Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout

Hao Tan **Licheng Yu** **Mohit Bansal**
UNC Chapel Hill
{haotan, licheng, mbansal}@cs.unc.edu

Abstract

A grand goal in AI is to build a robot that can accurately navigate based on natural language instructions, which requires the agent to perceive the scene, understand and ground language, and act in the real-world environment. One key challenge here is to learn to navigate in new environments that are unseen during training. Most of the existing approaches perform dramatically worse in unseen environments as compared to seen ones. In this paper, we present a generalizable navigational agent. Our agent is trained in two stages. The first stage is training via mixed imitation and reinforcement learning, combining the benefits from both off-policy and on-policy optimization. The second stage is fine-tuning via newly-introduced 'unseen' triplets (environment, path, instruction). To generate these unseen triplets, we propose a simple but effective 'environmental dropout' method to mimic unseen environments, which overcomes the problem of limited seen environment variability. Next, we apply semi-supervised learning (via back-translation) on these dropped-out environments to generate new paths and instructions. Empirically, we show that our agent is substantially better at generalizability when fine-tuned with these triplets, outperforming the state-of-art approaches by a large margin on the private unseen test set of the Room-to-Room task, and achieving the top rank on the leaderboard.[1]

1 Introduction

One of the important goals in AI is to develop a robot/agent that can understand instructions from humans and perform actions in complex environments. In order to do so, such a robot is required to perceive the surrounding scene, understand our spoken language, and act in a real-world house. Recent years have witnessed various types of embodied action based NLP tasks being proposed (Correa et al., 2010; Walters et al., 2007; Hayashi et al., 2007; Zhu et al., 2017b; Das et al., 2018; Anderson et al., 2018b).

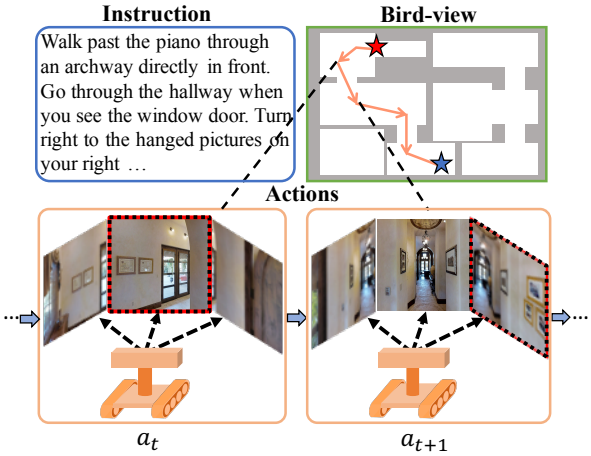


Figure 1: Room-to-Room Task. The agent is given an instruction, then starts its navigation from some staring viewpoint inside the given environment. At time t, the agent selects one view (highlighted as red dotted bounding boxes) from a set of its surrounding panoramic views to step into, as an action a_t.

In this paper, we address the task of instruction-guided navigation, where the agent seeks a route from a start viewpoint to an end viewpoint based on a given natural language instruction in a given environment, as shown in Fig. 1. The navigation simulator we use is the recent Room-to-Room (R2R) simulator (Anderson et al., 2018b), which uses real images from the Matterport3D (Chang et al., 2017) indoor home environments and collects complex navigable human-spoken instructions inside the environments, hence connecting problems in vision, language, and robotics. The instruction in Fig. 1 is *"Walk past the piano through an archway directly in front. Go through the hallway when you see the window door. Turn*

[1]Our code, data, and models publicly available at: https://github.com/airsplay/R2R-EnvDrop

Proceedings of NAACL-HLT 2019, pages 2610–2621
Minneapolis, Minnesota, June 2 - June 7, 2019.

right to the hanged pictures...”. At each position (viewpoint), the agent perceives panoramic views (a set of surrounding images) and selects one of them to step into. In this challenging task, the agent is required to understand each piece of the instruction and localize key views (*“piano”*, *“hallway”*, *“door”*, etc.) for making actions at each time step. Another crucial challenge is to generalize the agent’s navigation understanding capability to unseen test room environments, considering that the R2R task has substantially different unseen (test) rooms as compared to seen (trained) ones. Such generalization ability is important for developing a practical navigational robot that can operate in the wild.

Recent works (Fried et al., 2018; Wang et al., 2019, 2018a; Ma et al., 2019) have shown promising progress on this R2R task, based on speaker-follower, reinforcement learning, imitation learning, cross-modal, and look-ahead models. However, the primary issue in this task is that most models perform substantially worse in unseen environments than in seen ones, due to the lack of generalizability. Hence, in our paper, we focus on improving the agent’s generalizability in unseen environments. For this, we propose a two-stage training approach. The first stage is training the agent via mixed imitation learning (IL) and reinforcement learning (RL) which combines off-policy and on-policy optimization; this significantly outperforms using IL or RL alone.

The second, more important stage is semi-supervised learning with generalization-focused ‘environmental dropout’. Here, the model is fine-tuned using additional training data generated via back-translation. This is usually done based on a neural speaker model (Fried et al., 2018) that synthesizes new instructions for additional routes in the *existing* environments. However, we found that the bottleneck for this semi-supervised learning method is the limited variability of given (seen) environments. Therefore, to overcome this, we propose to generate novel and diverse environments via a simple but effective ‘environmental dropout’ method based on view- and viewpoint-consistent masking of the visual features. Next, the new navigational routes are collected from these new environments, and lastly the new instructions are generated by a neural speaker on these routes, and these triplets are employed to fine-tune the model training.

Overall, our fine-tuned model based on back-translation with environmental dropout substantially outperforms the previous state-of-the-art models, and achieves the most recent rank-1 on the Vision and Language Navigation (VLN) R2R challenge leaderboard’s private test data, outperforming all other entries in success rate under all evaluation setups (single run, beam search, and pre-exploration).[2] We also present detailed ablation and analysis studies to explain the effectiveness of our generalization method.

2 Related Work

Embodied Vision-and-Language Recent years are witnessing a resurgence of active vision. For example, Levine et al. (2016) used an end-to-end learned model to predict robotic actions from raw pixel data, Gupta et al. (2017) learned to navigate via mapping and planning, Sadeghi and Levine (2017) trained an agent to fly in simulation and show its performance in the real world, and Gandhi et al. (2017) trained a self-supervised agent to fly from examples of drones crashing. Meanwhile, in the intersection of active perception and language understanding, several tasks have been proposed, including instruction-based navigation (Chaplot et al., 2018; Anderson et al., 2018b), target-driven navigation (Zhu et al., 2017b; Gupta et al., 2017), embodied question answering (Das et al., 2018), interactive question answering (Gordon et al., 2018), and task planning (Zhu et al., 2017a). While these tasks are driven by different goals, they all require agents that can perceive their surroundings, understand the goal (either presented visually or in language instructions), and act in a virtual environment.

Instruction-based Navigation For instruction-based navigation task, an agent is required to navigate from start viewpoint to end viewpoint according to some given instruction in an environment. This task has been studied by many works (Tellex et al., 2011; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Andreas and Klein, 2015; Mei et al., 2016; Misra et al., 2017) in recent years. Among them, (Anderson et al., 2018b) differs from the others as it introduced a photo-realistic dataset – Room-to-Room (R2R), where all images are real ones taken by Matterport3D (Chang et al., 2017) and the instructions are also natural. In R2R

[2] https://evalai.cloudcv.org/web/challenges/challenge-page/97/overview

environments, the agent's ability to perceive real-world images and understanding natural language becomes even more crucial. To solve this challenging task, a lot of works (Fried et al., 2018; Wang et al., 2018a, 2019; Ma et al., 2019) have been proposed and shown some potential. The most relevant work to us is Fried et al. (2018), which proposed to use a speaker to synthesize new instructions and implement pragmatic reasoning. However, we observe there is some performance gap between seen and unseen environments. In this paper, we focus on improving the agent's generalizability in unseen environment.

Back-translation Back translation (Sennrich et al., 2016), a popular semi-supervised learning method, has been well studied in neural machine translation (Hoang et al., 2018; Wang et al., 2018b; Edunov et al., 2018; Prabhumoye et al., 2018). Given paired data of source and target sentences, the model first learns two translators – a forward translator from source to target and a backward translator from target to source. Next, it generates more source sentences using the back translator on an external target-language corpus. The generated pairs are then incorporated into the training data for fine-tuning the forward translator, which proves to improve the translation performance. Recently, this approach (also known as data augmentation) was applied to the task of instruction-based navigation (Fried et al., 2018), where the source and target sentences are replaced with instructions and routes.

3 Method

3.1 Problem Setup

Navigation in the Room-to-Room task (Anderson et al., 2018b) requires an agent to find a route $\mathbf{R}$ (a sequence of viewpoints) from the start viewpoint $\mathbf{S}$ to the target viewpoint $\mathbf{T}$ according to the given instruction $\mathbf{I}$. The agent is put in a photo-realistic environment $\mathbf{E}$. At each time step t, the agent's observation consists of a panoramic view and navigable viewpoints. The panoramic view o_t is discretized into 36 single views $\{o_{t,i}\}_{i=1}^{36}$. Each single view $o_{t,i}$ is an RGB image $v_{t,i}$ accompanied with its orientation $(\theta_{t,i}, \phi_{t,i})$, where $\theta_{t,i}$ and $\phi_{t,i}$ are the angles of heading and elevation, respectively. The navigable viewpoints $\{l_{t,k}\}_{k=1}^{N_t}$ are the N_t reachable and visible locations from the current viewpoint. Each navigable viewpoint $l_{t,k}$ is represented by the orientation $(\hat{\theta}_{t,k}, \hat{\phi}_{t,k})$ from current viewpoint to the next viewpoints. The agent needs to select the moving action a_t from the list of navigable viewpoints $\{l_{t,k}\}$ according to the given instruction $\mathbf{I}$, history/current panoramic views $\{o_\tau\}_{\tau=1}^{t}$, and history actions $\{a_\tau\}_{\tau=1}^{t-1}$. Following Fried et al. (2018), we concatenate the ResNet (He et al., 2016) feature of the RGB image and the orientation as the view feature $f_{t,i}$:

$$f_{t,i} = [\,\mathrm{ResNet}(v_{t,i}); \\ (\cos\theta_{t,i}, \sin\theta_{t,i}, \cos\phi_{t,i}, \sin\phi_{t,i})\,] \quad (1)$$

The navigable viewpoint feature $g_{t,k}$ is extracted in the same way.

3.2 Base Agent Model

For our base instruction-to-navigation translation agent, we implement an encoder-decoder model similar to Fried et al. (2018). The encoder is a bidirectional LSTM-RNN with an embedding layer:

$$\hat{w}_j = \mathrm{embedding}(w_j) \quad (2)$$

$$u_1, u_2, \cdots, u_{\mathrm{L}} = \text{Bi-LSTM}(\hat{w}_1, \cdots, \hat{w}_{\mathrm{L}}) \quad (3)$$

where u_j is the j-th word in the instruction with a length of L. The decoder of the agent is an attentive LSTM-RNN. At each decoding step t, the agent first attends to the view features $\{f_{t,i}\}$ computing the attentive visual feature $\tilde{f}_t$:

$$\alpha_{t,i} = \mathrm{softmax}_i(f_{t,i}^{\intercal} W_{\mathrm{F}}\, \tilde{h}_{t-1}) \quad (4)$$

$$\tilde{f}_t = \sum\nolimits_i \alpha_{t,i} f_{t,i} \quad (5)$$

The input of the decoder is the concatenation of the attentive visual feature $\tilde{f}_t$ and the embedding of the previous action $\tilde{a}_{t-1}$. The hidden output h_t of the LSTM is combined with the attentive instruction feature $\tilde{u}_t$ to form the instruction-aware hidden output $\tilde{h}_t$. The probability of moving to the k-th navigable viewpoint $p_t(a_{t,k})$ is calculated as softmax of the alignment between the navigable viewpoint feature $g_{t,k}$ and the instruction-aware hidden output $\tilde{h}_t$.

$$h_t = \mathrm{LSTM}\left([\tilde{f}_t; \tilde{a}_{t-1}], \tilde{h}_{t-1}\right) \quad (6)$$

$$\beta_{t,j} = \mathrm{softmax}_j\left(u_j^{\intercal} W_{\mathrm{U}}\, h_t\right) \quad (7)$$

$$\tilde{u}_t = \sum\nolimits_j \beta_{t,j}\, u_j \quad (8)$$

$$\tilde{h}_t = \tanh\left(W\,[\tilde{u}_t;\, h_t]\right) \quad (9)$$

$$p_t(a_{t,k}) = \mathrm{softmax}_k\left(g_{t,k}^{\intercal} W_{\mathrm{G}}\, \tilde{h}_t\right) \quad (10)$$

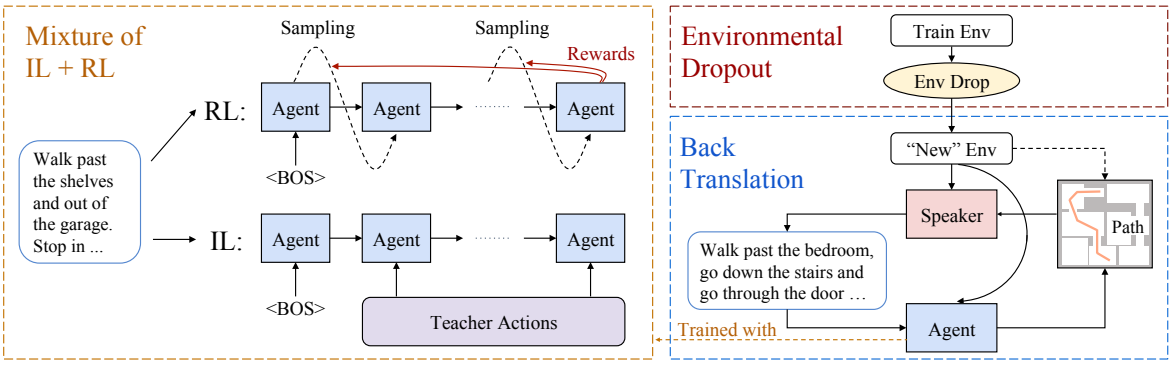


Figure 2: Left: IL+RL supervised learning (stage 1). Right: Semi-supervised learning with back translation and environmental dropout (stage 2).

Different from Fried et al. (2018), we take the instruction-aware hidden vector $\tilde{h}_{t-1}$ as the hidden input of the decoder instead of h_{t-1}. Thus, the information about which parts of the instruction have been attended to is accessible to the agent.

3.3 Supervised Learning: Mixture of Imitation+Reinforcement Learning

We discuss our IL+RL supervised learning method in this section.[3]

Imitation Learning (IL) In IL, an agent learns to imitate the behavior of a teacher. The teacher demonstrates a teacher action a_t^* at each time step t. In the task of navigation, a teacher action a_t^* selects the next navigable viewpoint which is on the shortest route from the current viewpoint to the target $\mathbf{T}$. The off-policy[4] agent learns from this weak supervision by minimizing the negative log probability of the teacher's action a_t^*. The loss of IL is as follows:

$$\mathcal{L}^{\text{IL}} = \sum_t \mathcal{L}_t^{\text{IL}} = \sum_t -\log p_t(a_t^*) \qquad (11)$$

For exploration, we follow the IL method of Behavioral Cloning (Bojarski et al., 2016), where the agent moves to the viewpoint following the teacher's action a_t^* at time step t.

Reinforcement Learning (RL) Although the route induced by the teacher's actions in IL is the shortest, this selected route is not guaranteed to satisfy the instruction. Thus, the agent using IL is biased towards the teacher's actions instead of finding the correct route indicated by the instruction. To overcome these misleading actions, the on-policy reinforcement learning method Advantage Actor-Critic (Mnih et al., 2016) is applied, where the agent takes a sampled action from the distribution $\{p_t(a_{t,k})\}$ and learns from rewards. If the agent stops within $3m$ around the target viewpoint $\mathbf{T}$, a positive reward $+3$ is assigned at the final step. Otherwise, a negative reward -3 is assigned. We also apply reward shaping (Wu et al., 2018): the direct reward at each non-stop step t is the change of the distance to the target viewpoint.

IL+RL Mixture To take the advantage of both off-policy and on-policy learners, we use a method to mix IL and RL. The IL and RL agents share weights, take actions separately, and navigate two independent routes (see Fig. 2). The mixed loss is the weighted sum of $\mathcal{L}^{\text{IL}}$ and $\mathcal{L}^{\text{RL}}$:

$$\mathcal{L}^{\text{MIX}} = \mathcal{L}^{\text{RL}} + \lambda_{\text{IL}} \mathcal{L}^{\text{IL}} \qquad (12)$$

IL can be viewed as a language model on action sequences, which regularizes the RL training.[5]

3.4 Semi-Supervised Learning: Back Translation with Environmental Dropout

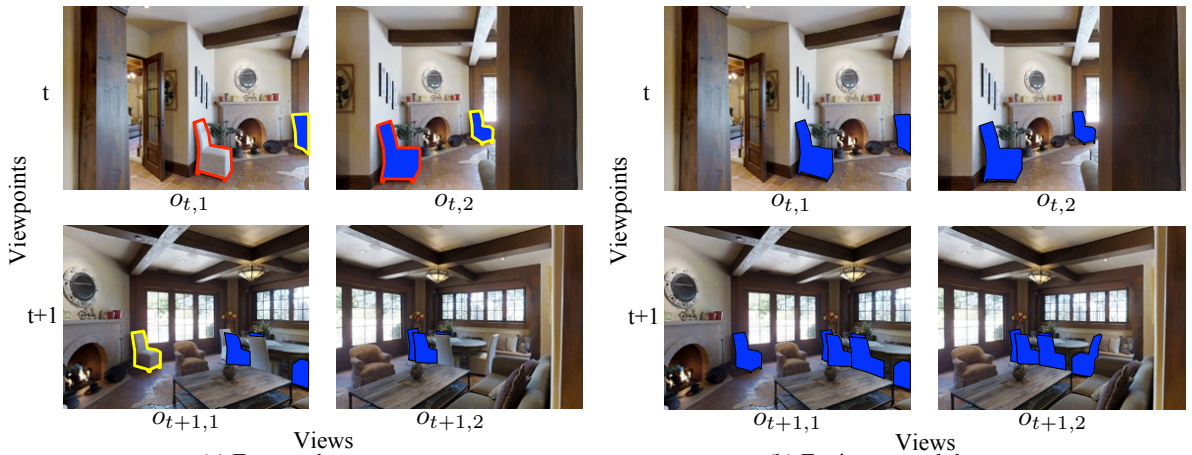
3.4.1 Back Translation

Suppose the primary task is to learn the mapping of $\mathbf{X} \rightarrow \mathbf{Y}$ with paired data $\{(\mathbf{X}, \mathbf{Y})\}$ and unpaired data $\{\mathbf{Y}'\}$. In this case, the back translation method first trains a forward model $P_{\mathbf{X}\rightarrow\mathbf{Y}}$ and a backward model $P_{\mathbf{Y}\rightarrow\mathbf{X}}$, using paired data $\{(\mathbf{X}, \mathbf{Y})\}$. Next, it generates additional datum $\mathbf{X}'$

[3] As opposed to semi-supervised methods in Sec. 3.4, in this section we view both imitation learning and reinforcement learning as supervised learning.

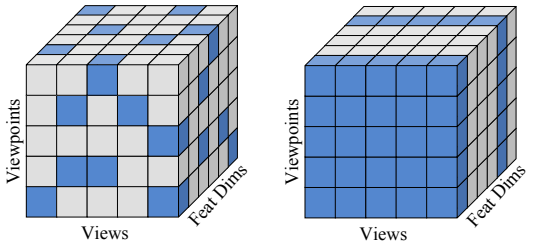
[4] According to Poole and Mackworth (2010), an off-policy learner learns the agent policy independently of the agent's navigational actions. An on-policy learner learns the policy from the agent's behavior including the exploration steps.

[5] This approach is similar to the method ML+RL in Paulus et al. (2018) for summarization. Recently, Wang et al. (2018a) combines purely supervised learning and RL training however, they use a different algorithm named MIXER (Ranzato et al., 2015), which computes cross entropy (XE) losses for the first k actions and RL losses for the remaining.



(a) Feature dropout (b) Environmental dropout

Figure 3: Comparison of the two dropout methods (based on an illustration on an RGB image).



(a) Feature dropout (b) Environmental dropout

Figure 4: Comparison of the two dropout methods (based on image features).

from the unpaired $\mathbf{Y}'$ using the backward model $P_{\mathbf{Y}\to\mathbf{X}}$. Finally, $(\mathbf{X}', \mathbf{Y}')$ are paired to further fine-tune the forward model $P_{\mathbf{X}\to\mathbf{Y}}$ as additional training data (also known as 'data augmentation').

Back translation was introduced to the task of navigation in Fried et al. (2018). The forward model is a navigational agent $P_{\mathbf{E},\mathbf{I}\to\mathbf{R}}$ (Sec. 3.2), which navigates inside an environment $\mathbf{E}$, trying to find the correct route $\mathbf{R}$ according to the given instruction $\mathbf{I}$. The backward model is a *speaker* $P_{\mathbf{E},\mathbf{R}\to\mathbf{I}}$, which generates an instruction $\mathbf{I}$ from a given route $\mathbf{R}$ inside an environment $\mathbf{E}$. Our speaker model (details in Sec. 3.4.3) is an enhanced version of Fried et al. (2018), where we use a stacked bidirectional LSTM-RNN encoder with attention flow.

For back translation, the Room-to-Room dataset labels around 7% routes $\{\mathbf{R}\}$ in the training environments[6], so the rest of the routes $\{\mathbf{R}'\}$ are unlabeled. Hence, we generate additional instructions $\mathbf{I}'$ using $P_{\mathbf{E},\mathbf{R}\to\mathbf{I}}(\mathbf{E}, \mathbf{R}')$, so to obtain the new triplets $(\mathbf{E}, \mathbf{R}', \mathbf{I}')$. The agent is then fine-tuned with this new data using the IL+RL method described in Sec. 3.3. However, note that the environment $\mathbf{E}$ in the new triplet $(\mathbf{E}, \mathbf{R}', \mathbf{I}')$ for semi-supervised learning is still selected from the *seen* training environments. We demonstrate that the limited amount of environments $\{\mathbf{E}\}$ is actually the bottleneck of the agent performance in Sec. 7.1 and Sec. 7.2. Thus, we introduce our environmental dropout method to mimic the "new" environment $\mathbf{E}'$, as described next in Sec. 3.4.2.

[6]The number of all possible routes (shortest paths) in the 60 existing training environments is 190K. Of these, the Room-to-Room dataset labeled around 14K routes with one navigable instruction for each, so the amount of labeled routes is around 7% of 190K.

3.4.2 Environmental Dropout

Failure of Feature Dropout Different from dropout on neurons to regularize neural networks, we drop raw feature dimensions (see Fig. 4a) to mimic the removal of random objects from an RGB image (see Fig. 3a). This traditional feature dropout (with dropout rate p) is implemented as an element-wise multiplication of the feature f and the dropout mask ξ^f. Each element ξ^f_e in the dropout mask ξ^f is a sample of a random variable which obeys an independent and identical Bernoulli distribution multiplied by $1/(1-p)$. And for different features, the distributions of dropout masks are independent as well.

$$\text{dropout}_p(f) = f \odot \xi^f \tag{13}$$

$$\xi^f_e \sim \frac{1}{1-p}\text{Ber}(1-p) \tag{14}$$

Because of this independence among dropout masks, the traditional feature dropout fails in augmenting the existing environments because the 'removal' is inconsistent in different views at the same viewpoint, and in different viewpoints.

To illustrate this idea, we take the four RGB views in Fig. 3a as an example, where the chairs are randomly dropped from the views. The removal of the left chair (marked with a red polygon) from view $o_{t,2}$ is inconsistent because it also appears in view $o_{t,1}$. Thus, the speaker could still refer to it and the agent is aware of the existence of the chair. Moreover, another chair (marked with a yellow polygon) is completely removed from viewpoint observation o_t, but the views in next viewpoint o_{t+1} provides conflicting information which would confuse the speaker and the agent. Therefore, in order to make generated environments consistent, we propose our environmental dropout method, described next.

Environmental Dropout We create a new environment $\mathbf{E}'$ by applying environmental dropout on an existing environment $\mathbf{E}$.

$$\mathbf{E}' = \text{envdrop}_p(\mathbf{E}) \tag{15}$$

The view feature $f'_{t,i}$ observed from the new environment $\mathbf{E}'$ is calculated as an element-wise multiplication of the original feature $f_{t,i}$ and the environmental dropout mask $\xi^{\mathbf{E}}$ (see Fig. 4b):

$$f'_{t,i} = f_{t,i} \odot \xi^{\mathbf{E}} \tag{16}$$

$$\xi_e^{\mathbf{E}} \sim \frac{1}{1-p}\text{Ber}(1-p) \tag{17}$$

To maintain the spatial structure of viewpoints, only the image feature $\text{ResNet}(v_{t,i})$ is dropped while the orientation feature $(\cos(\theta_{t,i}), \sin(\theta_{t,i}), \cos(\phi_{t,i}), \sin(\phi_{t,i}))$ is fixed. As illustrated in Fig. 3b, the idea behind environmental dropout is to mimic new environments by removing one specific class of object (e.g., the chair). We demonstrate our idea by running environmental dropout on the ground-truth semantic views in Sec. 7.3, which is proved to be far more effective than traditional feature dropout. In practice, we perform the environmental dropout on image's visual feature where certain structures/parts are dropped instead of object instances, but the effect is similar.

We apply the environmental dropout to the back translation model as mentioned in Sec. 3.4.1. Note the environmental dropout method still preserves the connectivity of the viewpoints, thus we use the same way (Fried et al., 2018) to collect extra unlabeled routes $\{\mathbf{R}'\}$. We take *speaker* to generate an additional instruction $\mathbf{I}'$=$P_{\mathbf{E},\mathbf{R}\rightarrow\mathbf{I}}(\mathbf{E}', \mathbf{R}')$ in the new environment $\mathbf{E}'$. At last, we use IL+RL (in Sec. 3.3) to fine-tune the model with this new triplet $(\mathbf{E}', \mathbf{R}', \mathbf{I}')$.

3.4.3 Improvements on Speaker

Our speaker model is an enhanced version of the encoder-decoder model of Fried et al. (2018), with improvements on the visual encoder: we stack two bi-directional LSTM encoders: a route encoder and a context encoder. The route encoder takes features of ground truth actions $\{a_t^*\}_{t=1}^T$ from the route as inputs. Each hidden state r_t then attends to surrounding views $\{f_{t,i}\}_{i=1}^{36}$ at each viewpoint. The context encoder then reads the attended features and outputs final visual encoder representations:

$$r_1, ..., r_{\text{T}} = \text{Bi-LSTM}^{\text{RTE}}(g_{1,a_1^*}, ..., g_{\text{T},a_{\text{T}}^*}) \tag{18}$$

$$\gamma_{t,i} = \text{softmax}_i(f_{t,i}^{\intercal} W_{\text{R}}\, r_t) \tag{19}$$

$$\hat{f}_t = \sum\nolimits_i \gamma_{t,i} f_{t,i} \tag{20}$$

$$c_1, ..., c_{\text{T}} = \text{Bi-LSTM}^{\text{CTX}}(\hat{f}_1, ..., \hat{f}_{\text{T}}) \tag{21}$$

The decoder is a regular attentive LSTM-RNN, which is discussed in Sec. 3.2. Empirically, our enhanced speaker model improves the BLEU-4 score by around 3 points.

4 Experimental Setup

Dataset and Simulator We evaluate our agent on the Matterport3D simulator (Anderson et al., 2018b). Navigation instructions in the dataset are collected via Amazon Mechanical Turk by showing them the routes in the Matterport3D environment (Chang et al., 2017). The dataset is split into training set (61 environments, 14,025 instructions), seen validation set (61 environments, 1,020 instructions), unseen validation set (11 environments, 2,349 instructions), and unseen test set (18 environments, 4,173 instructions). The unseen sets only involve the environments outside the training set.

Evaluation Metrics For evaluating our model, Success Rate (SR) is the primary metric. The execution route by the agent is considered a success when the navigation error is less than 3 meters. Besides success rate, we use three other metrics[7] : Navigation Length (NL), Navigation Error (NE), and Success rate weighted by Path Length (SPL) (Anderson et al., 2018a). Navigation Error (NE) is

[7] The Oracle Success Rate (OSR) is not included because it's highly correlated with the Navigation Length.

Models	Test Unseen (Leader-Board)								
	Single Run			Beam Search			Pre-Explore		
	NL	SR(%)	*SPL*	NL	*SR(%)*	SPL	NL	SR(%)	*SPL*
Random (Anderson et al., 2018b)	9.89	13.2	0.12	-	-	-	-	-	-
Seq-to-Seq (Anderson et al., 2018b)	8.13	20.4	0.18	-	-	-	-	-	-
Look Before You Leap (Wang et al., 2018a)	9.15	25.3	0.23	-	-	-	-	-	-
Speaker-Follower (Fried et al., 2018)	14.8	35.0	0.28	1257	53.5	0.01	-	-	-
Self-Monitoring (Ma et al., 2019)	18.0	48.0	0.35	373	61.0	**0.02**	-	-	-
Reinforced Cross-Modal (Wang et al., 2019)	12.0	43.1	0.38	358	63.0	**0.02**	9.48	60.5	0.59
Ours	11.7	**51.5**	**0.47**	687	**68.9**	0.01	9.79	**63.9**	**0.61**

Table 1: Leaderboard results under different experimental setups. NL, SR, and SPL are Navigation Length, Success Rate and Success rate weighted by Path Length. The primary metric for each setup is in italics. The best results are in bold font and the second best results are underlined.

the distance between target viewpoint $\mathbf{T}$ and agent stopping position.

Implementation Details Similar to the traditional dropout method, the environmental dropout mask is computed and applied at each training iteration. Thus, the amount of unlabeled semi-supervised data used is not higher in our dropout method. We also find that sharing the environmental dropout mask in different environments inside a batch will stabilize the training. To avoid overfitting, the model is early-stopped according to the success rate on the unseen validation set. More training details in appendices.

5 Results

In this section, we compare our agent model with the models in previous works on the Vision and Language Navigation (VLN) leaderboard. The models on the leaderboard are evaluated on a private unseen test set which contains 18 new environments. We created three columns in Table 1 for different experimental setups: single run, beam search, and unseen environments pre-exploration. For the result, our model outperforms all other models in all experimental setups.

Single Run Among all three experimental setups, single run is the most general and highly correlated to the agent performance. Thus, it is considered as the primary experimental setup. In this setup, the agent navigates the environment once and is not allowed[8] to: (1) run multiple trials, (2) explore nor map the test environments before starting. Our result is 3.5% and 9% higher than the second-best in Success Rate and SPL, resp.

Beam Search In the beam search experimental setup, an agent navigates the environment, collects multiple routes, re-ranks them, and selects the route with the highest score as the prediction. Besides showing an upper bound, beam search is usable when the environment is explored and saved in the agent's memory but the agent does not have enough computational capacity to fine-tune its navigational model. We use the same beam-search algorithm, state factored Dijkstra algorithm, to navigate the unseen test environment. Success Rate of our model is 5.9% higher than the second best. SPL metric generally fails in evaluating beam-search models because of the long Navigation Length (range of SPL is 0.01-0.02).

Pre-Exploration The agent pre-explores the test environment before navigating and updates its agent model with the extra information. When executing the instruction in the environment, the experimental setup is still "single run". The "pre-exploration" agent mimics the domestic robots (e.g., robot vacuum) which only needs to navigate the seen environment most of the time. For submitting to the leaderboard, we simply train our agent via back translation with environmental dropout on test unseen environments (see Sec.7.2). Our result is 3.4% higher than Wang et al. (2019) in Success Rate and 2.0% higher in SPL. [9]

6 Ablation Studies

Supervised Learning We first show the effectiveness of our IL+RL method by comparing it with the baselines (Table 2). We implement Behavioural Cloning[10] and Advantage Actor-Critic

[8] According to the Vision and Language Navigation (VLN) challenge submission guidelines

[9] To fairly compare with Wang et al. (2019), we exclude the exploration route in calculating Navigation Length.

[10] The Behavioral Cloning (IL) baseline is the same as the panoramic view baseline in Fried et al. (2018) except for two differences: (1) The agent takes the teacher action instead of the sampled action from the distribution (see "imitation learning" of Sec. 3.3), (2) The hidden input of the LSTM is the instruction-aware hidden from the previous step (see Sec. 3.2). We improve our baseline result with these modifi-

Models	Val Seen				Val Unseen			
	NL(m)	NE(m)	SR(%)	SPL	NL(m)	NE(m)	SR(%)	SPL
SUPERVISED LEARNING								
Behavioral Cloning (IL)	10.3	5.39	48.4	0.46	9.15	6.25	43.6	0.40
Advantage Actor-Critic (RL)	73.8	7.11	22.0	0.03	73.8	7.32	24.0	0.03
IL + RL	10.1	4.71	55.3	0.53	9.37	5.49	46.5	0.43
SEMI-SUPERVISED LEARNING								
Back Translation	10.3	4.19	58.1	0.55	10.5	5.43	48.2	0.44
+ Feat Drop	10.3	4.13	58.4	0.56	9.62	5.43	48.4	0.45
+ Env Drop (No Tying)	10.3	4.32	57.3	0.55	9.51	5.27	49.0	0.46
+ Env Drop (Tying)	11.0	3.99	62.1	0.59	10.7	5.22	52.2	0.48
FULL MODEL								
Single Run	11.0	3.99	62.1	0.59	10.7	5.22	52.2	0.48
Beam Search	703	2.52	75.7	0.01	663	3.08	69.0	0.01
Pre-Explore	9.92	4.84	54.7	0.52	9.57	3.78	64.5	0.61

Table 2: For the ablation study, we show the results of our different methods on validation sets. Our full model (single run) gets 8.6% improvement in validation unseen success rate above our baseline. And both the supervised learning (IL+RL) and semi-supervised learning methods (back translation + env drop) have substantial contributions to our final result.

as our imitation learning (IL) and reinforcement learning (RL) baselines, respectively. The mixture of IL+RL (see Sec. 3.3) outperforms the IL-only model and RL-only model by 2.9% and 22.5%, which means that our IL+RL could overcome the misleading teacher actions in IL and significantly stabilize the training of RL.

Semi-Supervised Learning We then fine-tune our best supervised model (i.e., IL+RL) with back translation. Besides providing a warm-up, IL+RL is also used to learn the new generated data triplets in back translation. As shown in Table 2, back translation with environmental dropout improves the best supervised model by 5.7%, where the improvement is 3 times more than the back translation without new environments. We then show the results of the alternatives to environmental dropout. The performance with feature dropout is almost the same to the original back translation, which is 3.8% lower than the environmental dropout. We also prove that the improvement from the environmental dropout method does not only come from generating *diverse* instructions introduced by dropout in the speaker, but also comes from using the same dropout mask in the follower agent too. To show this, we use two independent (different) environmental dropout masks for the speaker and the follower (i.e., no tying of the dropout mask), and the result drops a lot as compared to when we tie the speaker and follower dropout masks.

cations.

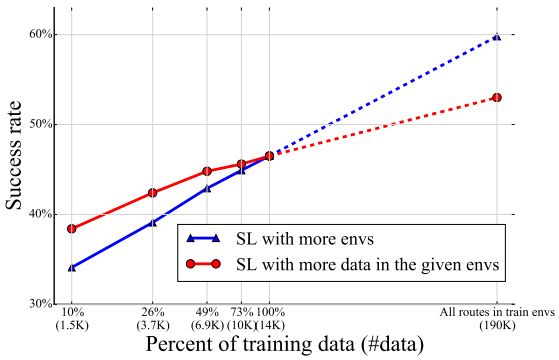
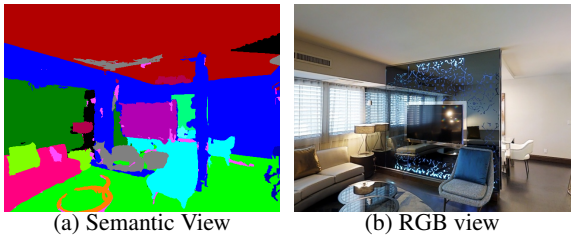


Figure 5: Success rates of agents trained with different amounts of data. X-axis in log-scale. The blue line represents the growth of results by gradually adding new environments to the supervised training method. The red line is trained with the same amounts of data as the blue line, but the data is randomly selected from all 60 training environments. The dashed lines are predicted.

Full Model Finally, we show the performance of our best agent under different experimental setups. The "single run" result is copied from the best semi-supervised model for comparison. The state-factored Dijkstra algorithm (Fried et al., 2018) is used for the beam search result. The method for pre-exploration is described in Sec. 7.2, where the agent applies back translation with environmental dropout on the validation unseen environment.

7 Analysis

In this section, we present analysis experiments that first exposed the limited environments bottleneck to us, and hence inspired us to develop our environmental dropout method to break this bottleneck.



(a) Semantic View (b) RGB view

Figure 6: Comparison of semantic and raw RGB views.

7.1 More Environments vs. More Data

In order to show that more environments are crucial for better performance of agents, in Fig. 5, we present the result of Supervised Learning (SL) with different amounts of data selected by two different data-selection methods. The first method gradually uses more #environments (see the blue line "SL with more envs") while the second method selects data from the whole training data with all 60 training environments (see the red line "SL with more data"). Note that the amounts of data in the two setups are the same for each plot point As shown in Fig. 5, the "more envs" selection method shows *higher growth rate* in success rate than the "more data" method. We also predict the success rates (in dashed line) with the prediction method in Sun et al. (2017). The predicted result is much higher when training with more environments. The predicted result (the right end of the red line) also shows that the upper bound of Success Rate is around 52% if all the 190K routes in the training environments is labeled by human (instead of being generated by *speaker* via back translation), which indicates the need for "new" environments.

7.2 Back Translation on Unseen Environments

In this subsection, we show that back translation could significantly improve the performance when it uses new data triplets from testing environments — the unseen validation environments where the agent is evaluated in. Back translation (w.o. Env Drop) on these unseen environments achieves a success rate of 61.9%, while the back translation on the training environments only achieves 46.5%. The large margin between the two results indicates the need of "new" environments in back translation. Moreover, our environmental dropout on testing environments could further improve the result to 64.5%, which means that the amount of environments in back translation is far from enough.

7.3 Semantic Views

To demonstrate our intuition of the success of environmental dropout (in Sec. 3.4.2), we replace the image feature $\mathrm{ResNet}(v_{t,i})$ with the semantic view feature. The semantic views (as shown in Fig. 6) are rendered from the Matterport3D dataset (Chang et al., 2017), where different colors indicate different types of objects. Thus, dropout on the semantic view feature would remove the object from the view. With the help of this additional information (i.e., the semantic view), the success rate of IL+RL is 49.5% on the unseen validation set. Back translation (without dropout) slightly improves the result to 50.5%. The result with feature dropout is 50.2% while the environmental dropout could boost the result to 52.0%, which supports our claim in Sec. 3.4.2.

8 Conclusion

We presented a navigational agent which better generalizes to unseen environments. The agent is supervised with a mixture of imitation learning and reinforcement learning. Next, it is fine-tuned with semi-supervised learning, with speaker-generated instructions. Here, we showed that the limited variety of environments is the bottleneck of back translation and we overcome it via 'environmental dropout' to generate new unseen environments. We evaluate our model on the Room-to-Room dataset and achieve rank-1 in the Vision and Language Navigation (VLN) challenge leaderboard under all experimental setups.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by ARO-YIP Award #W911NF-18-1-0336, ONR Grant #N00014-18-1-2871, and faculty awards from Google, Facebook, Adobe, Baidu, and Salesforce. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the funding agency.

References

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh

Mottaghi, Manolis Savva, et al. 2018a. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018b. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1165–1174.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

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

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2018. Gated-attention architectures for task-oriented language grounding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI*, volume 2, pages 1–2.

Andrew Correa, Matthew R Walter, Luke Fletcher, Jim Glass, Seth Teller, and Randall Davis. 2010. Multimodal interaction with an autonomous forklift. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 243–250. IEEE Press.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *CVPR)*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3318–3329.

Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. 2017. Learning to fly by crashing. *IROS*.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *CVPR*.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. 2017. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625.

Kotaro Hayashi, Daisuke Sakamoto, Takayuki Kanda, Masahiro Shiomi, Satoshi Koizumi, Hiroshi Ishiguro, Tsukasa Ogasawara, and Norihiro Hagita. 2007. Humanoid robots as a passive-social medium-a field experiment at a train station. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, pages 137–144. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *JMLR*.

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

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, volume 1, page 2.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

David L Poole and Alan K Mackworth. 2010. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proc. ACL*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. In *ICLR*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Fereshteh Sadeghi and Sergey Levine. 2017. CAD2RL: Real single-image flight without a single real image. *RSS*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 86–96.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 843–852. IEEE.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, volume 1, page 2.

Michael L Walters, Kerstin Dautenhahn, Sarah N Woods, and Kheng Lee Koay. 2007. Robotic etiquette: results from user studies involving a fetch and carry task. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 317–324. ACM.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. 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*.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018a. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, pages 38–55.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018b. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. 2018. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*.

Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. 2017a. Visual semantic planning using deep successor representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 483–492.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017b. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3357–3364. IEEE.

A Appendices

A.1 Implementation Details

We use ResNet-152 (He et al., 2016) pretrained on the ImageNet (Russakovsky et al., 2015) to extract the 2048-dimensional image feature. The agent model is first trained with supervised learning via the mixture of imitation and reinforcement learning. The model is then fine-tuned by back translation with environmental dropout. To stabilize the optimization of back translation, we calculate supervised loss for half of the batch and semi-supervised loss for the other half. We find that sharing the environmental dropout mask in different environments inside the same batch will stabilize the training.

The word embedding is trained from scratch with size 256 and the dimension of the action embedding is 64. The size of the LSTM units is set to 512 (256 for the bidirectional LSTM). In RL training, the discounted factor γ is 0.9. We use the reward shaping (Wu et al., 2018): the direct

reward r_t at time step t is the change of the distance $d_{t-1} - d_t$, supposing d_t is the distance to the target position at time step t. The maximum decoding action length is set to 35. For optimizing the loss, we use RMSprop (Hinton et al., 2012) (as suggested in Mnih et al. (2016)) with a fixed learning rate $1e-4$ and the batch size is 64. We applying dropout rate 0.4 to the environmental dropout and 0.5 to the dropout layers which regularize the network. The global gradient norm is clipped by 40. We tuned the hyper-parameters based on the Success Rate of the unseen validation set.

When working with the semantic view, the *key* labels (e.g., *wall*, *floor*, *ceil*) are not dropped, because they are the basic structure of the environment. Empirically, no improvement will be achieved when the *key* labels are dropped as well.