Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments

Jacob Krantz[1] Erik Wijmans[2,3] Arjun Majumdar[2] Dhruv Batra[2,3] Stefan Lee[1]

[1]Oregon State University [2]Georgia Institute of Technology [3]Facebook AI Research

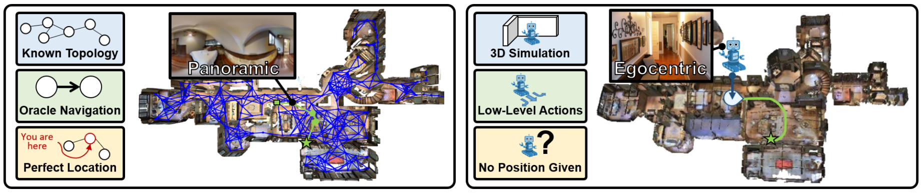
Abstract. We develop a language-guided navigation task set in a continuous 3D environment where agents must execute low-level actions to follow natural language navigation directions. By being situated in continuous environments, this setting lifts a number of assumptions implicit in prior work that represents environments as a sparse graph of panoramas with edges corresponding to navigability. Specifically, our setting drops the presumptions of known environment topologies, short-range oracle navigation, and perfect agent localization. To contextualize this new task, we develop models that mirror many of the advances made in prior settings as well as single-modality baselines. While some transfer, we find significantly lower absolute performance in the continuous setting – suggesting that performance in prior 'navigation-graph' settings may be inflated by the strong implicit assumptions. Code at jacobkrantz.github.io/vlnce

Keywords: Vision-and-Language Navigation, Embodied Agents

1 Introduction

Springing forth from the pages of science fiction and capturing the daydreams of weary chore-doers everywhere, the promise and potential of general-purpose robotic assistants that follow natural language instructions has been long understood. Taking a small step towards this goal, recent work has begun developing artificial agents that follow natural language navigation instructions in perceptually-rich, simulated environments [4, 6]. An example instruction might be "*Go down the hall and turn left at the wooden desk. Continue until you reach the kitchen and then stop by the kettle.*" and agents are evaluated by their ability to follow the described path in (potentially novel) simulated environments.

Many of these tasks have been developed from datasets of panoramic images captured in real scenes – e.g. Google StreetView images in Touchdown [6] or Matterport3D panoramas captured in homes in Vision-and-Language Navigation (VLN) [4]. This paradigm enables efficient data collection and high visual fidelity compared to 3D scanning or creating synthetic environments; however, scenes are only observed from a sparse set of points relative to the full 3D environment (∼117 viewpoints per environment in VLN). As a consequence, environments in these tasks are defined in terms of a navigation graph (or nav-graph for short) – a static topological representation of 3D space. As shown in Fig. 1(a), nodes in the nav-graph correspond to 360° panoramic images taken at fixed locations



(a) Vision-and-Language Navigation (VLN) (b) VLN in Continuous Environments (VLN-CE)

Fig. 1. The VLN setting **(a)** operates on a fixed topology of panoramic images (shown in blue) – assuming perfect navigation between nodes (often meters apart) and precise localization. Our VLN-CE setting **(b)** lifts these assumptions by instantiating the task in continuous environments with low-level actions – providing a more realistic testbed for robot instruction following.

and edges between nodes indicate navigability. This nav-graph based formulation introduces a number of assumptions that make it a poor proxy for what a robotic agent would encounter while navigating the real world.

Focusing our discussion on Vision-and-Language Navigation (VLN), the existence and common usage of the nav-graph imply the following assumptions:

- **Known topology.** Rather than continuous environments in which agents can move freely, agents operate on a fixed topology of traversable nodes (shown in blue in Fig. 1(a)). Aside from being a poor match to robot control, this also provides prior information about environment layout to agents – even in "unseen" test settings. For example, it is common practice to define agent actions by selecting directions in the current panorama and 'snapping' to the nearest adjacent nav-graph node in that direction. How an actual agent might acquire and update such a topology in new environments is an open question.
- **Oracle navigation.** Movement between adjacent nodes in the nav-graph is deterministic, implying the existence of an oracle navigator capable of accurately traversing multiple meters in the presence of obstacles – abstracting away the problem of visual navigation. Further, this movement between nodes is perceptually akin to teleportation – the current panorama is simply replaced by the panorama at the new location meters away. This is in contrast to the continuous stream of observations a real agent would encounter while moving.
- **Perfect localization.** Agents are given their precise location and heading at all times. Most works use this data to encode precise geometry between nodes in the nav-graph as part of the decision making process, e.g. moving 30°W and 1.12m forward from the previous node. Others use precise agent localization to construct spatial maps of the environment on which to reason about paths [3]. However, precise localization indoors is still a challenging problem.

Taken together, these assumptions make current settings poor reflections of the real world both in terms of control (ignoring actuation, navigation, and localization error) and visual stimuli (lacking the poor framing and long observation-sequences agents will encounter). In essence, the problem is reduced to that of visually-guided graph search. As such, closing the loop by transferring these trained agents to physical robotic platforms has not been examined.

These assumptions are often justified by invoking existing technologies as potential oracles. For example, simultaneous localization and mapping (SLAM) or odometry systems can offer strong localization in appropriate conditions [16,21]. Likewise, algorithms for path planning and control can navigate short distances in the presence of obstacles [11,25,31]. Further, it is reasonable to suggest that issuing commands at the level of relative waypoints (in analogy to nav-graph nodes) is the proper interface between language-guided AI navigators and lower-level agent control. However, these techniques are each independently far from perfect and such an agent would need to learn the limitations of these lower-level control systems – facing consequences when proposed waypoints cannot be reached effectively. Integrative studies that combine and evaluate techniques for control and mapping with learned AI agents are not possible in current nav-graph based problem settings. In this work, we develop a continuous setting that enables such studies and take a first step towards integrating VLN agents with control.

Vision-and-Language Navigation in Continuous Environments. In this work, we focus in on the Vision-and-Language Navigation (VLN) [4] task and lift these implicit assumptions by instantiating it in continuous 3D environments [5,19]. Consequently, we call this task Vision-and-Language Navigation in Continuous Environments (VLN-CE). Agents in our task are free to navigate to any unobstructed point through a set of low-level actions (e.g. move `forward` 0.25m, `turn-left` 15 degrees) rather than teleporting between fixed nodes. This setting introduces many challenges ignored in prior work. Agents in VLN-CE face significantly longer time horizons; the average number of actions along a path in VLN-CE is $\sim$55 compared to the 4-6 node hops in VLN (as illustrated in Fig. 1). Moreover, the views the agent receives along the way are not well-posed by careful human operators as in the panoramas, but rather a consequence of the agent's actions. Agents must also learn to avoid getting stuck on obstacles, something that is structurally impossible in VLN's navigability defined nav-graph. Further, agents are not provided their location or heading while navigating.

We develop agent architectures for this task and explore how popular mechanisms for VLN transfer to the VLN-CE setting. Specifically, we develop a simple sequence-to-sequence baseline architecture as well as a cross-modal attention-based model. We perform a number of input-modality ablations to assess the biases and baselines in this new setting (including models without perception or instructions as suggested in [27]). Unlike in VLN where depth is rarely used, our analysis reveals depth to be an integral signal for learning embodied navigation – echoing similar findings in point-goal navigation tasks [19,31]. We also apply existing training augmentations [17,24,26] popular in VLN to our setting, finding mixed results. Overall, our best performing agent successfully navigates to the goal in approximately a third of episodes in unseen environments.

To further examine the relationship between the nav-graph-based VLN task and VLN-CE, we also transfer paths from agents trained in continuous environments back to the nav-graph to provide a direct comparison. We find significant

Table 1. Comparison of language-guided visual navigation tasks. Ours is the only to provide unconstrained navigation in real environments for crowdsourced instructions.

Task	Instructions	Environment	Navigation
LANI [20]	Crowdsourced	Synthetic	Unconstrained
StreetNav [13]	Templated	Real	Nav-Graph Based
Touchdown [6]	Crowdsourced	Real	Nav-Graph Based
VLN [4]	Crowdsourced	Real	Nav-Graph Based
VLN-CE (ours)	Crowdsourced	Real	Unconstrained

gaps in performance between these settings indicative of the strong prior provided by the nav-graph. This suggests prior results in VLN may be overly optimistic in terms of progress towards instruction-following robots functioning in the wild.

Contributions. To summarize our contributions, we:

- Lift the VLN task to continuous 3D environments – removing many unrealistic assumptions imposed by the nav-graph-based representation.
- Develop model architectures for the VLN-CE task and evaluate a suite of single-input ablations to assess the biases and baselines of the setting.
- Investigate how a number of popular techniques in VLN transfer to this more challenging long-horizon setting – identifying significant gaps in performance.

2 Related Work

Language-guided Visual Navigation Tasks. Language-guided visual navigation tasks require agents to follow navigation directions in simulated environments. There have been a number of recent tasks proposed in this space [4,6,13,20]. Chen et al. [6] introduce the Touchdown task which studies outdoor language-guided navigation in Google Street View panoramas. Hermann et al. [13] investigates the same setting; however, the instructions are automatically generated from Google Map directions rather than being crowdsourced from human annotators. Both adopt a nav-graph setting due to the source data being panoramic images – constraining agent navigation to fixed points. Misra et al. [20] introduce a simulated environment with unconstrained navigation and a dataset of crowdsourced instructions; however, the environments are unrealistic, synthetic scenes. Most related to our work is the Vision-and-Language Navigation (VLN) task of Anderson et al. [4]. VLN provides nav-graph trajectories and crowdsourced instructions in Matterport3D [5] environments as the Room-to-Room (R2R) dataset. We build VLN-CE directly on these annotations – converting R2R panorama-based trajectories to fine-grained paths in continuous Matterport3D environments (Fig. 1(a) to Fig. 1(b)). This shift to continuous environments with unconstrained agent navigation lifts a number of unrealistic assumptions.

The variation in these tasks is primarily in the source of navigation instructions (crowdsourced from human annotators vs. generated via template), environment realism (hand-designed synthetic worlds vs. captures from real locations), and constraints on agent navigation (nav-graph based navigation vs. unconstrained agent motion). Tab. 1 provides a comparison between tasks along these axes. Our proposed VLN-CE task provides the first setting with crowdsourced instructions in realistic environments with unconstrained agent navigation.

Approaches to Vision-and-Language Navigation. VLN has seen considerable progress. Multimodal attention mechanisms have become popular to provide better grounding between instructions and the observations [29]. Orthogonal to new modeling architectures, improvements have also come from new training approaches and data augmentation methods. One prevalent technique is to utilize inverse "speaker" models to re-rank candidate trajectories or augment the available training data by generating instructions for novel trajectories [9]. Tan et al. [26] improve upon this idea by improving the diversity of the generated instructions. Ma et al. [17] show that an additional training signal can be gained by explicitly estimating progress toward the goal (referred to as self-monitoring). We adapt these methods to VLN-CE and examine their impact.

Other Language-based Embodied AI. A number of other embodied tasks have considered language-conditioned navigation. For instance, referring to specific rooms or objects that agents must then navigate to [7, 10, 30]. However, these settings use language to specify end-goals or query agent knowledge rather than to provide navigational directions. For example, specifying "*lamp*" or "*What color is the lamp in the living room?*" rather than multi-step, grounded navigation instructions. This loose coupling of intermediate agent action with the language instruction differentiates these tasks from language-guided navigation settings.

3 VLN in Continuous Environments (VLN-CE)

We consider a continuous setting for the vision-and-language navigation task which we refer to as Vision-and-Language Navigation in Continuous Environments (VLN-CE). Given a natural language navigation instruction, an agent must navigate from a start position to the described goal in a continuous 3D environment by executing a sequence of low-level actions based on egocentric perception alone. In overview, we develop this setting by transferring nav-graph-based Room-to-Room (R2R) [4] trajectories to reconstructed continuous Matterport3D environments in the Habitat simulator [19]. We discuss these details below.

Continuous Matterport3D Environments in Habitat. We set our problem in the Matterport3D (MP3D) [5] dataset, a collection of 90 environments captured through over 10,800 high-definition RGB-D panoramas. In addition to the panoramic images, MP3D also provides corresponding mesh-based 3D environment reconstructions. To enable agent interaction with these meshes, we develop the VLN-CE task on top of the Habitat Simulator [19], a high-throughput simulator that supports basic movement and collision checking for 3D environments including MP3D. In contrast to the simulator used in VLN [4], Habitat allows agents to navigate freely in the continuous environments.

Observations and Actions. We select observation and action spaces to emulate a ground-based, zero-turning radius robot with a single, forward-mounted RGBD camera, similar to a LoCoBot [1]. Agents perceive the world through egocentric RGBD images from the simulator with a resolution of 256×256 and a horizontal field-of-view of 90 degrees. Note that this is similar to the egocentric RGB perception in the original VLN task [4] but differs from the panoramic observation space adopted by nearly all follow-up work [9, 17, 26, 29].

While the simulator is quite flexible in terms of agent actions, we consider four simple, low-level actions for agents in VLN-CE – move `forward` 0.25m, `turn-left` or `turn-right` 15 degrees, or `stop` to declare that the goal position has been reached. These actions can easily be implemented on robotic agents with standard motion controllers. In contrast, actions to move between panoramas in [4] traverse 2.25m on average and can include avoiding obstacles.

3.1 Transferring Nav-Graph Trajectories

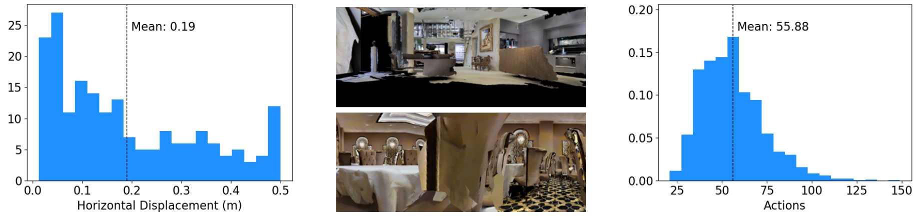
Rather than collecting a new dataset of trajectories and instructions, we instead transfer those from the nav-graph-based Room-to-Room dataset to our continuous setting. Doing so enables us to compare existing nav-graph-based techniques with our methods that operate in continuous environments on the same instructions.

Matterport3D Simulator and the Room-to-Room Dataset. The original VLN task is based on panoramas from Matterport3D (MP3D) [5]. To enable agent interaction with these panoramas, Anderson et al. [4] developed the Matterport3D Simulator. Environments in this simulator are defined as nav-graphs $E = \{\mathcal{V}, \mathcal{E}\}$. Each node $v \in \mathcal{V}$ corresponds to a panoramic image I captured by a Matterport camera at location x, y, z – i.e. $v = \{I, x, y, z\}$. Edges in the graph correspond to navigability between nodes. Navigability was defined by ray-tracing between node locations at varying heights to check for obstacles in the reconstructed MP3D scene and then manually inspected. Edges were manually added or removed based on judgement whether an agent could navigate between nodes – including by avoiding minor obstacles[1]. Agents act by teleporting between adjacent nodes in this graph. Based on this simulator, Anderson et al. [4] collect the Room-to-Room (R2R) dataset containing 7189 trajectories each with three human-generated instructions on average. These trajectories consist of a sequence of nodes $\tau = [v_1, \ldots, v_T]$ with length T averaging between 4 and 6 nodes.

Converting Room-to-Room Trajectories to Habitat. Given a mapping between the coordinate frames of Matterport3D Simulator and MP3D in Habitat, it is seemingly simple to transfer the Room-to-Room trajectories – after all, each node has a corresponding xyz location. However, node locations often do not correspond to reachable locations for a ground-based agent – existing at variable height depending on tripod configuration or placed on top of flat furniture like tables. Further, the reconstructions and panoramas may differ if objects are moved between camera captures.

For each node, $v = \{I, x, y, z\}$, we would like to identify the nearest, navigable point on the reconstructed mesh – i.e. the closest point that can be occupied by a ground-based agent represented by a 1.5m tall cylinder of diameter of 0.2m. Directly projecting to the nearest mesh location fails for 73% of nodes where

[1] Details included from correspondence with the author of [4]



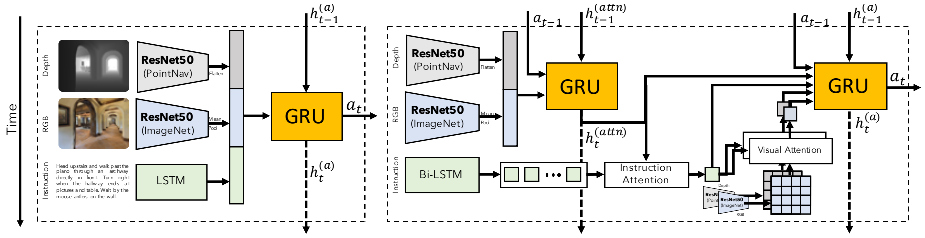
(a) Node Location Displacement (b) Discontinuities (c) Trajectory Length in Actions

Fig. 2. We successfully transfer 77% of the R2R trajectories. (a) Most panorama nodes transfer directly, but 3% require horizontal adjustment – with an average displacement of 0.19m. (b) Some trajectories are not navigable due to differences between the panoramas and reconstructed environments, e.g. holes in the 3D mesh (top) or objects like chairs being moved between panorama captures (bottom). (c) Optimal paths in our setting require 10x more agent actions per trajectory – 55.88 compared to 5 in R2R.

failure is projecting to distant (>0.5m) or non-navigable points. Many of these points project to surfaces other than the floor due to camera height. Instead, we cast a ray up to 2m directly downward from the node. At small, fixed intervals along this ray, we project to the nearest mesh point. If multiple navigable points are identified, we take the one with minimal horizontal displacement from the original location. If no navigable point is found with less than a 0.5m displacement, we consider this MP3D node unmappable to the 3D mesh and thus invalid. We manually reviewed invalid nodes and made corrections if possible, e.g. shifting nodes around furniture. After these steps, 98.3% of nodes transferred successfully. We refer to these transferred nodes as waypoint locations. In Fig. 2(a), points needing adjustment (3% of points) require small displacement, averaging 0.19m.

Given a trajectory of converted waypoints $\tau = [w_1, \ldots, w_T]$, we verify that an agent can actually navigate between each location. We employ an A*-based search algorithm to compute an approximate shortest path to a goal. We run this algorithm between each waypoint in a trajectory to the next (e.g. w_i to w_{i+1}). A trajectory is considered navigable if for each pairwise navigation, an agent can follow the shortest path to within 0.5m of the next waypoint (w_{i+1}). In total, we find 77% of the R2R trajectories navigable in the continuous environment.

Non-Navigable Trajectories. Among the 23% of trajectories that were not navigable, we observed two primary failure modes. First and most simply, 22% included one of the 1.7% of invalid nodes that could not be projected to MP3D 3D meshes. The remaining unnavigable trajectories spanned disjoint regions of the reconstruction – i.e. lacking a valid path from some waypoint w_i to w_{i+1}. As shown in Fig. 2(b), this may be due to holes or other mesh errors dividing the space. Alternatively, objects like chairs may be moved in between panorama captures – possibly resulting in a reconstruction that places the object mesh on top of individual panorama locations. Nodes in the R2R nav-graph were manually connected if there appeared to be a path between them, even if most other panoramas (and thus the reconstruction) showed blocking objects.



(a) Sequence-to-Sequence Baseline

(b) Cross-Modal Attention Model

Fig. 3. We develop a simple baseline agent (a) as well as an attentional agent (b) comparable to that in [29]. Both receive RGB and depth frames represented by pretrained networks for image classification [8] and point-goal navigation [31], respectively.

3.2 VLN-CE Dataset

In total, the VLN-CE dataset consists of 4475 trajectories converted from R2R train and validation splits. For each trajectory, we provide the multiple R2R instructions and a pre-computed shortest path following the waypoints via low-level actions. As shown in Fig. 2(c), the low-level action space of VLN-CE makes for a longer horizon task – with 55.88 steps on average compared to 4-6 in R2R.

4 Instruction-guided Navigation Models in VLN-CE

We develop two models for VLN-CE. A simple sequence-to-sequence baseline and a more powerful cross-modal attentional model. While there are many differences in the details, these models are conceptually similar to early [4] and more recent [29] work in the nav-graph based VLN task. Exploring these gives insight into the difficulty of this setting in isolation and by comparison relative to VLN. Further, these models allow us to test whether improvements from early to later architectures carry over to a more realistic setting. Both of our models make use of the same observation and instruction encodings described below.

Instruction Representation. We convert tokenized instructions to GLoVE [23] embeddings which are processed by recurrent encoders for each model. We denote these encoded tokens as $\mathbf{w}_1, \ldots, \mathbf{w}_T$ for a length T instruction.

Observation Encoding. For RGB, we apply a ResNet50 [12] pretrained on ImageNet [8] to collect semantic visual features. We denote the final spatial features of this model as $\mathcal{V} = \{\mathbf{v}_i\}$ where i indexes over spatial locations. Likewise for depth, we use a modified ResNet50 that was trained to perform point-goal navigation (i.e. to navigate to a location given in relative coordinates) [31] and denote these as $\mathcal{D} = \{\mathbf{d}_i\}$.

4.1 Sequence-to-Sequence Baseline

We consider a simple sequence-to-sequence model shown in Fig. 3(a). This model consists of a recurrent policy that takes visual observations (depth and RGB) and instructions at time step t to predict an action a. We can write the agent as

$$\bar{\mathbf{v}}_t = \text{mean-pool}\left(\mathcal{V}_t\right), \quad \bar{\mathbf{d}}_t = \left[\mathbf{d}_1, \dots, \mathbf{d}_{wh}\right], \quad \mathbf{s} = \text{LSTM}\left(\mathbf{w}_1, \dots, \mathbf{w}_T\right) \tag{1}$$

$$\mathbf{h}_t^{(a)} = \text{GRU}\left(\left[\bar{\mathbf{v}}_t, \bar{\mathbf{d}}_t, \mathbf{s}\right], \mathbf{h}_{t-1}^{(a)}\right), a_t = \underset{a}{\text{argmax}} \;\; \text{softmax}\left(W_a \mathbf{h}_t^{(a)} + \mathbf{b}_a\right) \tag{2}$$

where $[\cdot]$ denotes concatenation and $\mathbf{s}$ is the final hidden state of an LSTM instruction encoder. This model enables straight-forward input-modality ablations.

4.2 Cross-Modal Attention Model

The previous model lacks powerful inductive biases common to vision-and-language tasks including cross-modal attention and spatial reasoning which are intuitively important for language-guided visual navigation. In Fig. 3(b) we consider a model incorporating these mechanisms. This model consists of two recurrent networks – one tracking visual history and the other tracking attended instruction and visual features. We write the first recurrent network as:

$$\mathbf{h}_t^{(attn)} = \text{GRU}\left(\left[\bar{\mathbf{v}}_t, \bar{\mathbf{d}}_t, \mathbf{a}_{t-1}\right], \mathbf{h}_{t-1}^{(attn)}\right) \tag{3}$$

where $\mathbf{a}_{t-1} \in \mathbb{R}^{32}$ and is a learned linear embedding of the previous action. We encode instructions with a bi-directional LSTM and reserve all hidden states:

$$\mathcal{S} = \{\mathbf{s_1}, \dots, \mathbf{s_T}\} = \text{BiLSTM}\left(\mathbf{w}_1, \dots, \mathbf{w}_T\right) \tag{4}$$

We then compute an attended instruction feature $\hat{\mathbf{s}}_t$ over these representations which is then used to attend to visual ($\hat{\mathbf{v}}_t$) and depth ($\hat{\mathbf{d}}_t$) features. Concretely,

$$\hat{\mathbf{s}}_t = \text{Attn}\left(\mathcal{S}, \mathbf{h}_t^{(attn)}\right), \quad \hat{\mathbf{v}}_t = \text{Attn}\left(\mathcal{V}_t, \hat{\mathbf{s}}_t\right), \quad \hat{\mathbf{d}}_t = \text{Attn}\left(\mathcal{D}_t, \hat{\mathbf{s}}_t\right) \tag{5}$$

where Attn is a scaled dot-product attention [28]. For a query $\mathbf{q} \in \mathbb{R}^{1 \times d_q}$, $\hat{\mathbf{x}} = \text{Attn}(\{\mathbf{x}_i\}, \mathbf{q})$ is computed as $\hat{\mathbf{x}} = \sum_i \alpha_i \mathbf{x}_i$ for $\alpha_i = \text{softmax}_i((W_K \mathbf{x}_i)^T \mathbf{q} \, / \sqrt{d_q})$. The second recurrent network then takes a concatenation of these features including $\mathbf{a}_{t-1}$ and $\mathbf{h}_t^{(attn)}$ and predicts an action.

$$\mathbf{h}_t^{(a)} = \text{GRU}\left(\left[\hat{\mathbf{s}}_t, \hat{\mathbf{v}}_t, \hat{\mathbf{d}}_t, \mathbf{a}_{t-1}, \mathbf{h}_t^{(attn)}\right], \mathbf{h}_{t-1}^{(a)}\right) \tag{6}$$

$$a_t = \underset{a}{\text{argmax}} \;\; \text{softmax}\left(W_a \mathbf{h}_t^{(a)} + \mathbf{b}_a\right) \tag{7}$$

4.3 Auxiliary Losses and Training Regimes

Aside from modeling details, much of the remaining progress in VLN has come from adjusting the training regime – adding auxiliary losses / rewards [17, 29], mitigating exposure bias during training [4, 29], or incorporating synthetic data augmentation [9, 26]. We explore some common variants of these directions in

VLN-CE. We suspect addressing exposure bias and data sparsity will be important in VLN-CE where these issues may be amplified by lengthy action sequences.

Imitation Learning. A natural starting point for training is maximizing the likelihood of the ground truth trajectories. To do so, we perform teacher-forcing training with inflection weighting (IW). As described in [30], IW places emphasis on time-steps where actions change (i.e. $a_{t-1} \neq a_t$), adjusting loss weight proportionally to the rarity of such events. This was found to be helpful for navigation problems with long sequences of repeated actions. We observe a positive effect in early experiments and apply IW in all our experiments.

Coping with Exposure Bias. Imitation learning in auto-regressive settings suffers from a disconnect between training and test – agents are not exposed to the consequences of their actions during training. Prior work has shown significant gains by addressing this issue for VLN through scheduled sampling [4] or reinforcement learning fine-tuning [26, 29]. In this work, we apply Dataset Aggregation (DAgger) [24] towards the same end. While DAgger and scheduled sampling share many similarities, DAgger trains on the aggregated set of trajectories from all iterations 1 to n. Thus, the resulting policy after iteration n is optimized over all past experiences and not just those collected from iteration n.

Synthetic Data Augmentation. Another popular strategy is to learn a 'speaker' model that produces instructions given a trajectory. Both [26] and [9] use these models to generate new trajectory-instruction pairs and many following works have leveraged these additional trajectories. We convert ∼150k synthetic trajectories generated this way from [26] to our continuous environments.

Progress Monitor. An important aspect of success is identifying where to stop. Prior work [17] found improvements from explicitly supervising the agent with a progress-toward-goal signal. Specifically, agents are trained to predict their fraction through the trajectory at each time step. We apply progress estimation during training with a mean squared error loss term akin to [17].

5 Experiments

Setting and Metrics. We train and evaluate our models in VLN-CE. We perform early stopping based on val-unseen performance. We report standard metrics for visual navigation defined in [2, 4, 18] – trajectory length in meters (`TL`), navigation error in meters from goal at termination (`NE`), oracle success rate (`OS`), success rate (`SR`), success weighted by inverse path length (`SPL`), and normalized dynamic-time warping (`nDTW`). For full details on metrics, see [2, 4, 18].

Implementation Details. We utilize the Adam optimizer [15] with a learning rate of 2.5×10^{-4} and a batch size of 5 full trajectories. We set the inflection weighting coefficient [30] to 3.2 (inverse frequency of inflections in our ground-truth paths). We train on all ground-truth paths until convergence on val-unseen (at most 30 epochs). For DAgger [24], we collect the nth set by taking the oracle action with probability $\beta{=}0.75^n$ and the current policy action otherwise. We collect $5,000$ trajectories at each stage and then perform 4 epochs of imitation learning (with inflection weighting) over all collected trajectories. Once again, we

Table 2. No-learning baselines and input modality ablations for our baseline sequence-to-sequence model. Given the long trajectories involved, we find both random agents and single-modality ablations to perform quite poorly in VLN-CE.

				Val-Seen						Val-Unseen					
Model	Vision	Instr.	History	TL ↓	NE ↓	nDTW ↑	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	nDTW ↑	OS ↑	SR ↑	SPL ↑
Random	-	-	-	3.54	10.20	0.28	0.04	0.02	0.02	3.74	9.51	0.30	0.04	0.03	0.02
Hand-Crafted	-	-	-	3.83	9.56	0.33	0.05	0.04	0.04	3.71	10.34	0.30	0.04	0.03	0.02
Seq2Seq	RGBD	✓	✓	8.40	8.54	0.45	0.35	0.25	0.24	7.67	8.94	0.43	0.25	0.20	0.18
– No Image	D	✓	✓	7.77	8.55	0.46	0.31	0.24	0.23	7.87	9.09	0.41	0.23	0.17	0.15
– No Depth	RGB	✓	✓	4.93	10.76	0.29	0.10	0.03	0.03	5.54	9.89	0.31	0.11	0.04	0.04
– No Vision	-	✓	✓	4.26	11.07	0.26	0.03	0.00	0.00	4.68	10.06	0.30	0.07	0.00	0.00
– No Instruction	RGBD	-	✓	7.86	9.09	0.42	0.26	0.18	0.17	7.27	9.03	0.42	0.22	0.17	0.16

train to convergence on val-unseen (6 to 10 dataset collections, depending on the model). We implement our agents in PyTorch [22] and on top of Habitat [19].

5.1 Establishing Baseline Performance for VLN-CE

No-Learning Baselines. To establish context for our results, we consider random and hand-crafted agents in Tab. 2 (top two rows). The `random` agent selects actions according to the action distribution in train. [2] The `hand-crafted` agent picks a random heading and takes 37 `forward` actions (dataset average) before calling `stop`. Both these agents achieve a ∼3% success rate in val-unseen despite no learned components or input processing. A similar hand-crafted model in VLN yields a 16.3% success rate [4]. Though not directly comparable, this gap illustrates the strong structural prior provided by the nav-graph in VLN.

Seq2Seq and Single-Modality Ablations. Tab. 2 also shows performance for the baseline Seq2Seq model along with input ablations. All models are trained with imitation learning without data augmentation or any auxiliary losses. Our baseline Seq2Seq model significantly outperforms the random and hand-crafted baselines, successfully reaching the goal in 20% of val-unseen episodes.

As illustrated in [27], single modality models can be strong baselines in embodied tasks. We train models without access to the instruction (`No Instruction`) and with ablated visual input (`No Vision/Depth/Image`). All of these ablations under-perform the Seq2Seq baseline. We find depth is a very strong signal for learning – models lacking it (`No Depth` and `No Vision`) fail to outperform chance (≤1% success rates). We believe depth enables agents to quickly begin traversing environments effectively (e.g. without collisions) and without this it is very difficult to bootstrap to instruction following. The `No Instruction` model achieves 17% success, similarly to a hand-crafted agent in VLN, suggesting shared trajectory regularities between VLN and VLN-CE. While these regularities can be manually exploited in VLN via the nav-graph, they are implicit in VLN-CE as

[2] 68% `forward`, 15% `turn-left`, 15% `turn-right`, and 2% `stop`

Table 3. Performance in VLN-CE. We find that popular techniques in VLN have mixed benefit in VLN-CE; however, our best performing model combining all examined techniques succeeds nearly 1/3rd of the time in new environments. * denotes fine-tuning.

					Val-Seen						Val-Unseen					
#	Model	PM [17]	DA [24]	Aug. [26]	TL ↓	NE ↓	nDTW ↑	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	nDTW ↑	OS ↑	SR ↑	SPL ↑
1	Seq2Seq Baseline	-	-	-	8.40	8.54	0.45	0.35	0.25	0.24	7.67	8.94	0.43	0.25	0.20	0.18
2		✓	-	-	8.34	8.48	0.47	0.32	0.22	0.21	8.93	9.28	0.40	0.28	0.17	0.15
3		-	✓	-	9.32	7.09	0.53	0.44	0.34	0.32	8.46	7.92	0.48	0.35	0.26	0.23
4		-	-	✓	8.23	7.76	0.51	0.34	0.26	0.25	7.22	8.70	0.44	0.26	0.19	0.17
5		✓	✓*	✓	9.37	7.02	0.54	0.46	0.33	0.31	9.32	7.77	0.47	0.37	0.25	0.22
6	Cross-Modal Attention	-	-	-	8.26	7.81	0.49	0.38	0.27	0.25	7.71	8.14	0.47	0.31	0.23	0.22
7		✓	-	-	8.51	8.17	0.47	0.35	0.28	0.26	7.87	8.72	0.44	0.28	0.21	0.19
8		-	✓	-	8.90	7.40	0.52	0.42	0.33	0.31	8.12	8.00	0.48	0.33	0.27	0.25
9		-	-	✓	8.50	8.05	0.49	0.36	0.26	0.24	7.58	8.65	0.45	0.28	0.21	0.19
10		✓	✓*	✓	9.26	7.12	0.54	0.46	**0.37**	**0.35**	8.64	**7.37**	**0.51**	**0.40**	**0.32**	**0.30**
11		✓	-	✓	8.49	8.29	0.47	0.36	0.27	0.25	7.68	8.42	0.46	0.30	0.24	0.22
12		-	✓*	✓	9.32	**6.76**	**0.55**	**0.47**	**0.37**	0.33	8.27	7.76	0.50	0.37	0.29	0.26

evidenced by the significantly lower performance of our random and hand crafted agents which collide with and get stuck on obstacles. The `No Image` model also achieves 17% success, similarly failing to reason about instructions. This hints at the importance of grounding visual referents (through RGB) for navigation.

5.2 Model Performance in VLN-CE

Tab. 3 shows a comparison of our models (`Seq2Seq` and `Cross-Modal`) under three training augmentations (`Progress Monitor`, `DAgger`, `Data Augmentation`).

Cross-Modal Attention vs. Seq2Seq. We find the cross-modal attention model outperforms Seq2Seq under all settings for new environments. For example, in teacher-forcing training (row `1` vs. `6`), the cross-modal attention model improves from 0.18 to 0.22 SPL on val-unseen, an improvement of 0.04 SPL (22% relative). When applying all three augmentations (row `5` vs. `10`), the cross-modal model improves from 0.22 to 0.30 SPL, an improvement of 0.08 SPL (36% relative).

Training Augmentation. We find DAgger-based training impactful for both the Seq2Seq (row `1` vs. `3`) and Cross-Modal (row `6` vs. `8`) models – improving by 0.03-0.05 SPL in val-unseen. Contrary to findings in prior work, we observe negative effects from progress monitor auxiliary loss or data augmentation for both models (rows `2/4` and `7/9`) – dropping 0.01-0.03 SPL from standard training (rows `1/6`). Despite this, we find combining all three techniques to lead to significant performance gains for the cross-modal attention model (row `10`). Specifically, we pretrain with imitation learning, data augmentation, and the progress monitoring loss, then finetune using DAgger (with β=0.75^{n+1}) on the original data. This Cross-Modal Attention PM+DA*+Aug model achieves an SPL of 0.35 on val-seen and 0.30 on val-unseen – succeeding on 32% of episodes in new environments.

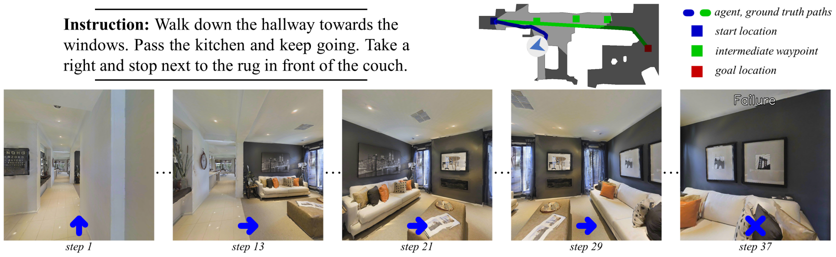

Instruction: Walk down the hallway towards the windows. Pass the kitchen and keep going. Take a right and stop next to the rug in front of the couch.
agent, ground truth paths
start location
intermediate waypoint
goal location
Failure
step 1
step 13
step 21
step 29
step 37

Fig. 4. Example of our Cross Modal Attention model taken in an unseen environment.

We explore this trend further for the Cross-Modal model. We examine the validation performance of PM+Aug (row 11) and find it to outperform Aug or PM alone (by 0.02-0.03 SPL). Next, we examine progress monitor loss on val-unseen for both PM and PM+Aug. We find that without data augmentation, the progress monitor over-fits considerably more (validation loss of 0.67 vs. 0.47) – indicating that the progress monitor can be effective in our continuous setting but tends to over-fit on the non-augmented training data, negatively affecting generalization. Finally, we examine the performance of DA*+Aug (row 12) and find that this outperforms DA (by 0.01-0.02 SPL), but is unable to match pre-training with the progress monitor and augmented data (row 10).

Example. We examine our Cross-Modal Attention PM+DA*+Aug model in an unseen environment (Fig. 4). The example demonstrates the increased difficultly of VLN-CE (37 actions vs. 4 hops in VLN). It also shows a failure of the agent – the agent navigates towards the wrong windows and fails to first "*pass the kitchen*" – stopping instead at the nearest couch. We observe failures when the agent never sees the instruction referent(s) – with a limited egocentric field-of-view, the agent must actively choose to observe the surrounding scene.

5.3 Examining the Impact of the Nav-Graph in VLN

To draw a direct comparison between the VLN and VLN-CE settings, we convert trajectories taken by our Cross-Modal Attention (PM+DA*+Aug.) model in continuous environments to nav-graph trajectories (details in the supplement) and then evaluate these paths on the VLN leaderboard.[3] We emphasize that the point of this comparison is not to outperform existing approaches for VLN, but rather to highlight how important the nav-graph is to the performance of existing VLN systems by contrasting them with our model. Unlike the approaches shown, our model does not benefit from the nav-graph during training or inference.

As shown in Tab. 4, we find significant gaps between our model and prior work in the VLN setting. Despite having similar cross-modal attention architectures, RCM [29] achieves an SPL of 0.38 in test environments while our model yields 0.21. Further, state-of-the-art on the test set is near 0.47 SPL, over 2x what we

[3] Note that the VLN test set is not publicly available except through this leaderboard.

Table 4. Comparison on the VLN validation and test sets with existing models. Note there is a significant gap between techniques that leverage the oracle nav-graph at train and inference (top set) and our best method in continuous environments.

		Val-Seen (VLN)					Val-Unseen (VLN)					Test (VLN)				
	Model	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑
VLN Task	VLN-Seq2Seq [4]	11.33	6.01	0.52	0.38	-	8.39	7.81	0.28	0.21	-	8.13	7.85	0.27	0.20	0.18
	Self-Monitoring [17]	-	3.18	0.77	0.68	0.58	-	5.41	0.68	0.47	0.34	18.04	5.67	0.59	0.48	0.35
	RCM [29]	10.65	3.53	0.75	0.66	-	11.46	6.09	0.50	0.42	-	11.97	6.12	0.495	0.43	0.38
	Back-Translation [26]	10.1	4.71	-	0.55	0.53	9.37	5.49	-	0.46	0.43	11.7	-	-	0.51	0.47
	Cross-Modal (PM+DA*+Aug.)	6.92	7.77	0.30	0.25	0.23	7.42	8.17	0.28	0.22	0.20	9.47	8.55	0.32	0.24	0.21

report. However, it is unclear if these gains could be realized on a real system given the strong assumptions set by the nav-graph. In contrast, our approach does not rely on external information and recent work has shown promising sim2real transferability for navigation agents trained in continuous simulations [14].

Caveats. Direct comparisons between drastically different settings are challenging, we note some caveats. About 20% of VLN trajectories are non-navigable in VLN-CE and thus our models cannot succeed on these. Further, continuous VLN-CE paths can translate poorly to nav-graph trajectories when traversing areas of the environment not well-covered by the sparse panoramas. Comparing VLN-CE val results in Tab. 3 with the same in Tab. 4 shows these effects account for a drop of ∼0.10 SPL. Even compensating for this possible underestimation, nav-graph-based approaches still outperform our continuous models significantly.

6 Discussion

In this work, we explore the problem of following navigation instructions in continuous environments with low-level actions – lifting many of the unrealistic assumptions in prior nav-graph-based settings. Our work lays the groundwork for future research into reducing the gap between simulation and reality for VLN agents. Crucially, setting our VLN-CE task in continuous environments (rather than a nav-graph) provides the community a testbed where integrative experiments studying the interface of high- and low-level control are possible. This includes studying the effect of imperfect actuation by leveraging recent features in the Habitat simulator [19], reasoning about (potentially dynamic) objects inserted in the 3D environment, or developing modular planner-controller architectures that leverage existing robot path planning algorithms.

Acknowledgements. We thank Anand Koshy for his implementation of nDTW. The GT effort was supported in part by NSF, AFRL, DARPA, ONR YIPs, ARO PECASE, Amazon. The OSU effort was supported in part by DARPA. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government, or any sponsor.

References

1. Locobot: An open source low cost robot (2019), https://locobot-website.netlify.com/ 5
2. Anderson, P., Chang, A., Chaplot, D.S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., et al.: On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757 (2018) 10
3. Anderson, P., Shrivastava, A., Parikh, D., Batra, D., Lee, S.: Chasing ghosts: Instruction following as bayesian state tracking. NeurIPS (2019) 2
4. Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., van den Hengel, A.: Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In: CVPR (2018) 1, 3, 4, 5, 6, 8, 9, 10, 11, 14
5. Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3D: Learning from RGB-D data in indoor environments. In: 3DV (2017), MatterPort3D dataset license available at: http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf 3, 4, 5, 6
6. Chen, H., Suhr, A., Misra, D., Snavely, N., Artzi, Y.: Touchdown: Natural language navigation and spatial reasoning in visual street environments. In: CVPR (2019) 1, 4
7. Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., Batra, D.: Embodied Question Answering. In: CVPR (2018) 5
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009) 8
9. Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., Darrell, T.: Speaker-follower models for vision-and-language navigation. In: NeurIPS (2018) 5, 6, 9, 10
10. Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., Farhadi, A.: IQA: Visual question answering in interactive environments. In: CVPR (2018) 5
11. Gupta, S., Davidson, J., Levine, S., Sukthankar, R., Malik, J.: Cognitive mapping and planning for visual navigation. In: CVPR (2017) 3
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: CVPR (2016) 8
13. Hermann, K.M., Malinowski, M., Mirowski, P., Banki-Horvath, A., Anderson, K., Hadsell, R.: Learning to follow directions in street view. AAAI (2020) 4
14. Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., Batra, D.: Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation. In: IROS (2020) 14
15. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. In: ICLR (2015) 10
16. Kohlbrecher, S., Meyer, J., von Stryk, O., Klingauf, U.: A flexible and scalable slam system with full 3d motion estimation. In: SSRR. IEEE (November 2011) 3
17. Ma, C.Y., Lu, J., Wu, Z., AlRegib, G., Kira, Z., Socher, R., Xiong, C.: Self-monitoring navigation agent via auxiliary progress estimation. ICLR (2019) 3, 5, 6, 9, 10, 12, 14
18. Magalhaes, G., Jain, V., Ku, A., Ie, E., Baldridge, J.: Effective and general evaluation for instruction conditioned navigation using dynamic time warping. arXiv preprint arXiv:1907.05446 (2019) 10
19. Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D.: Habitat: A Platform for Embodied AI Research. ICCV (2019) 3, 5, 11, 14

20. Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., Artzi, Y.: Mapping instructions to actions in 3d environments with visual goal prediction. In: EMNLP (2018) 4
21. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015) 3
22. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019) 11
23. Pennington, J., Socher, R., Manning, C.D.: GloVe: Global Vectors for Word Representation. In: EMNLP (2014) 8
24. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: AISTATS (2011) 3, 10, 12
25. Stentz, A.: Optimal and efficient path planning for partially known environments. In: Intelligent Unmanned Ground Vehicles, pp. 203–220. Springer (1997) 3
26. Tan, H., Yu, L., Bansal, M.: Learning to navigate unseen environments: Back translation with environmental dropout. In: NAACL HLT (2019) 3, 5, 6, 9, 10, 12, 14
27. Thomason, J., Gordon, D., Bisk, Y.: Shifting the baseline: Single modality performance on visual navigation & qa. In: NAACL HLT (2019) 3, 11
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017) 9
29. Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y.F., Wang, W.Y., Zhang, L.: Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In: CVPR (2019) 5, 6, 8, 9, 10, 13, 14
30. Wijmans, E., Datta, S., Maksymets, O., Das, A., Gkioxari, G., Lee, S., Essa, I., Parikh, D., Batra, D.: Embodied question answering in photorealistic environments with point cloud perception. In: CVPR (2019) 5, 10
31. Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., Batra, D.: DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In: ICLR (2020) 3, 8