Sim-2-Sim Transfer for Vision-and-Language Navigation in Continuous Environments

Jacob Krantz and Stefan Lee

Oregon State University
Project Page: jacobkrantz.github.io/sim-2-sim

Abstract. Recent work in Vision-and-Language Navigation (VLN) has presented two environmental paradigms with differing realism – the standard VLN setting built on topological environments where navigation is abstracted away [3], and the VLN-CE setting where agents must navigate continuous 3D environments using low-level actions [21]. Despite sharing the high-level task and even the underlying instruction-path data, performance on VLN-CE lags behind VLN significantly. In this work, we explore this gap by transferring an agent from the abstract environment of VLN to the continuous environment of VLN-CE. We find that this sim-2-sim transfer is highly effective, improving over the prior state of the art in VLN-CE by +12% success rate. While this demonstrates the potential for this direction, the transfer does not fully retain the original performance of the agent in the abstract setting. We present a sequence of experiments to identify what differences result in performance degradation, providing clear directions for further improvement.

Keywords: vision-and-language navigation (VLN), embodied AI

1 Introduction

Vision-and-Language Navigation (VLN) is a popular instruction-guided navigation task where a vision-equipped agent must navigate to a goal location in an never-before-seen environment by following a path described by a natural language instruction. In the standard VLN setting, the environment is abstracted as a topology of interconnected panoramic images (called a nav-graph) such that navigation amounts to traversing the graph by iteratively selecting among a small set of neighboring node locations at each time step. In effect, this induces a prior of possible locations for the agent and assumes perfect navigation between nodes. Recent work has identified that these assumptions do not reflect the challenges a deployed system would experience in a real environment. To reduce this gap, Krantz et al. [21] introduced the VLN in Continuous Environments (VLN-CE) setting which instantiates VLN in a 3D simulator and drops the nav-graph assumptions – requiring agents to navigate with low-level actions.

So far, VLN-CE has proven to be significantly more challenging than its more abstract counterpart, with published work reporting episode success rates less than half of those reported in standard VLN. Typically, models for VLN-CE have been end-to-end systems trained to directly predict low-level actions (or nearby waypoints) from language and observations. As such, the lower performance compared to VLN is commonly ascribed to the challenge of learning navigation and language grounding jointly in a long-horizon task. This suggests two avenues towards improvement – developing more capable models directly in VLN-CE or exploring transfer of knowledge from VLN to VLN-CE. Given the significant differences between the abstract and continuous settings, the potential of sim-2-sim transfer has so far been under-explored and uncertain.

In this work, we explore the sim-2-sim transfer of a VLN agent to VLN-CE. To support this transfer, we build a modular harness such that any VLN agent can be run in VLN-CE. Analogous to sim-2-real experiments from [2], this harness includes a local navigation policy and a subgoal generation module to mimic nav-graph provided candidates. We present a sequence of systematic experiments to quantify what differences between the two task settings are most impactful. Our final model demonstrates that transfer can lead to significant improvements over end-to-end VLN-CE-trained methods – achieving an increase of +12% success rate over prior published state-of-the-art (32% vs 44%) on the VLN-CE test set.

Along the way, our analysis quantifies the effect of (1) differences between VLN and VLN-CE data subsets, (2) the visual domain gap between real panoramas and simulated renders in VLN-CE, (3) error induced by navigation policies, and (4) subgoal candidate generation – identifying fruitful directions for future work. Through an analysis of our model's errors, we identify that episodes containing significant elevation change (i.e. stairs) are a challenge for our model.

Overall, our results demonstrate an alternative research direction for improving instruction-guided navigation agents in continuous environments. We show transfer with compelling results and present the community with clear remaining challenges. We release our model-agnostic VLN-2-VLNCE harness code[1] to facilitate research in this area – helping to unify progress between both settings.

Contributions. To summarize the contributions of this work:

- We present a first-of-its-kind demonstration that instruction-following agents trained in the abstract VLN setting can effectively transfer to VLN-CE — setting a new state of the art in VLN-CE by +12% success rate.
- We quantify performance loss for key steps of the transfer process – providing insight to existing gaps in our transfer paradigm that could be improved.
- We develop and release a modular VLN-2-VLNCE transfer harness to spur innovation on transfer techniques and help progress from VLN to VLN-CE.

2 Related Work

VLN Agents. The VLN task [3] has received significant attention in recent years with the continual improvement of benchmark models. A higher-level panoramic action space was introduced by Fried et al. [11] and widely adopted in

[1] github.com/jacobkrantz/Sim2Sim-VLNCE

follow-up works. Early efforts proposed cross-modal grounding of vision and language [27]. Pre-training and data augmentation methods can mitigate issues regarding limited data [11,26] and challenges of jointly processing vision, language, and action [14]. Recently, transformer-based architectures and multi-step training routines have demonstrated superior performance [22,16,8]. VLN↻BERT [16] is one such model that introduced a recurrent history context to a transformer-based agent. We adopt VLN↻BERT as the core VLN agent in this work. With continual interest in improving topological instruction followers, we establish expectations and infrastructure for their transfer to continuous environments.

VLN-CE Agents. Recognizing the unrealistic affordances of navigation graphs, Krantz et al. [21] proposed replacing the topological definition of VLN with lower-level actions in continuous environments. Initial end-to-end baselines that directly predict lower-level actions were modeled after sequence-to-sequence [3] and cross-modal [27] VLN methods and demonstrated significantly lower performance than in VLN. Raychaudhuri et al. [25] proposed language-aligned path supervision to better guide off-the-path decision-making. Krantz et al. [20] explored a higher-level waypoint action space trained with deep reinforcement learning, finding increased performance but still far below VLN. Other works exploited continuous environments to study topological map generation [7] and semantic mapping for navigation [18]. In the context of a velocity-based continuous action space, Irshad et al. [17] paired a discrete-action model with a learned control policy to demonstrate the benefit of hierarchy. Rather than train agents from scratch in VLN-CE as these prior works do, we analyze the transfer of pre-trained VLN agents to continuous environments.

Sim-2-Real Transfer of Instruction Following. Transferring skills learned in simulation to reality is a critical step for embodied agents [19,12,10,4]. The most similar to our work is that of Anderson et al. [2], which directly transferred VLN agents to reality. We analyze the intermediate step: sim-2-sim transfer from high-level simulation to more realistic, lower-level simulation. The study of Anderson et al. [2] emulated the VLN action space with subgoal candidate prediction and performed navigation with the Robot Operating System (ROS)[24]. They found a more than 2x drop in performance when evaluating agents in reality, showing that effective VLN sim-2-real transfer remains an open research question. We identify similar challenges in sim-2-sim transfer as Anderson et al. [2] did in sim-2-real. We diagnose and begin mitigating these issues in VLN-CE using methods infeasible in VLN and impractical in reality.

3 Task Descriptions

In this section, we describe both the vision-and-language navigation task (VLN) and its continuous-environment counterpart, VLN-CE.

Vision-and-Language Navigation (VLN). VLN was proposed by Anderson, et al. [3], in which agents navigate previously-unseen indoor environments. The task is to follow a path described in natural language and stop at the goal. Instructions are unstructured and crowd-sourced in English to form the Room-to-Room (R2R) dataset. VLN takes place in a topological simulator, where a pre-defined nav-graph dictates allowable navigation through edge connections.

At each time step t, a VLN agent receives the instruction $\mathcal{I}$, a panoramic RGB observation $\mathcal{O}_t$, and a set of n subgoal candidates $\mathcal{C}_t^{(1 \ldots n)}$. The instruction $\mathcal{I}$ is a sequence of L words $(w_0, w_1, \ldots, w_L)$. The panoramic observation $\mathcal{O}_t$ is represented as 36 views such that 12 equally-spaced horizontal headings (0°, 30°, ..., 360°) are represented at each of 3 elevation angles (-30°, 0°, +30°). Each image frame in $\mathcal{O}_t$ has a resolution of 480x640 with a horizontal field of view of 75°. Each subgoal candidate $\mathcal{C}_t^i$ is represented by a relative heading angle θ_t^i, an elevation angle ϕ_t^i, and the observation view frame $\mathcal{O}_t^j$ that most closely centers the candidate: $\mathcal{C}_t^i = (\theta_t^i, \phi_t^i, \mathcal{O}_t^j)$. The VLN agent then navigates by selecting a candidate. The environment then transitions and produces observations for time $t+1$. This repeats until the agent issues the `STOP` action.

VLN in Continuous Environments (VLN-CE). VLN-CE as proposed by Krantz et al. [21] converts the topologically-defined VLN task into a continuous-environment task more representative of real-world navigation. Instead of selecting and navigating by environment-provided subgoal candidates, agents in VLN-CE must navigate a continuous-valued 3D mesh by selecting actions from a lower-level action space (`FORWARD` 0.25m, `TURN-LEFT` 15°, `TURN-RIGHT` 15°, `STOP`). In this work, we seek to port VLN agents into continuous environments. Thus, we adopt what we can of the VLN observation space: the instruction $\mathcal{I}$ and the panoramic observation $\mathcal{O}_t$. We introduce a 2D laser scan which we describe in Sec. 4.3. This provides our subgoal generation module with 2D occupancy. Such a sensor is commonly used for both localization and mapping in real-world navigation stacks [24]. In our analysis below, we perform intermediate experiments that may allow oracle navigation or topological subgoal candidates but note that these experiments do not result in admissible VLN-CE agents.

Room-to-Room Dataset. The Room-to-Room dataset (R2R) consists of $7,189$ shortest-path trajectories across all train, validation, and test splits [3]. The VLN-CE dataset is a subset of R2R consisting of $5,611$ trajectories traversible in continuous environments (78% of R2R) [21]. For both tasks, we report performance on several validation splits. Val-Seen contains episodes with novel paths and instructions but from scenes observed in training. Val-Unseen contains novel paths, instructions, and scenes. Train* is a random subset of the training split derived by Hong et al. [16] to support further analysis.

Matterport3D Scene Dataset. Both VLN and VLN-CE use the Matterport3D (MP3D) Dataset [5] which consists of 90 scenes, 10,800 panoramic RGBD images, and 90 3D mesh reconstructions. VLN agents interact with MP3D through a connectivity graph that queries panoramic images. VLN-CE agents interact with the 3D mesh reconstructions through the Habitat Simulator [23].

Evaluation Metrics. We report evaluation metrics standard to the public leaderboards of both VLN and VLN-CE. These include trajectory length (`TL`), navigation error (`NE`), oracle success rate (`OS`), success rate (`SR`), and success weighted by inverse path length (`SPL`) as described in [1,3]. We consider `SR` the primary metric for comparison and `SPL` for evaluating path efficiency. Both `SR`

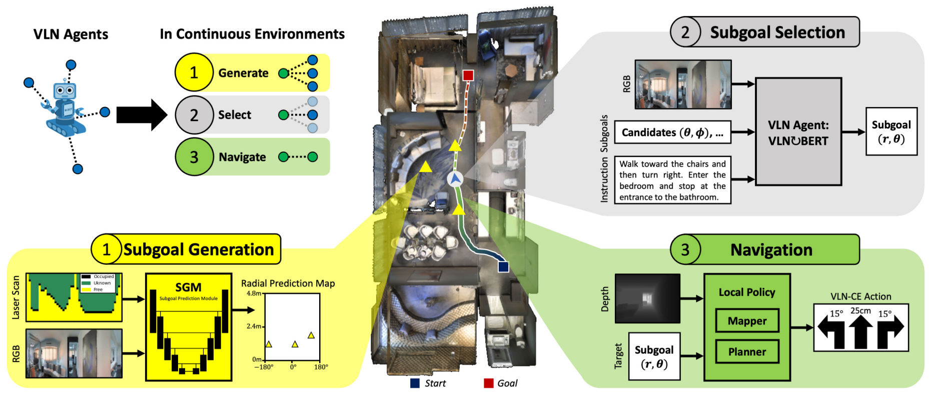


Fig. 1. We diagnose and quantify the VLN to VLN-CE gap by operating a topological VLN agent in continuous environments. We emulate the VLN observation space by generating subgoal candidates from egocentric observation (Sec. 4.3) and emulate the action space by calling a map-based navigation policy (Sec. 4.2).

and SPL determine success using a 3.0m distance threshold between the agent's terminal location and the goal. We report the same metrics across both VLN and VLN-CE to enable empirical diagnosis of performance differences. We note that distance is computed differently between VLN and VLN-CE resulting in subtle differences in evaluation for the same effective path. In VLN, the distance between nodes is Euclidean[2], whereas in VLN-CE, distances are geodesic as computed on the 3D navigation mesh. This tends to result in longer path lengths in VLN-CE and a drop in SPL.

4 Porting a VLN agent to Continuous Environments

To perform a sim-2-sim transfer, we emulate the VLN action space in VLN-CE by developing a harness consisting of a lower-level navigation policy and a subgoal generation module. An overview can be seen in Fig. 1. Our harness follows the structure used in sim-2-real experiments by Anderson et al. [2]. We develop our harness such that subgoal generation modules, VLN agents, and navigation policies are all modular components for drop-in replacement and evaluation. This not only enables the analysis in this work, but the released codebase will ease sim-2-sim transfer of future VLN agents to continuous environments.

4.1 Core VLN Agent

We adopt VLN↻BERT [16], a VLN agent that performs near the state-of-the-art[3] on R2R Test. This agent uses observation and action spaces typical of recent

[2] As defined by the Matterport3D Simulator used in VLN.
[3] eval.ai/web/challenges/challenge-page/97

work in VLN. VLN↻BERT has a recurrence-modified transformer architecture and encodes RGB vision with a ResNet-152 trained on Places365 [15,28]. We use VLN↻BERT as our core VLN agent for all experiments but note that our harness and related design choices are agnostic to the choice of VLN agent.

4.2 Navigation Policies

With the abstracted action space in VLN, navigation between subgoal locations is effectively performed by teleportation. In contrast, continuous settings require a navigation policy to convey the agent between subgoals. This navigation subtask can be considered a short-range version of PointGoal navigation [1] given that the average distance between nodes in the VLN nav-graph is 2.25m. To evaluate the impact of navigation between subgoals in VLN-CE, we first establish upper bounds with two oracle methods: teleportation and an oracle policy. We then evaluate with a local policy admissible in VLN-CE. We describe these navigation policies as follows.

Teleportation. A teleportation action involves translation to the target location and rotation to face away from the previous location. In our teleportation policy, the heading is snapped to the nearest global 30° increment to match the heading discretization in VLN. This policy matches the navigation assumption made in VLN and serves as both an upper bound on navigation performance and a confirmation that the VLN agent is operating as expected in VLN-CE.

Oracle Policy. An oracle navigation policy has access to the navigation mesh used by the simulator to take optimal actions from the VLN-CE action space. The input to our oracle policy is 3D coordinates that exist on the navigation mesh. Analytical search using the A* algorithm then determines an action plan. We set a stopping distance threshold of 0.15m to the goal which we empirically determine results in minimal navigation error and minimal action jitter near the goal. The Oracle Policy serves as an upper bound on navigation performance to policies operating under the same action space.

Local Policy. We employ a mapping and planning navigation policy based on [6] to convey the agent to selected subgoal locations as shown in Fig. 1 "Navigation". The Local Policy receives a target location specified by a distance and relative heading (r, θ). At each time step, a local occupancy map is aggregated from the geometric projection of an egocentric depth observation. The unexplored map area is assumed to be free space. We then plan a path to the target location using the Fast Marching Method (FMM) which produces position coordinates along the shortest path lying 0.25m away from the agent. Following [13], we greedily decode a discrete action to approach this point. We adopt the VLN-CE action space and stop once the agent is within 0.15m of the target or 40 actions have been taken. The occupancy map is re-initialized for each new target.

4.3 Subgoal Candidate Generation

The VLN action space requires a set of subgoal candidates to select from. Where VLN uses neighboring nodes in the nav-graph, such oracle information does not exist in VLN-CE. As shown in Fig. 1 "Subgoal Generation", we emulate the presence of a nav-graph by having a learned module predict subgoal candidates from observations. Anderson et al. [2] developed a subgoal generation module (SGM) for VLN sim-2-real transfer. We adopt SGM and modify it for VLN-CE.

The observation space of the SGM is panoramic RGB vision (same as the VLN agent), supplemented with a 2D laser scanner for range-finding. We emulate a 360° laser scanner in the Habitat Simulator mounted at 0.24m above ground level. We note that such scanners are commercially available. As shown in Fig. 1 "Laser Scan", the laser scan is represented as a radial obstacle map limited to a 4.8m range. The obstacle map size is 24x48 with 24 discrete range bins of size 0.2m and 48 heading bins of size 7.5°. The RGB panorama frames are encoded with the same ResNet-152 used by the VLN agent.

A U-Net architecture fuses these RGB features with the obstacle map to predict a radial map of subgoal candidates. This map follows the same range and heading discretization as the input. Gaussian non-maximum suppression filters localized predictions. The $k = 5$ candidates with the highest probability mass are converted into (r, θ) candidates consumed by the VLN agent.

Following Anderson et al. [2], we train the SGM to minimize Sinkhorn divergence [9] to ground-truth subgoal candidates on the VLN nav-graph. Nodes from Val-Unseen are held out for validation. To better match the visual domain of 3D-reconstructed environments, we train the SGM with panoramas rendered using the Habitat Simulator. We provide an ablation analysis motivating our changes to 360° laser scanning and reconstructed RGB vision in the supplementary.

5 Results and Analysis

We evaluate the transferability of VLN agents to continuous environments using the harness described above. We identify potential sources of performance degradation and evaluate their isolated impacts. Specifically, our analysis starts in VLN and moves toward VLN-CE, covering differences in dataset episodes (5.1), replacing MP3D panoramas with reconstructed vision (5.2), using a navigation policy to reach selected subgoals (5.3), and removing nav-graph subgoals (5.4). After demonstrating favorable performance over previous agents trained purely in VLN-CE (5.5), we diagnose a remaining failure mode to be addressed in both future VLN-2-VLNCE transfer and sim-2-real transfer efforts (5.6).

5.1 How different is the VLN-CE subset of R2R?

We find the VLN-CE subset is slightly more difficult but comparable to the full R2R dataset. This suggests that performance in VLN is accurate to, but slightly over-estimates, the upper bound on performance transferred to VLN-CE. We make this comparison in Tab. 1 by evaluating VLN↻BERT on the topological VLN task for the VLN and VLN-CE subsets of R2R. Differences in these datasets arise from conversion errors which cause the VLN-CE subset of R2R to contain 22% fewer trajectories than in VLN. VLN↻BERT performs worse on the VLN-CE subset by 3 SR in Val-Seen and 1 SR in Val-Unseen.

Table 1. Difficulty of Room-to-Room (R2R) episodes in the VLN-CE subset vs. all episodes of R2R. We find performance on the VLN-CE subset is slightly lower than full R2R; Val-Seen success drops by 3 points and Val-Unseen success drops by 1 point.

Task: VLN		Train*					Val-Seen					Val-Unseen				
#	Dataset	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑
1	Room-to-Room [3]	9.98	0.93	95	93	90	10.81	3.00	76	72	68	11.85	4.24	67	61	56
2	VLN-CE subset [21]	9.96	1.00	94	93	89	10.94	3.30	74	69	65	11.79	4.43	66	60	54

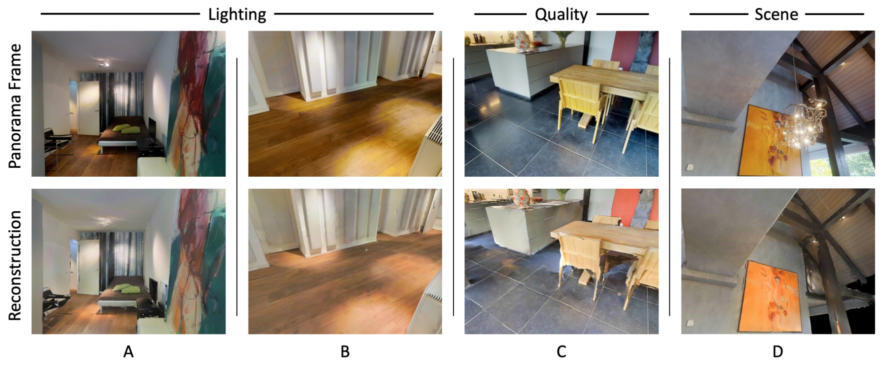


Fig. 2. Visual observation differences between VLN and VLN-CE. VLN renders observations from high-quality Matterport3D panoramas (top) and VLN-CE renders observations from 3D scene reconstructions (bottom). Examples of domain differences include lighting (A, B), furniture resolution (C), and object presence in the scene (D).

5.2 What is the visual domain gap from VLN to VLN-CE?

We find a significant visual domain gap from VLN to VLN-CE in previously-seen environments of 10-13 SR and a moderate gap in novel environments of 3 SR (Tab. 2). This suggests that VLN agents may be overfitting to visual representation in training. This visual domain gap exists despite VLN and VLN-CE being derived from the same real-world scenes and sharing paths. We demonstrate characteristic examples in Fig. 2. VLN observations rendered from Matterport3D panoramas (MP3D panos) are high quality and captured directly from Matterport Pro2 3D cameras [5]. On the other hand, VLN-CE observations are rendered from 3D scene reconstructions in Habitat-Sim (Habitat renders) which introduce reconstruction errors and a domain gap in lighting. We demonstrate that training the agent with Habitat renders eliminates this gap almost entirely.
Experiment Setup. We evaluate VLN↻BERT in VLN with MP3D panos and Habitat renders. To compute vision features from Habitat, we map each node in VLN to a matching location in VLN-CE using known camera poses. We capture a panorama in the scene reconstruction matching the camera height, angle, and

Table 2. Effect of the visual domain gap between Matterport3D panoramas (MP3D) and reconstruction renders (Recon.) on VLN performance. Reconstructed vision decreases success rate in Val-Unseen by 3 points (rows 1 vs. 2). This drop is mitigated by training VLN agents directly on reconstructed images (row 4).

Task: VLN	Camera		**Train***					**Val-Seen**					**Val-Unseen**				
#	Train	Eval	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑
1	MP3D	MP3D	9.98	0.93	95	93	90	10.81	3.00	76	72	68	11.85	4.24	67	61	56
2		Recon.	11.17	2.20	84	80	76	11.66	4.02	68	62	58	11.69	4.56	64	58	53
3	Recon.	MP3D	10.63	1.18	92	90	86	11.35	3.41	72	66	61	11.72	4.12	66	60	54
4		Recon.	9.76	0.48	98	98	96	10.56	2.88	77	72	69	11.16	4.14	66	60	56

resolution expected in VLN. We then encode the panorama with the same feature encoder used by VLN↻BERT, a ResNet-152 trained on Places365 [15,28]. These features are then used as a drop-in replacement for MP3D pano features.

Zero-Shot Generalization. We present zero-shot generalization performance in Tab. 2, row 1 vs. 2. Switching to Habitat renders results in success rate drops of 13 points in Train*, 10 points in Val-Seen, and 3 points in Val-Unseen. The significant performance degradation in seen environments (Train* and Val-Seen) demonstrates the impact of the visual domain gap between VLN and VLN-CE. The relatively smaller drop in Val-Unseen suggests that this gap is not as catastrophic when generalizing to novel environments and that VLN agents may be overfitting to visual representation in training environments.

Domain Transfer. We show in row 4 that re-training VLN↻BERT directly on reconstructed vision can entirely recover task performance in seen environments and recover all by 1 point in success in Val-Unseen (row 4 vs. 1). This reconstruction-trained model achieves a success rate of 60 in Val-Unseen regardless of using MP3D panos or Habitat renders (rows 3 and 4). This suggests that training VLN agents on Habitat renders may result in less visual overfitting.

5.3 What is the navigation gap between VLN and VLN-CE?

Conveying the VLN agent to selected subgoals using a realistic navigator causes a decrease in performance across all splits, highlighted by a 5 point drop in Val-Unseen success. The VLN task, which effectively teleports between subgoals, makes navigation assumptions of perfect obstacle avoidance and stopping precision. These challenges must be contended with in VLN-CE. In the worst case, navigation failure causes episode failure regardless of the decisions made by the VLN agent. In the best case, an unexpected pose and observation are produced at the next time step, a situation VLN agents do not encounter in training.

Experiment Setup. We evaluate our reconstruction-trained VLN↻BERT in VLN-CE with each navigation policy from Sec. 4.2 and provide known subgoal candidates from the VLN nav-graph. We note that navigation errors introduce ambiguity in graph localization which must be resolved to provide the next

Table 3. Impact of navigation policies on VLN agents operating in VLN-CE. We assume a known nav-graph but require lower-level actions between nodes. The Oracle Policy results in better performance than the Local Policy, with a 5 SR Val-Unseen gap.

Task: VLN-CE		**Train***					**Val-Seen**					**Val-Unseen**				
#	Navigator	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑
1	Teleportation	10.04	0.58	97	97	88	11.28	3.24	75	70	63	11.98	4.06	66	60	52
2	Oracle Policy	10.09	0.72	97	96	87	11.19	3.10	75	69	61	12.04	4.07	68	60	52
3	Local Policy	10.58	1.57	90	88	78	11.37	3.49	72	66	58	12.28	4.51	63	55	47

subgoal candidates. In our experiments, we keep the agent at its navigation-terminated location but provide subgoal candidates by assuming it is located at the nearest node on the nav-graph by geodesic distance. This reflects the consequences of poor navigation where a large navigation error may cause the VLN agent to observe candidates from a different graph node than intended. We first evaluate the impact of using the VLN-CE action space instead of teleportation. We then evaluate the effect of a realistic navigation policy. Results are in Tab. 3.

Teleportation vs. VLN-CE Actions. We use the Oracle Policy to represent ideal actions within the VLN-CE action space. We find that performance under the Oracle Policy matches performance under Teleportation in Val-Unseen (row 2 vs. 1). This suggests that the navigation gap between VLN and VLN-CE can be minimized with performant navigation policies. This is an expected result due to the navigation verification that originally culled episodes for the VLN-CE dataset (see [21]). We additionally record the navigation error of the Oracle Policy for each short-range navigation performed, finding an average navigation error of 0.11m in Val-Seen and 0.08m in Val-Unseen. This demonstrates that the VLN agent is robust to position jitter about official graph nodes.

Optimal Navigation vs. Realistic Navigation. In row 3, we present performance under our Local Policy. Compared to the Oracle Policy, we find a drop in success in both Val-Seen (3 points) and Val-Unseen (5 points). The SPL drop is identical, suggesting that path efficiency remains high in successful episodes. The average short-range navigation error is 0.26m in Val-Seen and 0.40m in Val-Unseen, which are higher than that of the Oracle Policy. We observe that most calls to the Local Policy perform similarly to the Oracle Policy. However, the Local Policy has a longer tail of significant navigation failures (error >0.5m). Specifically, the Local Policy has a >0.5m failure rate that is 15x higher than the Oracle Policy. We expand on this error comparison in the supplementary.

5.4 What is the subgoal candidate gap?

In the previous experiments, VLN agents were selecting from subgoal candidates provided by the nav-graph, but these are not available in VLN-CE. Here, we predict subgoal candidates online (at each time step) and observe significant

Table 4. Comparing VLN-CE performance when subgoal candidates are provided by the nav-graph (rows 1,2) vs. predicted by a subgoal generation module (SGM)(rows 3,4). Rows 3 and 4 demonstrate a complete harness for a VLN agent admissible in VLN-CE, consisting of both subgoal candidate generation and lower-level navigation.

Task: VLN-CE			**Train***					**Val-Seen**					**Val-Unseen**				
#	Subgoal Candidates	F-tune	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑
1	Nav-Graph	-	10.58	1.57	90	88	78	11.37	3.49	72	66	58	12.28	4.51	63	55	47
2	Optimal SGM	-	11.40	2.07	86	81	72	12.69	3.78	71	63	55	14.56	4.96	61	49	41
3	SGM	-	13.12	3.54	71	65	55	12.69	4.51	60	51	44	13.74	5.83	51	41	35
4		✓	11.15	3.77	73	66	56	11.18	4.67	61	52	44	10.69	6.07	52	43	36

performance degradation (14-23 SR). Fine-tuning the VLN agent on the online distribution of subgoal candidates slightly mitigates this drop, recovering 1-2 SR.

Experiment Setup. We replace the nav-graph with our subgoal generation module (SGM: Sec. 4.3) as the source of subgoal candidates. We join our SGM with reconstruction-trained VLN↻BERT and our Local Policy as navigator. In Tab. 4, we evaluate the impact of the prediction space and the SGM predictions.

Optimal SGM Predictions. We first consider the case where SGM predictions optimally match the nav-graph (Tab. 4, row 2). We convert nav-graph subgoals into a radial prediction map matching the SGM output space. This discretizes subgoals into polar bins with a 0.20m range and 7.5° resolution. This prediction space accounts for a non-trivial drop in performance: a 3 point drop in Val-Seen success rate and a 6 point drop in Val-Unseen success rate (row 2 vs. 1).

Performance with SGM. Row 3 demonstrates subgoal predictions from the SGM with no nav-graph reliance. This experiment is compliant with the VLN-CE task definition; subgoal candidates are predicted from egocentric observations, a VLN agent selects a candidate, and a local policy conveys the agent using VLN-CE actions. We find a decrease of 12 SR in Val-Seen and 8 SR in Val-Unseen. These drops indicate a large domain gap between the nav-graph and the SGM. This result parallels findings in sim-2-real transfer by Anderson et al. [2] that point to an alignment problem between the VLN nav-graph and the SGM.

Fine-Tuning VLN↻BERT with SGM. We seek to determine if fine-tuning a VLN agent on the SGM candidates can reclaim nav-graph performance. To motivate, consider how learned priors in VLN could result in detrimental behavior under a different distribution of subgoals. For example, a slight change in the distribution of subgoal distances could break a learned time horizon prior – the VLN agent may choose to stop either too early or too late. Such a prior can be very strong in VLN since all paths require 5-7 actions. We fine-tune our reconstruction-trained VLN↻BERT via imitation learning in VLN-CE. Specifically, we train with teacher forcing to maximize the probability of predicting the subgoal candidate that greedily minimizes the geodesic distance to the goal. We set a batch size of 12, a learning rate of 1e-7, and train with cross-entropy

Table 5. Results on the VLN-CE Challenge Leaderboard. Our submission outperforms previously-published results, demonstrating a 12 point improvement over the next best model in success rate on Test (a 38% relative improvement).

Task: VLN-CE		**Val-Seen**					**Val-Unseen**					**Test**				
#	Model	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑	TL ↓	NE ↓	OS ↑	SR ↑	SPL ↑
1	**VLN-CE↻BERT** (ours)	11.18	**4.67**	**61**	**52**	**44**	10.69	**6.07**	**52**	**43**	**36**	11.43	**6.17**	**52**	**44**	**37**
2	HPN+DN [20]	8.54	5.48	53	46	43	7.62	6.31	40	36	34	8.02	6.65	37	32	30
3	WPN+DN [20]	9.52	6.23	45	37	33	9.86	6.93	40	33	29	9.68	7.49	36	29	25
4	LAW [25]	9.34	6.35	49	40	37	8.89	6.83	44	35	31	9.67	7.69	38	28	25
5	CMA+PM+DA+Aug [21]	9.06	7.21	44	34	32	8.27	7.60	36	29	27	8.85	7.91	36	28	25

loss and early stopping. In Tab. 4 row 4, we present the result of fine-tuning on SGM predictions. Performance increases slightly across all validation splits, highlighted by a 2-point increase in Val-Unseen success rate. This still leaves a 6 point gap in success rate to predicting optimal subgoal candidates and a 12 point gap in success rate to candidates directly from the nav-graph. Fine-tuning experiments such as this one cannot be performed on the rigid topology of VLN. Neither can they be performed practically in reality, making VLN-CE a promising setting for generalizing VLN agents for sim-2-sim or sim-2-real transfer.

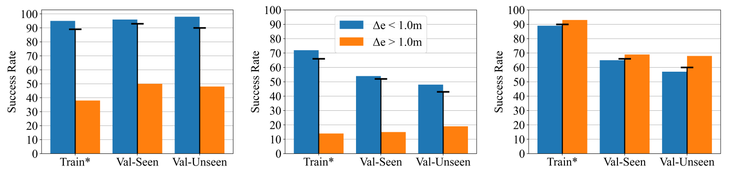
5.5 Comparison With Previous VLN-CE Models

We compare our best agent against previously published methods on the VLN-CE Challenge Leaderboard[4]. This agent includes subgoal candidates from the SGM, reconstruction-trained VLN↻BERT fine-tuned on SGM candidates, and Local Policy navigation. We label our submission VLN-CE↻BERT. As shown in Tab. 5, VLN-CE↻BERT surpasses the success rate of all previous models on all splits including Test. Notably, all previous methods trained exclusively in VLN-CE, with the best amongst them requiring extensive compute (7000+ GPU-hours) to achieve a 32 `SR`. With just 12 GPU-hours of fine-tuning, our model surpasses this by 12 `SR` to achieve a 44 `SR` in Test (a relative improvement of 38%). This result shows that VLN to VLN-CE transfer is a highly promising avenue for making progress on the VLN-CE task and closing the gap to VLN.

5.6 Failures of Subgoal Candidate Generation

Starting from VLN↻BERT operating in VLN, we diagnosed a 1 `SR` drop from dataset differences, a 1 `SR` drop from the visual domain gap, a 5 `SR` drop from navigation policies, and a 12 `SR` drop from subgoal candidate generation. We now look into what failure modes constitute the subgoal candidate drop. We find that half of these failures (6/12 `SR`) are caused by episodes that require the agent to gain or lose at least 1.0m of elevation (i.e. traversing stairs). In such

[4] eval.ai/web/challenges/challenge-page/719



(a) Greedy Oracle (b) VLN-CE ↻BERT (c) VLN: rt-VLN ↻BERT

	Δe >1.0m (%)	
Split	VLN-CE	VLN
Train	8.8	19.6
Train*	11.0	23.2
Val-Seen	6.2	15.9
Val-Unseen	16.0	21.5

(d) Episodes

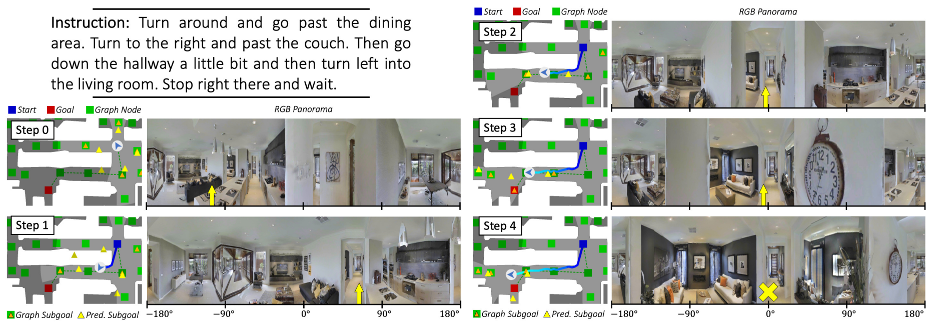
Fig. 3. Success rates in episodes with high (>1.0m) vs. low (<1.0m) elevation delta. Black bars indicate success rate irrespective of elevation. (a,b) use SGM subgoal candidates in VLN-CE and perform worse with elevation. However in VLN (c), our reconstruction-trained agent performs better with elevation. (d) shows the percentage of episodes that have a >1.0m delta.

episodes, an oracle VLN agent (a greedy-optimal subgoal selector) paired with the Oracle Policy for navigation is successful less than 50% of the time. This indicates that the SGM has a recall problem in generating elevation-changing subgoals necessary for success. Details and experimental support are below.

Oracle Selection and Navigation. We first consider whether candidates produced by the SGM can lead to episode success independent of failures stemming from selection or navigation issues. We define a greedy oracle VLN agent that selects the subgoal candidate with minimal geodesic distance to the goal. `STOP` is called when no candidate is closer to the goal than the current position. We run this oracle over SGM candidates and navigate via the Oracle Policy. We find success rates of 89% in Train*, 93% in Val-Seen, and 90% in Val-Unseen (Fig. 3(a)). However, failures disproportionately stem from episodes requiring elevation change. This is exemplified in Train* where episodes requiring at least a 1.0m elevation change result in 38% success – a highly restrictive upper bound.

Elevation-Based Performance of VLN-CE↻BERT. We compare success rates for VLN-CE↻BERT in episodes requiring a high (>1.0m) vs. low (<1.0m) elevation delta (Fig. 3(b)). In Val-Unseen, there is a 29 `SR` gap (19 `SR` vs. 48 `SR`). This is unsurprising given the poor high-elevation performance of the oracle VLN agent. The failure of the SGM to predict high-elevation subgoals (a recall problem) may be related to training data; subgoal candidate locations are highly correlated with free-space in the 2D laser scan. However, 2D free-space is a poor predictor of elevation subgoals, which are relatively rare in training (Fig. 3(d)).

Elevation-Based Performance in VLN. Elevation-based VLN evaluation provides an upper bound on VLN-CE performance in elevation episodes. Surprisingly, we find in Fig. 3(c) that our reconstruction-trained VLN↻BERT has a higher success rate in episodes requiring a high elevation delta (68 `SR` vs. 57 `SR` in Val-Unseen). We suspect this result stems from episodes that feature stairs have a smaller search space; nav-graph nodes on stairs have a smaller degree. From this result, we estimate that solving the elevation problem in VLN-CE can mitigate 6 points of the 12 `SR` drop attributable to subgoal candidate generation, leaving a cumulative 12 `SR` gap between VLN and VLN-CE.



Instruction: Turn around and go past the dining area. Turn to the right and past the couch. Then go down the hallway a little bit and then turn left into the living room. Stop right there and wait.

Fig. 4. Qualitative example of our VLN-CE↻BERT agent operating in VLN-CE.

5.7 Qualitative Example.

We present a qualitative example of VLN-CE↻BERT successfully navigating in a novel environment within VLN-CE (Fig. 4). This example demonstrates all components of our agent: subgoal candidate generation, subgoal selection, and navigation. In each step, subgoal candidates (large, yellow triangles) enable navigation in primary directions. However, not all can be reached, like in Step 1 where a candidate is centered on the kitchen countertop. The SGM candidates diverge from nav-graph node locations. This leads the VLN agent to observe the environment from poses off the nav-graph, yet proper candidate selections are still made over the course of 5 actions. Between steps 0 and 1, the selected subgoal cannot be navigated to in a straight line. Our Local Policy demonstrates obstacle avoidance while planning and executing a path that traverses around the countertop. This example demonstrates that through modular construction, VLN agents can operate successfully and efficiently in continuous environments.

6 Conclusion

In summary, we explore the sim-to-sim transfer of instruction-following agents from topological VLN to unconstrained VLN-CE. Our transfer results in an absolute improvement of 12% over the prior state-of-the-art in VLN-CE. We diagnose the remaining VLN-to-VLNCE gap, identifying subgoal candidate generation as a primary hindrance to transfer. We outline the problem of generating candidates in multi-floor environments to guide future work. Operating VLN agents in continuous environments enables a new interplay between language and navigation topologies. This can lead not only to higher performance in realistic environments, but also to the development of more robust topological navigators.

Acknowledgements. This work was supported in part by the DARPA Machine Common Sense program. 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. 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)
2. Anderson, P., Shrivastava, A., Truong, J., Majumdar, A., Parikh, D., Batra, D., Lee, S.: Sim-to-real transfer for vision-and-language navigation. In: CoRL (2020)
3. 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)
4. Blukis, V., Terme, Y., Niklasson, E., Knepper, R.A., Artzi, Y.: Learning to map natural language instructions to physical quadcopter control using simulated flight. In: CoRL (2020)
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
6. Chaplot, D.S., Gandhi, D., Gupta, S., Gupta, A., Salakhutdinov, R.: Learning to explore using active neural slam. In: ICLR (2020)
7. Chen, K., Chen, J.K., Chuang, J., Vázquez, M., Savarese, S.: Topological planning with transformers for vision-and-language navigation. In: CVPR (2021)
8. Chen, S., Guhur, P.L., Schmid, C., Laptev, I.: History aware multimodal transformer for vision-and-language navigation. In: NeurIPS (2021)
9. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. NeurIPS (2013)
10. Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., Salvador, J., Schwenk, D., VanderBilt, E., Wallingford, M., et al.: Robothor: an open simulation-to-real embodied ai platform. In: CVPR (2020)
11. 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)
12. Gordon, D., Kadian, A., Parikh, D., Hoffman, J., Batra, D.: Splitnet: sim2sim and task2task transfer for embodied visual navigation. In: CVPR (2019)
13. Hahn, M., Chaplot, D.S., Tulsiani, S., Mukadam, M., Rehg, J.M., Gupta, A.: No rl, no simulation: learning to navigate without navigating. In: NeurIPS (2021)
14. Hao, W., Li, C., Li, X., Carin, L., Gao, J.: Towards learning a generic agent for vision-and-language navigation via pre-training. In: CVPR (2020)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
16. Hong, Y., Wu, Q., Qi, Y., Rodriguez-Opazo, C., Gould, S.: Vln bert: a recurrent vision-and-language bert for navigation. In: CVPR (2021)
17. Irshad, M.Z., Ma, C.Y., Kira, Z.: Hierarchical cross-modal agent for robotics vision-and-language navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2021)
18. Irshad, M.Z., Mithun, N.C., Seymour, Z., Chiu, H.P., Samarasekera, S., Kumar, R.: Sasra: Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. In: International Conference on Pattern Recognition (ICPR) (2022)
19. 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)

20. Krantz, J., Gokaslan, A., Batra, D., Lee, S., Maksymets, O.: Waypoint models for instruction-guided navigation in continuous environments. In: ICCV (2021)
21. Krantz, J., Wijmans, E., Majumdar, A., Batra, D., Lee, S.: Beyond the nav-graph: vision-and-language navigation in continuous environments. In: ECCV (2020)
22. Majumdar, A., Shrivastava, A., Lee, S., Anderson, P., Parikh, D., Batra, D.: Improving vision-and-language navigation with image-text pairs from the web. In: ECCV (2020)
23. 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. In: ICCV (2019)
24. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
25. Raychaudhuri, S., Wani, S., Patel, S., Jain, U., Chang, A.X.: Language-aligned waypoint (law) supervision for vision-and-language navigation in continuous environments. In: EMNLP (2021)
26. Tan, H., Yu, L., Bansal, M.: Learning to navigate unseen environments: back translation with environmental dropout. In: NAACL HLT (2019)
27. 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)
28. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. TPAMI (2017)