next time step. The green star (★) indicates the *cross-modal matching* and the *past decision encoding* (§3.3).

When the agent decides to stop (a_t=`stop`), a final reward is assigned depending on if the agent successfully complete the task:

$$r_t^{D,final} = \begin{cases} +2.0, & D_t < 3.0 \\ -2.0, & \text{otherwise} \end{cases} \tag{7}$$

Overall, the agent will receive a positive reward if it approaches the target and completes the task (by stopping within 3 meters to the target viewpoint), while it will be penalised with a negative reward if it moves away from the target or stops at a wrong viewpoint.

Path Fidelity Rewards The *Progress Reward* encourages the agent to approach the target, but it does not constrain the agent to take the shortest path. As there could be multiple routes to the target, the agent could learn to take a longer path or even a cyclic path to maximise the total reward. To address the problem, we apply the normalised dynamic time warping reward [9], a measurement of the similarity between the ground-truth path and the predicted path, to urge the agent to follow the instruction accurately. Let P_t be the normalised dynamic time warping [9] at time t, and $\Delta P_t = P_t - P_{t-1}$ to be the change of P caused by action a_t, the reward for $a_t \neq$ `stop` is defined as:

$$r_t^{P,step} = \Delta P_t \tag{8}$$

and the reward for a_t=`stop`:

$$r_t^{P,final} = \begin{cases} +2.0 P_t, & D_t < 3.0 \\ +0.0, & \text{otherwise} \end{cases} \tag{9}$$

Moreover, as suggested by many previous works, there exists a large discrepancy between the agent's oracle success rate and success rate, indicating that it does not learn well to stop accurately. To address this issue, we introduce a negative stopping reward r_t^S which will be triggered whenever the agent first approaches the target but then departs from it. To be precise, if $D_{t-1} \leqslant 1.0$ and $\Delta D_t > 0.0$:

$$r_t^S = -2.0 \times (1.0 - D_{t-1}) \tag{10}$$

The normalised dynamic time warping reward r_t^P and the stopping reward r_t^S together form the path fidelity rewards which can encourage the agent to navigate efficiently. In summary, the overall reward at each step during navigation can be expressed as:

$$r_t = \begin{cases} r_t^{D,step} + r_t^{P,step} + r_t^S, & a_t \neq \texttt{stop} \\ r_t^{D,final} + r_t^{P,final}, & a_t = \texttt{stop} \end{cases} \tag{11}$$

Ablation Study We perform ablation experiments on training the OSCAR-based and the PREVALENT-based VLN↻BERT with and without the path fidelity rewards. As shown in Table 2, the models trained with the path fidelity rewards achieve higher Success Rate (SR) and lower Trajectory Length (TL), leading to higher Success weighted by Path Length (SPL). Despite the improvements in SR, the gap between the Oracle Success Rate (OSR) and SR is kept roughly the same for the PREVALENT-based model and is reduced by about 1.45% the OSCAR-based models. These results suggest that the path-fidelity rewards benefit the agent to navigate more accurate and efficient. Note that, comparing to the results that are shown in Table 1 and Table 3 of our Main Paper, such reward shaping only contributes to a slight gain of the improvement, whereas the structure of the recurrent BERT is much more influential.

Models	R2R Validation Seen				R2R Validation Unseen			
	TL	OSR	SR↑	SPL↑	TL	OSR	SR↑	SPL↑
OSCAR								
r^D only	11.15	77.86	70.71	66.64	12.62	66.92	58.32	52.48
$r^D + r^P + r^S$	10.79	76.79	**71.11**	**67.23**	11.86	65.86	**58.71**	**53.41**
PREVALENT								
r^D only	12.24	77.28	69.93	64.46	12.89	69.52	62.15	55.65
$r^D + r^P + r^S$	11.13	78.16	**72.18**	**67.72**	12.01	70.24	**62.75**	**56.84**

Table 2. Performance of OSCAR-based and PREVALENT-based VLN↻BERT trained with and without the path fidelity rewards.

Appendix C. Language Attention

C.1. Additional Explanation of the Averaged Language Attention Weights (§4.1)

In Figure 3 of the Main Paper, we visualised the averaged attention weights over all instructions in validation unseen split during navigation. Notice that in this visualisation, we interpolate the instruction to 80 words and the trajectory to 15 steps for each sample to compute the averaged attention weights. Since a large portion of instructions in R2R are less than 80 words, the last few tokens in the plot are corresponding to the full stop punctuation and the [SEP] token, which are less likely to provide valuable information to the state and hence receive very little weight. However, for the last step in the bottom plot, the selected visual token for stopping could have a high correspondence to the full stop and [SEP] which are strong indications of the end of navigation.

C.2. Self-Attended Language Features with Gradient Accumulation (§4.2)

To evaluate the influence of performing different language self-attention under the same batch size, we train *Emb-Attn*, *Init-Attn* and *Re-Attn* with gradient accumulation to achieve batch size of 16 (same as *Ours*) while keeping learning rate unchanged. Comparing to the results reported in Table 4 of the Main Paper, the performance of all the three methods has been significantly improved. On the validation unseen split, *Emb-Attn* even outperforms *Ours* which does not re-attend the language at each time step. However, a downside for accumulating gradient is that the training speed is significantly reduced, which is about 2.5 times slower for accumulating 4 batches of size 4. Nevertheless, this result suggests that agent's performance can be further improved by training with larger batch size, potentially including *Ours*. Therefore, whether it is necessary to perform language self-attention at each navigational step when there is sufficient computational power remains a question worth investigating. We will leave it as a future work.

Models	R2R Validation Seen				R2R Validation Unseen				Accu-Grad	Speed
	TL	NE↓	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑		
Emb-Attn	11.36	**2.97**	70.91	66.26	12.22	**4.15**	**61.00**	**54.85**	✓	1.43
Init-Attn	12.11	3.78	64.84	59.49	12.32	4.37	58.62	52.53	✓	1.33
Re-Attn	9.71	4.44	54.16	51.51	10.19	5.28	47.89	44.64	✓	1.38
Ours	10.79	3.11	**71.11**	**67.23**	11.86	4.29	58.71	53.41	✗	**3.40**

Table 3. Comparison of performing language self-attention with gradient accumulation. We set the batch size of the first three methods as 4 and accumulate the gradient for 4 iterations, so that the overall batch size for each update is 16, same as *Outs*. Unit for *Speed* is 1,000 iterations per hour (including the time spent on evaluation).

Appendix D. Visualisation (§4.1)

As shown in Figure 2, 3 and 4, we visualise the language-to-language and the state/vision-to-state/vision/language attention weights of a sample in the validation unseen split. As shown by the panoramas in Fig. 3 and the given instruction "*Exit the bedroom. Walk the opposite way of the picture hanging on the wall through the kitchen. Turn right at the long white countertop. Stop when you get past the two chairs.*", the agent needs to understand the complex contextual clues, including different scene clues (*bedroom*, *kitchen*), different object clues (*picture*, *wall*, *countertop*, *chair*) and various directional clues (*forward*, *opposite*, *left/right*, *stop*) to complete the task.

Language Self-Attention Fig. 2 shows the language self-attention attention weights at some selected heads at initialisation (t=0); different heads demonstrate different functions and the attentions at different layers behave very differently. Plot (1) and (4) show the general pattern of attentions at shallow (Layer 0) and deep layers (Layer 8); words in shallow layers tend to collect information from the entire sentence, while words at deep layers have higher correspondence to the adjacent words since they are semantically more relevant. It is very interesting to see that the attention head in Plot (2) learns to attend the adjectives and the action-related terms, which describe the objects and scenes, for the initialised state representation (`[CLS]`). This head also learns about the co-occurrence between different entities, for example, *picture* has higher correspondence to *wall*, and *countertop* has higher correspondence to *kitchen*. In contrast, the attention head in Plot (3) learns to extract the important landmarks such as *bedroom*, *picture*, *kitchen* and *chairs*. Attention head in Plot (5) learns about the `[SEP]` token, which indicates the ending of the instruction. Attention weights in Plot (6) show the most frequent pattern of the attention heads at the final (l=12-th) layer, those heads seem to aggregate information from the punctuations in the instruction. This implies the heads could have learnt about breaking the sentence into multiple sub-sentences. Refer to the idea of sub-instruction proposed by Hong *et al*. [8], such attention pattern could be beneficial for matching the current observation to a particular and the most relevant part of the instruction.

State/Vision Step-wise Attention Fig. 3 shows the trajectory of the agent starting from a bedroom, taking a series of actions and eventually stopping at the target location. It also displays the averaged attention weights at the final (l=12-th) layer for the state and visual tokens. Note that the averaged attention for state/vision tokens is representative since we apply the averaged attention weights for *visual-textual matching* to state and use it as the action prob-

abilities (see Eq. 12 and *Decision Making* in §3.3, respectively). As shown in Plot (1a-6a), the attention shifts from the beginning of the instruction to the end, which agrees with the agent's navigation progress. It is also interesting to see that at the final layer, the state token is more influential to the predicted action at the first two steps, while the language tokens are more influential at the later steps. The state/vision self-attention in Plot (1b-6b) reflects the action prediction at each time step. The first row in each plot (attention of state with respect to candidate actions) shows the prediction result, we can see that the agent is very confident in each decisions. Moreover, starting from Step 3, the information from state and from different views are aggregated to support choosing the correct direction.

State/Vision Layer-wise Attention To better understand how the visual and language features are aggregated to support action prediction, we visualise the layer-wise attention at Step 4 of the trajectory (t=4). As shown by Fig. 4 Plot (1-12), at the first two layers, candidate views collect information from the entire instruction. But as the signals propagate to deeper layers (Layer 3-6), the visual tokens attend more the middle part of the instruction, which should be more relevant to the current observations. Interestingly, starting from Layer 6, the visual features tend to dominate the attention, and information aggregates toward the visual token at the predicted direction (which is the correct direction). We can see that, at Layer 7-9, the network still has some doubts about the candidate directions that are spatially closer to the correct direction. But after implicitly reasoning in deeper layers, the network becomes very confident about choosing the correct direction.

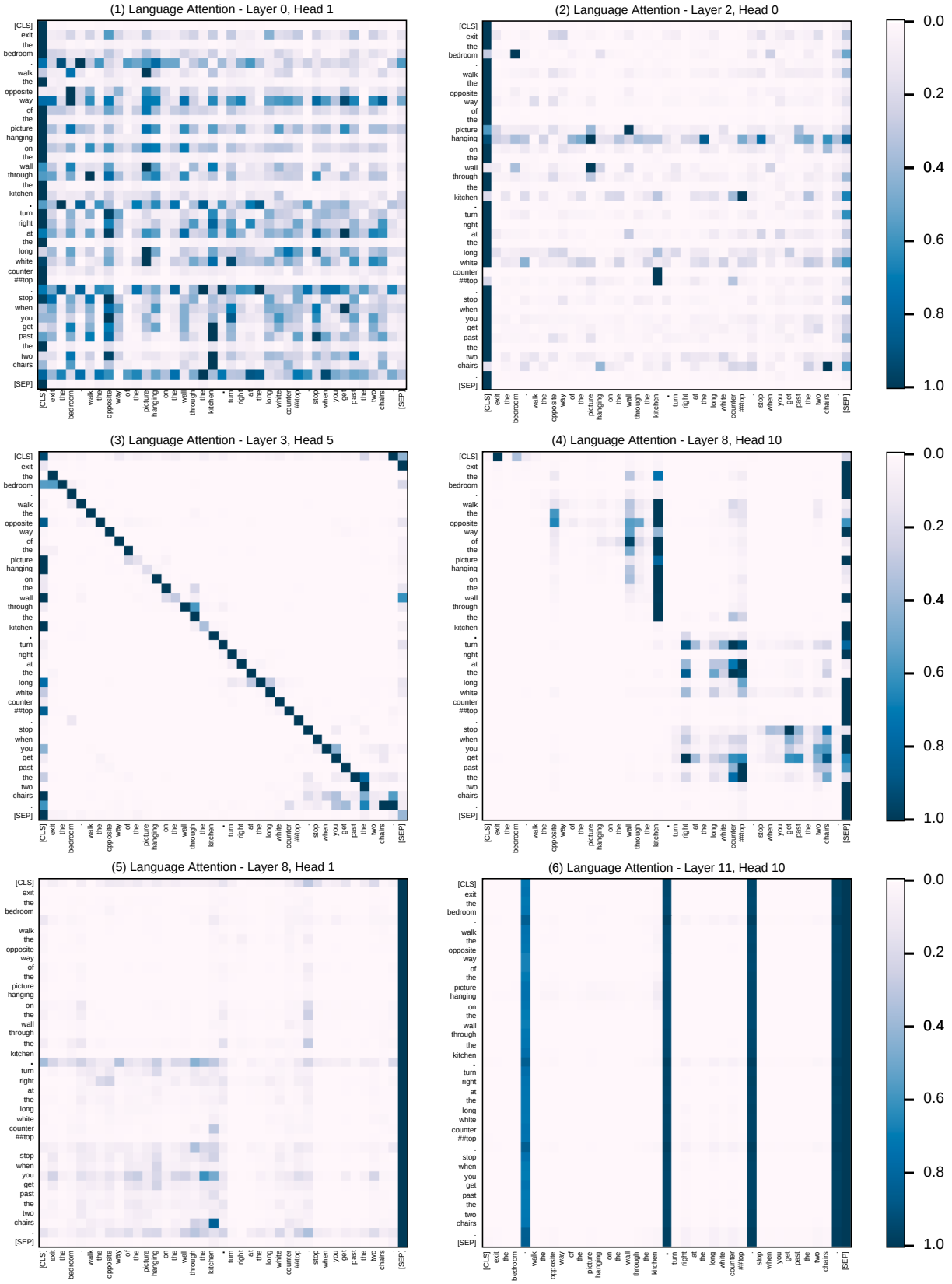

Figure 2. Language self-attention weights of some selected heads at initialisation. The attention weights are normalised for each row.

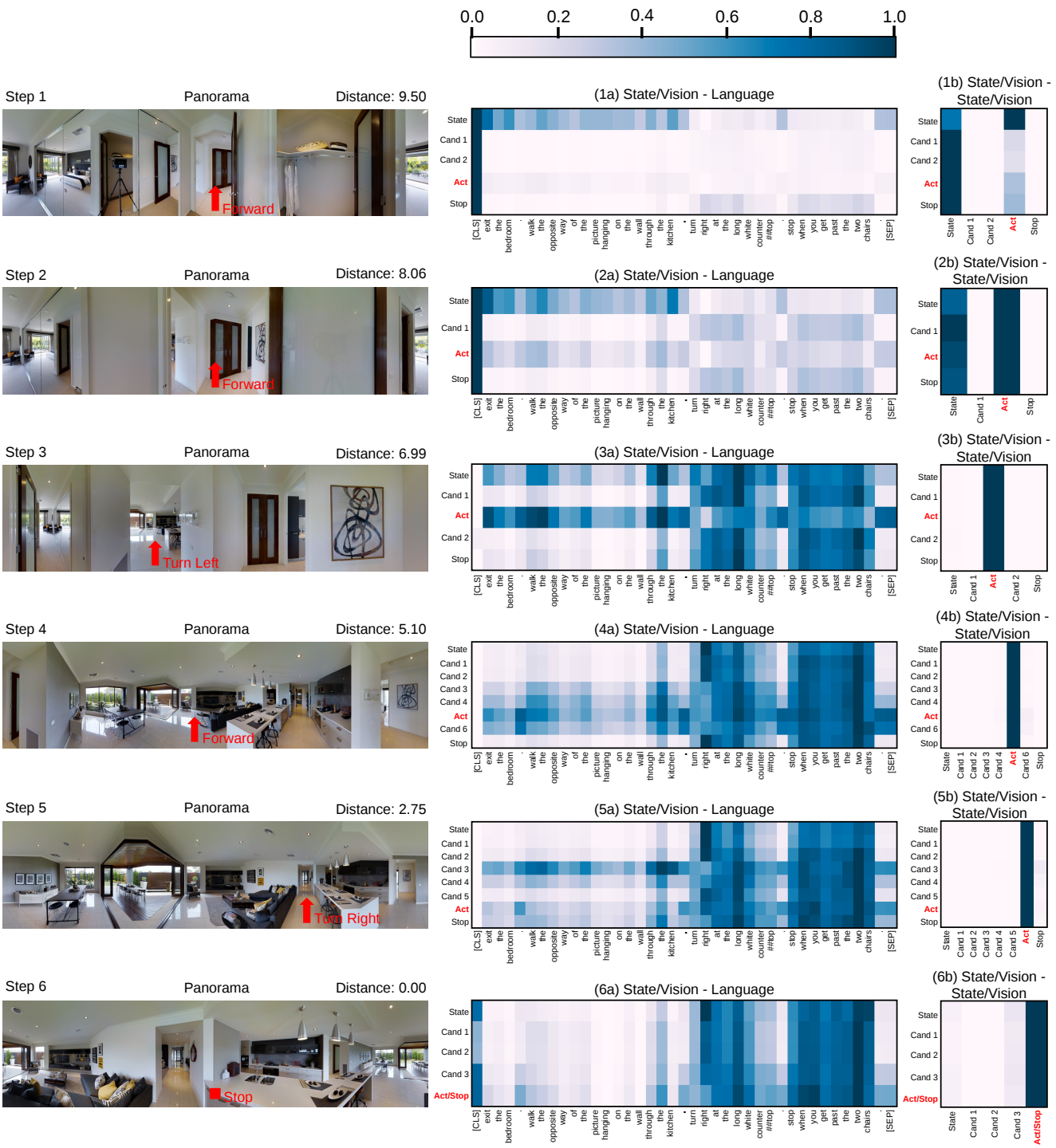

Figure 3. Visualisation of a trajectory and the averaged attention weights at the final (l=12-th) layer. The centre of each panorama is roughly the agent's heading direction at the corresponding time step. *Distance* is the agent's distance to target in meters. The attention weights are normalised for each row. Texts in red indicates the predicted action (corresponds to a candidate view) at each time step.

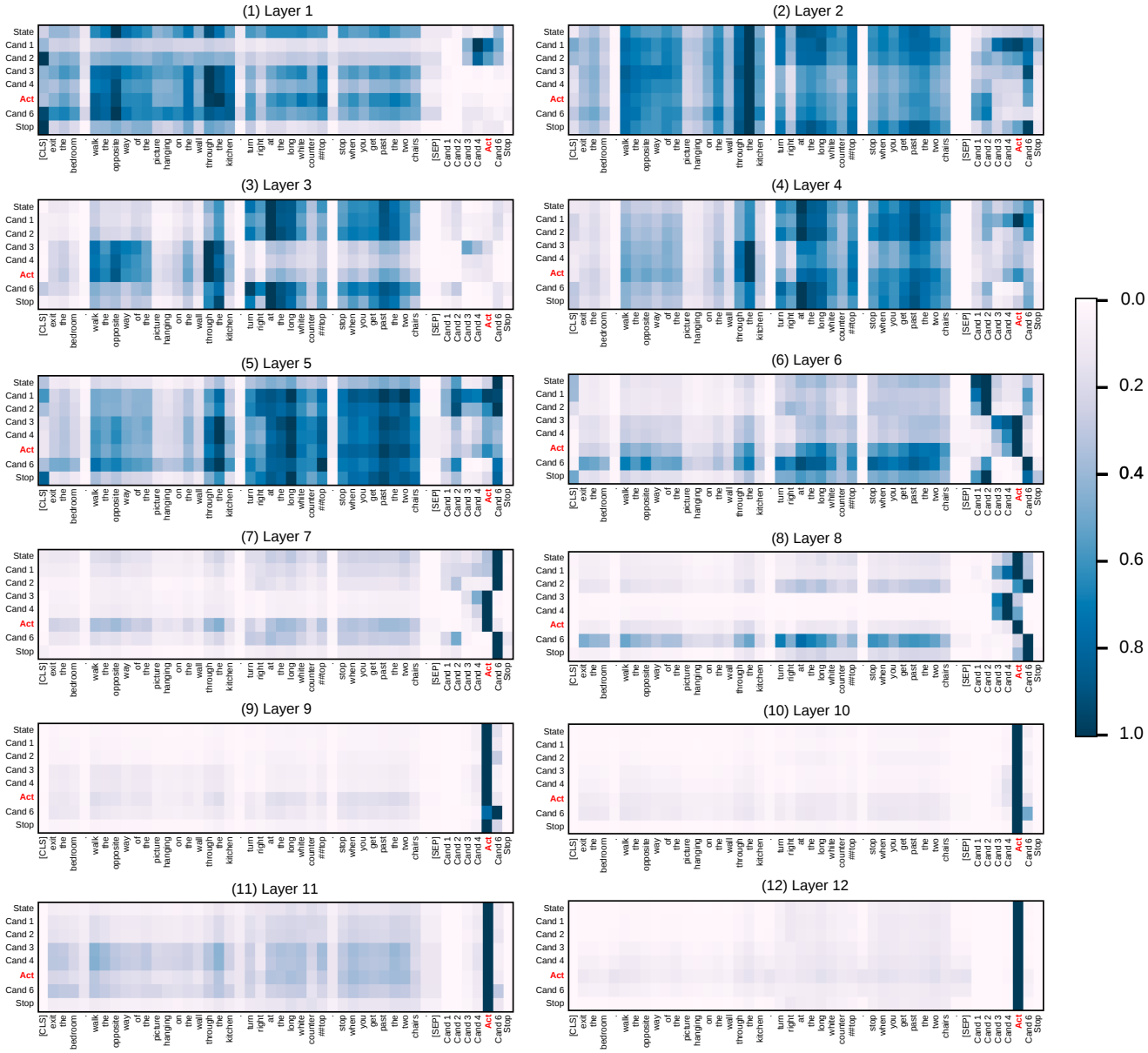

Figure 4. Averaged state/vision-to-state/vision/language attention weights at each layer at Step 4 of the trajectory. The attention weights are normalised for each row. Texts in red indicates the predicted action (corresponds to a candidate view) at the current time step (t=4).

References

[1] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018. 1

[2] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. 1, 2

[3] 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 *2017 International Conference on 3D Vision (3DV)*, pages 667–676. IEEE, 2017. 1

[4] 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 *Advances in Neural Information Processing Systems*, pages 3314–3325, 2018. 1, 2

[5] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146, 2020. 1, 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[7] Yicong Hong, Cristian Rodriguez, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[8] Yicong Hong, Cristian Rodriguez, Qi Wu, and Stephen Gould. Sub-instruction aware vision-and-language navigation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3360–3376, Online, Nov. 2020. Association for Computational Linguistics. 4

[9] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *arXiv preprint arXiv:1907.05446*, 2019. 2, 3

[10] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017. 1

[11] 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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016. 2

[13] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020. 1, 2

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[15] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. 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 5103–5114, 2019. 2

[16] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of NAACL-HLT*, pages 2610–2621, 2019. 1, 2

[17] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315, 2018. 2

[18] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017. 1