Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction

Dipendra Misra **Andrew Bennett** **Valts Blukis**
Eyvind Niklasson **Max Shatkhin** **Yoav Artzi**

Department of Computer Science and Cornell Tech, Cornell University, New York, NY, 10044
{dkm, awbennett, valts, yoav}@cs.cornell.edu
{een7, ms3448}@cornell.edu

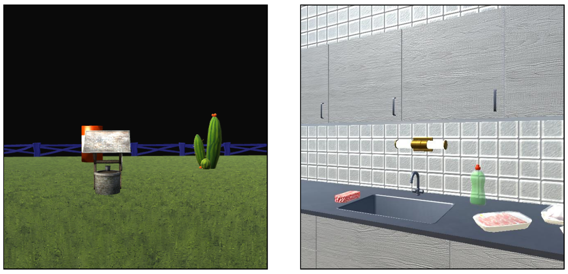
Abstract

We propose to decompose instruction execution to goal prediction and action generation. We design a model that maps raw visual observations to goals using LingUNet, a language-conditioned image generation network, and then generates the actions required to complete them. Our model is trained from demonstration only without external resources. To evaluate our approach, we introduce two benchmarks for instruction following: Lani, a navigation task; and Chai, where an agent executes household instructions. Our evaluation demonstrates the advantages of our model decomposition, and illustrates the challenges posed by our new benchmarks.

1 Introduction

Executing instructions in interactive environments requires mapping natural language and observations to actions. Recent approaches propose learning to directly map from inputs to actions, for example given language and either structured observations (Mei et al., 2016; Suhr and Artzi, 2018) or raw visual observations (Misra et al., 2017; Xiong et al., 2018). Rather than using a combination of models, these approaches learn a single model to solve language, perception, and planning challenges. This reduces the amount of engineering required and eliminates the need for hand-crafted meaning representations. At each step, the agent maps its current inputs to the next action using a single learned function that is executed repeatedly until task completion.

Although executing the same computation at each step simplifies modeling, it exemplifies certain inefficiencies; while the agent needs to decide what action to take at each step, identifying its goal is only required once every several steps or even once per execution. The left instruction in Figure 1 illustrates this. The agent can compute its goal once given the initial observation, and given this goal can then generate the actions required. In this paper, we study a new model that explicitly distinguishes between goal selection and action generation, and introduce two instruction following benchmark tasks to evaluate it.



After reaching the hydrant head towards the blue fence and pass towards the right side of the well.

Put the cereal, the sponge, and the dishwashing soap into the cupboard above the sink.

Figure 1: Example instructions from our two tasks: Lani (left) and Chai (right). Lani is a landmark navigation task, and Chai is a corpus of instructions in the Chalet environment.

Our model decomposes into goal prediction and action generation. Given a natural language instruction and system observations, the model predicts the goal to complete. Given the goal, the model generates a sequence of actions.

The key challenge we address is designing the goal representation. We avoid manually designing a meaning representation, and predict the goal in the agent's observation space. Given the image of the environment the agent observes, we generate a probability distribution over the image to highlight the goal location. We treat this prediction as image generation, and develop LingUNet, a language conditioned variant of the U-Net image-to-image architecture (Ronneberger et al., 2015). Given the visual goal prediction, we generate actions using a recurrent neural network (RNN).

Our model decomposition offers two key advantages. First, we can use different learning methods as appropriate for the goal prediction and action

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2667–2678
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

generation problems. We find supervised learning more effective for goal prediction, where only a limited amount of natural language data is available. For action generation, where exploration is critical, we use policy gradient in a contextual bandit setting (Misra et al., 2017). Second, the goal distribution is easily interpretable by overlaying it on the agent observations. This can be used to increase the safety of physical systems by letting the user verify the goal before any action is executed. Despite the decomposition, our approach retains the advantages of the single-model approach. It does not require designing intermediate representations, and training does not rely on external resources, such as pre-trained parsers or object detectors, instead using demonstrations only.

We introduce two new benchmark tasks with different levels of complexity of goal prediction and action generation. LANI is a 3D navigation environment and corpus, where an agent navigates between landmarks. The corpus includes 6,000 sequences of natural language instructions, each containing on average 4.7 instructions. CHAI is a corpus of 1,596 instruction sequences, each including 7.7 instructions on average, for CHALET, a 3D house environment (Yan et al., 2018). Instructions combine navigation and simple manipulation, including moving objects and opening containers. Both tasks require solving language challenges, including spatial and temporal reasoning, as well as complex perception and planning problems. While LANI provides a task where most instructions include a single goal, the CHAI instructions often require multiple intermediate goals. For example, the household instruction in Figure 1 can be decomposed to eight goals: opening the cupboard, picking each item and moving it to the cupboard, and closing the cupboard. Achieving each goal requires multiple actions of different types, including moving and acting on objects. This allows us to experiment with a simple variation of our model to generate intermediate goals.

We compare our approach to multiple recent methods. Experiments on the LANI navigation task indicate that decomposing goal prediction and action generation significantly improves instruction execution performance. While we observe similar trends on the CHAI instructions, results are overall weaker, illustrating the complexity of the task. We also observe that inherent ambiguities in instruction following make exact goal identification difficult, as demonstrated by imperfect human performance. However, the gap to human-level performance still remains large across both tasks. Our code and data are available at `github.com/clic-lab/ciff`.

2 Technical Overview

Task Let $\mathcal{X}$ be the set of all *instructions*, $\mathcal{S}$ the set of all *world states*, and $\mathcal{A}$ the set of all *actions*. An instruction $\bar{x} \in \mathcal{X}$ is a sequence $\langle x_1, \ldots, x_n \rangle$, where each x_i is a token. The agent executes instructions by generating a sequence of actions, and indicates execution completion with the special action STOP.

The sets of actions $\mathcal{A}$ and states $\mathcal{S}$ are domain specific. In the navigation domain LANI, the actions include moving the agent and changing its orientation. The state information includes the position and orientation of the agent and the different landmarks. The agent actions in the CHALET house environment include moving and changing the agent orientation, as well as an object interaction action. The state encodes the position and orientation of the agent and all objects in the house. For interactive objects, the state also includes their status, for example if a drawer is open or closed. In both domains, the actions are discrete. The domains are described in Section 6.

Model The agent does not observe the world state directly, but instead observes its pose and an RGB image of the environment from its point of view. We define these observations as the agent context $\tilde{s}$. An agent model is a function from an agent context $\tilde{s}$ to an action $a \in \mathcal{A}$. We model goal prediction as predicting a probability distribution over the agent visual observations, representing the likelihood of locations or objects in the environment being target positions or objects to be acted on. Our model is described in Section 4.

Learning We assume access to training data with N examples $\{(\bar{x}^{(i)}, s_1^{(i)}, s_g^{(i)})\}_{i=1}^{N}$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $s_g^{(i)}$ is the goal state. We decompose learning; training goal prediction using supervised learning, and action generation using oracle goals with policy gradient in a contextual bandit setting. We assume an instrumented environment with access to the world state, which is used to compute rewards during training only. Learning is described in Section 5.

Evaluation We evaluate task performance on a test set $\{(\bar{x}^{(i)}, s_1^{(i)}, s_g^{(i)})\}_{i=1}^{M}$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $s_g^{(i)}$ is the goal state. We evaluate task completion accuracy and the distance of the agent's final state to $s_g^{(i)}$.

3 Related Work

Mapping instruction to action has been studied extensively with intermediate symbolic representations (e.g., Chen and Mooney, 2011; Kim and Mooney, 2012; Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Misra et al., 2015, 2016). Recently, there has been growing interest in direct mapping from raw visual observations to actions (Misra et al., 2017; Xiong et al., 2018; Anderson et al., 2018; Fried et al., 2018). We propose a model that enjoys the benefits of such direct mapping, but explicitly decomposes that task to interpretable goal prediction and action generation. While we focus on natural language, the problem has also been studied using synthetic language (Chaplot et al., 2018; Hermann et al., 2017).

Our model design is related to hierarchical reinforcement learning, where sub-policies at different levels of the hierarchy are used at different frequencies (Sutton et al., 1998). Oh et al. (2017) uses a two-level hierarchy for mapping synthetic language to actions. Unlike our visual goal representation, they use an opaque vector representation. Also, instead of reinforcement learning, our methods emphasize sample efficiency.

Goal prediction is related to referring expression interpretation (Matuszek et al., 2012a; Krishnamurthy and Kollar, 2013; Kazemzadeh et al., 2014; Kong et al., 2014; Yu et al., 2016; Mao et al., 2016; Kitaev and Klein, 2017). While our model solves a similar problem for goal prediction, we focus on detecting visual goals for actions, including both navigation and manipulation, as part of an instruction following model. Using formal goal representation for instruction following was studied by MacGlashan et al. (2015). In contrast, our model generates a probability distribution over images, and does not require an ontology.

Our data collection is related to existing work. LANI is inspired by the HCRC Map Task (Anderson et al., 1991), where a leader directs a follower to navigate between landmarks on a map. We use a similar task, but our scalable data collection process allows for a significantly larger corpus. We also provide an interactive navigation environment, instead of only map diagrams. Unlike Map Task, our leaders and followers do not interact in real time. This abstracts away interaction challenges, similar to how the SAIL navigation corpus was collected (MacMahon et al., 2006). CHAI instructions were collected using scenarios given to workers, similar to the ATIS collection process (Hemphill et al., 1990; Dahl et al., 1994). Recently, multiple 3D research environments were released. LANI has a significantly larger state space than existing navigation environments (Hermann et al., 2017; Chaplot et al., 2018), and CHALET, the environment used for CHAI, is larger and has more complex manipulation compared to similar environments (Gordon et al., 2018; Das et al., 2018). In addition, only synthetic language data has been released for these environment. An exception is the Room-to-Room dataset (Anderson et al., 2018) that makes use of an environment of connected panoramas of house settings. Although it provides a realistic vision challenge, unlike our environments, the state space is limited to a small number of panoramas and manipulation is not possible.

4 Model

We model the agent policy as a neural network. The agent observes the world state s_t at time t as an RGB image $\mathbf{I}_t$. The agent context $\tilde{s}_t$, the information available to the agent to select the next action a_t, is a tuple $(\bar{x}, \mathbf{I}_P, \langle(\mathbf{I}_1, p_1), \ldots, (\mathbf{I}_t, p_t)\rangle)$, where $\bar{x}$ is the natural language instructions, $\mathbf{I}_P$ is a panoramic view of the environment from the starting position at time $t = 1$, and $\langle(\mathbf{I}_1, p_1), \ldots, (\mathbf{I}_t, p_t)\rangle$ is the sequence of observations $\mathbf{I}_t$ and poses p_t up to time t. The panorama $\mathbf{I}_P$ is generated through deterministic exploration by rotating 360° to observe the environment at the beginning of the execution.[1]

The model includes two main components: goal prediction and action generation. The agent uses the panorama $\mathbf{I}_P$ to predict the goal location l_g. At each time step t, a projection of the goal location into the agent's current view $\mathbf{M}_t$ is given as input to an RNN to generate actions. The probability of an action a_t at time t decomposes to:

$$P(a_t \mid \tilde{s}_t) = \sum_{l_g} \Big(P(l_g \mid \bar{x}, \mathbf{I}_P) \\ P(a_t \mid l_g, (\mathbf{I}_1, p_1), \ldots, (\mathbf{I}_t, p_t)) \Big) \ ,$$

where the first term puts the complete distribution mass on a single location (i.e., a delta function). Figure 2 illustrates the model.

[1] The panorama is a concatenation of deterministic observations along the width dimension. For simplicity, we do not include these deterministic steps in the execution.

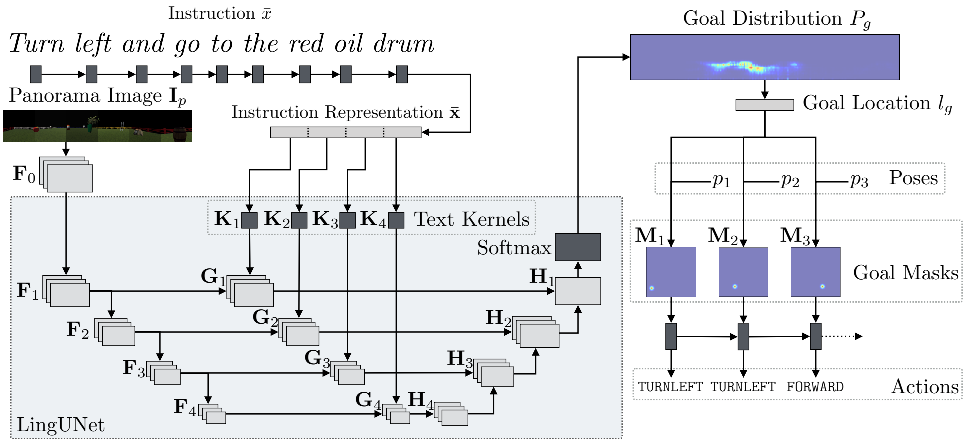


Figure 2: An illustration for our architecture (Section 4) for the instruction *turn left and go to the red oil drum* with a LINGUNET depth of $m = 4$. The instruction $\bar{x}$ is mapped to $\bar{\mathbf{x}}$ with an RNN, and the initial panorama observation $\mathbf{I}_P$ to $\mathbf{F}_0$ with a CNN. LINGUNET generates $\mathbf{H}_1$, a visual representation of the goal. First, a sequence of convolutions maps the image features $\mathbf{F}_0$ to feature maps $\mathbf{F}_1,\ldots,\mathbf{F}_4$. The text representation $\bar{\mathbf{x}}$ is used to generate the kernels $\mathbf{K}_1,\ldots,\mathbf{K}_4$, which are convolved to generate the text-conditioned feature maps $\mathbf{G}_1,\ldots,\mathbf{G}_4$. These feature maps are de-convolved to $\mathbf{H}_1,\ldots,\mathbf{H}_4$. The goal probability distribution P_g is computed from $\mathbf{H}_1$. The goal location is the inferred from the max of P_g. Given l_g and p_t, the pose at step t, the goal mask $\mathbf{M}_t$ is computed and passed into an RNN that outputs the action to execute.

Goal Prediction To predict the goal location, we generate a probability distribution P_g over a feature map $\mathbf{F}_0$ generated using convolutions from the initial panorama observation $\mathbf{I}_P$. Each element in the probability distribution P_g corresponds to an area in $\mathbf{I}_P$. Given the instruction $\bar{x}$ and panorama $\mathbf{I}_P$, we first generate their representations. From the panorama $\mathbf{I}_P$, we generate a feature map $\mathbf{F}_0 = [\text{CNN}_0(\mathbf{I}_P); \mathbf{F}^p]$, where CNN_0 is a two-layer convolutional neural network (CNN; LeCun et al., 1998) with rectified linear units (ReLU; Nair and Hinton, 2010) and $\mathbf{F}^p$ are positional embeddings.[2] The concatenation is along the channel dimension. The instruction $\bar{x} = \langle x_1, \cdots x_n \rangle$ is mapped to a sequence of hidden states $\mathbf{l}_i = \text{LSTM}_x(\psi_x(x_i), \mathbf{l}_{i-1})$, $i = 1, \ldots, n$ using a learned embedding function ψ_x and a long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) RNN LSTM_x. The instruction representation is $\bar{\mathbf{x}} = \mathbf{l}_n$.

We generate the probability distribution P_g over pixels in $\mathbf{F}_0$ using LINGUNET. The architecture of LINGUNET is inspired by the U-NET image generation method (Ronneberger et al., 2015), except that the reconstruction phase is conditioned on the natural language instruction. LINGUNET first applies m convolutional layers to generate a sequence of feature maps $\mathbf{F}_j = \text{CNN}_j(\mathbf{F}_{j-1})$, $j = 1 \ldots m$, where each CNN_j is a convolutional layer with leaky ReLU non-linearities (Maas et al., 2013) and instance normalization (Ulyanov et al., 2016). The instruction representation $\bar{\mathbf{x}}$ is split evenly into m vectors $\{\bar{\mathbf{x}}_j\}_{j=1}^{m}$, each is used to create a 1×1 kernel $\mathbf{K}_j = \text{AFFINE}_j(\bar{\mathbf{x}}_j)$, where each AFFINE_j is an affine transformation followed by normalizing and reshaping. For each $\mathbf{F}_j$, we apply a 2D 1×1 convolution using the text kernel $\mathbf{K}_j$ to generate a text-conditioned feature map $\mathbf{G}_j = \text{CONVOLVE}(\mathbf{K}_j, \mathbf{F}_j)$, where CONVOLVE convolves the kernel over the feature map. We then perform m deconvolutions to generate a sequence of feature maps $\mathbf{H}_m,\ldots,\mathbf{H}_1$:

$$
\begin{aligned}
\mathbf{H}_m &= \text{DECONV}_m(\text{DROPOUT}(\mathbf{G}_m)) \\
\mathbf{H}_j &= \text{DECONV}_j([\mathbf{H}_{j+1}; \mathbf{G}_j]) \ .
\end{aligned}
$$

DROPOUT is dropout regularization (Srivastava et al., 2014) and each DECONV_j is a deconvolution operation followed a leaky ReLU nonlinearity and instance norm.[3] Finally, we generate P_g by applying a softmax to $\mathbf{H}_1$ and an additional learned scalar bias term b_g to represent events where the goal is out of sight. For example, when the agent already stands in the goal position and therefore the panorama does not show it.

We use P_g to predict the goal position in the environment. We first select the goal pixel in $\mathbf{F}_0$ as the pixel corresponding to the highest probability element in P_g. We then identify the corresponding 3D location l_g in the environment using backward camera projection, which is computed given the

[2] We generate $\mathbf{F}^p$ by creating a channel for each deterministic observation used to create the panorama, and setting all the pixels corresponding to that observation location in the panorama to 1 and all others to 0. The number of observations depends on the agent's camera angle.

[3] DECONV_1 does deconvolution only.

camera parameters and p_1, the agent pose at the beginning of the execution.

Action Generation Given the predicted goal l_g, we generate actions using an RNN. At each time step t, given p_t, we generate the goal mask $\mathbf{M}_t$, which has the same shape as the observed image $\mathbf{I}_t$. The goal mask $\mathbf{M}_t$ has a value of 1 for each element that corresponds to the goal location l_g in $\mathbf{I}_t$. We do not distinguish between visible or occluded locations. All other elements are set to 0. We also maintain an out-of-sight flag o_t that is set to 1 if (a) l_g is not within the agent's view; or (b) the max scoring element in P_g corresponds to b_g, the term for events when the goal is not visible in $\mathbf{I}_P$. Otherwise, o_t is set to 0. We compute an action generation hidden state y_t with an RNN:

$$y_t = \text{LSTM}_A\left(\text{AFFINE}_A([\text{FLAT}(\mathbf{M}_t); o_t]), y_{t-1}\right) \ ,$$

where FLAT flattens $\mathbf{M}_t$ into a vector, AFFINE$_A$ is a learned affine transformation with ReLU, and LSTM$_A$ is an LSTM RNN. The previous hidden state y_{t-1} was computed when generating the previous action, and the RNN is extended gradually during execution. Finally, we compute a probability distribution over actions:

$$P(a_t \mid l_g, (\mathbf{I}_1, p_1), \ldots, (\mathbf{I}_t, p_t)) = \text{SOFTMAX}(\text{AFFINE}_p([y_t; \psi_T(t)])) \ ,$$

where ψ_T is a learned embedding lookup table for the current time (Chaplot et al., 2018) and AFFINE$_p$ is a learned affine transformation.

Model Parameters The model parameters θ include the parameters of the convolutions CNN$_0$ and the components of LINGUNET: CNN$_j$, AFFINE$_j$, and DECONV$_j$ for $j = 1, \ldots, m$. In addition we learn two affine transformations AFFINE$_A$ and AFFINE$_p$, two RNNs LSTM$_x$ and LSTM$_A$, two embedding functions ψ_x and ψ_T, and the goal distribution bias term b_g. In our experiments (Section 7), all parameters are learned without external resources.

5 Learning

Our modeling decomposition enables us to choose different learning algorithms for the two parts. While reinforcement learning is commonly deployed for tasks that benefit from exploration (e.g., Peters and Schaal, 2008; Mnih et al., 2013), these methods require many samples due to their high sample complexity. However, when learning with natural language, only a relatively small number of samples is realistically available. This problem was addressed in prior work by learning in a contextual bandit setting (Misra et al., 2017) or mixing reinforcement and supervised learning (Xiong et al., 2018). Our decomposition uniquely offers to tease apart the language understanding problem and address it with supervised learning, which generally has lower sample complexity. For action generation though, where exploration can be autonomous, we use policy gradient in a contextual bandit setting (Misra et al., 2017).

We assume access to training data with N examples $\{(\bar{x}^{(i)}, s_1^{(i)}, s_g^{(i)})\}_{i=1}^N$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $s_g^{(i)}$ is the goal state. We train the goal prediction component by minimizing the cross-entropy of the predicted distribution with the gold-standard goal distribution. The gold-standard goal distribution is a deterministic distribution with probability one at the pixel corresponding to the goal location if the goal is in the field of view, or probability one at the extra out-of-sight position otherwise. The gold location is the agent's location in $s_g^{(i)}$. We update the model parameters using Adam (Kingma and Ba, 2014).

We train action generation by maximizing the expected immediate reward the agent observes while exploring the environment. The objective for a single example i and time stamp t is:

$$J = \sum_{a \in \mathcal{A}} \pi(a \mid \tilde{s}_t) R^{(i)}(s_t, a) + \lambda H(\pi(. \mid \tilde{s}_t)) \ ,$$

where $R^{(i)} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is an example-specific reward function, $H(\cdot)$ is an entropy regularization term, and λ is the regularization coefficient. The reward function $R^{(i)}$ details are described in details in Appendix B. Roughly speaking, the reward function includes two additive components: a problem reward and a shaping term (Ng et al., 1999). The problem reward provides a positive reward for successful task completion, and a negative reward for incorrect completion or collision. The shaping term is positive when the agent gets closer to the goal position, and negative if it is moving away. The gradient of the objective is:

$$\begin{aligned} \nabla J &= \sum_{a \in \mathcal{A}} \pi(a \mid \tilde{s}_t) \nabla \log \pi(a \mid \tilde{s}_t) R(s_t, a) \\ &\quad + \lambda \nabla H(\pi(. \mid \tilde{s}_t) \ . \end{aligned}$$

We approximate the gradient by sampling an action using the policy (Williams, 1992), and use the gold goal location computed from $s_g^{(i)}$. We perform several parallel rollouts to compute gradients and update the parameters using Hogwild! (Recht et al., 2011) and Adam learning rates.

Dataset Statistic	Lani	Chai
Number paragraphs	6,000	1,596
Mean instructions per paragraph	4.7	7.70
Mean actions per instruction	24.6	54.5
Mean tokens per instruction	12.1	8.4
Vocabulary size	2,292	1,018

Table 1: Summary statistics of the two corpora.

6 Tasks and Data

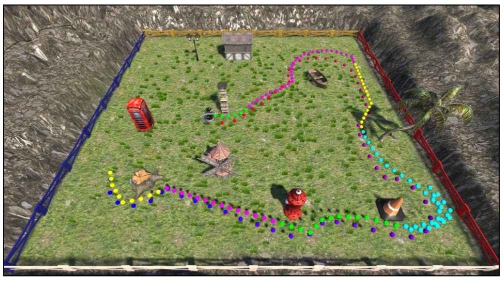
6.1 Lani

The goal of Lani is to evaluate how well an agent can follow navigation instructions. The agent task is to follow a sequence of instructions that specify a path in an environment with multiple landmarks. Figure 1 (left) shows an example instruction.

The environment is a fenced, square, grass field. Each instance of the environment contains between 6–13 randomly placed landmarks, sampled from 63 unique landmarks. The agent can take four types of discrete actions: `FORWARD`, `TURNRIGHT`, `TURNLEFT`, and `STOP`. The field is of size 50×50, the distance of the `FORWARD` action is 1.5, and the turn angle is 15°. The environment simulator is implemented in Unity3D. At each time step, the agent performs an action, observes a first person view of the environment as an RGB image, and receives a scalar reward. The simulator provides a socket API to control the agent and the environment.

Agent performance is evaluated using two metrics: task completion accuracy, and stop distance error. A task is completed correctly if the agent stops within an aerial distance of 5 from the goal.

We collect a corpus of navigation instructions using crowdsourcing. We randomly generate environments, and generate one reference path for each environment. To elicit linguistically interesting instructions, reference paths are generated to pass near landmarks. We use Amazon Mechanical Turk, and split the annotation process to two tasks. First, given an environment and a reference path, a worker writes an instruction paragraph for following the path. The second task requires another worker to control the agent to perform the instructions and simultaneously mark at each point what part of the instruction was executed. The recording of the second worker creates the final data of segmented instructions and demonstrations. The generated reference path is displayed in both tasks. The second worker could also mark the paragraph as invalid. Both tasks are done from an overhead view of the environment, but workers are instructed to provide instructions for a robot that observes the environment from a first person view. Figure 3 shows a reference path and the written instruction. This data can be used for evaluating both executing sequences of instructions and single instructions in isolation.



[*Go around the pillar on the right hand side*] [*and head towards the boat, circling around it clockwise.*] [*When you are facing the tree, walk towards it, and the pass on the right hand side,*] [*and the left hand side of the cone. Circle around the cone,*] [*and then walk past the hydrant on your right,*] [*and the the tree stump.*] [*Circle around the stump and then stop right behind it.*]

Figure 3: Segmented instructions in the Lani domain. The original reference path is marked in red (start) and blue (end). The agent, using a drone icon, is placed at the beginning of the path. The follower path is coded in colors to align to the segmented instruction paragraph.

Table 1 shows the corpus statistics.[4] Each paragraph corresponds to a single unique instance of the environment. The paragraphs are split into train, test, and development, with a 70% / 15% / 15% split. Finally, we sample 200 single development instructions for qualitative analysis of the language challenge the corpus presents (Table 2).

6.2 Chai

The Chai corpus combines both navigation and simple manipulation in a complex, simulated household environment. We use the Chalet simulator (Yan et al., 2018), a 3D house simulator that provides multiple houses, each with multiple rooms. The environment supports moving between rooms, picking and placing objects, and opening and closing cabinets and similar containers. Objects can be moved between rooms and in and out of containers. The agent observes the world in first-person view, and can take five actions: `FORWARD`, `TURNLEFT`, `TURNRIGHT`, `STOP`, and `INTERACT`. The `INTERACT` action acts on objects. It takes as argument a 2D position in the agent's view. Agent performance is evaluated with two metrics: (a) stop distance, which measures the distance of the agent's final state to the final annotated position; and (b) manipulation accuracy, which compares the set of manipulation actions

[4] Appendix A provides statistics for related datasets.

Category	Count		Example
	Lani	Chai	
Spatial relations between locations	123	52	Lani: *go to the* ***right side of the*** *rock* Chai: *pick up the cup* ***next to the*** *bathtub and place it on . . .*
Conjunctions of two more locations	36	5	Lani: *fly between* ***the mushroom and the yellow cone*** Chai: *. . . set it on the table next to* ***the juice and milk****.*
Temporal coordination of sub-goals	65	68	Lani: *at the mushroom* ***turn right and move forward towards the statue*** Chai: ***go back to*** *the kitchen* ***and put the glass in the sink****.*
Constraints on the shape of trajectory	94	0	Lani: *go past the house* ***by the right side of the apple***
Co-reference	32	18	Lani: *turn around* ***it*** *and move in front of fern plant* Chai: *turn left, towards the kitchen door and move through* ***it****.*
Comparatives	2	0	Lani: *. . . the small stone* ***closest to the blue and white fences*** *stop*

Table 2: Qualitative analysis of the Lani and Chai corpora. We sample 200 single development instructions from each corpora. For each category, we count how many examples of the 200 contained it and show an example.

Scenario

You have several hours before guests begin to arrive for a dinner party. You are preparing a wide variety of meat dishes, and need to put them in the sink. In addition, you want to remove things in the kitchen, and bathroom which you don't want your guests seeing, like the soaps in the bathroom, and the dish cleaning items. You can put these in the cupboards. Finally, put the dirty dishes around the house in the dishwasher and close it.

Written Instructions

[*In the kitchen, open the cupboard above the sink.*] [*Put the cereal, the sponge, and the dishwashing soap into the cupboard above the sink.*] [*Close the cupboard.*] [*Pick up the meats and put them into the sink.*] [*Open the dishwasher, grab the dirty dishes on the counter, and put the dishes into the dishwasher.*]

Figure 4: Scenario and segmented instruction from the Chai corpus.

to a reference set. When measuring distance, to consider the house plan, we compute the minimal aerial distance for each room that must be visited. Yan et al. (2018) provides the full details of the simulator and evaluation. We use five different houses, each with up to six rooms. Each room contains on average 30 objects. A typical room is of size 6×6. We set the distance of FORWARD to 0.1, the turn angle to 90°, and divide the agent's view to a 32×32 grid for the INTERACT action.

We collected a corpus of navigation and manipulation instructions using Amazon Mechanical Turk. We created 36 common household scenarios to provide a familiar context to the task.[5] We use two crowdsourcing tasks. First, we provide workers with a scenario and ask them to write instructions. The workers are encouraged to explore the environment and interact with it. We then segment the instructions to sentences automatically. In the second task, workers are presented with the segmented sentences in order and asked to execute them. After finishing a sentence, the workers request the next sentence. The workers do not see the original scenario. Figure 4 shows a scenario and the written segmented paragraph. Similar to Lani, Chai data can be used for studying complete paragraphs and single instructions.

Table 1 shows the corpus statistics.[6] The paragraphs are split into train, test, and development, with a 70% / 15% / 15% split. Table 2 shows qualitative analysis of a sample of 200 instructions.

7 Experimental Setup

Method Adaptations for Chai We apply two modifications to our model to support intermediate goal for the Chai instructions. First, we train an additional RNN to predict the sequence of intermediate goals given the instruction only. There are two types of goals: NAVIGATION, for action sequences requiring movement only and ending with the STOP action; and INTERACTION, for sequence of movement actions that end with an INTERACT action. For example, for the instruction *pick up the red book and go to the kitchen*, the sequence of goals will be ⟨INTERACTION, NAVIGATION, NAVIGATION⟩. This indicates the agent must first move to the object to pick it up via interaction, move to the kitchen door, and finally move within the kitchen. The process of executing an instruction starts with predicting the sequence of goal types. We call our model (Section 4) separately for each goal type. The execution concludes when the final goal is completed. For learning, we create a separate example for each intermediate goal and train the additional RNN separately. The second modification is replacing the backward camera projection for inferring the goal location with ray casting to iden-

[5] We observed that asking workers to simply write instructions without providing a scenario leads to combinations of repetitive instructions unlikely to occur in reality.

[6] The number of actions per instruction is given in the more fine-grained action space used during collection. To make the required number of actions smaller, we use the more coarse action space specified.

	Lani		Chai	
Method	SD	TC	SD	MA
Stop	15.37	8.20	2.99	37.53
RandomWalk	14.80	9.66	2.99	28.96
MostFrequent	19.31	2.94	3.80	37.53
Misra17	10.54	22.9	2.99	32.25
Chaplot18	9.05	31.0	2.99	37.53
Our Approach (OA)	**8.65**	**35.72**	**2.75**	37.53
OA w/o RNN	9.21	31.30	3.75	37.43
OA w/o Language	10.65	23.02	3.22	37.53
OA w/joint	11.54	21.76	2.99	36.90
OA w/oracle goals	2.13	94.60	2.19	41.07

Table 3: Performance on the development data.

tify `INTERACTION` goals, which are often objects that are not located on the ground.

Baselines We compare our approach against the following baselines: (a) Stop: Agent stops immediately; (b) RandomWalk: Agent samples actions uniformly until it exhausts the horizon or stops; (c) MostFrequent: Agent takes the most frequent action in the data, `FORWARD` for both datasets, until it exhausts the horizon; (d) Misra17: the approach of Misra et al. (2017); and (e) Chaplot18: the approach of Chaplot et al. (2018). We also evaluate goal prediction and compare to the method of Janner et al. (2018) and a Center baseline, which always predict the center pixel. Appendix C provides baseline details.

Evaluation Metrics We evaluate using the metrics described in Section 6: stop distance (SD) and task completion (TC) for Lani, and stop distance (SD) and manipulation accuracy (MA) for Chai. To evaluate the goal prediction, we report the real distance of the predicted goal from the annotated goal and the percentage of correct predictions. We consider a goal correct if it is within a distance of 5.0 for Lani and 1.0 for Chai. We also report human evaluation for Lani by asking raters if the generated path follows the instruction on a Likert-type scale of 1–5. Raters were shown the generated path, the reference path, and the instruction.

Parameters We use a horizon of 40 for both domains. During training, we allow additional 5 steps to encourage learning even after errors. When using intermediate goals in Chai, the horizon is used for each intermediate goal separately. All other parameters and detailed in Appendix D.

8 Results

Tables 3 and 4 show development and test results. Both sets of experiments demonstrate similar trends. The low performance of Stop, RandomWalk, and MostFrequent demonstrates the challenges of both tasks, and shows the tasks are robust to simple biases. On Lani, our approach outperforms Chaplot18, improving task completion (TC) accuracy by 5%, and both methods outperform Misra17. On Chai, Chaplot18 and Misra17 both fail to learn, while our approach shows an improvement on stop distance (SD). However, all models perform poorly on Chai, especially on manipulation (MA).

	Lani		Chai	
Method	SD	TC	SD	MA
Stop	15.18	8.29	3.59	39.77
RandomWalk	14.63	9.76	3.59	33.29
MostFrequent	19.14	3.15	4.36	39.77
Misra17	10.23	23.2	3.59	36.84
Chaplot18	8.78	31.9	3.59	39.76
Our Approach	**8.43**	**36.9**	**3.34**	**39.97**

Table 4: Performance on the held-out test dataset.

	Lani		Chai	
Method	Dist	Acc	Dist	Acc
Center	12.0	19.51	3.41	19.0
Janner et al. (2018)	9.61	30.26	2.81	28.3
Our Approach	8.67	35.83	2.12	40.3

Table 5: Development goal prediction performance. We measure distance (Dist) and accuracy (Acc).

To isolate navigation performance on Chai, we limit our train and test data to instructions that include navigation actions only. The Stop baseline on these instructions gives a stop distance (SD) of 3.91, higher than the average for the entire data as these instructions require more movement. Our approach gives a stop distance (SD) of 3.24, a 17% reduction of error, significantly better than the 8% reduction of error over the entire corpus.

We also measure human performance on a sample of 100 development examples for both tasks. On Lani, we observe a stop distance error (SD) of 5.2 and successful task completion (TC) 63% of the time. On Chai, the human distance error (SD) is 1.34 and the manipulation accuracy is 100%. The imperfect performance demonstrates the inherent ambiguity of the tasks. The gap to human performance is still large though, demonstrating that both tasks are largely open problems.

The imperfect human performance raises questions about automated evaluation. In general, we observe that often measuring execution quality with rigid goals is insufficient. We conduct a human evaluation with 50 development examples from Lani rating human performance and our approach. Figure 5 shows a histogram of the ratings. The mean rating for human followers is 4.38, while our approach's is 3.78; we observe a similar trend to before with this metric. Using

Category	Present	Absent	p-value
Spatial relations	8.75	10.09	.262
Location conjunction	10.19	9.05	.327
Temporal coordination	11.38	8.24	.015
Trajectory constraints	9.56	8.99	.607
Co-reference	12.88	8.59	.016
Comparatives	10.22	9.25	.906

Table 6: Mean goal prediction error for Lani instructions with and without the analysis categories we used in Table 2. The p-values are from two-sided t-tests comparing the means in each row.

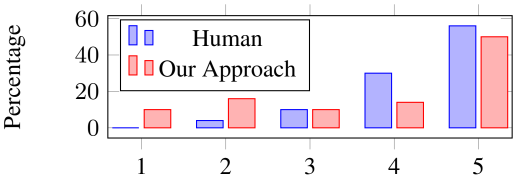
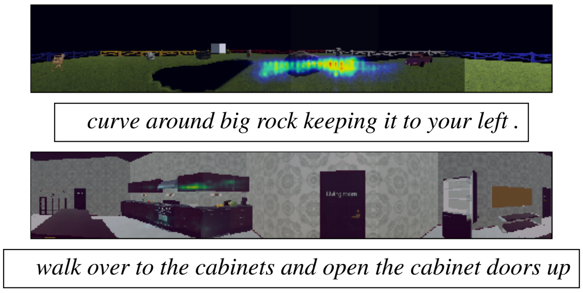


Figure 5: Likert rating histogram for expert human follower and our approach for Lani.


curve around big rock keeping it to your left .

walk over to the cabinets and open the cabinet doors up

Figure 6: Goal prediction probability maps P_g overlaid on the corresponding observed panoramas $\mathbf{I}_P$. The top example shows a result on Lani, the bottom on Chai.

judgements on our approach, we correlate the human metric with the SD measure. We observe a Pearson correlation -0.65 (p=5e-7), indicating that our automated metric correlates well with human judgment.[7] This initial study suggests that our automated evaluation is appropriate for this task.

Our ablations (Table 3) demonstrate the importance of each of the components of the model. We ablate the action generation RNN (w/o RNN), completely remove the language input (w/o Language), and train the model jointly (w/joint Learning).[8] On Chai especially, ablations results in models that display ineffective behavior. Of the ablations, we observe the largest benefit from decomposing the learning and using supervised learning for the language problem.

We also evaluate our approach with access to oracle goals (Table 3). We observe this improves navigation performance significantly on both tasks. However, the model completely fails to learn a reasonable manipulation behavior for Chai. This illustrates the planning complexity of this domain. A large part of the improvement in measured navigation behavior is likely due to eliminating much of the ambiguity the automated metric often fails to capture.

Finally, on goal prediction (Table 5), our approach outperforms the method of Janner et al. (2018). Figure 6 and Appendix Figure 7 show example goal predictions. In Table 6, we break down Lani goal prediction results for the analysis categories we used in Table 2 using the same sample of the data. Appendix E includes a similar table for Chai. We observe that our approach finds instructions with temporal coordination or co-reference challenging. Co-reference is an expected limitation; with single instructions, the model can not resolve references to previous instructions.

9 Discussion

We propose a model for instruction following with explicit separation of goal prediction and action generation. Our representation of goal prediction is easily interpretable, while not requiring the design of logical ontologies and symbolic representations. A potential limitation of our approach is cascading errors. Action generation relies completely on the predicted goal and is not exposed to the language otherwise. This also suggests a second related limitation: the model is unlikely to successfully reason about instructions that include constraints on the execution itself. While the model may reach the final goal correctly, it is unlikely to account for the intermediate trajectory constraints. As we show (Table 2), such instructions are common in our data. These two limitations may be addressed by allowing action generation access to the instruction. Achieving this while retaining an interpretable goal representation that clearly determines the execution is an important direction for future work. Another important open question concerns automated evaluation, which remains especially challenging when instructions do not only specify goals, but also constraints on how to achieve them. Our resources provide the platform and data to conduct this research.

Acknowledgments

This research was supported by NSF (CRII-1656998), Schmidt Sciences, and cloud computing credits from Microsoft. We thank John Langford, Claudia Yan, Bharath Hariharan, Noah Snavely, the Cornell NLP group, and the anonymous reviewers for their advice.

[7] We did not observe this kind of clear anti-correlation comparing the two results for human performance (Pearson correlation of 0.09 and p=0.52). The limited variance in human performance makes correlation harder to test.

[8] Appendix C provides the details of joint learning.

References

Anne H Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry S. Thompson, and Regina Weinert. 1991. The HCRC map task corpus. *Language and Speech*, 34.

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

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

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.

Yonatan Bisk, Daniel Marcu, and William Wong. 2016. Towards a dataset for human computer communication via grounded language acquisition. In *Proceedings of the AAAI Workshop on Symbiotic Cognitive Systems*.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2018. Gated-attention architectures for task-oriented language grounding.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the workshop on Human Language Technology*.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

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. *CoRR*, abs/1806.02724.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. 2017. Grounded language learning in a simulated 3D world. *CoRR*, abs/1706.06551.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9.

Michael Janner, Karthik Narasimhan, and Regina Barzilay. 2018. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L. Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Joohyun Kim and Raymond Mooney. 2012. Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Nikita Kitaev and Dan Klein. 2017. Where is misty? interpreting spatial descriptors by modeling regions in space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. 2014. What are you talking about? text-to-image coreference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jayant Krishnamurthy and T. Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the international conference on machine learning*.

James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael L. Littman, Smaranda Muresan, S Bertel Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. 2015. Grounding english commands to reward functions. In *Robotics: Science and Systems*.

Matthew MacMahon, Brian Stankiewics, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2016. Generation and Comprehension of Unambiguous Object Descriptions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012a. A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning*.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2012b. Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics*.

Hongyuan Mei, Mohit Bansal, and R. Matthew Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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

Dipendra K. Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2016. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35.

Kumar Dipendra Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the international conference on machine learning*.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*.

Junhyuk Oh, Satinder P. Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the international conference on machine learning*.

Jan Peters and Stefan Schaal. 2008. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15.

Alane Suhr and Yoav Artzi. 2018. Situated mapping of sequential instructions to actions with single-step reward observation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1998. Intra-option learning about temporally abstract actions. In *Proceedings of the international conference on machine learning*.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8.

Wenhan Xiong, Xiaoxiao Guo, Mo Yu, Shiyu Chang, Bowen Zhou, and William Yang Wang. 2018. Scheduled policy optimization for natural language communication with intelligent agents. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

Claudia Yan, Dipendra Kumar Misra, Andrew Bennett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. 2018. Chalet: Cornell house agent learning environment. *CoRR*, abs/1801.07357.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. Modeling context in referring expressions. In *Proceedings of the European Conference on Computer Vision*.