Multimodal Attention Networks for Low-Level Vision-and-Language Navigation

Federico Landi,∗∗, Lorenzo Baraldi, Marcella Cornia, Massimiliano Corsini, Rita Cucchiara

aUniversity of Modena and Reggio Emilia, Italy
bISTI - CNR, Italy

ABSTRACT

Vision-and-Language Navigation (VLN) is a challenging task in which an agent needs to follow a language-specified path to reach a target destination. The goal gets even harder as the actions available to the agent get simpler and move towards low-level, atomic interactions with the environment. This setting takes the name of low-level VLN. In this paper, we strive for the creation of an agent able to tackle three key issues: multi-modality, long-term dependencies, and adaptability towards different locomotive settings. To that end, we devise “Perceive, Transform, and Act” (PTA): a fully-attentive VLN architecture that leaves the recurrent approach behind and the first Transformer-like architecture incorporating three different modalities – natural language, images, and low-level actions for the agent control. In particular, we adopt an early fusion strategy to merge lingual and visual information efficiently in our encoder. We then propose to refine the decoding phase with a late fusion extension between the agent’s history of actions and the perceptual modalities. We experimentally validate our model on two datasets: PTA achieves promising results in low-level VLN on R2R and achieves good performance in the recently proposed R4R benchmark. Our code is publicly available at https://github.com/aimagelab/perceive-transform-and-act.

1. Introduction

Effective instruction-following and contextual decision-making can open the door to a new world for researchers in embodied AI. Deep neural networks have the potential to build complex reasoning rules that enable the creation of intelligent agents, and research on this subject could also help to empower the next generation of collaborative robots (Savva et al., 2019; Xia et al., 2018). In this scenario, Vision-and-Language Navigation (VLN) (Anderson et al., 2018c) plays a significant part in current research. This task requires to follow natural language instructions through unknown environments, discovering the correspondences between lingual and visual perception step by step. Additionally, the agent needs to progressively adjust navigation in light of the history of past actions and explored areas. Even a small error while planning the next move can lead to failure because perception and actions are unavoidably entangled; indeed, we must perceive in order to move, but we must also move in order to perceive (Gibson, 2014). For this reason, the agent can succeed in this task only by efficiently combining the three modalities – language, vision, and actions.

Recent literature identifies two main operating settings for VLN (Landi et al., 2019), called high-level action space and low-level action space (Fig. 1a, b). The concept of a high-level, panoramic action space was first proposed by Fried et al. (2018). In this setting, navigation takes place on a graph whose connectivity is known a priori and the nodes are represented by different viewpoints (i.e., the locations where the agent can step and look at the surroundings). High-level agents predict the path to the goal as a sequence of connected viewpoints, and move through the environment using a teleporting system. This aspect limits adaptability to real-world applications and prevents current research on high-level VLN from having a practical impact on embodied navigation robots. Instead, low-level methods make predictions over the agent locomotor system, hence performing actions with a one-to-one correspondence with the robot control system – rotate $X^\circ$, tilt up/down, and step forward are examples of low-level actions. Even though low-level navigation can still be performed on a graph-like environment (with viewpoints as nodes), the agent is not aware of it and does not exploit any knowledge related to the structure of the underlying simulating platform. This setting is more...
in line with recent research on embodied AI platforms (Savva et al., 2019; Xia et al., 2018), which is moving towards realistic and low-level interactions with the environment and continuous control of the agent. Since the adaptability to real-world applications represents an important challenge in this scenario, we tackle the task of low-level VLN, in which abstract reasoning (i.e., teleporting from a viewpoint to the next and knowledge of the connectivity graph) is no longer available to the agent.

Encouraged by the success of attention in many vision-and-language tasks (Devlin et al., 2019; Lu et al., 2019; Vaswani et al., 2017), we propose a new model for low-level VLN that exploits fully-attentive networks to merge the knowledge coming from different domains. In this work, we devise Perceive, Transform, and Act (PTA), in which the different modalities (text, vision, and actions) can be conditioned on the full history of previous observations. While all the previous approaches to VLN rely on a recurrent policy to track the agent’s internal status through time, we directly infer the state from the observations via attention and avoid any form of recurrence (Fig. 1b). For this reason, our agent can model the dependencies tied to navigation more efficiently and generalize to longer episodes better than other models.

At the present time, there is no study exploring the possibility for a given architecture to switch between the high-level and the low-level action spaces. In this work, we experimentally show that methods born for high-level navigation experience a drop in performance when adapted for low-level VLN. Indeed, high-level reasoning and abstraction from the physical environment is too heavily exploited to let the agent walk on its own. This is not true for PTA, which is designed for low-level use but can easily adapt to high-level scenarios. We summarize our main contributions as follows:

- We propose a novel multimodal framework for low-level VLN that replaces any form of recurrence with attention mechanisms, using them to tackle both long-term dependencies and multi-modality. To the best of our knowledge, our model is the first Transformer-like architecture to merge visuo-linguistic perception with information coming from the agent action system;
- We technically describe how it is possible to switch from a high-level output space to a low-level locomotor system and vice versa. Experimental results on this subject are the first to analyze the mutual relationships between low-level and high-level VLN, and validate the hypothesis that high-level architectures are not easily adaptable to the low-level counterpart. Such results highlight the need for more experiments in this direction for future works;
- Experimental results show that PTA achieves good performance on low-level VLN. We validate this claim on two different benchmarks of increasing instruction length and complexity. Since our setting is closer to real-world applications and requires to decode fine-grained atomic actions, we believe that low-level VLN represents the next testbed for embodied agents aiming to perform Vision-and-Language Navigation.

2. Related Work

There is a wide area of research devoted to bridge natural language processing and image understanding. Image captioning (Anderson et al., 2018b; Vinyals et al., 2015; Xu et al., 2015), visual question answering (Antol et al., 2015; Goyal et al., 2017), and visual dialog (Das et al., 2017a; b) are examples of active research areas in this field. At the same time, visual navigation (Gupta et al., 2017; Shen et al., 2019; Xia et al., 2018) and goal-oriented instruction following (Chen et al., 2019; Fu et al., 2019; Qi et al., 2020b) represent an important part of current work on embodied AI (Das et al., 2018a, b; Savva et al., 2019; Yang et al., 2019). In this context, Vision-and-Language Navigation (VLN) (Anderson et al., 2018c) constitutes a peculiar challenge, as it enriches traditional navigation with a set of visually rich environments and detailed instructions. Additionally, all the scenes are photo-realistic and unknown to the agent beforehand.

High-level Vision-and-Language Navigation. The idea of a high-level action space was first proposed by Fried et al. (2018).
and immediately allowed for an important boost in terms of performance. Following work includes visual and textual co-grounding with progress inference \cite{Ma2019b} and back-tracking with learned heuristics \cite{Ma2019a}. Other methods implement a speaker module which strengthens consistency between the chosen path and the instruction \cite{Fried2018, Wang2019}. \cite{Wang2019} propose a reinforced cross-modal matching critic, together with a new self-supervised imitation learning setting. \cite{Jan2019} devise a novel environmental dropout method to improve traditional features dropout for VLN. \cite{Ke2019} propose a FAST navigation agent which improves the performance both over greedy decoding of the next action and over beam search. Very recently, \cite{Zhu2020} exploit auxiliary reasoning tasks and the rich semantic given by the navigation in their model, while \cite{Hao2020} investigate an efficient pre-training for generic VLN agents. While pragmatic approaches with high-level reasoning allow for a boost in performance, architectures built for high-level VLN rely heavily on the information coming from the underlying simulating platform. Even when the environment is supposed to be unknown \( \text{e.g. during test} \) the agent can get a priori knowledge from the connectivity graph and exploit this information for a more efficient navigation. Recently proposed benchmarks and new evaluation metrics \cite{Jan2019} show that traditional approaches hardly adapt to longer trajectories. Indeed, the recurrent nature of previous methods exacerbates the difficulty of learning long-term dependencies \cite{Bengio1994} both in the instruction and in the navigation.

After the initial submission of this paper, new methods have been proposed to deal with VLN on a high-level perspective: a recent line of work designs graph operations to boost planning capabilities \cite{Deng2020} or to model visuo-linguistic relationships in the graph nodes \cite{Hong2020}. \cite{Zhang2020} propose to employ two levels of attention-guided co-grounding, together with a new learning scheme alternating teacher-forcing and student-forcing. \cite{Qi2020a} design an architecture taking advantage from both visual tokens and action tokens in the instructions. Visual tokens are employed to identify meaningful visual features in the environment, while action tokens consider only the agent state \( \text{represented by coordinates features} \). In this work, we leverage the same intuition in our multi-modal decoder. In fact, we propose an additional decoding branch that does not employ visual features, but focuses on action clues provided in the sole instruction.

**Low-level Vision-and-Language Navigation.** In low-level VLN, the agent takes move in the environment by using actions such as \texttt{rotate}, \texttt{tilt up}, and \texttt{step ahead}. So far, only a small portion of literature has taken this direction. \cite{Anderson2018c} build on a traditional sequence-to-sequence architecture, while \cite{Wang2018} employ a mixture of model-free and model-based reinforcement learning. In these works the agent perceives only the first person view of the surrounding environment. More recently, \cite{Landi2019} propose a sequence-to-sequence model which exploits dynamic convolution to make the visual representation more compact and informative for the agent. In this last work, the agent perceives the 360° image of the surroundings. This generalization does not hurt adaptability to real-world scenarios, since it is relatively easy to enrich the agent with additional RGB cameras.

### 3. Perceive, Transform, and Act

Our goal is to navigate unseen environments using low-level actions with the only help of natural language instructions and egocentric visual observations. To merge multimodal knowledge coming from the environment, we devise a two-stage encoder. In the first stage, we focus on encoding the instruction \( \text{– this step can be done once per episode as the natural language indication remains the same throughout the navigation} \). In the second stage, we use spatial attention to encode the visual observation and then employ the encoded instruction coming from the previous phase to enrich the agent representation of the surrounding environment. At each time step, the agent selects a move to progress towards the goal. To determine the next action, we fuse visuo-linguistic information with the history of actions via attention and build a multimodal decoder which merges the three modalities: actions, images, and text. We then decode a probability distribution over a low-level output space in which possible actions are atomic moves like \texttt{turn} or \texttt{step ahead}. After a first phase in which we train the agent with classical imitation learning, we implement an extrinsic reward function to promote coherence between ground-truth and predicted trajectories. We are the first, to the best of our knowledge, to build a VLN architecture without recurrence. Each component of our model is end-to-end trainable. Our architecture is depicted in Fig. 2 and detailed next.

### 3.1. Two-stage Encoder

At the beginning of each navigation episode, the agent receives a natural language instruction \( \{w_0, w_1, \ldots, w_{n-1}\} \) of variable length \( n \). The agent also perceives a panoramic 360° image of the surroundings \( I_t \) at each timestep \( t \). Our encoder consists of a single branch for each modality: text and images, and then employs attention to create a fused representation which specifically models the relevance of the source instruction into the visual observation.

**Instruction Encoding.** To encode the textual instruction, we employ an attention mechanism with multiple heads, followed by a feed-forward network. As a first step, we filter stop words and apply GloVe embeddings \cite{Pennington2014} to obtain a meaningful representation for each word. We then apply the following transformation:

\[
\bar{X} = \text{LayerNorm} \left( \text{max}(0, XW_x + b_x) \right),
\]

where \( X \) is the GloVe embedding for the natural language instruction, \( W_x \in \mathbb{R}^{d_{\text{in}} \times d_{\text{head}}} \) and \( b_x \in \mathbb{R}^{d_{\text{head}}} \) are learnable parameters, and \( \text{LayerNorm}() \) stands for layer normalization. Since the instruction encoder has no recurrence, we must inject information about the relative position of the words in the sentence. Such information is added in the form of positional encoding to the input embeddings. The positional encodings have the same dimension as the embeddings, so that the two
can be summed. We employ sine and cosine functions of different frequencies, in line with [Vaswani et al., 2017]:
\[
    PE_{(pos,2)} = \sin(pos/10000^{2j/d_{model}}) \\
    PE_{(pos,2+1)} = \cos(pos/10000^{2j/d_{model}})
\]
where \(pos\) is the position in the sequence and \(j\) is the channel index. At this point we use multi-head attention to create a representation that models temporal dependencies inside the instruction. Multi-head attention is defined as:
\[
    \text{MH}(Q, K, V) = \text{Concat}(h_1, h_2, \ldots, h_h) W^O
\]
with \(h_i = \text{Attention}(Q W^Q, K W^K, V W^V)\),
where \(W^Q \in \mathbb{R}^{d_{model} \times d_k}\), \(W^K \in \mathbb{R}^{d_{model} \times d_k}\), \(W^V \in \mathbb{R}^{d_{model} \times d_v}\), and \(d_k = d_v = d_{model} / h\). In each head, we employ the scaled dot-product attention defined by [Vaswani et al., 2017]:
\[
    \text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V.
\]
The attention mechanism described by Eq. [4] computes a weighted sum of the values \(V\) basing on the similarity between the keys and the queries \((K, Q)\). In the self-attention, the same source sequence \(X\) (in this case) is employed to model the \((Q, K, V)\) triplet of Eq. [3]. Following the attention layer, we place a feed-forward multilayer perceptron:
\[
    \text{FF}(\tilde{X}) = \max(0, \tilde{X} W_1 + b_1) W_2 + b_2.
\]
where \(W_1 \in \mathbb{R}^{d_{model} \times d_k}\), \(W_2 \in \mathbb{R}^{d_k \times d_{ff}}\), \(b_1 \in \mathbb{R}^{d_k}\), \(b_2 \in \mathbb{R}^{d_{ff}}\). At the end of this step, we obtain the attended representation for the current instruction \(X = \{x_0, x_1, \ldots, x_{n-1}\}\), that we use both during image encoding and in our multimodal decoder.

Image Encoding. As a first step, we discretize the 360° panoramic image of the surroundings \(I_t\) in 36 squared locations and we extract the corresponding visual features with a ResNet-152 [He et al., 2016] trained on ImageNet (Deng et al., 2009). Each viewpoint covers 30° in the equiangular image representing the agent surroundings, hence the image representation takes the form of a 3 × 12 grid. We then project visual features with a transformation similar to Eq. [1] but instead of using sinusoidal positional encodings, we append a coordinate vector given by:
\[
    \text{coord} = (\sin \phi_i, \cos \phi_i, \sin \theta_i),
\]
where \(\phi_i \in (-\pi, \pi)\) and \(\theta_i \in [-\frac{\pi}{2}, \frac{\pi}{2}]\) are the heading and elevation angles for each viewpoint in the 3 × 12 grid relative to the agent position at timestep \(t\). We then apply multi-head self-attention according to Eq. [3] to help modeling concepts such as relative positions between objects. In this layer, the input sequence modeling \((Q, K, V)\) is composed by the features extracted from the 36 squared regions of \(I_t\).

After this step, we aim to create an image representation enriched with the textual concepts expressed by the attended instruction \(X\). We use cross-attention to achieve this goal, and employ \(\tilde{X}\) as keys and values for multi-head attention (Eq. [5]), while the queries come from the output of the previous self-attention layer. Using cross-attention, we enrich visual information with a weighted sum of the instruction tokens. From the resulting representation it is possible to draw concepts such as the tableness or the redness of an image region, given an instruction that refers to concepts such as table or red. Finally, a feed-forward network as in Eq. [5] is applied to obtain the attended visual observation \(I_t\).

3.2. Multimodal Decoder

Our decoder predicts the next action to perform among the following instructions: turn right/left 30°, tilt up/down, step forward, and end episode – to signal that it has reached the goal.

Contextual History for Action Decoding. The first part of our decoder takes into account the history of past actions. While previous methods employ a recurrent neural network to keep track of previous steps (see for instance [Anderson et al., 2018c]).
Ma et al. (2019a); Wang et al. (2019)), we explicitly model $H_t = \{a_0, a_1, \ldots, a_{t-1}\}$ as the set of actions performed before the current timestep $t$. Note that $a_0$ coincides with the $<\text{start}>$ token. We add sinusoidal positional encoding (Eq. 2) to provide temporal information and apply multi-head self-attention to obtain an attended history representation $\tilde{H}_t = \{\tilde{a}_0, \tilde{a}_1, \ldots, \tilde{a}_{t-1}\}$.

**Late Fusion of Perception and Action.** At this point, $\tilde{H}_t$ contains the relevant information regarding the action history of the navigation episode. However, this information must be enriched with the perception coming from the environment. We merge textual and visual information with $\tilde{H}_t$ via attention, allowing mutual influence between perception and motion. We build two branches of multi-head cross-attention accepting respectively $X$ and $I$, as key/value pairs and using $\tilde{H}_t$ as query. The image-action cross-attention is motivated by the fact that the agent needs to look around before decoding the next action. Since $\tilde{I}_t$ already contains information coming from the instruction, this cross-attention layer is sufficient to achieve decent results on the VLN task (as demonstrated by our ablation studies). However, we find out that adding a separate text-action cross-attention layer helps generalization in unseen environments. After this step, we concatenate the two representations and apply a FC layer to obtain the output sequence before decoding the next action.

**Action Selection.** To select the next low-level action, we project the final representation $\tilde{a}_t$ in a six-dimensional space corresponding with the agent locomotor space containing the following actions: $\text{turn right/left } 30^\circ$, $\text{tilt up/down}$, $\text{step forward}$, and $\text{end episode}$. The output probability distribution over the action space can therefore be written as:

$$p_t = \text{softmax}(\tilde{a}_t W_p + b_p),$$

where $W_p \in \mathbb{R}^{d_{\text{model}} \times n_{\text{actions}}}$ and $b_p \in \mathbb{R}^{n_{\text{actions}}}$ are learned parameters ($n_{\text{actions}} = 6$). During training, we sample the next action to perform $a_t$ from $p_t$, while we select $a_t = \arg\max(p_t)$ during evaluation and test.

3.3. Training

Our training setup includes two distinct objective functions. The first estimates the policy by imitation learning, while the second enforces similarity between the ground-truth and predicted trajectories via reinforcement learning.

**Imitation Learning.** To approximate a good policy, we first train our agent using strong supervision. At each timestep $t$, the simulator outputs the ground-truth action $y_t$. In the low-level setup, the ground-truth action is the one that allows getting to the next target viewpoint in the minimum amount of steps. In this phase, we aim to minimize the cross-entropy loss of the predicted distribution $p_t$ w.r.t. the ground-truth action $y_t$.

**Extrinsic Reward.** After a first training phase with supervised learning, we finetune our agent using an extrinsic reward function. Recently, Magalhaes et al. (2019) propose to employ Dynamic Time Warping (DTW) (Berndt and Clifford, 1994) to evaluate the trajectories performed by navigation agents. In particular, they define the normalized Dynamic Time Warping (nDTW) as:

$$\text{nDTW}(R, Q) = \exp\left(\frac{-\text{DTW}(R, Q)}{|R| \cdot d_{\text{th}}}\right),$$

where $R$ and $Q$ are respectively the reference and the query paths, $|R|$ is the length of the reference path, and $d_{\text{th}}$ is the success threshold distance. At each navigation step $t$, the agent receives a reward equal to the gain in terms of nDTW:

$$R_t = \text{nDTW}(q_0, \ldots, t, R) - \text{nDTW}(q_0, \ldots, t-1, R).$$

Additionally, we give an episode-level reward to the agent if it terminates the navigation within a success threshold distance $d_{\text{goal}}$ from the goal, given by $R_e = \max(0, 1 - d_{\text{goal}}/d_{\text{th}})$, where $d_{\text{goal}}$ is the final distance between the agent and the target. We can write our final reinforcement learning objective function as:

$$L_{rl} = -E_{a_t \sim \pi_t}[A_t].$$

where the advantage function $A_t = R_t + R_e$. Based on REINFORCE algorithm (Williams, 1992), we derive the gradient of our reward-based objective as:

$$\nabla_{\theta} L_{rl} = -A_t \nabla \log \pi_t(a_t|s_t).$$

4. Low-level and High-Level Navigation

Section 3 describes our approach to low-level VLN. Here, we discuss the main technical differences with the high-level counterpart and explain how PTA can switch from one setting to the other. Differently from the low-level architectures, a high-level method aims to predict the next node to traverse in the navigation graph, as physical navigation takes place with a teleport mechanism. The choice at time step $t$ is done with a similarity measure between the agent internal state $s_t$ and the appearance vector for the navigable locations $v_t$. This similarity function is normally mapped into a bilinear dot-product:

$$p_t = \text{softmax}(f(s_t)^T g(v_t))$$

where $f(\cdot)$ and $g(\cdot)$ are generic transformations.

In principle, it is possible to substitute the final softmax classifier of a low-level architecture (Eq. 7) with Eq. 12 and change the corresponding action space. According to this observation, we can swap the action space of a model to test its adaptability to different navigation settings. While traditional approaches start from the hidden state of the recurrent policy to estimate the agent’s internal state $s_t$, we can derive it directly from $\tilde{a}_t$:

$$s_t = \tilde{a}_t W_s + b_s,$$

where $W_s$ and $b_s$ are learned parameters. As $v_t$, we select the unattended visual features augmented with the coordinate vector described by Eq. 8 and apply the following transformation:

$$g(v_t) = \max(0, v_t W_v + b_v),$$
where $W_i$ and $b_i$ are learned parameters.

In our architecture, $\tilde{a}_t$ can fit to represent any kind of information about the current navigation. This is because it can draw knowledge from the perceptual modalities and the history of past actions directly and without the bottleneck represented by a recurrent network. Our experiments on this subject (Sec.5.5) show that our model stands out from the literature in terms of adaptability. In other words, PTA can adapt to a different action space because it does not make any assumptions on the underlying simulating platform. Instead, our architecture relies on efficient visuo-linguistic fusion mechanisms designed to be agnostic towards the final action space. We will see that methods making stronger assumptions on the action space experience a larger drop in performance than PTA.

5. Experiments and Discussion

5.1. Experimental Setup

Datasets. In our experiments, we primarily test our architecture on the R2R dataset for VLN (Anderson et al. 2018c). This dataset builds on the Matterport3D dataset of spaces (Chang et al. 2017), which contains complete scans of 90 different buildings. The visual data is enriched with more than 7,000 navigation paths and 21,000 natural language instructions. The episodes are divided into a training set, two validation splits (validation-seen, with environments that the agent has already seen during training, and validation-unseen, containing only unexplored buildings), and a test set. The testing phase takes place in previously unseen environments and is accessible via a test-server with a public leaderboard. While the instructions in R2R are quite long and complex (about 29 words on average), navigation episodes usually involve a limited number of steps – max 6 steps for high-level action space and max 23 steps for the low-level setup. In the R4R dataset, Jain et al. (2019), merge the paths in R2R to create a more complex and challenging setup. Episodes become considerably longer, pushing the navigation trajectories, as this metric is also sensitive to intermediate nodes in the reference path. Additionally, Magalhaes et al. (2019) propose Dynamic Time Warping (DTW) and derived metrics (Normalized DTW and Success weighted by normalized DTW) to measure the similarity between reference and predicted paths. These three last metrics are more meaningful on the R4R dataset than SR and SPL (Jain et al. 2019).

Implementation Details. In the instruction encoder, $d_{\text{in}} = 300$. In each component of our model, we project the input features into a $d_{\text{model}}$-dimensional space, with $d_{\text{model}} = 512$. For multi-head attention, we employ $h = 8$ heads, thus $d_h = d_c = d_{\text{model}}/h = 64$. The internal representation of feed-forward networks has size $d_f = 2048$. After each sub-module, we add a residual connection followed by layer normalization. We also apply dropout (Srivastava et al. 2014) with drop probability $p = 0.1$ after each linear layer. During training, we use Adam optimizer (Kingma and Ba 2015) with learning rate $10^{-4}$, we set the batch size to 32 and reduce the learning rate by a factor 10 if the SPL on the validation unseen split does not improve for 5 consecutive epochs. We stop the training after 30 epochs without improvement on the same metric. When finetuning using REINFORCE, we set the initial learning rate to $10^{-7}$.

5.2. Ablation Study

In our ablation study, we experimentally validate the importance of each module in our architecture. First, we ablate modality in our decoder and we do not apply late fusion before decoding the next action. In a second experiment, we remove cross-attention between visual and linguistic information in the encoder. Finally, we show the impact of synthetic data augmentation (Fried et al. 2018) and the role of REINFORCE. Results are shown in Table I and discussed below.

Multimodal Decoder. In our first ablation study, we use only one of the two decoder branches at the time, and we do not perform late fusion between linguistic and visually-grounded information. When removing the textual branch (Table I line 3), our agent performs worse on unseen environments, hence losing potential in terms of generalization. When removing the visual modality, our PTA agent is blinded and can only count on the natural language instruction. This setup leads to success only when the instruction does not involve references to objects or visual properties of the environment – a nearly empty subset of the dataset. Indeed, the metrics for our blind agent are extremely low, and they do not vary between seen and unseen environments (Table I line 4). This result is meaningful in light of recent studies proving that some single-modality agents perform better than their multimodal version by removing the visual perception and overfitting on dataset biases (Thomason et al. 2018).

Early Fusion of Textual and Visual Perception. As a second experiment, we remove the early fusion mechanism, namely the cross-attention layer between the textual and visual branches of our encoder, to check its contribution. If this fusion layer is redundant, we expect that the late fusion stage will compensate for the loss. Instead, we experience a drop in performance: −12% in SPL in unseen environments (Table I line 5). We thus prove the importance of early textual and visual fusion in our architecture for VLN.
Contextual History for Action Decoding. \( H_i \) stores past actions as a series of one-hot vectors, and it is extremely helpful to model navigation history. It acts as a sort of memory for the agent, so that it knows what actions have already been made. A similar trick in LSTM-based VLN consists in adding the last action as input to the policy RNN at each step. In our model, removing \( H_i \) and using only the last action (losing all the history) causes a drop in performance: −14% and −17% on SPL and SR respectively for the Val-Unseen split (Table 1 line 6).

Data Augmentation. In line with previous literature, we find the use of additional synthetic instructions useful to initialize our agent. The synthetic training set was provided by [Fried et al. 2018] using a Speaker module. After a first training with the full set of instructions (synthetic and human-generated), we finetune using only the original R2R train set. Results are reported in Table 1 line 7.

Extrinsic Reward. While imitation learning allows approximating a good policy, there is still room for improvement via reinforcement learning. [Wang et al. 2019] were the first to use REINFORCE in the context of VLN to refine their navigation policy based on cross-modal matching. In line with them, we find REINFORCE beneficial for our model: our final agent sticks more closely to the reference trajectory and penalizes long navigations (Table 1 line 8).

5.3. Results on R2R

In our experiments on the R2R dataset [Anderson et al. 2018c], we test the ability of our agent to navigate unseen environments in light of previously unseen natural language instructions. The main test-bed for this experiment is represented by the R2R evaluation leaderboard, which is publicly available online.

Comparison with SOTA. In Table 2, we report our results on the R2R test set, together with the results achieved by other state-of-the-art architectures on VLN. Other methods that operate in the low-level action space are the sequence-to-sequence architecture proposed by [Anderson et al. 2018c], the RPA model using a mixture of model-free and model-based reinforcement learning [Wang et al. 2018], and the recurrent architecture with dynamic convolutional filters proposed by [Landi et al. 2019]. Our method overcomes the state-of-the-art on low-level VLN by a large margin (5% in terms of SPL and SR).

Although a direct comparison between the two settings is not feasible, we notice that PTA performs better than some high-level architectures in terms of SPL. Notably, we achieve this result without making any assumption on the underlying simulating platform and decoding a longer sequence of atomic moves, instead of target viewpoints. Moreover, high-level architectures can often count on efficient graph-search methods (impractical when dealing with continuous controls) to decode the final trajectory, and on additional modules that are not present in our method. While these are effective for high-level VLN, their generalizability to a low-level setup, closer to real-world application, is yet to be tested.

Switching from Low-level to High-level. Our second experiment on R2R aims to test the effects of retraining existing models after switching their final action spaces (from high-level to low-level and vice-versa). To that end, we change the final classifier of PTA as described in Section 3. In this new setting, the output of the action decoder becomes a probability distribution over the adjacent nodes of the navigation graph. Once the agent decides where to go, the displacements are made automatically and there is no need to decode lower-level actions such as rotations. We train PTA from scratch in this setup, without any further hyperparameter tuning. In Table 3 we detail the full set of metrics obtained using PTA with the high-level classifier.
and compare with the model incorporating the low-level control system. The small gap between the metrics in the two setups suggests that PTA does not take any particular advantage from the underlying action space. Of course, metrics that directly evaluate the final trajectory (like DTW-based metrics) benefits from using high-level actions with automatic oracle displacements.

In principle, every model should exhibit a decent level of elasticity towards different locomotor settings. In practice, we find out that architectural choices that strongly help high-level VLN often end up hindering the other setup. This is especially true when the agent exploits high-level reasoning and makes strong assumptions on the nature of the underlying simulator. As a result, current high-level methods experience a drop in performance when adopting a simple, atomic action space (see Figure 3). PTA, instead, does not rely on such assumptions and builds on more efficient modules to merge multi-modal information entailed in the VLN task. The plots in Figure 3 show that our model exhibits far greater flexibility to the final action space than other architectures. The considerably narrow step between the blue and the gray bars (representing the low-level and the high-level action spaces respectively) denotes that a change in the final action space does not prevent PTA from reaching its goal. We compare with the Speaker-Follower (Fried et al., 2018) and the Self-Monitoring agent (Ma et al., 2019a) from the high-level setup, which experience a sizeable loss in performance. In fact, results drop of 11% and 13% respectively in terms of SR when adapted for low-level use. We also compare PTA with a recurrent architecture exploiting dynamic convolution (Landi et al., 2019) from the low-level category. The lower degree of adaptability shown by this competitor is motivated by the fact that it operates a strong compression on the visual input basing on the current instruction. In this step, much information that could ease high-level action selection is lost.

To conduct this experiment we adjust the codes from Landi et al. (2019) and Ma et al. (2019a), which are publicly available online, and report the results in the paper for Fried et al. (2018). We choose the Speaker-Follower and the Self-Monitoring agents because they are flexible frameworks by design, and for this reason they are the most suitable models among their high-level peers for this comparison. We believe that the findings and insights provided in this experiment will motivate further experiments in this direction, and help to unravel the main reasons of improvements in new architectures for VLN.

**Qualitative Results.** In Fig. 4, we report a qualitative result from the R2R val-unseen set. Remarkably, PTA is able to ground concepts such as “the second doorway on your left” and terminates the navigation episode successfully. Since our agent operates in a low-level setup, it needs to orientate towards the next viewpoint before stepping ahead, making the decoding phase more challenging.

### 5.4. Results on R4R

R4R (Jain et al., 2019) builds upon R2R and aims to provide an even more challenging setting for embodied navigation agents. While navigation in R2R is usually direct and takes the shortest path between the starting position and the goal viewpoint, trajectories in R4R may bend and return on the agent’s previous steps. This change calls for adaptation in evaluation metrics: SPL and SR are now less indicative because the agent might stop close the goal in the first half of the navigation and still fail to complete the second part. In this sense, an important role is played by recently proposed metrics: CLS (Jain et al., 2019) and nDTW (Magalhaes et al., 2019) take into account the agent’s steps and are sensitive to intermediate errors in the navigation path. For this reason, these last metrics are more meaningful when evaluating navigation agents on R4R.

**Comparison with SOTA.** In this experiment, we compare PTA with other state-of-the-art architectures for VLN and report the results in Table 4. In the low-level setup, we compare to the recurrent architecture with dynamic convolution proposed by Landi et al. (2019). Results show that our approach performs better on all of the main metrics. In particular, a lower NE and a

![Fig. 3. Visualization of the navigation error (left) and success rate (right) on the R2R val-unseen split. A larger difference between the blue and gray bars denotes a lower degree of adaptability. The metric gap is reduced when using PTA.](image-url)
higher CLS indicate that our agent tends to get closer to the goal while sticking to the natural language instruction better than the competitor. We also report the results obtained by our model incorporating the high-level decision space. We compare with Speaker-Follower (Fried et al., 2018) and RCM (Wang et al., 2019), as implemented in (Jain et al., 2019). PTA performs better than its high-level competitors on the majority of the metrics. In particular, the higher CLS score shows that PTA can generally select a path that follows the instruction better than the competitors. When considering the reference metrics proposed for R4R (Jain et al., 2019), our architecture achieves the best results on both the setups.

6. Conclusion

In this paper, we have presented Perceive, Transform, and Act (PTA), the first fully-attentive model for VLN. In particular, we tackle the challenging task of low-level VLN, in which high-level information about the environment is no longer accessible to the agent. We also report the results obtained by our model incorporating the high-level decision space. We compare with Speaker-Follower (Fried et al., 2018) and RCM (Wang et al., 2019), as implemented in (Jain et al., 2019). PTA performs better than its high-level competitors on the majority of the metrics. In particular, the higher CLS score shows that PTA can generally select a path that follows the instruction better than the competitors. When considering the reference metrics proposed for R4R (Jain et al., 2019), our architecture achieves the best results on both the setups.

Acknowledgement

This work has been supported by “Fondazione di Modena” and by the national project “IDEEA: Innovation for Data Elaboration in Heritage Areas” (PON ARS01_00421), cofunded by the Italian Ministry of University and Research.

References

Anderson, P., Chang, A., Chaplot, D.S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., et al., 2018a. On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757.
Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L., 2018b. 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.
Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., S¨underhauf, N., Reid, I., Gould, S., van den Hengel, A., 2018c. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C.L., Parikh, D., 2015. VQA: Visual Question Answering, in: Proceedings of the International Conference on Computer Vision.
Bengio, Y., Simard, P., Frasconi, P., et al., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5, 157–166.
Berndt, D.J., Clifford, J., 1994. Using dynamic time warping to find patterns in time series, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining.
Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., Zhang, Y., 2017. Matterport3D: Learning from RGB-D Data in Indoor Environments, in: Proceedings of the International Conference on 3D Vision.
