VLN⋃BERT: A Unified Object and Scene Aware Transformer for Vision-and-Language Navigation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Natural language instructions for visual navigation use *object references* and *scene descriptions* to provide a breadcrumb trail to a goal location. This work presents a multimodal transformer for Vision-and-Language Navigation (VLN) that processes visual observations using both an object detector and a scene classification network, which produce features that mirror these two distinct types of visual cues. In our approach, scene features provide high-level contextual information to support object-level processing. With this design, our model is able to use vision-and-language pretraining – i.e., learning the alignment between images and text from large-scale web data – to substantially improve performance over a strong baseline on the Room-to-Room (R2R) [1] and Room-Across-Room (RxR) [2] benchmarks.

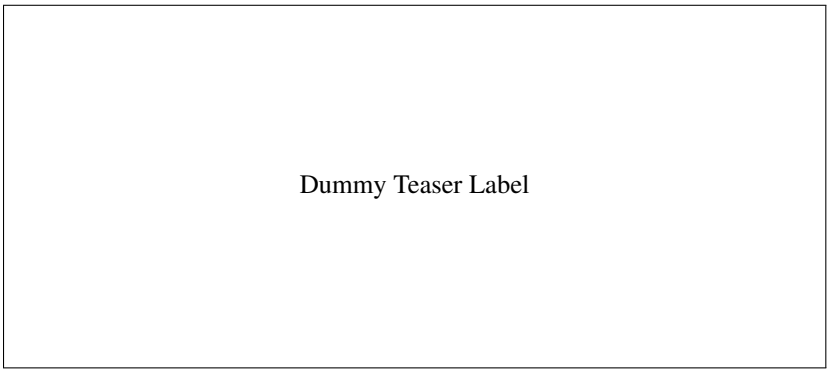

Figure 1: Dummy Teaser caption

1 Introduction

The vision-and-language navigation (VLN) task [1] requires an agent to find a goal location within an environment by following natural language navigation instructions. A central component of this task is associating the instruction with visual landmarks in the environment. Figure 1 provides an illustrative example from the Room-to-Room (R2R) dataset [1] in which the agent needs to follow the indoor navigation instructions: *"Exit the bedroom and turn left. Continue down the hall and into the room straight ahead and stop before the desk with two green chairs."* The visual landmarks in this instruction range from scene-level descriptions (e.g., *"bedroom"* and *"hall"*) to specific object references (e.g., *"desk"* and *"two green chairs"*). Accordingly, VLN agents should be able to recognize this diverse range of visual cues.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

In most VLN methods [1, 3–12], visual observations are first encoded with a convolutional network that was trained to solve an image classification task – either using ImageNet [13] or the Places [14] scene recognition dataset. While ImageNet features may identify objects mentioned in the instructions and Places features might match the scene descriptions, neither solution was explicitly trained to recognize both types of visual cues. Another limitation of standard VLN methods is that the alignment (or grounding) between instructions and visual features is typically learned using only the limited supervision provided by VLN datasets (e.g., the 14k or 42k path-instruction pairs from the R2R [1] and RxR [2] datasets, respectively). Recent work has overcome this challenge by transferring visual grounding from large-scale web corpora to VLN models operating over either scene [15] or object [16] features. We advance this line of research by developing a VLN model that transfers visual grounding to make effective use of both scene and object features.

Our proposed approach extends the VLN↻BERT framework [15] by incorporating object features as an input to the model and for action prediction. We find that naïvely using both scene and object features does not significantly improve VLN performance over a VLN↻BERT baseline. Thus, we propose architectural changes that allow the model to take better advantage of these two distinct types of visual information. Specifically, our model treats scene features as contextual information that is used for object-level processing. This design is similar to the object-level processing paradigm that has been adopted by several recent vision-and-language pretraining (VLP) methods (e.g., [17, 18]), which learn visual grounding by pretraining on large image-text datasets collected from the web. Accordingly, unlike VLN↻BERT, our model consistently uses object features during vision-and-language pretraining and VLN task-specific fine-tuning.

We experiment with our proposed approach on the Room-to-Room (R2R) [1] and Room-Across-Room (RxR) [2] datasets. Empirically we find that our model substantially improves VLN metrics over a strong baseline on R2R and outperforms state-of-the-art methods on English language instructions in RxR. Concretely, our proposed approach improves Success Rate (SR) on the unseen validation split in the R2R dataset by 1.4 absolute percentage points. On RxR, which is a more challenging dataset due to indirect paths and larger variations in path length, we see even larger improvements in SR of 3.7 absolute percentage points, alongside a gain of 1.9 absolute percentage points on the Normalized Dynamic Time Warping (NDTW) metric. Through ablation experiments we discover that vision-and-language pretraining is vital to our approach, suggesting that strong visual grounding is key for using object features in VLN.

To summarize, we make the following contributions:

- We propose a unified multi-modal transformer model for vision-and-language navigation that leverages vision-and-language pretraining for jointly processing scene and object features.
- We show that our approach outperforms strong baselines on the R2R and RxR datasets. [SL] Would be nice to make these a little punchier and include key results.
- We provide analysis that demonstrates the effectiveness of our design decisions and highlights the importance of vision-and-language pretraining to the success of our model.

2 Related Work

Vision-and-Language Navigation. The Room-to-Room (R2R) [1] and Room-Across-Room (RxR) [2] datasets both situate the VLN task within Matterport3D [19] indoor environments. Since the release of R2R there has been steady improvement in VLN task performance [3–12, 15]. Some of the key innovations include using instruction-generation via *"speaker"* models for data augmentation [3, 8], combining imitation and reinforcement learning [4], using auxiliary losses [5, 11], and different pretraining strategies [10, 12, 15]. All of these methods have one thing in common – they process visual observations with a convolutional network pretrained to solve an image classification task (using either the ImageNet [13] or Places [14] datasets). In contrast, this work explores using a combination of features from visual encoders pretrained for scene classification and object detection.

Object Features for VLN. Intuitively, object detections should naturally match the object cues mentioned in VLN instructions. Indeed, several recent studies [16, 20–22] have demonstrated the utility of using object detectors for VLN. In [20, 21] detections are converted into object *"tags"* (i.e., object classification labels), which are encoded using a GLoVe [23] embedding. Similarly, [22]

convert detections into a feature vector using the classification label, object area, and detector confidence. Unlike these methods, our approach directly uses object features produced by a detector and takes advantage of vision-and-language pretraining to transfer visual grounding. In [16], object features are used in a model that solves a path selection task in VLN, which requires pre-exploring an environment before executing the navigation task. By comparison, this work focuses on combining scene and object features to solve instruction-guided visual navigation without pre-exploration.

3 Preliminaries

[AM] TODO: Update this paragraph.

In this section, we review the Vision-and-Language Navigation (VLN) task and the recently proposed VLN↻BERT [15] model for this task. We describe our approach that builds on top of the VLN↻BERT architecture in Section 4.

3.1 Vision-and-Language Navigation

In VLN, an agent is placed in a photo-realistic 3D environment and must navigate to a goal location that is specified through natural language navigation instructions $\boldsymbol{I}$ (as illustrated in Figure 1). At each timestep t, the agent receives a panoramic observation $\boldsymbol{O}_t = \{\boldsymbol{o}_{t,i}\}_{i=1}^{36}$ (introduced by Fried et al. [3]), which is composed of RGB images from 36 viewing angles (12 headings $\times$ 3 elevations). In this work we focus on the *"nav-graph"* setting, in which the agent has access to a navigation graph that specifies a discrete set of navigable locations for each viewpoint in the environment. Using this information the agent selects an action from the set $\boldsymbol{A}_t = \{a_{t,i}\}_{i=0}^{N_t}$ consisting of N_t navigable locations and the `stop` action. The agent is successful if it calls `stop` within 3m of the goal location.

3.2 Visual Encoders for VLN

The goal of this work is to develop a VLN model that can process both scene and object features (as discussed in Section 1). This is accomplished by processing each RGB image $\boldsymbol{o}_{t,i}$ with two different visual encoders that are adopted from prior work. The feature extraction methods used by VLN↻BERT and our model are summarized here; additional details are provided in the Appendix.

Scene Features. Following [1], scene features are produced using a ResNet-152 [24] CNN that was trained on the Places [14] scene recognition dataset. In VLN, understanding the relative angle to each image $\boldsymbol{o}_{t,i}$ is important for following instructions with directional cues (e.g., *"to your left"*). Accordingly, the relative heading $\theta_{t,i}$ and elevation $\phi_{t,i}$ to each RGB image is encoded into a 4-dimensional vector $[\sin\theta_{t,i}, \cos\theta_{t,i}, \sin\phi_{t,i}, \cos\phi_{t,i}]$ and combined with the CNN features to construct the final scene feature vector $\boldsymbol{f}_{t,i}$.

Object Features. To generate object features we use the approach taken in [16], in which a Faster R-CNN [25] detector trained on Visual Genome [26] using the training procedure from [27] is used to produce a set of M region features for each RGB image. Heading and elevation angles to each image region are encoded (as above) and combined with the region features, resulting in a set of object features $\{\boldsymbol{g}_{t,i,j}\}_{j=1}^{M}$ for each RGB image.

3.3 Multimodal Transformers

Here we provide a brief overview of multimodal transformers (e.g., OSCAR [17]), which provide a basis for VLN↻BERT and our approach. Multimodal transformers are an extension of transformer-based language models such as BERT [28] that process paired image and text data. Like their language counterparts, the inputs to these models are a set of tokens, which are processed by a series of transformer encoder layers [29] to output representations for each token. Commonly, language tokens are constructed using the approach taken in BERT [28] and *"visual tokens"* are generated using region features from an object detector (such as the specific model described above). Additional details can be found in [17], which describes OSCAR – the multimodal transformer used in this work.

[AM] TODO: add a sentence about how multimodal transformers are used in our work

3.4 Multimodal Transformers for VLN

VLN requires sequentially following navigation instructions (e.g., *"Exit the bedroom..."* then *"Continue down the hall..."* then *"stop before the desk with two green chairs."*). Accordingly, maintaining a history of the agent's state is helpful for understanding which sub-instruction to follow at each timestep. Traditional VLN agents use recurrent neural networks (e.g., LSTM [30]) to model state history. In contrast, VLN↻BERT [15] introduces a generic recurrence mechanism that can, in principle, be added to any multimodal transformer model to refashion it for the VLN task. In this work we extend the OSCAR [17] instantiation of VLN↻BERT, which allows us to study the effect of vision-and-language pretraining on a VLN agent that processes both scene and object features.

VLN↻BERT [15] uses a multimodal transformer to process a panoramic observation $\boldsymbol{O}_t$ alongside the navigation instructions $\boldsymbol{I}$ to selection an action from $\boldsymbol{A}_t$. The recurrence mechanism in VLN↻BERT is operationalized using a state token $\boldsymbol{s}_t$, which is passed as an input to the multimodal transformer and updated at each timestep using self-attention (as discussed below).

Initialization. The state token is initialized by passing the word tokens from the navigation instructions $\boldsymbol{I}$ along with special `[CLS]` and `[SEP]` tokens through the multimodal transformer. The output corresponding to the `[CLS]` token is used to set $\boldsymbol{s}_0$ and the outputs for the word tokens (denoted as $\boldsymbol{X}$) are used as the language representation during navigation.

Visual Tokens. For each panoramic observation $\boldsymbol{O}_t = \{\boldsymbol{o}_{t,i}\}_{i=1}^{36}$, VLN↻BERT runs a Places [14] CNN on each of the 36 RGB images to extract scene features $\{\boldsymbol{f}_{t,i}\}_{i=1}^{36}$ (see Section 3.2). Since the agent can only navigate to one of N_t navigable viewpoints, only features for images pointing towards these locations are used as visual tokens. To facilitate the `stop` action an all zeros feature vector is added to the set of visual tokens (denoted as $\boldsymbol{F}_t$).

Navigation. At each timestep ($t > 0$), the input to the multimodal transformer is composed of the previous state $\boldsymbol{s}_{t-1}$, the encoded instruction $\boldsymbol{X}$, and the visual tokens $\boldsymbol{F}_t$. To reduce computational complexity, the instruction tokens are not updated during navigation and only serve as keys within the attention-based processing. Actions are selected using attention scores for the visual tokens from the last layer of the multimodal transformer model. Specifically, attention scores for the visual tokens are calculated with respects to the state token (i.e., the state token serves as the query and visual tokens serve as keys). [AM] Should we add an equation describing this? The scores are passed through a softmax function and treated as the probability of moving to that navigable viewpoint or stopping. Finally, the state history is maintained by using the output representation for the state token $\boldsymbol{s}_{t-1}$ as the state token input at the next timestep (after a *"refinement"* procedure described in [15]).

4 Approach

One key limitation of VLN↻BERT is that it only uses high-level scene features, which might fail to encode specific object references such as *"desk"* or *"two green chairs"* (see Figure 1). Second, there is a mismatch between the pretraining and fine-tuning paradigm while training the model. OSCAR [17] is pre-trained on Conceptual Captions [31] using image region features from an object detector as input. However, during fine-tuning on the VLN dataset, VLN↻BERT simply feeds high-level scene features as input and hopes that the model can adapt to the new feature distribution.

To overcome these two limitations, we present a unified multi-modal transformer model which uses both scene and object features as shown in Figure 2. Interestingly, we find that simply adding objects as additional inputs in VLN↻BERT does not help. To address this challenge, we propose a selective attention mechanism that only refines a subset of the input tokens. This in turn shifts the processing in our model closer to the base model (i.e., OSCAR [17]) used for vision-and-language pretraining.

4.1 Input Tokens

Our approach expands the set of input tokens used in VLN↻BERT to include visual tokens for object features.[1] Specifically, we use object features from the Faster R-CNN detector described in Section 3.2. Similar to the filtering done on scene features, only object features corresponding with navigable viewpoints are retained to form the set of object tokens $\boldsymbol{G}_t$. Unlike scene tokens $\boldsymbol{F}_t$, each

[1] [AM] Add a note about how VLN↻BERT using object features.

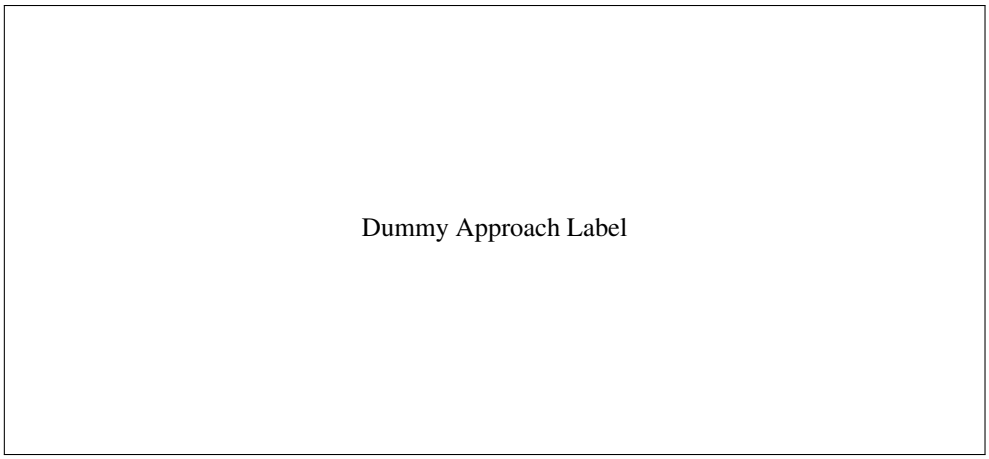

Figure 2: Dummy Approach caption

navigable viewpoint may include zero or more object tokens (based on the outputs from the object detector). Thus, we modify the VLN↻BERT approach for calculating action probabilities (discussed in Section 4.3). At each timestep, inputs to the multimodal transformer consists of four sets of tokens: the previous state $\boldsymbol{s}_{t-1}$, encoded instruction $\boldsymbol{X}$, scene tokens $\boldsymbol{F}_t$, and object tokens $\boldsymbol{G}_t$.

4.2 Selective Object Attention

In initial experiments (detailed in Section 5) we found that simply adding object tokens to the set of inputs used by VLN↻BERT did not improve performance. We hypothesize that scene tokens dominate the VLN training, which does not allow the model to properly learn how to use object features. To mitigate this issue, we modify the attention pattern used within the multimodal transformer as illustrated in Figure 2. With our attention pattern, both the language $\boldsymbol{I}$ and scene $\boldsymbol{F}_t$ tokens only serve as keys (and values) during the attention-based processing, while the state token $\boldsymbol{s}_t$ and object tokens $\boldsymbol{G}_t$ operate as queries, keys and values (which is standard in transformer models).

Intuitively, this attention pattern focuses processing on the object tokens, while allowing the scene tokens to providing high-level contextual information to support this object-level processing. Additionally, the scene tokens are still used for action prediction (see section 4.3), which provides the model with flexibility in using both sources of information. We hypothesize that this design leads to more effective transfer learning (demonstrated in Section 5), because the object-level processing more closely matches the pretraining setup, in which the model only receives language and object tokens as input.

4.3 Action Prediction

Recall that in VLN↻BERT actions are selected using attention scores over the scene tokens $\boldsymbol{F}_t$ (see Section 3.4), which correspond to each of the navigable viewpoints and the `stop` action. These attention scores are also used to update the state token (as described in [15]), so all of the values (not just the maximum) are used by the model. In our approach attention scores are also available for object tokens $\boldsymbol{G}_t$ that correspond to image regions pointing towards each of the navigable viewpoints. However, it is not immediately evident how the scene and object attention scores should be combined.

Through initial experiments, we discovered that aggregating attention scores for each viewpoint by selecting the maximum over the scene and object scores yielded the best results. Intuitively, this approach allows the model to either select a relevant object (which may be mentioned in the instructions such as *"green chairs"*) or the full scene (which might match scene descriptions such as *"hallway"*) to represent each navigable viewpoint. Accordingly, the model is able to select actions using features that match both types of visual cues.

[AM] We tried many alternatives here, should we provide some examples?

Table 1: R2R and RxR val-unseen results. † indicates reproduced results.

		R2R				RxR			
	Methods	TL	NE ↓	SR ↑	SPL ↑	NE ↓	SR ↑	SDTW ↑	NDTW ↑
1	Random	9.77	9.23	16	-	9.5	5.1	3.8	27.6
2	Human	-	-	-	-	1.32	90.4	74.3	77.7
3	RxR baseline[2]	-	-	37	32	10.1	25.6	20.3	41.3
4	EnvDrop [8]	10.70	5.22	52	48	-	-	-	-
5	PREVALENT [12]	10.19	4.71	58	53	-	-	-	-
6	VLN↻BERT [15] (init. OSCAR)	11.86	4.29	59	53	-	-	-	-
7	VLN↻BERT [15] (init. PREVALENT)	12.01	3.93	63	57	-	-	-	-
8	VLN↻BERT [15] †	12.16	4.40	58	51	7.31	40.5	50.8	65.9
9	Ours	12.15	**4.28**	**59**	**53**	**6.72**	**44.2**	**54.6**	**67.8**

5 Experiments

5.1 Training and Evaluation

Datasets. We initialize our model with a pre-trained OSCAR model [17] for training. We evaluate our method on R2R [1] and RxR [2] datasets. R2R dataset contains 21,567 instruction-path pairs which are divided into four splits: training (14,025), val-seen (1,020), val-unseen (2,349) and test-unseen (4,173). Val-seen split uses environments from the training split but the instruction-path pairs are novel. Val-unseen and test-unseen tests the generalization capacity of agents on unseen environments with novel instruction-path pairs. For R2R, we also use augmented dataset generated from speaker model as done in prior works [8, 12, 15]. RxR [2] is a recently introduced multi-lingual VLN dataset. It contains total 126K instruction-path pairs in 3 languages (Hindi, English and Telugu) on the same 90 Matterport scenes which were used in R2R. Moreover, it contains English instructions from two regions: India (`en-IN`) and US (`en-US`). Since our model is pre-trained on Conceptual Captions dataset which is in English language, we just use English instructions (combining `en-IN` and `en-US` instructions) from RxR dataset for training and evaluation. Specifically, we use 26,464 English instructions from RxR train split for training and test on 4,551 English instructions from RxR val-unseen split.

Evaluation. We follow standard evaluation protocol for R2R and RxR datasets. On R2R, we report (↑ indicates higher is better and ↓ indicates lower is better): Trajectory Length (TL), Navigation Error (NE ↓), Success Rate (SR ↑) and Success weighted by Path Length (SPL ↑). On RxR dataset, in addition to NE and SR metrics, we also report Normalized Dynamic Time Warping (NDTW ↑) and Success weighted by normalized Dynamic Time Warping (SDTW ↑) metrics which explicity measure path adherence. We refer the reader to [1, 32, 33] for a detailed explanation of these metrics.

5.2 Implementation Details

Our model has been implemented in PyTorch [34] on a single Nvidia TitanX GPU. For both R2R and RxR, we fine-tune our model with behaviour cloning and reinforcement learning objectives. Each minibatch consists of 50% rollouts from behaviour cloning and 50% from reinforcement learning (policy gradient). We train using a constant learning rate of 1e-5 with AdamW optimizer and batch size 16 for 300k iterations. We extract visual features for scene tokens with a pre-trained ResNet-152 on Places [14] dataset. For object tokens, we extract object proposals from a pre-trained bottom-up attention model [27] and adopt filtering procedure from [16] to discard redundant proposals. We use the exact same hyperparameters for VLN↻BERT baseline and our approach in all the experiments.

5.3 Results and Analysis

How does our method compare to state-of-the-art methods? We compare our approach with the recently published state-of-the-art approaches in Table 1 on R2R [35] and RxR [2] val-unseen splits. EnvDrop [8] trains an encoder-decoder model [3] with mixture of imitation and reinforcement learning on augmented data (in addition to R2R training split) with back-translated instructions and "dropped out" environments in order to generalize well to unseen environments. PREVALENT [12]

Table 2: R2R test results. † indicates reproduced results.

	Methods	R2R Test Unseen TL	NE ↓	SR ↑	SPL ↑
1	Random	9.89	9.79	13	12
2	Human	11.85	1.61	86	76
3	EnvDrop [8]	11.66	5.23	51	47
4	PREVALENT [12]	10.51	5.30	54	51
5	VLN↻BERT [15] (init. OSCAR)	12.34	4.59	57	53
6	VLN↻BERT [15] (init. PREVALENT)	12.35	4.09	63	57
7	VLN↻BERT [15] †	12.78	4.55	58	52
8	Ours	12.26	4.49	**58**	**53**

Table 3: R2R Ablations.

	Models	Object Features	Score Aggregation	Selective Attention	Val Seen TL	NE ↓	SR ↑	SPL ↑	Val Unseen TL	NE ↓	SR ↑	SPL ↑
1	Baseline [15]				11.02	3.11	69.93	65.65	12.16	4.40	57.90	51.43
2		✓			11.05	2.89	71.69	67.22	12.25	4.35	57.26	50.96
3		✓	✓		11.12	3.24	70.03	66.59	12.05	4.34	57.85	51.73
4	Ours	✓	✓	✓	11.81	3.63	62.78	58.01	**12.15**	**4.28**	**58.71**	**53.24**

builds on top of EnvDrop method by pre-training a multi-modal transformer model on back-translated augmented VLN data. After pre-training, PREVALENT method feeds contextual word embeddings from this pre-trained transformer model into EnvDrop [8] encoder-decoder model for fine-tuning on R2R dataset. VLN↻BERT extends PREVALENT by directly fine-tuning the pre-trained multi-modal transformer model for navigation instead of using an ensemble of encoder-decoder model and a multi-modal transformer.

RxR is more challenging dataset than R2R since it contains much longer instructions and paths (which are often not shortest) with dense object and scene references. On RxR, our approach (row 6) outperforms the previous state-of-the-art RxR method (row 1) by 26.33% absolute on NDTW and 19.03% absolute on Success Rate (SR) metrics. We also train and evaluate VLN↻BERT baseline on RxR dataset with the same hyperparameters as ours. Our unified approach VLN⋃BERT which uses both object and scene features for navigation also beats (only scene features) VLN↻BERT baseline (row 5) by 3.7% on SR, 3.8% on SDTW and 1.9% on NDTW metrics. Moreover, the gap between our approach (row 6) and VLN↻BERT baseline (row 5) is significantly increased in RxR than R2R which highlights that object features significantly help as you move to a larger dataset with dense instructions containing more entity references. On R2R, our approach (row 6) shows 1.81% gain in SPL over the reproduced VLN↻BERT baseline (row 5) and is also competitive with the published VLN↻BERT result.

Table 2 reports results on R2R test-unseen split which again demonstrates that our method is competitive with the previous state-of-the-art approaches across all the metrics. For completeness, we also report VLN↻BERT with PREVALENT initialization in Table 1 (row 5) and Table 1 (row 4) since it achieves state-of-the-art results on R2R. However, comparison with it is unfair since it is pre-trained and fine-tuned with high-level scene features on augmented VLN data which has much less domain gap between pre-training and fine-tuning than VLN↻BERT with OSCAR initialization (row 4) which is pre-trained on Conceptual Captions and fine-tuned on VLN data. However, the focus of our work is to show how we can effectively leverage pre-training on large-scale V&L data to boost performance on VLN task *without* any additional pre-training on VLN data. Pre-training on VLN data requires large number of instruction-path pairs which is very expensive collection process [1, 2]. PREVALENT resort to using noisy augmented data generated by speaker model for pre-training which leads to improvements over OSCAR pre-training. We posit that this gap can be overcome by pre-training OSCAR [17] model on larger V&L data whose collection process is relatively much cheaper and scalable than VLN.

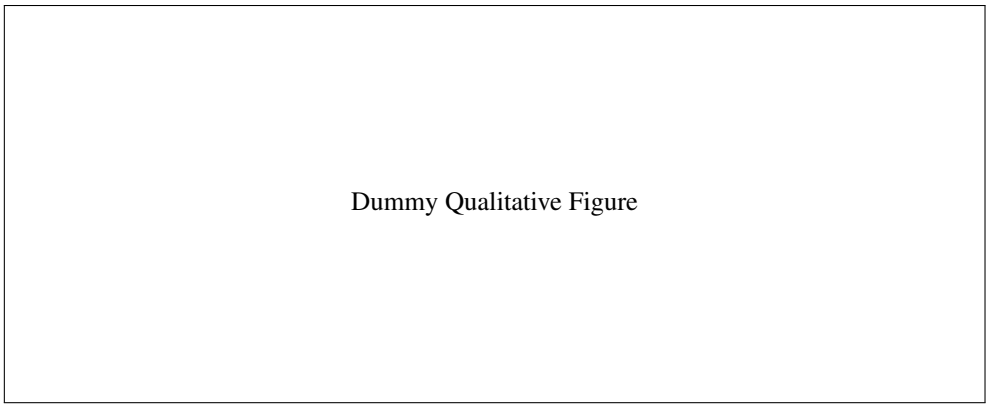

Figure 3: Dummy caption

Learning curves. Figure 4 plots the learning curves for VLN↻BERT baseline and our method on RxR val-unseen split. [AB] Compare with human baseline on NDTW metric. Our model gives NDTW score 69 and human baseline is at 77.

Do object features help just as additional input? In Table 3 (row 2), we show the results of an ablation experiment on val-seen and val-unseen split of R2R dataset in which we used object features just as additional input to VLN↻BERT baseline. In this ablation, all visual tokens (object and scene tokens) get refined via self-attention through multiple layers in the tranformer and we get action probabilities and next hidden state in the similar fashion as VLN↻BERT baseline [15]. We find out that adding object tokens as additonal input does not help in generalization – we see improvements by $\sim$2% in SPL and SR on val-seen but it is slightly worse than baseline on val-unseen by 0.64% on SR and 0.47% on SPL. Since additional detected object proposals are noisy, this experiment demonstrates that updating high-scene tokens from these object tokens introduces additional noise and causes overfitting. Moreover, during navigation, state token `[CLS]` gets updated from these updated scene tokens in experiment 2 which introduces more noise. In 3 (row 3), we fix this noise in `[CLS]` token by aggregating scores from objects and scenes for getting action probabilities and using corresponding token embeddings for updating state token `[CLS]`. Incorporating score aggregation (row 3) improves object features alation (row 2) SPL by 0.77% on val-unseen and reduces overfitting since it has lower gap between val-seen and val-unseen metrics.

Do object features help without any pre-training? Table 4 reports results of VLN↻BERT baseline and our approach without V&L pre-training on R2R val-useen split. As evident from the Table 4, both the approaches perform similar across all the metrics. This further demonstrates that object grounding is crucial for our approach to work; our approach improves performance on RxR and R2R datasets compared to VLN↻BERT scene features baseline predominantly because it effectively utilizes the large-scale V&L pre-training.

Table 4: Results on R2R without V&L pre-training.

		R2R Val Unseen			
	Models	TL	NE ↓	SR ↑	SPL↑
1	VLN↻BERT [15]	10.31	5.21	49.21	45.53
2	Ours	12.00	4.99	50.36	44.69

How often does the model pick up objects rather than scenes for navigation?

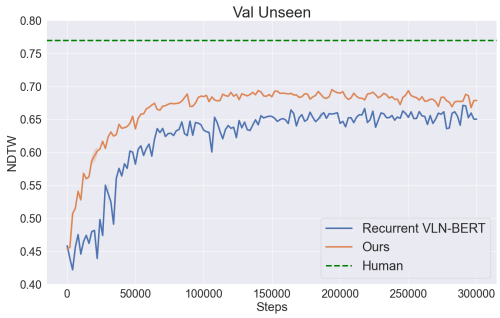

Figure 4: NDTW learning curve comparing Recurrent VLN-BERT, our method and human baseline on RxR val-unseen split.

6 Conclusion

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018.

[2] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2020.

[3] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, pages 3314–3325, 2018.

[4] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019.

[5] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*, 2019.

[6] Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6741–6749, 2019.

[7] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *CVPR*, pages 6732–6740, 2019.

[8] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195*, 2019.

[9] Peter Anderson, Ayush Shrivastava, Devi Parikh, Dhruv Batra, and Stefan Lee. Chasing ghosts: Instruction following as bayesian state tracking. *arXiv*, 2019.

[10] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah Smith, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. *arXiv preprint arXiv:1909.02244*, 2019.

[11] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10012–10022, 2020.

[12] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146, 2020.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[14] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[15] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation, 2020.

[16] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web, 2020.

[17] Xiujun Li, Xi Yin, C. Li, X. Hu, Pengchuan Zhang, Lei Zhang, Longguang Wang, H. Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *ECCV*, 2020.

[18] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, 2019.

[19] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision (3DV)*, 2017.

[20] Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. Are you looking? grounding to multiple modalities in vision-and-language navigation, 2019.

[21] Yicong Hong, Cristian Rodriguez-Opazo, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation, 2020.

[22] Yubo Zhang, Hao Tan, and Mohit Bansal. Diagnosing the environment bias in vision-and-language navigation. *arXiv preprint arXiv:2005.03086*, 2020.

[23] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP*, 2014.

[24] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[26] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*, 2016.

[27] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. *arXiv preprint arXiv:1707.07998*, 2017.

[28] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997.

[31] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018.

[32] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents, 2018.

[33] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping, 2019.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[35] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **[TODO]**
   (b) Did you describe the limitations of your work? **[TODO]**
   (c) Did you discuss any potential negative societal impacts of your work? **[TODO]**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[TODO]**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[TODO]**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[TODO]**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **[TODO]**
   (b) Did you mention the license of the assets? **[TODO]**
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[TODO]**

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[TODO]**

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

Table 5: Notation

Symbol	Description
$\boldsymbol{I}$	Natural language navigation instruction
$\boldsymbol{O}_t = \{\boldsymbol{o}_{t,i}\}_{i=1}^{N_t}$	Panoramic observation at time t for N_t navigable viewpoints
$\boldsymbol{A}_t = \{a_{t,k}\}_{i=0}^{N_t}$	Actions available at time t consisting of N_t navigable viewpoints and `stop`
$\theta_{t,i}$	Heading angle for view i at time t
$\phi_{t,i}$	Elevation angle for view i at time t
$\boldsymbol{f}_{t,i}$	CNN scene features for view i at time t
$\boldsymbol{g}_{t,i,j}$	Faster R-CNN features for region j in view i at time t
$p(a_{t,j})$	Probability of taking action $a_{t,k}$ at time t
$\boldsymbol{s}_t$	Agent state at time t

Role of attention pattern on performance. [AB] TODO: Update this paragraph when all the experiments finish.

Table 6: R2R attention ablations. [AB] TODO: Update numbers when experiments finish.

				Val Seen				Val Unseen			
	Models	Input Tokens	Query Tokens	TL	NE ↓	SR ↑	SPL ↑	TL	NE ↓	SR ↑	SPL ↑
1	Baseline [15]	$\langle \boldsymbol{s}, \boldsymbol{X}, \boldsymbol{V} \rangle$	$\langle \boldsymbol{s}, \boldsymbol{V} \rangle$	11.02	3.11	69.93	65.65	12.16	4.40	57.90	51.43
2	Scene attention	$\langle \boldsymbol{s}, \boldsymbol{X}, \boldsymbol{V}, \boldsymbol{O} \rangle$	$\langle \boldsymbol{s}, \boldsymbol{V} \rangle$	11.12	3.24	70.03	66.59	12.05	4.34	57.85	51.73
3	All attention	$\langle \boldsymbol{s}, \boldsymbol{X}, \boldsymbol{V}, \boldsymbol{O} \rangle$	$\langle \boldsymbol{s}, \boldsymbol{V}, \boldsymbol{O} \rangle$	-	-	-	-	-	-	-	-
4	Object attention	$\langle \boldsymbol{s}, \boldsymbol{X}, \boldsymbol{O} \rangle$	$\langle \boldsymbol{s}, \boldsymbol{O} \rangle$	-	-	-	-	-	-	-	-
5	Ours	$\langle \boldsymbol{s}, \boldsymbol{X}, \boldsymbol{V}, \boldsymbol{O} \rangle$	$\langle \boldsymbol{s}, \boldsymbol{O} \rangle$	11.81	3.63	62.78	58.01	12.15	4.28	58.71	53.24