Embodied Vision-and-Language Navigation with Dynamic Convolutional Filters

Federico Landi
federico.landi@unimore.it
Lorenzo Baraldi
lorenzo.baraldi@unimore.it
Massimiliano Corsini
massimiliano.corsini@unimore.it
Rita Cucchiara
rita.cucchiara@unimore.it

University of Modena and Reggio Emilia, Italy

Abstract

In Vision-and-Language Navigation (VLN), an embodied agent needs to reach a target destination with the only guidance of a natural language instruction. To explore the environment and progress towards the target location, the agent must perform a series of low-level actions, such as rotate, before stepping ahead. In this paper, we propose to exploit dynamic convolutional filters to encode the visual information and the lingual description in an efficient way. Differently from some previous works that abstract from the agent perspective and use high-level navigation spaces, we design a policy which decodes the information provided by dynamic convolution into a series of low-level, agent friendly actions. Results show that our model exploiting dynamic filters performs better than other architectures with traditional convolution, being the new state of the art for embodied VLN in the low-level action space. Additionally, we attempt to categorize recent work on VLN depending on their architectural choices and distinguish two main groups: we call them *low-level actions* and *high-level actions* models. To the best of our knowledge, we are the first to propose this analysis and categorization for VLN.

1 Introduction

Imagine finding yourself in a large conference hall, with an assistant giving you instructions on how to reach the room for your talk. You are likely to hear something like: *turn right at the end of the corridor, head upstairs and reach the third floor: your room is immediately on the left*. Succeeding in the task of finding your target location is rather nontrivial because of the length of the instruction and its sequential nature: the flow of actions must be coordinated with a series of visual examinations – like recognizing the end of the corridor or the floor number. Furthermore, navigation complexity dramatically increases if the environment is unknown, and no prior knowledge, such as a map, is available.

Vision-and-Language Navigation (VLN) [3] is a challenging task that demands an embodied agent to reach a target location by navigating unseen environments, with a natural language instruction as its only clue. Similarly to the previous example, the agent must

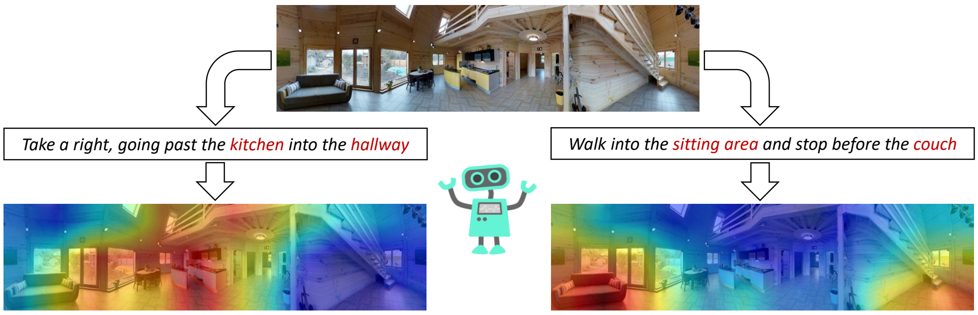

Take a right, going past the kitchen into the hallway
Walk into the sitting area and stop before the couch

Figure 1: Given a fixed visual observation, dynamic convolutional filters can extract a subset of specific features depending on the leading instruction. In this example, the agent focuses on two different parts of the same environment (best viewed in color).

assess different sub-tasks to succeed. First, a fine-grained comprehension of the given instruction is needed. Then, the agent must be able to map parts of the description into the visual perception. For example, *walking past the piano* requires to find and focus on the piano, rather than considering other objects in the scene. Finally, the agent needs to understand when the navigation has been completed and send a stop signal.

VLN has been first proposed by Anderson *et al.* [3], with the aim of connecting the research efforts on vision-and-language understanding [2, 4, 6, 7, 13, 23, 26] with the raising area of embodied AI [1, 8, 9, 25]. This is particularly challenging, as embodied agents must deal with a series of issues that do not belong to traditional vision and language tasks [1], like contextual decision-making and planning. Recent works on VLN [11, 17, 18, 21] integrate the agent with a simplified action space in which it "*only needs to make high-level decisions as to which navigable direction to go next*" [11]. In this scenario, the agent does not need to infer the sequence of actions to progress in the environment (*e.g.*, turn right 30 degrees, then move forward) but it exploits a navigation graph to teleport itself to an adjacent location. The adoption of this high-level action space allowed for a significant boost in success rates, while partly depriving the task of its embodied nature, and leaving space for little more than pure visual and language understanding. We claim that this type of approach is inconvenient, as it strongly relies on prior knowledge on the environment. Depending on information such as the position and the availability of navigable directions, it reduces the task to a pure graph navigation. Moreover, it ignores the active role of the agent, as it only perceives the surrounding scene and selects the next target viewpoint from a limited set. We claim instead that the agent should be the principal component of embodied VLN [1]. Consequently, the output space should match with the low-level set of movements that the agent can perform.

In this paper, we propose a novel architecture for embodied VLN which employs dynamic convolutional filters [16] to identify the next target direction, without getting any information about the navigable viewpoints from the simulator. Convolutional filters are produced via an attention mechanism which follows the given instruction, and are in turn used to attend relevant directions of the scene towards which the agent should move. We then rely on a policy network to predict the sequence of low-level actions.

Dynamic convolutional filters, proposed by Li *et al.* [16], were first conceived to identify and track people by a natural language specification. They were then successfully employed in other computer vision tasks, such as actor and action video segmentation from a sentence [12]. Nonetheless, these works considered mainly short descriptions, while we deal

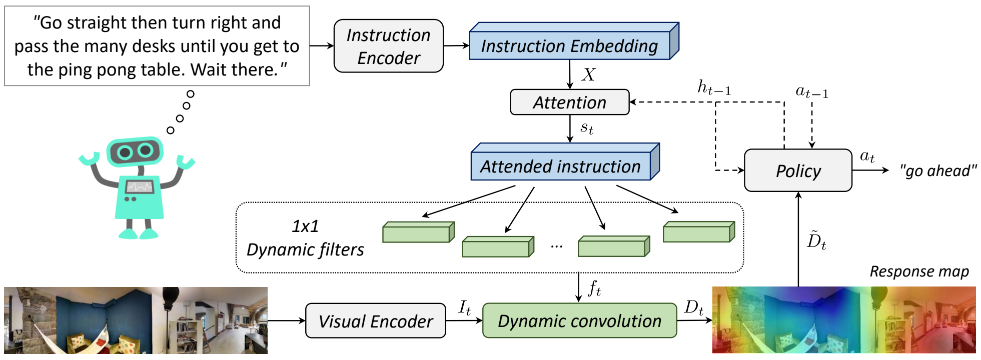


Figure 2: Schema of the proposed architecture for VLN. The input instruction is encoded via an attentive mechanism. We then generate dynamic convolutional filters that let the agent attend relevant regions of the input scene. The resulting visual information is used to feed a policy network controlling the movements of the embodied agent in a low-level action space.

with complex sentences and long-term dependencies. We generate dynamic filters according to the given instruction, to extract only the relevant information from the visual context. In this way, the same observation can lead to different feature maps, depending on the part of the instruction that the agents must complete (Fig. 1).

The proposed method is competitive with prior work that performs high-level navigation exploiting information about the reachable viewpoints (*i.e*. the navigation graph). Additionally, our approach is fully compliant with recent recommendations for embodied navigation [1]. When compared with models that are compliant with the VLN setup, we overcome the current state of the art by a significant margin.

To sum up, our contributions are as follows:

- We propose a new encoder-decoder architecture for embodied VLN, which for the first time employs dynamic convolutional filters to attend relevant regions of the visual scene and control the actions of the agent.
- We show, through extensive experimental evaluations, that in a mutable environment with shifting goals dynamic convolutional filters can provide better performance than traditional convolutional filters. Results show that our proposed architecture overcomes the state of the art on the embodied VLN task.
- As a complementary contribution, we categorize previous work on VLN basing on their level of abstraction and generalizability. We distinguish a group of works that strongly relies on the simulating platform and on the navigation graph, we call them *high-level actions* models. A second group, named *low-level actions* models, includes methods that are more agnostic on the underlying implementation and that directly predicts agent actions. With this categorization, we hope to encourage further research to consider low-level and high-level action spaces as distinct fields of application when dealing with VLN.

2 Method

We propose an encoder-decoder architecture for vision-and-language navigation. Our work employs dynamic convolutional filters conditioned on the current instruction to extract the relevant piece of information from the visual perception, which is in turn used to feed a policy network which controls the actions performed by the agent. The output of our model is a probability distribution over a low-level action space $\mathbf{A} = \{a_i\}_{i=1}^{6}$, which comprises the following actions: *turn 30° left*, *turn 30° right*, *raise elevation*, *lower elevation*, *go ahead*, <*end episode*>. The output probability distribution at a given step, $p_t = P(a_t|X, V_t, h_{t-1})$, depends on a natural language instruction X, the current visual observation V_t, and on the policy hidden state at time step $t-1$. Our architecture is depicted in Fig. 2 and detailed next.

2.1 Encoder

To represent the two inputs of the architecture, *i.e.* the instruction and the visual input at time t, we devise an instruction and a visual encoder. The instruction encoder provides a representation of the navigation instructions that is employed to guide the whole navigation episode. On the other hand, the visual encoding module operates at every single step, building a representation of the current observation which depends on the agent position.

Instruction Encoding. The given natural language instruction is split into single words via tokenization, and stop words are filtered out to obtain a shorter description. Differently from previous works that train word embeddings from scratch, we rely on word embeddings obtained from a large corpus of documents. Beside providing semantic information which could not be learned purely from VLN instructions, this also let us handle words that are not present in the training set (see Sec. 3.2 for a discussion). Given an instruction with length N, we denote its embeddings sequence as $L = (l_1, l_2, ..., l_N)$, where l_i indicates the embedding for the i-th word. Then, we adopt a Long Short-Term Memory (LSTM) network to provide a timewise contextual representation of the instruction:

$$X = (x_1, x_2, ..., x_N) = \mathrm{LSTM}(L), \tag{1}$$

where each x_i denotes the hidden state of the LSTM at time i, thus leading to a final representation with shape (N, d), where d is the size of the LSTM hidden state.

Visual Features Encoding. As visual input, we employ the panoramic 360° view of the agent, and discretize the resulting equirectangular image in a 12×3 grid, consisting of three different elevation levels and 30° heading shift from each other. Each location of the grid is then encoded via the 2048-dimensional features extracted from a ResNet-152 [14] pre-trained on ImageNet [10]. We also append to each cell vector a set of coordinates relative to the current agent heading and elevation:

$$coord_t = (\sin\phi_t, \cos\phi_t, \sin\theta_t), \tag{2}$$

where $\phi_t \in (-\pi, \pi]$ and $\theta_t \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ are the heading and elevation angles *w.r.t.* the agent position. By adding $coord_t$ to the image feature map, we encode information related to concepts such as *right*, *left*, *above*, *below* into the agent observation.

2.2 Decoder

Given the instruction embedding X for the whole episode, we use an attention mechanism to select the next part of the sentence that the agent has to fulfill. We denote this encoded piece of instruction as s_t. We detail our attentive module in the next section.

Dynamic Convolutional Filters. Dynamic filters are different from traditional, fixed filters typically used in CNN, as they depend on an input rather than being purely learnable parameters. In our case, we can think about them as specialized feature extractors reflecting the semantics of the natural language specification. For example, starting from an instruction like "head towards the red chair" our model can learn specific filters to focus on concepts such as *red* and *chair*. In this way, our model can rely on a large ensemble of specialized kernels and apply only the most suitable ones, depending on the current goal. Naturally, this approach is more efficient and flexible than learning a fixed set of filters for all the navigation steps. We use the representation of the current piece of instruction s_t to generate multiple 1×1 dynamic convolutional kernels, according to the following equation:

$$f_t = \ell_2[\tanh(W_f s_t + b_f)], \tag{3}$$

where $\ell_2[\cdot]$ indicates L2 normalization, and f_t is a tensor of filters reshaped to have the same number of channels as the image feature map. We then perform the dynamic convolution over the image features I_t, thus obtaining a response map for the current timestep as follows:

$$D_t = f_t * I_t. \tag{4}$$

As the aforementioned operation is equivalent to a dot product, we can conceive the dynamic convolution as a specialized form of dot-product attention, in which I_t acts as key and the filters in f_t act as time-varying queries. Following this interpretation, we rescale D_t by $\sqrt{d_f}$, where d_f is the dynamic filter size [22] to maintain dot products smaller in magnitude.

Action Selection. We use the response maps dynamically generated as input for the policy network. We implement it with an LSTM whose hidden state at time step t is employed to obtain the action scores. Formally,

$$h_t = \text{LSTM}([\tilde{D}_t, a_{t-1}], h_{t-1}), \quad p_t = \text{softmax}(W_a h_t + b_a), \tag{5}$$

where $[\cdot,\cdot]$ indicates concatenation, a_{t-1} is the one-hot encoding of the action performed at the previous timestep, and $\tilde{D}_t$ is the flattened tensor obtained from D_t. To select the next action a_t, we sample from a multinomial distribution parametrized with the output probability distribution during training, and select $a_t = \arg\max p_t$ during the test. In line with previous work, we find out that sampling during the training phase encourages exploration and improves overall performances.

Note that, as previously stated, we do not employ a high-level action space, where the agent selects the next viewpoint in the image feature map, but instead make the agent responsible of learning the sequence of low-level actions needed to perform the navigation. The agent can additionally send a specific stop signal when it considers the goal reached, as suggested by recent standardization attempts [1].

2.3 Encoder-Decoder Attention

The navigation instructions are very complex, as they involve not only different actions but also temporal dependencies between them. Moreover, their high average length represents an additional challenge for traditional embedding methods. For these reasons, we enrich our architecture with a mechanism to attend different locations of the sentence representation, as the navigation moves towards the goal. In line with previous work on VLN [3, 11], we employ an attention mechanism to identify the most relevant parts of the navigation instruction. We employ the hidden state of our policy LSTM to get the information about our progress in the navigation episode and extract a time-varying query $q_t = W_q h_{t-1} + b_q$. We then project our sentence embedding into a lower dimensional space to obtain key vectors, and perform a scaled dot-product attention [22] among them.

$$\alpha_t = \frac{q_t K^T}{\sqrt{d_{att}}}, \text{ where } K = W_k X + b_k \tag{6}$$

After a softmax layer, we obtain the current instruction embedding s_t by matrix multiplication between the initial sentence embedding and the softmax scores.

$$s_t = \text{softmax}(\alpha_t)X \tag{7}$$

At each timestep of the navigation process s_t is obtained by attending the instruction embedding at different locations. The same vector is in turn used to obtain a time-varying query for attending spatial locations in the visual input.

2.4 Training

Our training sample consists of a batch of navigation instructions and the corresponding ground truth paths coming from the R2R (*Room-to-Room*) dataset [3] (described in section 3). The path denotes a list of discretized viewpoints that the agent has to traverse to progress towards the goal. The agent spawns in the first viewpoint, and its goal is to reach the last viewpoint in the ground truth list. At each step, the simulator is responsible for providing the next ground truth action in the low-level action space that enables the agent to progress. Specifically, the ground truth action is computed by comparing the coordinates of the next target node in the navigation graph with the agent position and orientation. At each time step t, we minimize the following objective function:

$$L = -\sum_t y_t \log p_t \tag{8}$$

where p_t is the output of our network, and y_t is the ground truth low-level action provided by the simulator at time step t. We train our network with a batch size of 128 and use Adam optimizer [15] with a learning rate of 10^{-3}. We adopt early stopping to terminate the training if the mean success rate does not improve for 10 epochs.

3 Experiments

3.1 Experimental Settings

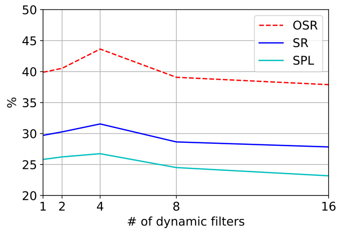
For our experiments, we employ the R2R (*Room-to-Room*) dataset [3]. This challenging benchmark builds upon Matterport3D dataset of spaces [6] and contains $7,189$ different navigation paths in 90 different scenes. For each route, the dataset provides 3 natural language instructions, for a total of 21,567 instructions with an average length of 29 words. The R2R dataset is split into 4 partitions: training, validation on seen environments, validation on unseen scenes, and test on unseen environments.

Evaluation Metrics. We adopt the same evaluation metrics employed by previous work on the R2R dataset: navigation error (NE), oracle success rate (OSR), success rate (SR), and success rate weighted by path length (SPL). NE is the mean distance in meters between the final position and the goal. SR is fraction of episodes terminated within no more than 3 meters from the goal position. OSR is the success rate that the agent would have achieved if it received an oracle stop signal in the closest point to the goal along its navigation. SPL is the success rate weighted by normalized inverse path length and penalizes overlong navigations.

Implementation Details. For each LSTM, we set the hidden size to 512. Word embeddings are obtained with GloVe [19]. In our visual encoder, we apply a bottleneck layer to reduce the dimension of the image feature map to 512. We generate dynamic filters with 512 channels using a linear layer with dropout [20] ($p = 0.5$). In our attention module, q and K

	Validation-Seen				Validation-Unseen			
Method	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Random agent	9.45	15.9	21.4	-	9.23	16.3	22.0	-
Baseline w/ traditional convolution [3]	6.01	38.6	52.9	-	7.81	21.8	28.4	-
Ours w/o encoder-decoder attention	5.86	41.3	51.2	36.3	7.72	22.0	29.3	19.3
Ours w/o pre-trained embedding	5.62	42.0	54.0	36.3	7.32	25.8	33.3	22.1
Ours w/ dynamic filters	**4.68**	**53.1**	**66.1**	**46.0**	**6.65**	**31.6**	**43.6**	**26.8**

Table 1: Ablation study for our architecture on the validation sets of R2R. The full model works better than when attention is removed or when conventional filters are used.



# of filters	Validation-Unseen			
	NE ↓	SR ↑	OSR ↑	SPL ↑
1	6.79	29.7	39.9	25.8
2	6.77	30.3	40.5	26.2
4	**6.65**	**31.6**	**43.6**	**26.8**
8	7.19	28.7	39.1	24.5
16	7.03	27.8	37.9	23.2

Figure 3: Comparison with different numbers of dynamic filters on the validation-unseen set of R2R. The best results for all the metrics are obtained using four different dynamic filters.

have 128 channels and we apply a ReLU non-linearity after the linear transformation. For our action selection, we apply dropout with $p = 0.5$ to the policy hidden state before feeding it to the linear layer.

3.2 Ablation Study

In our ablation study, we test the influence of our implementation choices on VLN. As a first step, we discuss the impact of dynamic convolution by comparing our model with a similar *seq2seq* architecture that employs fixed convolutional filters. We then detail the importance of using an attention mechanism to extract the current piece of instruction to be fulfilled. Finally, we compare the results obtained using a pre-trained word embedding instead of learning the word representation from scratch. Results are reported in Table 1.

Static Filters Vs. Dynamic Convolution. As results show, dynamic convolutional filters surpass traditional fixed filters for VLN. This because they can easily adapt to new instructions and reflect the variability of the task. When compared to a baseline model that employs traditional convolution [3], our method performs 14.5% and 9.8% better, in terms of success rate, on the val-seen and val-unseen splits respectively.

Fixed Instruction Representation Vs. Attention. The navigation instructions are very complex and rich. When removing the attention module from our architecture, we keep the last hidden state h_N as instruction representation for the whole episode. Even with this limitation, dynamic filters achieve better results than static convolution, as the success rate is higher for both of the validation splits. However, our attention module further increases the success rate by 11.8% and 9.6%.

Word Embedding from Scratch Vs. Pre-trained Embedding. Learning a meaningful word embedding is nontrivial and requires a large corpus of natural language descriptions. For this reason, we adopt a pre-trained word embedding to encode single words in our instructions. We then run the same model while trying to learn the word embedding from scratch. We discover that a pre-trained word embedding significantly eases VLN. Our model with GloVe [19] obtains 11.1% and 5.8% more on the val-seen and val-unseen splits respectively, in terms of success rate.

	Validation-Seen				Validation-Unseen				Test (Unseen)			
Low-level Actions Methods	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Random	9.45	0.16	0.21	-	9.23	0.16	0.22	-	9.77	0.13	0.18	0.12
Student-forcing [3]	6.01	0.39	0.53	-	7.81	0.22	0.28	-	7.85	0.20	0.27	0.18
RPA [24]	5.56	0.43	0.53	-	7.65	0.25	0.32	-	7.53	0.25	0.33	0.23
Ours	4.68	0.53	0.66	0.46	6.65	0.32	0.44	0.27	7.14	0.31	0.42	0.27
Ours w/ data augmentation	**3.96**	**0.58**	**0.73**	**0.51**	**6.52**	**0.34**	**0.43**	**0.29**	**6.55**	**0.35**	**0.45**	**0.31**
High-level Actions Methods	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Speaker-Follower [11]	3.36	0.66	0.74	-	6.62	0.36	0.45	-	6.62	0.35	0.44	0.28
Self-Monitoring [17]	**3.22**	0.67	**0.78**	0.58	5.52	0.45	0.56	0.32	5.99	0.43	0.55	0.32
Regretful [18]	3.23	**0.69**	0.77	**0.63**	**5.32**	**0.50**	**0.59**	**0.41**	**5.69**	**0.48**	**0.56**	**0.40**

Table 2: Comparison with state-of-art methods for VLN. For compliance with the evaluation server, we report success rates as fractions. The results for high-level models comprehend data augmentation with synthetic data provided by [11], as in our final setup. Our method outperforms comparable models by a large margin, while being competitive with or even better than some high-level actions architectures.

3.3 Multi-headed Dynamic Convolution

In this experiment, we test the impact of using a different number of dynamically-generated filters. We test our architecture when using 1, 2, 4, 8, and 16 dynamic filters. We find out that the best setup corresponds to the use of 4 different convolutional filters. Results in Fig. 3 show that the success rate and the SPL increase linearly with the number of dynamic kernels for a small number of filters, reaching a maximum at 4. The metrics then decrease when adding new parameters to the network. This suggests that a low number of dynamic filters can represent a wide variety of natural language specifications. However, as the number of dynamic filters increase, the representation provided by the convolution becomes less efficient.

3.4 Comparison with the State-of-the-art

Finally, we compare our architecture with the state-of-the-art methods for VLN. Results are reported in Table 2. We distinguish two main categories of models, depending on their output space: the first, to which our approach belongs, predicts the next atomic action (*e.g. turn right*, *go ahead*). We call architectures in this category *low-level actions methods*. The second, instead, searches in the visual space to match the current instruction with the most suitable navigable viewpoint. In these models, atomic actions are not considered, as the agent displacements are done with a teleport system, using the next viewpoint identifier as target destination. Hence, we refer to these works as *high-level actions methods*. While the latter achieve better results, they make strong assumptions on the underlying simulating platform and on the navigation graph. Our method, exploiting dynamic convolutional filters and predicting atomic actions, outperforms comparable architectures and achieves state of the art results for *low-level actions* VLN. Our final implementation takes advantage of the synthetic data provided by Fried *et al.* [11] and overcomes comparable methods [3, 24] by 15% and 10% success rate points on the R2R test set. Additionally, we note that our method is competitive with some *high-level actions* models, especially in terms of SPL. When considering the test set, we notice in fact that our model outperforms Speaker-Follower [11] by 3%, while performing only 1% worse than [17].

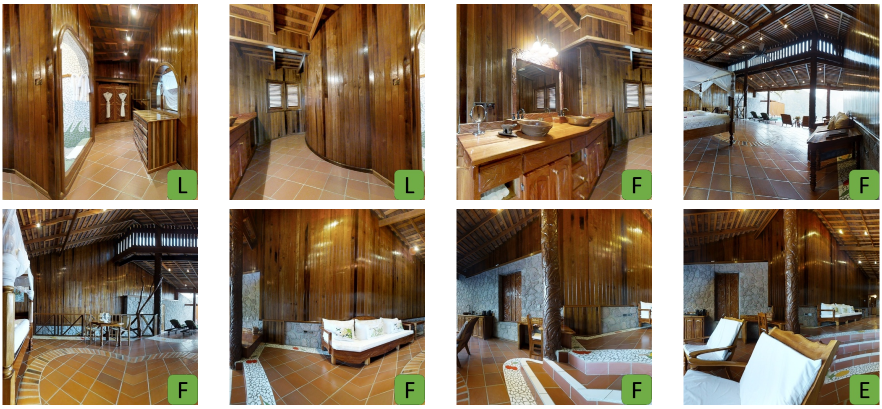
Low-level Action Space or High-level Navigation Space? While previous work on VLN never considered this important difference, we claim that it is imperative to categorize navigation architectures depending on their output space. In our opinion, ignoring this aspect would lead to inappropriate comparisons and wrong conclusions. Considering the results in

Table 2, we separate the two classes of work and highlight the best results for each category. Please note that the random baseline was initially provided by [3] and belongs to *low-level actions* architectures (a random *high-level actions* agent was never provided by previous work). We immediately notice that, with this new categorization, intra-class results have less variance and are much more aligned to each other. We believe that future work on VLN should consider this new taxonomy in order to provide meaningful and fair comparisons.

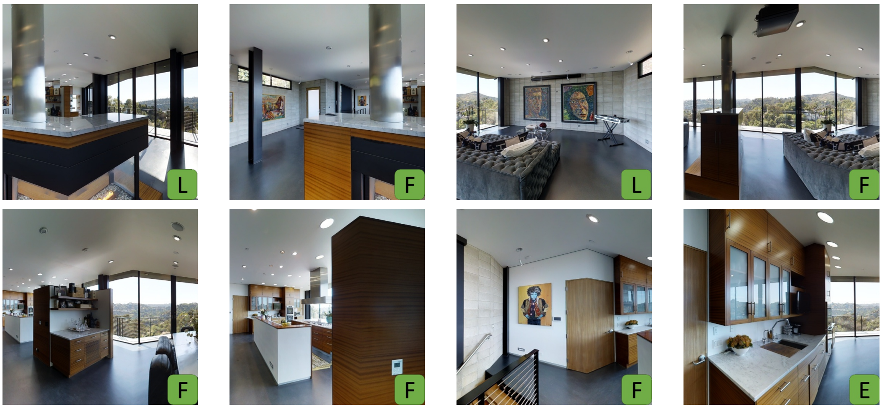
3.5 Qualitative Results

Fig. 4 shows two navigation episodes from the R2R validation set. We display the predicted action in a green box on the bottom-right corner of each image. Both examples are successful.

Legend: L *left* R *right* F *forward* E *end episode*



Instruction: *From bathroom, enter bedroom and walk straight across down two steps, wait at loungers.*



Instruction: *Walk past the fireplace and to the left. Stop in the entryway of the kitchen.*

Figure 4: Qualitative results from the R2R validation set. Each episode is detailed by eight pictures, representing the current position of the agent and containing the next predicted action (from left to right, top to bottom). To make the visualization more readable, we do not display the 360° panoramic images.

4 Conclusion

In this paper, we propose dynamic convolution for embodied Vision-and-Language Navigation. Instead of relying on a high-level action space, where the agent is teleported from one viewpoint to the other, we predict a series of action in an agent friendly action space. Basing on this substantial difference, we propose a new categorization based on the model output space. We then separate previous VLN architectures into *low-level actions* and *high-level actions* methods. We claim that comparisons made considering this new taxonomy are more fair and reasonable than previous analysis. Our method with dynamic convolutional filters achieves state-of-the-art results for the *low-level actions* category, and it is competitive with *high-level actions* architectures that rely on much more information and have a higher level of abstraction during the navigation episode. We hope this work encourages further research on low-level VLN, and in general we consider this a step towards the use of more realistic action spaces for this task. While our experiments show promising results in this setting, much work remains to inspect the possible connections between low-level and high-level Vision-and-Language Navigation.

Acknowledgements: This work was partially supported by the *Fondazione Cassa di Risparmio di Modena* project "*AI for Digital Humanities*" (Prot. n. 505.18.8b del 18/10/2018 - Pratica Sime n. 2018.0390). We also want to thank the anonymous reviewers for their insightful remarks and their constructive criticism.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[2] 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*, 2018.

[3] 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*, 2018.

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *Proceedings of the International Conference on Computer Vision*, 2015.

[5] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *International Conference on 3D Vision*, 2017.

[6] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[7] Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the International Conference on Computer Vision*, 2017.

[8] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[9] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. In *Conference on Robot Learning*, 2018.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[11] 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*, 2018.

[12] Kirill Gavrilyuk, Amir Ghodrati, Zhenyang Li, and Cees GM Snoek. Actor and action video segmentation from a sentence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[13] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[14] 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*, 2016.

[15] Diederik Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

[16] Zhenyang Li, Ran Tao, Efstratios Gavves, Cees GM Snoek, and Arnold WM Smeulders. Tracking by natural language specification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[17] 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*, 2019.

[18] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[19] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[23] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[24] 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. In *Proceedings of the European Conference on Computer Vision*, 2018.

[25] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[26] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, 2015.