SOAT: A Scene- and Object-Aware Transformer for Vision-and-Language Navigation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Natural language instructions for visual navigation tend to use *scene descriptions* (e.g., *'bedroom'*) and *object references* (e.g., *'green chairs'*) 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 method, scene features contribute 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 on the Room-to-Room (R2R) [1] and Room-Across-Room (RxR) [2] benchmarks. Specifically, our approach leads to improvements of 1.8% absolute in SPL on R2R and 3.7% absolute in SR on RxR. Our analysis reveals larger gains on navigation instructions with many object references – further suggesting that our approach is better able to exploit object features and align them with references in the instructions.

1 Introduction

The vision-and-language navigation (VLN) task [1] requires an agent to follow a path through an environment that is specified by 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 instruction from the Room-to-Room (R2R) dataset [1]: *'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.'* Note that 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'*). To be successful, a VLN agent should be able to recognize this diverse range of visual cues.

Learning appropriate groundings between these referring expressions in language and their visual correspondence is made difficult in the VLN setting due to limited visual diversity. The VLN training set consists of only 61 unique environments and models simply cannot learn the long-tail of visual cues appearing in new environments. To address this challenge, recent work has shown the promise of transferring multimodal representation models pretrained on large static image-text corpuses before finetuning for the embodied VLN task [3, 4].

In most VLN methods [1, 5–14], visual observations are first encoded with a convolutional network that was trained to solve an image classification task – either using ImageNet [15] or the Places [16] 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, which is a limitation we address in this paper. Further, pretrained multimodal models have typically been trained using features extracted from object detection networks

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

Figure 1: Illustrative VLN Instruction. Visual landmarks in the instruction include scene descriptions (e.g., *'bedroom'* and *'hall'*) and object references (e.g., *'desk'* and *'green chairs'*). Unlike most previous methods, our model uses both scene (△) and object features (○) to match the visual references in the instructions. We employ a novel attention mechanism and view aggregation strategy to select the most relevant features for action prediction.

rather than the image-level representations described above. Despite this, prior work leveraging these models [3] has used image-level features – leading to a significant domain shift between the input during pretraining and VLN finetuning.

In this work, we address these issues by (a) explicitly encoding the inductive bias that the world is composed of objects and scenes and (b) using object-level features from detection models to better align with pretrained multimodal representation models. The starting point for our work is a VLN↻BERT [3] model, which processes scene-level features from a Places [16] CNN with a multimodal transformer – i.e., a transformer-based [17] model that process images and text – modified with a recurrence mechanism designed for the VLN task. Our work extends this approach by including object features as an additional input to the model. However, we find that simply adding object features to VLN↻BERT does not improve (but rather slightly reduces) VLN performance. Thus, we propose architectural changes that allow the model to take better advantage of these two distinct types of visual information. Specifically, we change the attention pattern within the the transformer to effectively freeze the scene representations and focus the processing on the object-level features. The result is a new model for VLN that produces contextualized representations of objects – leveraging scene features to provide high-level contextual cues.

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 on VLN metrics over our VLN↻BERT baseline on R2R and outperforms state-of-the-art methods on English language instructions in RxR. Specifically, our proposed approach improves Success weighted by Path Length (SPL) on the unseen validation split in R2R by 1.8 absolute percentage points.

On RxR – a more challenging dataset due to indirect paths and greater variations in path length – we see even larger improvements. Success Rate (SR) improves by 3.7 absolute percentage points, alongside a gain of 2.4 absolute percentage points on the Normalized Dynamic Time Warping (NDTW) metric. Through ablation experiments we find that (consistent with the observations in [4]) vision-and-language pretraining is vital to our approach, which suggests that strong visual grounding is key for using object-level features in VLN. Additionally, on RxR instructions that heavily mention objects (6 or more object references in the instruction), our method gives an even stronger improvements over VLN↻BERT of 7.9 absolute percentage points on SR.

To summarize, we make the following contributions:

- We propose a scene- and object-aware transformer (SOAT) model for vision-and-language navigation that uses both scene-level and object-level features – explicitly encoding the inductive bias that the world is composed of objects and scenes. Our model uses a novel attention masking technique and view aggregation strategy, which both improve performance.

- We demonstrate that our approach outperforms strong baselines by 1.8% absolute in SPL on R2R and by 3.7% absolute in SR on RxR.
- We show that our method has significantly stronger performance on instructions that heavily mention objects. Our model is able to exploit object features and align them with corresponding references in the instruction to improve SR by 7.91% absolute on object-heavy instructions.

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 [18] indoor environments. Since the release of R2R there has been steady improvement in VLN task performance [3, 5–14]. Some of the key innovations include using instruction-generation via *'speaker'* models for data augmentation [5, 10], combining imitation and reinforcement learning [6], using auxiliary losses [7, 13], and different pretraining strategies [3, 12, 14]. All of these methods have one thing in common – they process visual observations with a single convolutional network pretrained to solve an classification task (using either the ImageNet [15] or Places [16] datasets). In contrast, this work explores using a combination of features from visual encoders pretrained for scene classification and object detection.

Object Detectors in VLN. Intuitively, object detections should naturally match the object cues (e.g., *'green chairs'*) mentioned in VLN instructions. Indeed, several recent studies [4, 19–21] have demonstrated the utility of using object detectors for VLN. In [19, 20] object classification labels from an object detector are encoded using a GLoVe [22] embedding. Similarly, [21] convert detections into a feature vector using the classification label, object area, and detector confidence. Unlike these methods, our model directly uses object features produced by a detector, which provide a richer, high-dimensional representation of each region. Additionally, our approach takes advantage of vision-and-language pretraining, which eases the burden of learning the grounding between language and object representations from scratch using only VLN data, as is done in these prior methods. In [4], 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 navigating without pre-exploration.

3 Preliminaries: VLN, Visual Encoders, Multimodal Transformers

This section reviews the vision-and-language navigation (VLN) task and describes how multimodal transformers are used in the recently proposed VLN↻BERT [3] model. Our approach builds on VLN↻BERT and is described in Section 4.

3.1 Vision-and-Language Navigation

In VLN, agents are placed in a photo-realistic 3D environment and must navigate to a goal location that is specified through natural language navigational instructions $\boldsymbol{I}$ (illustrated in Figure 1). At each timestep t, the agent receives a set of panoramic observations $\boldsymbol{O}_t = \{\boldsymbol{o}_{t,i}\}_{i=1}^{36}$ composed of RGB images from 36 viewing angles (12 headings $\times$ 3 elevations). We follow the VLN with known navigation graph setting [1, 2] – agents have access to a graph that specifies a set of navigable locations from each viewpoint in the environment. We agree with the limitations of this setting discussed in [23]. However, we report results for the nav-graph setting to be consistent with the long line of prior work in this area. We plan on generalizing to continuous environments in the future. Using the nav-graph, agents select 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 Multimodal Transformers

Here we provide a brief overview of multimodal transformers (e.g., OSCAR [24]), which provide a basis for the VLN↻BERT architecture. Multimodal transformers are an extension of transformer-based [17] language models such as BERT [25] that process image-text pairs. Like their language counterparts, one of the inputs to these models is a set of word tokens that are encoded (as in

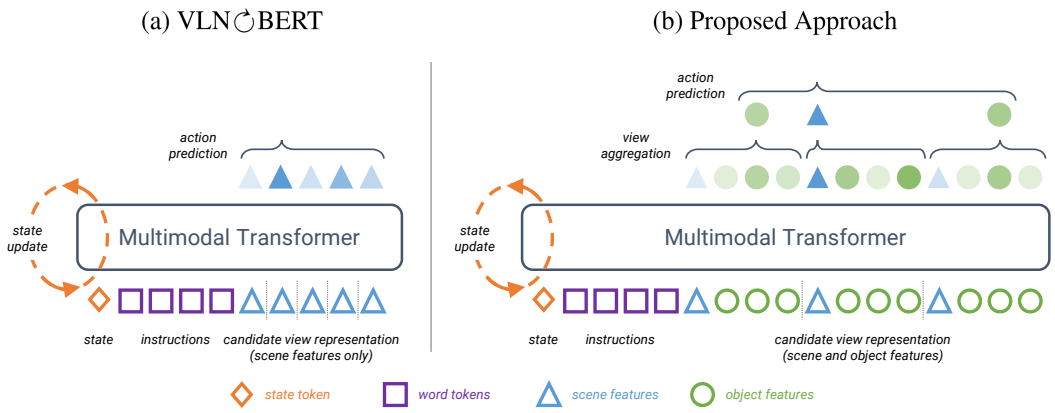

Figure 2: Model Architecture. (a) VLN↻BERT uses a multimodal transformer to encode state history, instruction tokens and scene features from candidate views. The candidate view corresponding to the scene features with the highest attention score is chosen as the next action. (b) We extend VLN↻BERT to *include object features* from all candidate views as additional input. In the view aggregation step, the model picks the visual features (either object or scene) with the maximum attention score to represent each candidate view. Finally, for action prediction, the model chooses the candidate view with the highest attention score as the next action.

BERT [25]) with a learned embedding and combined with positional information (i.e., word order). Commonly, the visual input (i.e., the image) is first processed by an object detector (e.g., a Faster R-CNN [26] model trained on Visual Genome [27]) to produce a set of region features that are combined with spatial information (e.g., the offset to the bounding box in the image). To summarize, the multimodal input consisting of L word tokens $\{w_1, \ldots, w_L\}$ and M image regions $\{r_1, \ldots, r_M\}$ can be written as

$$\texttt{[CLS]}, w_1, \ldots, w_L, \texttt{[SEP]}, r_1, \ldots, r_M$$

where [CLS] is a special token used as a global representation of the input and the [SEP] token separates modalities. This set of multimodal inputs are fed to a series of transformer encoder layers [17] to produce contextualized representations for each input using attention-based processing.

3.3 Multimodal Transformers for VLN

The language instructions and panoramic image at one timestep in VLN match the input interface to multimodal transformers. However, 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 [28]) to model state history. In contrast, VLN↻BERT [3] 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 [24] instantiation of VLN↻BERT.

As shown in Figure 2a, VLN↻BERT [3] uses a multimodal transformer to process word tokens from the navigation instructions and scene features. Rather than processing all 36 panoramic observations, VLN↻BERT considers only the subset of observation that align best with directions towards navigable next locations (average ∼4 such observations). Note that each view corresponds to a navigation action in $\boldsymbol{A}_t$. Accordingly, output representations for each view are used for action prediction. The recurrence mechanism in VLN↻BERT is operationalized using a state token $\boldsymbol{s}_t$, which is fed as an input to the multimodal transformer and maintained throughout navigation.

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

Visual Inputs. For each RGB image in the N_t candidate views, VLN↻BERT runs a convolutional network trained on Places [16] to extract high-level scene features. To facilitate the `stop` action an all-zeros feature vector is added to the set of visual inputs. We denoted this set of visual inputs as $\boldsymbol{V}_t$.

Navigation. At each timestep, the input to the multimodal transformer is composed of the state token $\boldsymbol{s}_t$, the encoded instruction $\psi(\boldsymbol{I})$, and the visual inputs $\boldsymbol{V}_t$. To reduce computation, the instruction representation is not updated during navigation; the encoded word tokens simply serve as keys and values when processed by the multimodal transformer.

Actions are predicted using "state-conditioned" attention scores for the scene features. These scores are produced at the final layer of the multimodal transformer. Concretely, let the final layer outputs for the state token $\boldsymbol{s}_t$ and scene features $\boldsymbol{V}_t$ be denoted as $\psi(\boldsymbol{s}_t) \in \mathbb{R}^d$ and $\psi(\boldsymbol{V}_t) \in \mathbb{R}^{(N_t+1)\times d}$ where N_t is the number of navigable locations and d is the model's hidden dimension size. State-conditioned attention scores are calculate using a scaled dot-product as

$$\alpha(\boldsymbol{V_t}) = \frac{\psi(\boldsymbol{s}_t)\psi(\boldsymbol{V}_t)^T}{\sqrt{d}} \tag{1}$$

These scores $\alpha(\boldsymbol{V_t})$ are normalized using the softmax function $\tilde{a}(\boldsymbol{V_t}) = \texttt{softmax}(\alpha(\boldsymbol{V_t}))$ and used as the probability of moving to one of the N_t navigable locations or stopping. Finally, the state history is maintained by constructing the next state token $\boldsymbol{s}_{t+1}$ with the output representation for the state token $\psi(\boldsymbol{s}_t)$ using a "refinement" procedure detailed in the Appendix.

4 Approach

Motivated by the observation that VLN instructions often use scene descriptions (e.g., *'bedroom'*) and object references (e.g., *'green chairs'*) to provide visual cues (discussed in Section 1), our goal is to design a model that effectively uses both scene and object features for VLN. One straightforward approach is to include object features as an additional input to VLN↻BERT [3]. However, we find that this simple solution does not improve VLN performance (see Table 3 row 2), which is perhaps why object features were not used for navigation in [3].[1] Thus, we redesign the VLN↻BERT architecture to create a scene- and object-aware transformer (SOAT) model that benefits from using both scene and object features.

First, we modify the processing in the model to focus on object features by designing the attention mask shown in Figure 3. The mask only allows the state token and object features to be updated by the multimodal transformer. Second, as illustrated in Figure 2b, the output from our model includes scene and object representations for each candidate view. Thus, we design a simple approach to aggregate this information into a single representation for each view, which is used for action prediction.

4.1 Visual Inputs

To generate scene and object features we use two different visual encoders to process RGB images from the panoramic observations. Matching VLN↻BERT, we use a CNN trained on the Places [16] dataset to extract scene features for each candidate view, and use an all-zeros feature vector for the `stop` action to produce the set of scene features $\boldsymbol{V}_t$. For object features, we use the same object detector that was used to pretrain the multimodal transformer (i.e., OSCAR [24]) that we use to initialize our model – avoiding any additional domain gap between pretraining and fine-tuning. Specifically, we use a Faster R-CNN [26] detector trained on Visual Genome [27] using the training procedure from [30]. We denote the set of object features for detections from all of the candidate views as $\boldsymbol{G}_t$. Note that for a given image the detector may or may not detect any objects. As a result, in our model, each candidate view is represented with one scene feature and zero or more object feature vectors. As illustrated in Figure 2b, the full set of inputs to our model include the state token $\boldsymbol{s}_t$, encoded instructions $\psi(\boldsymbol{I})$, and the visual inputs consisting of scene $\boldsymbol{V}_t$ and object $\boldsymbol{G}_t$ features.

4.2 Selective Object Attention

As discussed, simply expanding the set of inputs to the model by including object features does not improve performance. To overcome this challenge, we adjust the processing in the model to

[1] Object features were only used in [3] for the REVERIE task [29] in which ground-truth proposals and target object supervision are available.

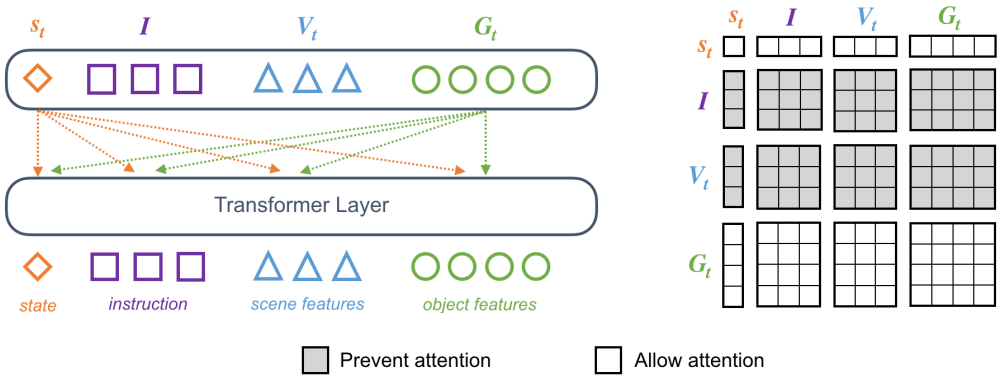

Figure 3: Selective Attention Mechanism: During navigation, the encoded navigation instruction and the scene features are kept frozen – i.e., they are not updated in the transformer layers and just serve as keys and values. The state token and object features are updated through self-attention by attending to *all* input tokens.

emphasize the object feature inputs. This is accomplished using the attention mask show in Figure 3 (right). Like VLN↻BERT, with this attention mask, the encoded navigation instructions $\psi(\boldsymbol{I})$ are not updated in the multimodal transformer. However, in our approach the scene features $\boldsymbol{V}_t$ are also kept frozen. As a result, these inputs only serve as keys and values during the attention based processing, while the state token $\boldsymbol{s}_t$ and object features $\boldsymbol{G}_t$ operate as queries, keys and values (which is standard in transformer-based models).

Intuitively, this attention mask allows scene features to provide contextual information to support the refinement of object representations. Note that the (unaltered) scene features are still used for action prediction (discussed in Section 4.3), which provides flexibility in how the model uses both types of visual representations. We hypothesize that this design leads to more effective transfer learning (demonstrated in Section 5), because the object-centric processing more closely matches the pretraining setup, in which the model only receives language and object feature inputs.

4.3 View Aggregation

At each timestep of navigation, the multimodal transformer uses selective object attention (described above) to produce contextualized representations of the state $\psi(\boldsymbol{s}_t)$ and object features $\psi(\boldsymbol{G}_t)$. As illustrated in Figure 2b, the object representations $\psi(\boldsymbol{G}_t)$ alongside the unaltered scene features $\boldsymbol{V}_t$ are used for view aggregation and action prediction. Specifically, state-conditioned attention scores are calculated for the scene and object features using a scaled dot-product (as in Equation (1)) as

$$\alpha(\boldsymbol{V_t}) = \frac{\psi(\boldsymbol{s}_t)\boldsymbol{V}_t^T}{\sqrt{d}}, \qquad \alpha(\boldsymbol{G_t}) = \frac{\psi(\boldsymbol{s}_t)\psi(\boldsymbol{G}_t)^T}{\sqrt{d}} \tag{2}$$

where d is the model's hidden dimension size. As a result, each candidate view is represented with one scene feature attention score and zero or more object feature attention scores.

For view aggregation, we select the maximum attention score for all of the scene and object features in a given view, and use the corresponding visual input to represent that view. 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 (i.e., candidate view). Finally, for action prediction, we take the softmax over the selected attention scores and use these normalized scores to represent the probability of moving to each navigable location or stopping.

5 Experiments

Datasets. We evaluate our method on the Room-to-Room (R2R) [1] and Room-Across-Room (RxR) [2] datasets. R2R is built using Matterport3D (MP3D) [18] indoor scenes and contains 21,567 path-instruction 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 uses environments from the training split but the path-instruction pairs are novel. Val-unseen and test-unseen use new environments and new path-instruction pairs to evaluate generalization performance. We augment the R2R training data with 1M instructions generated by the speaker model from [14].

RxR [2] is a recently introduced multi-lingual VLN dataset that also uses MP3D scenes. It contains 126K path-instruction pairs in 3 languages (Hindi, English and Telugu). English instructions are collected from two regions: India (`en-IN`) and US (`en-US`). Since our model is pretrained on English language data, we focus on the English language subset of RxR (both `en-IN` and `en-US`), which includes 26,464 path-instruction pairs for training and 4,551 pairs in the val-unseen split.

Evaluation. We follow the standard evaluation protocols for R2R and RxR as shown in Table 1. When applicable, we use an ↑ to indicate higher is better and a ↓ to indicate lower is better. On R2R, we report: Trajectory Length (TL), Navigation Error (NE ↓) - average distance between the target and the agent's final position, Success Rate (SR ↑) - percentage of trajectories in which agent stopped within 3 meters of the target, and Success weighted by normalized inverse of Path Length (SPL ↑). On RxR dataset, in addition to the NE and SR metrics, we also report the Normalized Dynamic Time Warping (NDTW ↑) and Success weighted by normalized Dynamic Time Warping (SDTW ↑) metrics, which explicitly measure path adherence. Additional details are included in the Appendix.

Baselines. We compare our approach (Row 9) with recent state-of-the-art methods. EnvDrop [10] (Row 4) trains an encoder-decoder model [5] on augmented data (in addition to R2R training data) generated with back-translation using "dropped out" scenes in order to generalize well to unseen environments. PREVALENT [14] (Row 5) builds on EnvDrop by pretraining a multimodal transformer model on augmented data. After pretraining, PREVALENT feeds contextualized word embeddings from this pretrained transformer into the EnvDrop [10] encoder-decoder to fine-tune on R2R dataset. VLN↻BERT (Row 6-8) extends PREVALENT by directly fine-tuning the pretrained multimodal transformer for navigation. VLN↻BERT uses two initializations – OSCAR (Row 6) and PREVALENT (Row 7). For fair comparison, we also report results reproduced with the released implementation[2] of VLN↻BERT by training from the OSCAR initialization (Row 8). Importantly, all these methods use *either* scene or object features as visual inputs for navigation.

Implementation Details. We implemented our model in PyTorch [31] and trained on a single Nvidia TitanX GPU. Consistent with VLN↻BERT [3], we initialize our model with a pretrained OSCAR model [24]. For both R2R and RxR, we fine-tune with behaviour cloning and reinforcement learning (RL) objectives adapted from prior work [3, 10, 14]. As in [3], 50% of each batch consists of behaviour cloning rollouts and 50% from RL (policy gradient). We train with a constant learning rate of 1e-5 using the AdamW optimizer with a batch size of 16 for 300k iterations. Like VLN↻BERT, we extract scene features with a ResNet-152 model [32] pretrained on the Places dataset [16]. For object features, we extract detections from a pretrained bottom-up attention model [30], and adopt the filtering procedure from [4] to discard redundant detections. In all of our experiments we use the same hyperparameters for the VLN↻BERT baseline and our approach.

5.1 Main Results

Performance on R2R and RxR. Rows 8 and 9 of Table 1 provide a comparison with the VLN↻BERT baseline on R2R (left) and RxR (right). RxR is a more challenging dataset because there is more variation in path lengths and agents often need to follow indirect paths to a goal. On RxR, our approach (row 9), which uses both scene and object features, beats the scene features only VLN↻BERT baseline (row 8) by **3.7%** on SR, **3.4%** on SDTW and **2.4%** on NDTW. On R2R, our approach (row 9) provides a **1.8%** gain in SPL over the VLN↻BERT baseline (row 8). The larger gains on RxR may result from a greater number of object references in RxR instructions (which is suggested by the linguistic analysis in [2]), which are required to describe the more complex paths. This hypothesis is consistent with our analysis in section 5.2 that shows larger gains from our method on instructions that contain 6 or more object references. These significant improvements highlight the benefit of using object features for challenging datasets like RxR.

Table 1 rows 3-7 show the results for state-of-the-art methods on R2R and RxR. On RxR, our method outperforms the previous state-of-the-art established by the RxR baseline (row 3) by **13.5%** absolute on NDTW and **18.6%** absolute on SR. On R2R, our approach is competitive with the previous state-

[2] `https://github.com/YicongHong/Recurrent-VLN-BERT`

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

	Methods	Pretraining		R2R				RxR			
		Web	VLN	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 [10]			10.70	5.22	52	48	-	-	-	-
5	PREVALENT [14]		✓	10.19	4.71	58	53	-	-	-	-
6	VLN↻BERT [3] (init. OSCAR)	✓		11.86	4.29	59	53	-	-	-	-
7	VLN↻BERT [3] (init. PREVALENT)		✓	12.01	**3.93**	**63**	**57**	-	-	-	-
8	VLN↻BERT [3] (init. OSCAR) †	✓		12.16	4.40	58	51	7.31	40.5	33.0	52.4
9	Ours	✓		12.15	**4.28**	**59**	**53**	**6.72**	**44.2**	**36.4**	**54.8**

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

	Methods	Pretraining		R2R Test Unseen			
		Web	VLN	TL	NE ↓	SR ↑	SPL ↑
1	Random			9.89	9.79	13	12
2	Human			11.85	1.61	86	76
3	EnvDrop [10]			11.66	5.23	51	47
4	PREVALENT [14]		✓	10.51	5.30	54	51
5	VLN↻BERT [3] (init. OSCAR)	✓		12.34	4.59	57	53
6	VLN↻BERT [3] (init. PREVALENT)		✓	12.35	**4.09**	**63**	**57**
7	VLN↻BERT [3] (init. OSCAR) †	✓		12.78	4.55	**58**	52
8	Ours	✓		12.26	4.49	**58**	**53**

of-the-art across all metrics. Table 2 reports results on the R2R test-unseen split, which again shows that our approach is competitive with prior work. In both tables VLN↻BERT with a PREVALENT initialization outperforms our approach. However, we note that PREVALENT [14] uses an alternative pretraining setup with scene features and augmented VLN data, which is orthogonal to our goals of using object features and transferring visual grounding for VLN. Combining these two different pretraining strategies is an interesting direction for future work.

Table 3: Ablation Study on R2R.

	Models	Object Features	View Aggregation	Selective Attention	Val Seen				Val Unseen			
					TL	NE ↓	SR ↑	SPL ↑	TL	NE ↓	SR ↑	SPL ↑
1	Baseline [3]				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**

5.2 Ablations and Analysis

Do object features help just as an additional input? Table 3 shows the importance of each design choice in our model. In Table 3 (row 2), we show the results of an ablation experiment on R2R in which object features are only used as additional input to VLN↻BERT. In this ablation, all of the visual inputs (object and scene features) get refined via self-attention and action probabilities and the next state token are calculated in a similar fashion to VLN↻BERT [3]. We see that merely feeding object tokens as an additional input without view aggregation or selective attention leads to slightly reduced performance on val-unseen (row 1 vs. 2). Row 3 demonstrates that using view aggregation (Section 4.3) reverses this trend. Furthermore, row 4 shows that our proposed selective attention mechanism (Section 4.2) leads to an additional improvements that cumulatively results in a **1.8%** gain in SPL over the VLN↻BERT baseline. To summarize, using object features as an additional input does not improve performance without the architectural changes proposed in this work.

Do object features help without vision-and-language pretraining? Table 4 reports results of the VLN↻BERT baseline and our approach without vision-and-language pretraining on the R2R val-unseen split. As evident from Table 4, both the approaches perform similarly across all metrics. This demonstrates that transferring visual grounding is crucial for our approach to work; our approach improves performance compared to the VLN↻BERT baseline (Table 1 Row 7 vs Row 8) predominantly because it more effectively utilizes the large-scale vision-and-language pretraining.

Table 4: Results on R2R without vision-and-language pretraining.

		R2R Val Unseen			
	Models	TL	NE ↓	SR ↑	SPL↑
1	VLN↻BERT [3]	10.31	5.21	49.21	45.53
2	Ours	12.00	4.99	50.36	44.69

How does our method perform on instructions that heavily mention objects? The RxR dataset contains long instructions that frequently mention objects and scenes. Specifically, RxR contains 6 entity references per instruction on average, which is nearly double the amount of entities referenced in R2R (3.7) [2]. Here, we evaluate our model on a subset of RxR instructions that heavily contain object references. To find this subset, we first randomly pick 500 instructions from val-unseen split in RxR. Then, we extract noun entities from the instructions using a standard NLP parser [33]. Next, we manually inspect these nouns and remove direction, scene, or other non-object references. This results in a list that only contains object references. Finally, we split the randomly selected 500 instructions into two sets containing 6 or more object object references (i.e., object heavy instructions) or less than 6 object references (i.e., not object heavy instructions). The threshold of 6 was selected to match the average number of entity references in RxR instructions [2]. With this process 354 of the 500 instructions were selected for object heavy set.

Results of our method and VLN↻BERT baseline on these subsets are reported in Table 5. On the object heavy subset, our approach gives *even stronger performance improvements* over VLN↻BERT baseline than seen on the full RxR val-unseen split. Specifically, we obtain an absolute improvement of **7.91%** for Success Rate (SR), **6.56%** for SDTW score and **5.51%** for NDTW. These gains are approximately twice the improvement on val-unseen split and larger that the ∼**3%** improvements on the complementary subset containing 146 instructions with less that 6 object references (Table 5 right). These results suggest that our approach is able to exploit object features and appropriately align them with object references to follow visually grounded navigation instructions.

Table 5: Results on a subset of RxR instructions divided by the number of object references.

		Object Heavy Instructions				Not Object Heavy Instructions			
	Models	NE ↓	SR ↑	SDTW ↑	NDTW ↑	NE ↓	SR ↑	SDTW ↑	NDTW ↑
1	VLN↻BERT [3]	8.46	33.89	26.36	46.95	4.07	58.21	50.33	67.75
2	Ours	**7.27**	**41.80**	**32.92**	**52.46**	**3.77**	**61.64**	**53.87**	**69.37**

6 Conclusion

In this work, we propose a novel scene- and object-aware transformer (SOAT) model, which uses scene and object features for vision-and-language navigation. We introduce a selective attention pattern in the transformer for processing these two distinct types of visual inputs such that scene tokens only provide contextual information for object-centric processing. As a result, our approach improves overall performance over a strong baseline on the R2R and RxR benchmarks. Our analysis shows that vision-and-language pretraining is crucial for exploiting the newly added object features. We obtain even larger performance improvement on instructions that heavily mention objects, which suggests that our approach is able to effectively ground object references in the instructions to the visual object features.

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 *EMNLP*, 2020.

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

[4] 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. In *ECCV*, 2020.

[5] 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*, 2018.

[6] 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 *CVPR*, 2019.

[7] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019.

[8] 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 *CVPR*, 2019.

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

[10] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL*, 2019.

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

[12] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A. Smith, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. In *EMNLP*, 2019.

[13] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR*, 2020.

[14] 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 *CVPR*, 2020.

[15] 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.

[16] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *PAMI*, 2017.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[18] 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 *3DV*, 2017.

[19] 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. In *ACL*, 2019.

[20] Yicong Hong, Cristian Rodriguez, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation. In *NeurIPS*, 2020.

[21] Yubo Zhang, Hao Tan, and Mohit Bansal. Diagnosing the environment bias in vision-and-language navigation. *IJCAI*, 2020.

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

[23] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, 2020.

[24] 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.

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

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

[27] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017.

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

[29] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *CVPR*, 2020.

[30] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018.

[31] 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 *NeurIPS*. 2019.

[32] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[33] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. *spaCy: Industrial-strength Natural Language Processing in Python*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No] We include this discussion in supplemental material.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We include this discussion in supplemental material.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
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)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We include results with multiple random seeds in supplemental material.

(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)? [Yes]

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? [Yes]

   (b) Did you mention the license of the assets? [Yes]

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We used standard datasets for vision-and-language navigation, but did not check these datasets for personally identifiable information or offensive content.

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]