Robust and Interpretable Grounding of Spatial References with Relation Networks

Tsung-Yen Yang[1], **Andrew S. Lan**[2], **Karthik Narasimhan**[1]
[1]Princeton University, Princeton, NJ
[2] University of Massachusetts, Amherst, MA
{ty3, karthikn}@princeton.edu, andrewlan@cs.umass.edu

Abstract

Learning representations of spatial references in natural language is a key challenge in tasks like autonomous navigation and robotic manipulation. Recent work has investigated various neural architectures for learning multi-modal representations for spatial concepts. However, the lack of explicit reasoning over entities makes such approaches vulnerable to noise in input text or state observations. In this paper, we develop effective models for understanding spatial references in text that are robust and interpretable, without sacrificing performance. We design a text-conditioned *relation network* whose parameters are dynamically computed with a cross-modal attention module to capture fine-grained spatial relations between entities. This design choice provides interpretability of learned intermediate outputs. Experiments across three tasks demonstrate that our model achieves superior performance, with a 17% improvement in predicting goal locations and a 15% improvement in robustness compared to state-of-the-art systems. [1]

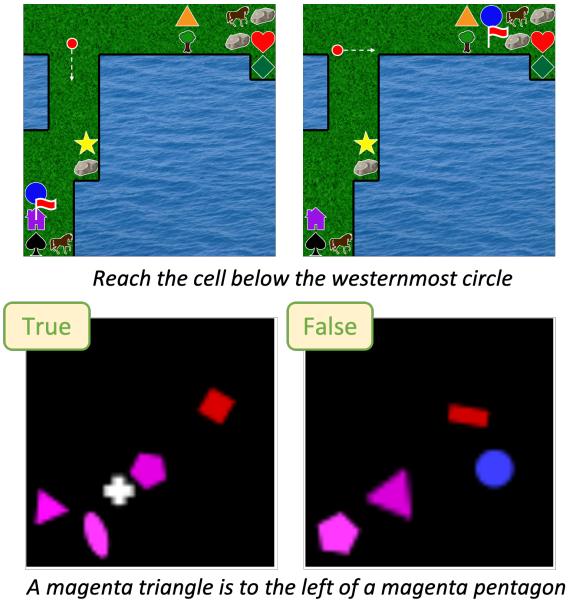


Figure 1: Two different tasks requiring joint spatial reasoning over observation and text – **(top)** the same instruction may specify different goal locations, depending on the map (red flags = goals); **(bottom)** the same statement may be true or false depending on the image.

1 Introduction

Grounding spatial references in text is essential for effective human-machine communication through natural language. Spatial reasoning is ubiquitous in many scenarios such as autonomous navigation (MacMahon et al., 2006; Vogel and Jurafsky, 2010), situated dialog (Skubic et al., 2002) and robotic manipulation (Landsiedel et al., 2017). Despite tremendous applicability, understanding spatial references is a highly challenging task for current natural language processing (NLP) systems, requiring a solid contextual understanding of language dependent on other observations from the environment. Figure 1 demonstrates two tasks where the interpretation of the instruction or statement changes completely with the observation provided. In the first example, the *westernmost circle* may lie to the left or right of the navigating agent's starting location (red dots). In the second, the validity of the statement depends on the relative orientation of the relevant objects – the *triangle* and the *pentagon*.

Some of the earliest work in this field (Herskovits, 1987; Regier, 1996) investigated the grounding of spatial prepositions (*e.g.,* "above", "below") to perceptual processes like visual signals. Such early grounding efforts were limited by computational bottlenecks but several deep neural architectures have been recently proposed that jointly process text and visual input (Janner et al., 2018; Misra et al., 2017; Bisk et al., 2016; Liu et al., 2019; Jain et al., 2019; Gaddy and Klein, 2019; Hristov et al., 2019; Yu et al., 2018). While these approaches have made significant advances in improving the ability of agents at following spatial

[1]Code is available at https://sites.google.com/view/robust-relation-net/home.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1908–1923

instructions, they are either not easily interpretable or require pre-specified parameterization to induce interpretable modules (Bisk et al., 2018). Moreover, their end-to-end formulations and lack of explicit entity reasoning make them susceptible to noise and perturbations in the input data, as we demonstrate in our experiments (Section 5).

In this paper, we develop a model to perform robust and interpretable grounding of spatial references in text. Our objective is to understand and analyze how to ground spatial concepts in a robust and interpretable way. In particular, we focus on the class of *deictic spatial references* (Logan and Sadler, 1996), which specify a location or object in terms of one or more *reference objects*. Our key idea is to decompose the spatial reasoning process into two important steps: (1) identifying the reference object(s) (*e.g., rock, circle*) from the instructions, and (2) accurately inferring the spatial direction (*e.g., left, top-right*) of the goal with reference to that object.

We use a relation network (Santoro et al., 2017) to implicitly enable this factorization by computing representations for each location in the environment based on its interactions with neighboring entities. The parameters of the relation network are dynamically derived from a vector representation of the input text followed by an attention module conditioned on the observations of the environment. This architecture provides three key benefits: (1) the dynamically computed parameters of the relation network enable fine-grained modeling of spatial references, (2) since the model considers relations between pairs of entities, it is more robust to noisy inputs, and (3) the explicit multi-modal representations learned for each entity pair are highly interpretable.

We empirically test our model on three different task settings – classification, value map regression and reinforcement learning (RL) for navigation, and compare its performance to existing state-of-the-art methods. We find that our approach is competitive with or outperforms the baselines under several different evaluation metrics. For example, in the navigation task with RL, our model obtains up to 13.5% relative improvement in policy quality over the best performing baseline. Our approach is also more robust to noisy inputs – for instance, after adding unseen objects as noise to a value map regression task, our model's performance degrades by only around 10% compared to over 20% for the best baseline. Finally, we also present several visualizations of relation and value maps produced by the model, which demonstrate appropriate grounding of reference objects as well as spatial words.

2 Related Work

The role of language in spatial reasoning has been explored since the 1980s (Herskovits, 1987; Logan and Sadler, 1996; Regier, 1996; Regier and Carlson, 2001). Most early papers dealt with the question of representing spatial prepositions (Herskovits, 1987; Coventry et al., 2004; Coventry and Garrod, 2004) and grounding them to spatial templates (Logan and Sadler, 1996). Regier and Carlson (2001) introduced the influential attention vector-sum model which accurately predicted human spatial judgements for words like "above" and "below". The use of neural networks to computationally ground spatial terms to geometric orientations was first explored by Regier (1996) and later by Cangelosi et al. (2005). While spatial reasoning in general is a wide-ranging problem, in this paper, we focus on grounding *deictic spatial references* in third person, which involve referring to a goal location using one or more referent objects (Logan and Sadler, 1996).

Spatial Reasoning in Text. Reasoning about spatial references has been explored in various contexts such as instruction following for 2-D and 3-D navigation (MacMahon et al., 2006; Vogel and Jurafsky, 2010; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Kim and Mooney, 2013; Andreas and Klein, 2015; Fried et al., 2018; Liu et al., 2019; Jain et al., 2019; Gaddy and Klein, 2019; Hristov et al., 2019; Chen et al., 2019) and situated dialog for robotic manipulation (Skubic et al., 2002; Kruijff et al., 2007; Kelleher and Costello, 2009; Landsiedel et al., 2017). Most of these approaches utilize supervised data, either in the form of policy demonstrations or target geometric representations.

More recent work has demonstrated the use of RL in navigation tasks that require spatial reasoning for goal prediction. Misra et al. (2017) use a factored approach to process both text and visual observations in parallel, before fusing the representations to capture correspondences. Janner et al. (2018) use a recurrent network to generate vector representations for the text, which then serve as parameters for an observation processing module (*e.g.,* convolutional neural network (CNNs)). A similar architecture called LingUNet was employed to process observation frames in 3-D navi-

gation tasks (Misra et al., 2018; Blukis et al., 2018), producing probability distributions over goals that are used by the agent to predict action sequences. These pieces of work can be viewed as using forms of feature-wise transformations (Dumoulin et al., 2018) in neural architectures. While we also employ a similar form of text-conditioning, our model processes observations using a relation network (Santoro et al., 2017) (instead of convolutions) which allows us to capture spatial references in a fine-grained manner, robust to noise.

Interpretable Spatial Reasoning. Ramalho et al. (2018) learn viewpoint-invariant representations for spatial relations expressed in natural language. They propose a multi-modal objective to generate images of scenes from text descriptions, and show that the learned representations generalize well to varying viewpoints (and their corresponding descriptions). Bisk et al. (2018) learn interpretable spatial operators for manipulating blocks in a 3-D world. They build a model to explicitly predict the manipulation operators and objects to manipulate, which are used as inputs to another neural network that predicts the final location of each object. The manipulation operators can then be associated with canonical spatial descriptions (*e.g.,* below, south). We focus on demonstrating learned associations between the text representations and visual observations instead of the manipulation operators (actions). Moreover, we also consider the RL setting while both papers above require full supervision.

3 Framework and Design

3.1 Setup

We consider 2-D map-like fully observable environments, where the accompanying text contains spatial references that are key to understanding the goal location. This text contains references to objects or landmarks in the world, as well as relative spatial positions such as *"above"* or *"to the bottom left of"*. We do not assume access to any ontology of entities or spatial references – the agent has to learn representations using feedback from the task. We consider two settings – (1) supervised learning and (2) reinforcement learning.

Supervised Learning. In the supervised scenario, we assume access to data with ground-truth annotation for the quantity to be predicted. This can be either (1) classification labels (*e.g.,* as in ShapeWorld (Andreas et al., 2018)), or (2) value maps (*e.g.,* as in PuddleWorld (Janner et al., 2018)). In this case, the model takes the inputs of an observation map $s \in S$ and a text instruction $x \in X$ and predicts the required outputs.

Reinforcement Learning. We also consider an instruction-following scenario where the main source of supervision is a scalar reward provided upon successful completion of the task. We employ a standard Markov decision process (MDP) framework $< S, A, X, T, R >$, where S is the set of states, A is the set of the actions, X is the set of possible text instructions, T is the transition probability of the environment and R is the reward function. Given a text instruction $x \in X$ and the current state s, the agent takes actions according to a policy $\pi(a|s, x) : S \times X \rightarrow A$, which transitions the environment to the next state s' according to the state transition model $T(s'|s, a, x)$. For simplicity, we assume T is deterministic. This RL setup is inherently harder than supervised learning due to the sparse and weak feedback.

3.2 Model

Any model that can ground spatial references must have the ability to learn flexible, compositional representations for text and effectively fuse it with visual observations. Prior work has explored neural architectures with feature-wise conditioning, where the text representations are used to dynamically induce parameters of Convolutional Neural Networks (CNNs) that process the observations (Janner et al., 2018; Misra et al., 2018). While CNNs are useful to capture spatial invariances, they do not provide fine-grained reasoning abilities since the convolution operates at the coarse level.

To this end, we use a text-conditioned Relation Network to compute representations over the observations. Unlike the relation network in Santoro et al. (2017), the parameters of our relation network are dynamically initialized using the text instruction provided, allowing us to implicitly factorize the reasoning process into locating the reference object(s), and inferring the goal location relative to the reference object(s). Our architecture consists of three main components – *(a) a text encoder ψ, (b) a text-conditioned relation network (RNet) and (c) a task-dependent output network τ*. Figure 2 provides an overview of the entire architecture. For ease of exposition, we will first describe the RNet, followed by the other modules.

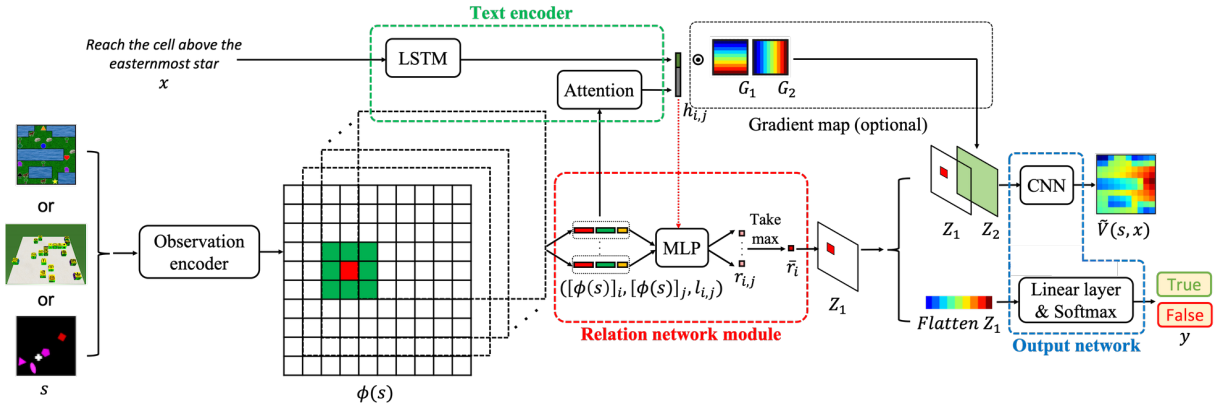


Figure 2: (Model overview) Our architecture consists of three parts – (1) a relation network takes a tensor $\phi(s)$ provided by the observation encoder as input and produces a relation map Z_1; (2) a text encoder (LSTM+attention) converts the text into a vector $h_{i,j}$ for each pair of cells $\langle s_i, s_j \rangle$ in the observation. This $h_{i,j}$ is split and reshaped into the parameters of the relation network module; (3) an output network that takes Z_1 as input and produces the final outputs to be predicted, depending on the task. The max operator in the relation network allows the model to attend to the pair that has a highest value. This enables robustness to the noise and interpretability of the model.

(a) Relation Network (RNet). Assume the input observation s to be a 2-D matrix.[2] First, we convert this matrix into a 3-D tensor $\phi(s)$ by encoding each cell as a vector through an observation encoder (similar to a word embedding). If the element in s is the index of the entity, the observation encoder is an embedding network. On the other hand, if s is a raw image pixels, the observation encoder is a CNN. Next, we feed this tensor into our relation network module f, which computes representations for each cell in the 2-D grid as a function of its neighboring cells. This is done using a multilayer perceptron (MLP) that computes a scalar *relation score* $r_{i,j}$ for each pair of neighboring cells s_i, s_j as:

$$r_{i,j} = f([\phi(s_i)], [\phi(s_j)], l_{i,j})$$

where $[\phi(s_i)] \in \mathbb{R}^k$ is the embedding for cell s_i, and $l_{i,j} \in \mathbb{R}$ is the encoding for the relative location of these two cells. Intuitively, this relation score represents the relevance of the pair of objects (and their positions) in locating the goal. We then perform max operator over all the r-scores associated with each cell to build a relation map $Z_1 \in \mathbb{R}^{m \times n}$, each cell of which is computed as:

$$[Z_1]_i = \max_{j \in \mathcal{N}(i)} r_{i,j},$$

where m and n are the size of the input observation, and $\mathcal{N}(i)$ is the set of neighbors of cell i.[3] The max operator allows the model to attend to the pair that has the highest relation score. Finally, since the processing of the observation should depend on the instruction, we dynamically predict all the parameters of the RNet (*i.e.,* f) for each input s_i, s_j using the text provided (details below).

(b) Text Encoder. We use an LSTM recurrent network (Hochreiter and Schmidhuber, 1997) to convert the instruction text x into a vector h, which is taken to be the weighted combination of the output state of the LSTM after processing the entire instruction. This vector h is used to dynamically initialize all the parameters of the relation network module. We simply choose the size of h to be equal to the total number of parameters in the RNet and split and reshape h accordingly. For example, if RNet f is a two layers of a MLP with the size of (a_1, a_2), the size of vector h is $(2k+1) \cdot a_1 + a_1 \cdot a_2 + a_2 \cdot 1$. We then take the first $(2k+1) \cdot a_1$ components of h and reshape it into a 2-D matrix which is the weight of f in the first layer, and so on. As a result, the computation of the first layer of f is $W_1[\phi(s_i); \phi(s_j); l_{i,j}]$ followed by an activation function, where $W_1 \in \mathbb{R}^{a_1 \times (2k+1)}$. More examples are in Appendix A. Therefore, for each different instruction, RNet will process the observations using a different set of parameters.

Attention: To encourage tighter coupling between the processing of text and observations, we also add an attention module. We compute the text

[2] Though similar architectures can be built for 3-D inputs, we focus on 2-D observations in this paper.

[3] Depending on the tasks, our approach can be extended to handle longer-range relations by considering cells that are more than 2 cells away (though this may require some form of refinement (*e.g.,* beam search) to choose appropriate entities).

representation h as a weighted combination of each word's representation, where the weights are influenced by the current pair of cells being considered by RNet. Specifically, when processing cells s_i, s_j, we compute $h_{i,j}$ as:

$$h_{i,j} = \sum_{k=1}^{L} \alpha_{i,j}(k) h(k),$$

where $\alpha_{i,j}(k) \propto \mathrm{e}^{h(k)^T([\phi(s_i)],[\phi(s_j)])}$ and L is the length of the instruction text. Note that this means the MLP parameters (*i.e.,* the relation network module f) will depend not only on the instruction, but also on the pair of cells s_i, s_j.

(c) Output Network. The final component of our architecture is a task-dependent output network τ, whose form varies according to the task and the type of supervision. For the tasks we consider, we develop two variants of this:
(1) For classification tasks, we simply flatten Z_1 into a vector and pass it through a linear layer followed by a Softmax function to predict the class.
(2) For predicting value maps (in both supervised and reinforcement learning), we use convolution operations. Following Janner et al. (2018), we add two global gradient maps (horizontal and vertical) which have been shown to help global spatial references (*e.g.,* "*the easternmost house*"). We use the last three elements of h to produce the coefficients $\beta_1, \beta_2, \beta_3$ and produce a map Z_2:

$$Z_2 = \beta_1 \cdot G_1 + \beta_2 \cdot G_2 + \beta_3 \cdot J,$$

where $J \in \mathbb{R}^{m \times n}$ is an all-ones matrix and βs are predicted using the text encoder as described above. Then, we concatenate Z_1 (the output of RNet) with Z_2 and feed this tensor ($[Z_1; Z_2] \in \mathbb{R}^{m \times n \times 2}$) into a convolutional layer to predict the value function $\tilde{V}(s, x)$. We call our model *t-RNetAttn*.

3.3 Learning

For all cases below, we train all the parameters of our model jointly, including those in the text encoder, RNet and output networks.

Supervised Learning. As previously mentioned, we consider two supervised learning scenarios – classification and value map prediction (a regression task). For classification, we train our model using softmax loss:

$$\mathcal{L}_1(\Theta) = -\mathrm{E}_{s,x\sim\mathcal{D}}[y \log(p)],$$

where y is the ground truth label, and p is the predicted probability of certain class.

For value maps prediction, we minimize the mean squared error (MSE) between the model's prediction and the ground truth:

$$\mathcal{L}_2(\Theta) = \mathrm{E}_{s,x\sim\mathcal{D}}\left[\left(\tilde{V}_\Theta(s,x) - V(s,x)\right)^2\right],$$

where Θ denotes the parameters in the entire model, and $V(s, x)$ is the ground truth.

Reinforcement Learning. In the RL scenario, we explore the environment using the predicted value map. With the collected trajectories, we then perform fitted Value iteration (Munos and Szepesvári, 2008):

$$\mathcal{L}_3(\Theta) = \mathrm{E}_{(s,a,r,s')\sim\tau}\Bigg[\tilde{V}_\Theta(s,x) - \left(r + \gamma \max_a E_{s'\sim T(s'|s,a)} \tilde{V}_{\Theta'}(s',x)\right)\Bigg],$$

where Θ denotes the parameters of the entire model, and Θ' denotes a set of target parameters that are periodically synced with Θ.

4 Experimental Setup

Tasks. We perform several empirical studies and compare our model with prior work in terms of accuracy and robustness. As previously mentioned, we focus on deictic spatial references, which involve talking about a location or object in terms of other referent objects. We consider three different prediction tasks with accompanying text. These tasks all involve joint reasoning over both observations and the text in order for a system to perform well. The tasks are as follows:
1. *Classification*: Given an image and a text statement containing spatial references, predict whether the statement is applicable. We use data from ShapeWorld (Andreas et al., 2018), which contains images of abstract objects in various shapes and sizes, each paired with a statement about their relative positions and a True/False annotation.
2. *Value map regression*: In this task, the input is a top-down view of a navigation environment along with a text instruction describing a goal location. The aim is to produce the optimal value function map with a value $V(s)$ for each location in the map with respect to the goal. For this task, we use two recently proposed instruction following datasets – ISI (Bisk et al., 2016) and PuddleWorld (Janner

	SW
	ACC ↑
t-VGG (Andreas et al., 2018)	0.71
t-RNetAttn (ours)	**0.72**

(a) Classification

	PW local		PW global		ISI	
	PQ ↑	MD ↓	PQ ↑	MD ↓	PQ ↑	MD ↓
t-CNN (Janner et al., 2018)	0.89	**2.03**	0.91	**3.80**	0.74	3.94
t-UVFA (Schaul et al., 2015)	0.56	4.71	0.62	6.28	0.15	4.61
t-RNetAttn (ours)	**0.91**	2.10	**0.93**	4.23	**0.84**	**3.79**

(b) Goal navigation with RL

	PW local			PW global			ISI	
	MSE ↓	PQ ↑	MD ↓	MSE ↓	PQ ↑	MD ↓	MSE ↓	MD ↓
t-CNN (Janner et al., 2018)	0.25	**0.94**	2.34	0.41	0.89	**3.81**	**0.15**	**3.14**
t-UVFA (Schaul et al., 2015)	3.23	0.57	4.97	1.90	0.62	5.31	–	4.61
t-RNetAttn (ours)	**0.22**	**0.94**	**1.95**	**0.40**	**0.91**	3.82	**0.15**	3.43

(c) Value map regression

Table 1: Performance of all models on all three tasks – classification, value map regression and goal navigation with RL (PW: PuddleWorld, SW: ShapeWorld, PQ: policy quality, MD: Manhattan distance, MSE: mean squared error)). Arrows denote higher or lower scores being better. Best values are in bold.

et al., 2018). ISI contains a set of blocks, each with a different company logo or number, with the instruction being to move a particular block to a desired location. PuddleWorld (PW) is a navigation environment with a positive reward for reaching the goal and negative rewards for stepping in puddles. PW consists of local instructions with local neighborhoods in references *e.g., "two cells to the left of the triangle"* and global instructions with description about the entire map *e.g., "the westernmost rock."* The ground truth value maps for each instance are obtained using the value iteration algorithm (Sutton and Barto, 1998).

3. *Goal navigation with RL*: This is a variant of the previous task where the agent is not provided with ground truth value maps, and instead has to explore the environment, receive rewards (both positive and negative) and learn a policy to navigate to the goal conditioned on the text instruction. We make use of the same datasets as above, sans the value maps.

These three tasks cover different application settings for spatial reasoning. While the prediction objective is different in each one, the input observations are also varied – ShapeWorld uses pixels, while ISI and PuddleWorld are grid worlds. All three are fully observable 2-D environments, and do not contain first-person points of view or certain kinds of spatial references such as intrinsic relations (Logan and Sadler, 1996). However, these environments are sufficient to demonstrate the accuracy, robustness and interpretability of our approach and there is nothing inherently preventing the model from generalizing to more scenarios (*e.g.,* 3-D, different points of view, etc.). More statistics on the datasets including train-test splits are provided in Table 2 in the appendix.

Evaluation Metrics. We use several different quantitative metrics to evaluate the models:

1. *Accuracy* (ACC) of predictions for the binary classification task.
2. *Mean square error* (MSE) between the predicted and ground truth value maps for the regression task.
3. *Policy quality* (PQ), which is a normalized reward score obtained by the agent's policy compared to the optimal policy (Schaul et al., 2015).
4. *Manhattan distance* (MD) which measures the distance between the agent's final position and the ground truth goal location.

The last two measures (PQ and MD) are naturally applicable to the navigation task with RL, but we also apply them to the regression task by inducing a policy from the predicted value map as

$$\pi(s) = \arg\max_a R(s,a) + \gamma T(s'|s,a)\tilde{V}(s').$$

Baselines. For binary classification on ShapeWorld, we use the text-VGG net (t-VGG) from Andreas et al. (2018), which contains a convolution network and two dense layers of size (512, 512) with $tanh$ activation functions, followed by a softmax layer. For the other two tasks, we compare with a text-conditioned *universal value function approximator* (UVFA) (Schaul et al., 2015), and the text-conditioned CNN (t-CNN) architecture of Janner et al. (2018) which has been shown to obtain state-of-the-art performance on the PW and ISI datasets. Both models learn multi-modal representations by either concatenating text and observation

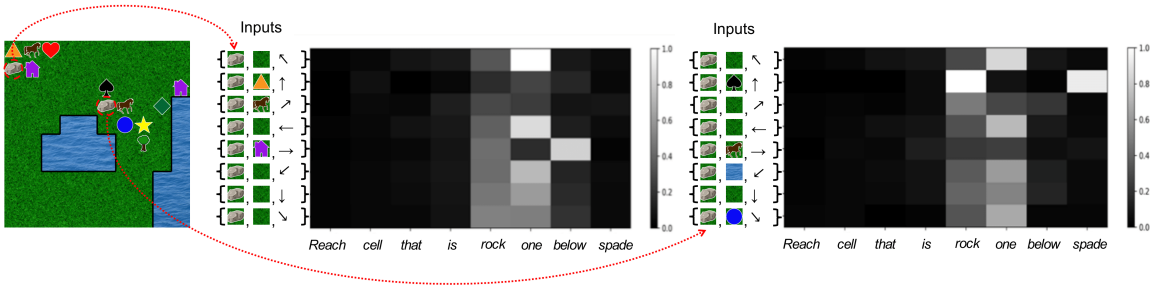
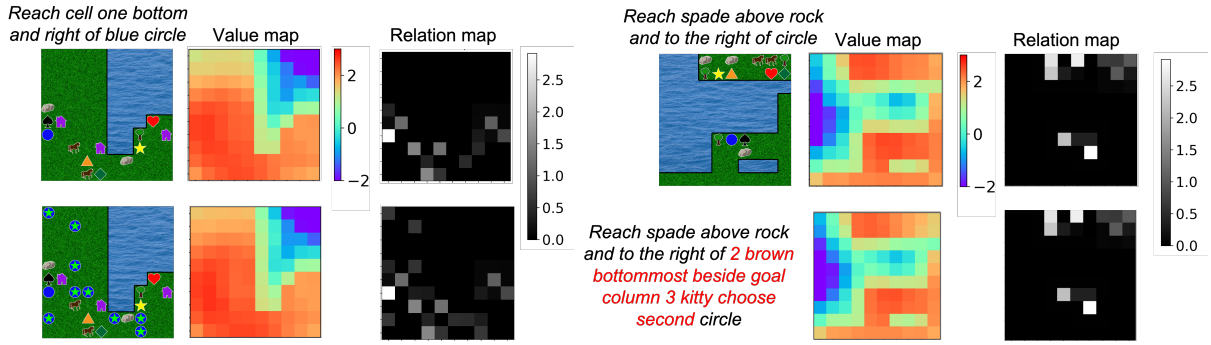


Figure 3: Attention weights in the text encoder for different cell pairs in the observation and the instruction "*Reach cell that is rock one below spade*". The attention weights on the left are for the rock in the upper left corner, and the ones on the right are for the rock in the middle. We see that the model attends correctly to *rock* and *spade* on the right, helping it locate the correct goal. (Note that in the attention weights on the right, the values of weights on the words "*rock*" and "*spade*" are large simultaneously when the spatial relation is mentioned in the text.)



(a) Observational noise **(b)** Textual noise

Figure 4: Visualization of value maps and relation maps after taking absolute values $|Z_1|$ from t-RNetAttn, without (top) and with observation and textual noise (bottom) in the PuddleWorld environment. Blue stars with circle are unseen objects which are not presented during training. Our approach produces sharper (magnitude-wise) $|Z_1|$ values for goal location and referent objects, and is almost undisturbed by noise.

vectors or using text vectors as a kernel in a convolution operation over the observation. Further details on the models are in Appendix A.

5 Results

5.1 Overall performance

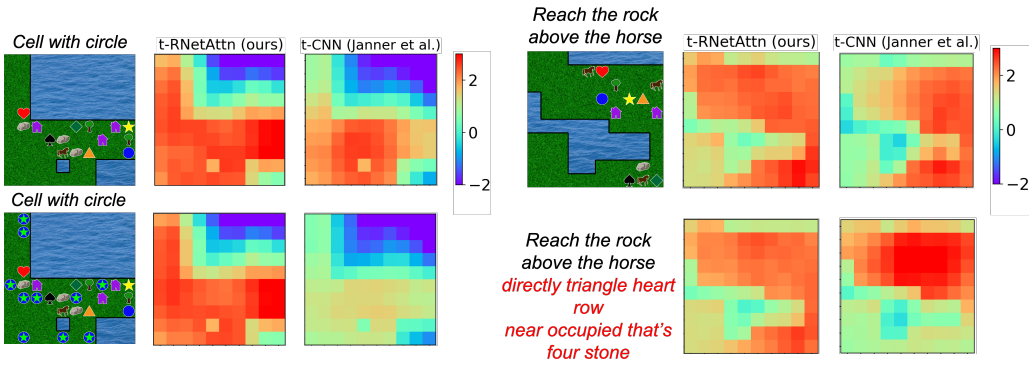
Table 1 details the performance of our model t-RNetAttn, along with the baselines for all three tasks. Our model obtains a slightly higher accuracy of 72% on classification compared to 71% by t-VGG, and outperforms the baselines in MSE and policy quality (PQ) in all settings. Under MD, our model is competitive with the baselines and achieves significantly higher scores in some cases.

In the RL task, the performance gap is particularly pronounced for ISI (0.84 for t-RNetAttn vs. 0.74 for t-CNN in policy quality). For PuddleWorld, we observe that t-RNetAttn achieves better policy quality than t-CNN. This is because that RNet computes the relation score of the pair of objects individually without using a convolution kernel that takes every object into account as in t-CNN. This allows our model to capture spatial relations at a fine-grained level.

In the value map regression task, we observe that the proposed model achieves better performance six out of eight across all metrics. Compared to the RL setting, we find that the regression setting achieves better performance. This is because the RL setting requires better exploration and more interaction with the environment *i.e.,* it is harder than supervised learning of the value maps. Overall, these observations shows that the proposed model achieves superior or competitive performance.

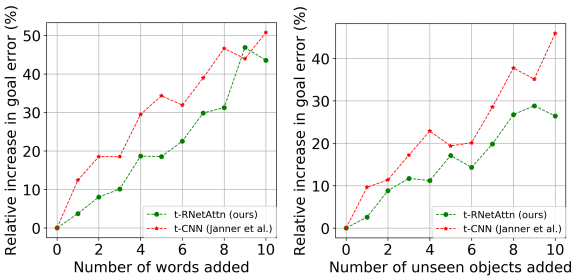
5.2 Interpretability

We now provide some insight into the inner workings of our model by analyzing the intermediate representations produced by both the LSTM and the relation network. Here, we focus on models trained for PuddleWorld; additional analyses for other domains are provided in the appendix.



(a) Observational noise **(b)** Textual noise

Figure 5: Visualization of value maps from t-RNetAttn and t-CNN in PuddleWorld, without (top) and with observational and textual noise (bottom). Blue stars with circle are unseen objects which are not presented during training. t-RNetAttn is more robust and exhibits less degradation in the value map compared to t-CNN.



(a) Words **(b)** Unseen objects

Figure 6: Relative robustness of t-RNetAttn and t-CNN under observational and textual noise in PuddleWorld, in terms of increase in goal localization error for RL goal navigation. Our model is more robust.

Text Encoder Attention. To understand how the text encoder handles the spatial information given by the instructions, we visualize the attention weights over the instruction text. Figure 3 shows an example of attention weights conditioned on two different locations of rocks in the environment. We observe that the model assigns higher attention weights to the correct rock instance (right), where the rock appears below a spade, thereby demonstrating correct grounding for both relevant objects in the instruction.

Relation Map. We also visualize the value map and the relation map $|Z_1|$ produced by the RNet module. Figure 4 shows two examples from PuddleWorld (top row for both (a) and (b))). We observe that the relation network assigns sharper weights (in absolute magnitude) in $|Z_1|$ to objects mentioned in the text as well as their neighboring cells. For instance, in the first example *"Reach cell one bottom and right of blue circle,"* only the circle on the map is referenced by the instruction and $|Z_1|$ shows most extreme weights for that circle.

5.3 Robustness

Next, we investigate the performance of our model under two kinds of noise:

1. *Observational noise:* Here, we add (up to 10 different) unseen objects which are not presented during training to the observations at test time. Models that can ignore such objects while computing representations will be more robust to this type of noise.

2. *Textual noise:* We also add random words, *unrelated* to goal locations, into the instruction text. We randomly choose one position in the text instruction, and insert 1 to 10 words including verbs (*e.g.,* locate, reach, go), articles (*e.g.,* the, a), and irrelevant objects (*e.g.,* car, stone). We aim to test the ability of a model to ignore unhelpful words and focus on the informational content in the text.

Figure 6 plots the relative increase in goal error (MD) for both t-CNN and t-RNetAttn as a function of the amount of observational or textual noise. We see that our model (green line) suffers less from both types of noise, with a drop of 30% vs. $> 45\%$ for t-CNN under observational noise. This fact is further highlighted by Figure 5 which shows the change in value maps when noise is added to both models. While the value maps of t-CNN change drastically, our model is less affected, especially in the prediction of the goal location (highest value). This observation is also strengthened by Figure 4 which shows the change in relation map when noise is added. We observe that in Figure 4(a) the relation maps are similar except that there are small values

on the top-right corner of the map. In addition, in Figure 4(b) the relation maps are almost identical. These results demonstrate that the proposed model can focus on the relevant parts of the observation map. In contrast, t-CNN computes a coarser global representation by taking every nearby object on the map into account. A small noise leads to a failure in capturing correct multi-modal representations. The computational costs, the failure cases, and border impact are provided in Appendix C.

6 Conclusion

We have presented an approach to learn robust and interpretable models for handling spatial references in text. We use a text-conditioned relation network to capture fine-grained spatial concepts between entity pairs, with dynamically computed weights using a cross-modal attention layer. Our empirical experiments over various domains demonstrate that our model matches or outperforms existing state-of-the-art systems on several metrics, e.g. achieving up to 16.7% improvement in goal localization error. Further, we show that our approach is more robust to noise compared to the baselines, in terms of both unseen objects (observational noise) and randomly injected words (textual noise). Finally, we demonstrate that our model's intermediate representations provide a way to interpret its predictions.

Future research can explore other types of spatial relations as well as techniques to scale relation networks to larger observations spaces (*e.g.,* egocentric vision tasks) in a computationally efficient manner. In addition, our approach can be readily extended to more scenarios by considering longer-range cells and incorporating the object detector to extract the object in the scene.

Acknowledgments

The authors would like to thank the anonymous reviewers, area chairs, Ameet Deshpande, Michael Hu, and Princeton NLP members for their valuable feedback.

References

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

Jacob Andreas, Dan Klein, and Sergey Levine. 2018. Learning with latent language. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2166–2179.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, and Devi Lawrence Zitnick, C .and Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

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

Yonatan Bisk, Kevin J. Shih, Yejin Choi, and Daniel Marcu. 2018. Learning interpretable spatial operations in a rich 3d blocks world. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016. Natural language communication with robots. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761.

Valts Blukis, Dipendra Misra, Ross A. Knepper, and Yoav Artzi. 2018. Mapping navigation instructions to continuous control actions with position-visitation prediction. In *Proceedings of the Conference on Robot Learning*, pages 505–518.

Angelo Cangelosi, Kenny R. Coventry, Rohana Rajapakse, Dan Joyce, Alison Bacon, Lynn Richards, and Steven N. Newstead. 2005. Grounding language in perception: A connectionist model of spatial terms and vague quantifiers. In *Modeling Language, Cognition And Action*, pages 47–56. World Scientific.

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

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12538–12547.

Kenny R. Coventry, Angelo Cangelosi, Rohanna Rajapakse, Alison Bacon, Stephen Newstead, Dan Joyce, and Lynn V. Richards. 2004. Spatial prepositions and vague quantifiers: Implementing the functional geometric framework. In *Proceedings of the International Conference on Spatial Cognition*, pages 98–110. Springer.

Kenny R. Coventry and Simon C. Garrod. 2004. *Saying, seeing and acting: The psychological semantics of spatial prepositions*. Psychology Press.

Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. 2018. Feature-wise transformations. *Distill*. Https://distill.pub/2018/feature-wise-transformations.

Hugh Durrant-Whyte and Tim Bailey. 2006. Simultaneous localization and mapping: part i. *IEEE robotics and Automation Magazine*, 13(2):99–110.

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 of the International Conference on Neural Information Processing Systems*, pages 3318–3329.

David Gaddy and Dan Klein. 2019. Pre-learning environment representations for data-efficient neural instruction following. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1946–1956.

Annette Herskovits. 1987. *Language and spatial cognition*. Cambridge university press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yordan Hristov, Daniel Angelov, Michael Burke, Alex Lascarides, and Subramanian Ramamoorthy. 2019. Disentangled relational representations for explaining and learning from demonstration. In *Proceedings of the Conference on Robot Learning*.

Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. 2019. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1862–1872.

Michael Janner, Karthik Narasimhan, and Regina Barzilay. 2018. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6:49–61.

John D. Kelleher and Fintan J. Costello. 2009. Applying computational models of spatial prepositions to visually situated dialog. *Computational Linguistics*, 35(2):271–306.

Joohyun Kim and Raymond J. Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *Proceedings of the Association for Computational Linguistics*, pages 218–227.

Geert-Jan M. Kruijff, Hendrik Zender, Patric Jensfelt, and Henrik I. Christensen. 2007. Situated dialogue and spatial organization: What, where... and why? *International Journal of Advanced Robotic Systems*, 4(1):16.

Christian Landsiedel, Verena Rieser, Matthew Walter, and Dirk Wollherr. 2017. A review of spatial reasoning and interaction for real-world robotics. *Advanced Robotics*, 31(5):222–242.

Ninghao Liu, Mengnan Du, and Xia Hu. 2019. Representation interpretation with spatial encoding and multimodal analytics. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 60–68. ACM.

Gordon D. Logan and Daniel D. Sadler. 1996. A computational analysis of the apprehension of spatial relations.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the Conference on Artificial Intelligence*, pages 1475–1482.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678.

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*, pages 1004–1015.

Rémi Munos and Csaba Szepesvári. 2008. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857.

Tiago Ramalho, Tomáš Kociskỳ, Frederic Besse, S. M. Eslami, Gábor Melis, Fabio Viola, Phil Blunsom, and Karl Moritz Hermann. 2018. Encoding spatial relations from natural language. *arXiv preprint arXiv:1807.01670*.

Terry Regier. 1996. *The human semantic potential: Spatial language and constrained connectionism*. MIT Press.

Terry Regier and Laura A. Carlson. 2001. Grounding spatial language in perception: An empirical and computational investigation. *Journal of experimental psychology: General*, 130(2):273.

Adam Santoro, David Raposo, David G. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances of the Neural Information Processing Systems*, pages 4967–4976.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal value function approximators. In *Proceedings of the International Conference on Machine Learning*, pages 1312–1320.

Marjorie Skubic, Dennis Perzanowski, Alan Schultz, and William Adams. 2002. Using spatial language in a human-robot dialog. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 4143–4148. IEEE.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to reinforcement learning*. MIT Press.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the Association for Computational Linguistics*, pages 806–814.

Haonan Yu, Haichao Zhang, and Wei Xu. 2018. Interactive grounded language acquisition and generalization in a 2d world. In *Proceedings of the International Conference on Learning Representations*.

Appendix

Outline. Section A describes the architecture of the models, the implementation details, the instructions for executing the code, and computational resources. Section B details three datasets considered in the paper. Section C provides an additional analysis of interpretability and robustness in ISI and ShapeWorld. In addition, we discuss the potential weakness of the proposed approach. Finally, Section D discusses the border impact and the risk of deploying the proposed approach in real applications.

A Implementation Details

Architectures. We briefly describe the architectures of t-RNetAttn, t-CNN, and t-UVFA.

(1) t-RNetAttn: In PuddleWorld the relation network is a multilayer perceptron (MLP) with layers of $\{10, 10, 1\}$ neurons. We use tanh as an activation function. The size of each object embedding $[\phi(e)]$ is 7. This results in a size 263 for the text vector h (This is because that $(7 \cdot 2 + 1) \cdot 10 + 10 \cdot 10 + 10 \cdot 1 + 3 = 263$, where the last 3 components of h are used for a gradient map). The text encoder is an LSTM with the size 15 for input layers and the size 30 for the hidden layers, followed by a linear decoder. In the first 260 components of h, the first 150 components are reshaped into a 2-D matrix $W_1 \in \mathbb{R}^{10\times15}$ for the first layer, the following 100 components are reshaped into a 2-D matrix $W_2 \in \mathbb{R}^{10\times10}$ for the second layer, and the rest of the components are reshaped into a 2-D matrix $W_3 \in \mathbb{R}^{1\times10}$ for the final output layer. As a result, we compute $r_{i,j}$ by

$$\begin{aligned} r_{i,j} &= f([\phi(s_i), \phi(s_j), l_{i,j}]) \\ &= W_3(\sigma(W_2\sigma(W_1[\phi(s_i), \phi(s_j), l_{i,j}]))), \end{aligned} \tag{1}$$

where $\sigma(\cdot)$ is an activation function. The remaining 3 components are used for the gradient map Z_2. Finally, the relation map Z_1, and the gradient map Z_2 are concatenated, resulting in the size $(10, 10, 2)$ tensor, where the first two numbers are the size of the map and the last number is the size of the channel. We then use a convolution operation with the kernel size 3 and a relu activation function to get the final value maps.

In ISI we use the same architectures in PuddleWorld except for $[\phi(e)]$ being 13. This results in a size 383 for text vector h. (This is because that $(13 \cdot 2 + 1) \cdot 10 + 10 \cdot 10 + 10 \cdot 1 + 3 = 383$, where the last 3 components of h are used for a gradient map.)

In ShapeWorld we use the same architectures in PuddleWorld except for replacing the final convolution layer with a dense layer followed by a softmax function to predict labels.

(2) t-CNN: In PuddleWorld we use a convolution filter with a size $(3, 3)$. (We also increase the size of filters to match the number of parameters in the proposed model. We find that the baseline performance drops by 5% since there are too many entities on a single filter.) This results in a size 66 for h (This is because that $3 \cdot 3 \cdot 7 + 3 = 66$, where the last 3 components of h are used for a gradient map). We use the first 63 components of h to be a convolution kernel of size $(3, 3, 7)$ on an environment map, where the value 7 is the size of the object embedding. This results in a text-conditioned map Z_1 with a size $(10, 10, 1)$ The remaining components are used to construct a gradient map. Finally, the relation map Z_1, and the gradient map Z_2 are concatenated, resulting in the size $(10, 10, 2)$ tensor. We then use a convolution operation with the kernel size 3 and a relu activation function to get the final value maps.

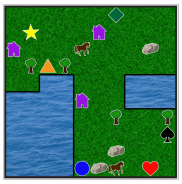
In ISI we use the same architectures in PuddleWorld. We use a convolution filter with a size 3. This creates a size 120 for h (This is because that $3 \cdot 3 \cdot 13 + 3 = 120$).

(3) t-UVFA: For all three datasets, the size of h is 7, followed by concatenating h and state representations for the map to obtain a multi-modal representation. This representation is then fed into a deconvolution layer used to decode and reconstruct value maps.

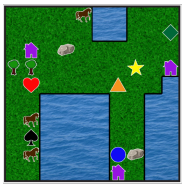
For all the models and datasets, we use the batch size 512. The step size is 10^{-3}. We choose Adam algorithm to train the models.

Hyperparameter Search. We conduct a grid search on the embedding size of the object $\{4, 5, 6, 7, 8, 9, 10\}$, the step size $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$, the batch size $\{100, 500, 1000\}$. We test the performance (*i.e.,* the policy quality) of each combination of the hyperparameter on the validation set separated from the training dataset.

Instructions for Reproducibility. To reproduce the results, first install the libraries for python3 such as numpy, scipy and PyTorch. Then download the



Go two spaces above the heart

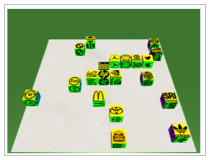


Reach the northernmost house

(a) Local instruction **(b)** Global instruction

Figure 7: Examples of the local and the global instructions in PuddleWorld. A local instruction specifies the goal by using nearby objects. A global instruction specifies the goal by using a global viewpoint.

Put the Toyota block in the same row as the Twitter block



Move Benz to the immediate right of BMW, evenly aligned and slightly separated

(a) **(b)**

Figure 8: Examples in ISI. The goal is to place the block in the right position specified by the instructions.

package from https://github.com/anonymous. Put the code on the designated folder. Finally, go to the example folder and execute the code using python command.

Computational Resources. We conduct the experiments on a machine with an Intel Core i7 CPU and no GPU is used. We find that in general it takes about 6 hours to complete the experiments across all three models. This shows that the proposed model does not have a substantial computation cost compared to the other baselines. All experiments are performed using PyTorch[4].

B Datasets

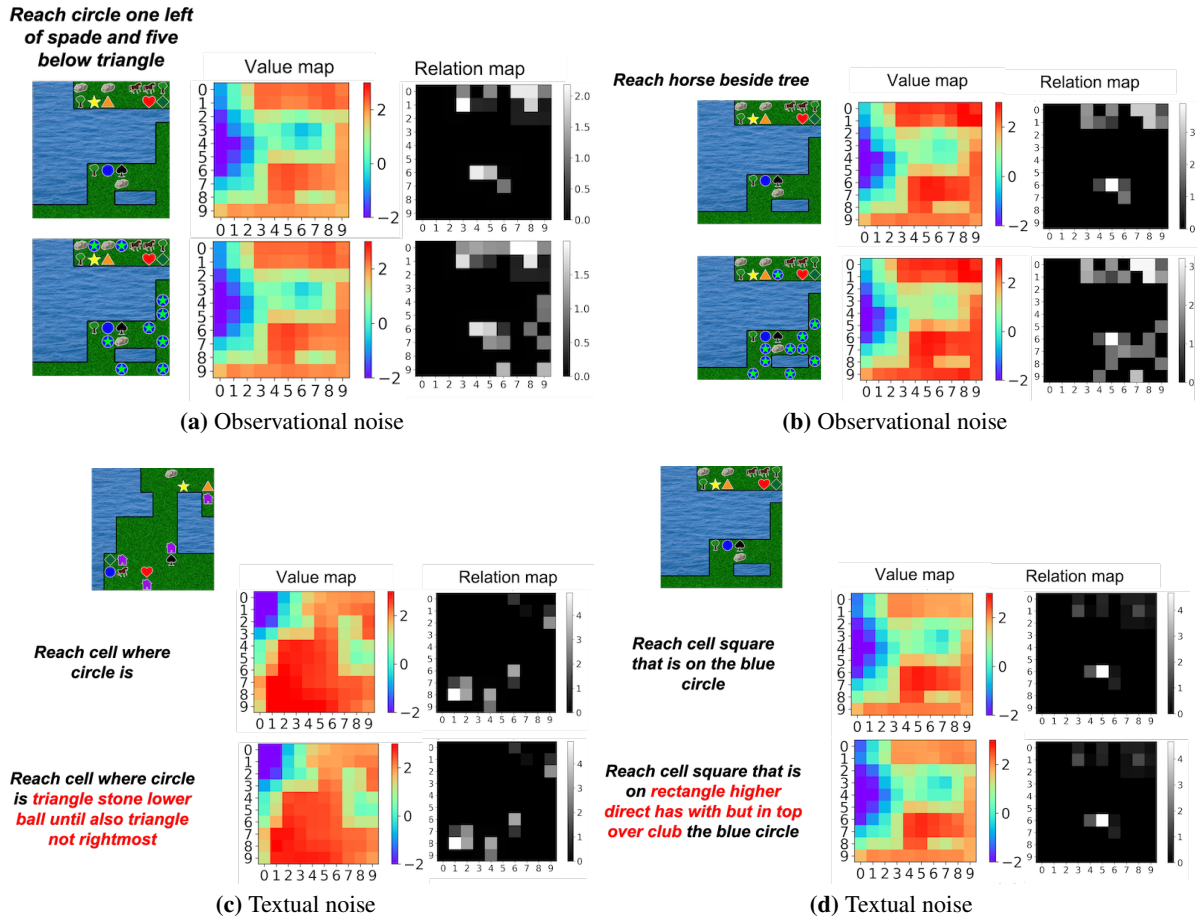
PuddleWorld. A $10 \cdot 10$ grid represents the states. The cell is placed with either a water or a grass. In addition, a grass may be placed with six unique objects (triangle, star, diamond, circle, heart, and spade) appearing once per map or the non-unique objects (rock, tree, horse, and house). Two types of instructions are provided: a local instruction that describes the goal location with the nearby objects, *e.g.,* "*two cells to the left of the triangle*," and a global instruction that specifies the goal location with the global viewpoint, *e.g.,* "*the westernmost rock*." Figure 7 shows the examples of the local and global instructions. We give a reward of $+3$ when the agent reaches the final goal location. The reward design is to encourage the agent to produce the accurate value maps. Please refer to Janner et al. (2018) for more discussion on data collection process.

ISI. The environment contains up to 20 blocks marked with logos (*e.g.,* Toyota, BMW) or digits. Each instruction specifies the goal location of the object, *e.g.,* "*Move Toyota to the immediate right of SRI, evenly aligned and slightly separated*." Figure 8 shows the examples. Please refer to Bisk et al. (2018) for more discussion on data collection process.

ShapeWorld. Each scene contains 4 or 5 non-overlapping objects. Unlike the object in the Puddle and ISI that has a unique identifier, the object in the ShapeWorld is a pixel image. This is to demonstrate that the proposed approach can operate on the raw images. The instruction describes the spatial relationships between pairs of objects specified by shape, color, or both, *e.g.,* "*a red ellipse is to the right of an ellipse*." There are 8 colors and 8 shapes in total. Unlike the previous two tasks that predict a target location, the task in the ShapeWorld is to classifier whether the instruction matches the scene. Figure 1 (bottom) shows the examples. Please refer to Andreas et al. (2018) for more discussion on data collection process.

Dataset Splitting. We follow the same splitting scheme as in Janner et al. (2018); Bisk et al. (2018);

[4] https://pytorch.org



(a) Observational noise **(b)** Observational noise

(c) Textual noise **(d)** Textual noise

Figure 9: Visualization of value maps and relation maps after taking absolute values $|Z_1|$ from t-RNetAttn, without (top) and with observation and textual noise (bottom) in the PuddleWorld environment. Blue stars are unseen objects. Our approach produces sharper (magnitude-wise) $|Z_1|$ values for goal location and referent objects, and is almost undisturbed by noise.

Andreas et al. (2018). We show dataset statistics in table 2.

Dataset	Train	Test
PW local	1566	399
PW global	1071	272
ISI	11871	3177
SW	9000	500

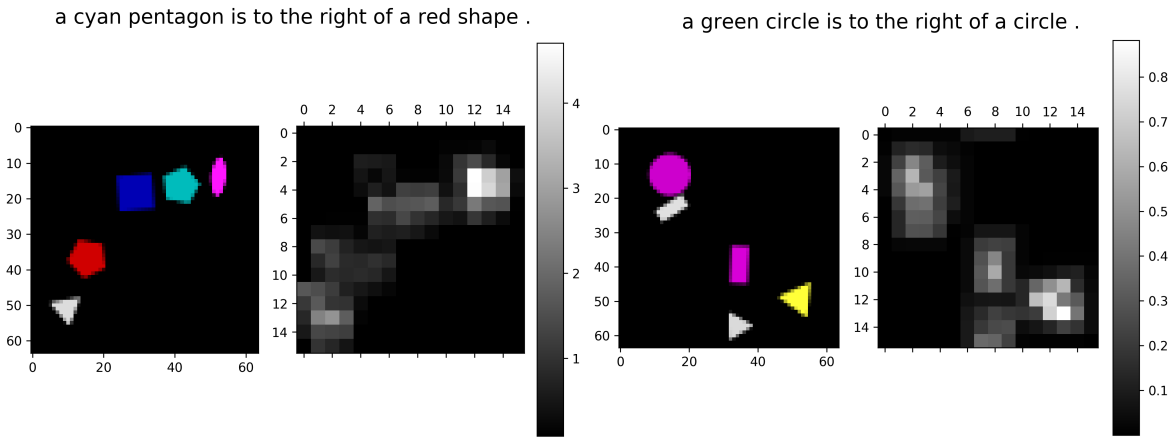
Table 2: Statistics of PuddleWorld (PW), ISI language grounding (ISI), and ShapeWorld (SW).

C Additional Results

C.1 Interpretability

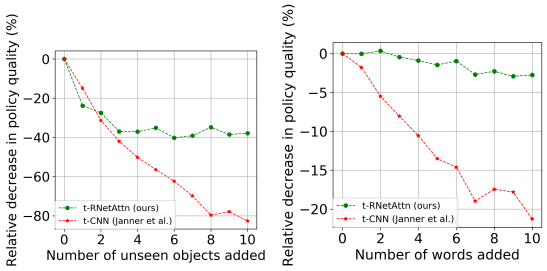
PuddleWorld. To show the proposed model can increase the interpretability, we provide additional visualization examples of relation map in PuddleWorld as shown in Figure 9. We observe that the proposed model assigns a larger magnitude of the weights to the objects mentioned in the text in the relation map. For example, in Figure 9(d) we observe that the model successfully attends to "*circle*", which is specified by the instruction. In addition, under the observational and textual noise the value maps are almost undisturbed by noise. For example, in Figure 9(a) and (b) we observe that the relation maps are almost unchanged after adding observational noise except for relatively small values on the unseen object. On the other hand, in Figure 9(c) and (d) we observe that the relation maps are almost identical after adding textual noise. These observations imply that explicitly computing the relation score of each entity pairs allows the model to attend to objects that are most relevant to the instruction. The results here are also consistent with the results in Figure 4.

ShapeWorld. To show the proposed model can increase the interpretability, we provide visualization examples of the relation map in ShapeWorld as shown in Figure 10. We observe that t-RNetAttn also assigns a larger magnitude of the weights to the objects mentioned in the text. For example,



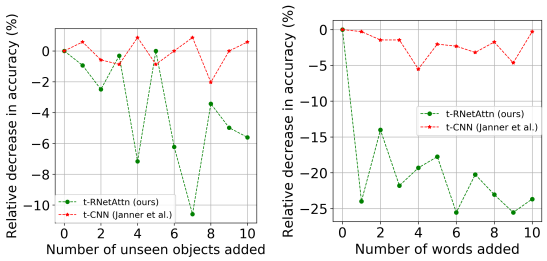
(a) **(b)**

Figure 10: Visualization of value maps and relation maps after taking absolute values $|Z_1|$ from t-RNetAttn in the ShapeWorld environment. Our approach produces sharp (magnitude-wise) $|Z_1|$ values for goal location and referent objects. Note that the size of the environment map is different from the size of the relation map since we reduce the size of the environment map by using a convolution operation, reducing the computational cost.



(a) Observational noise **(b)** Textual noise

Figure 11: Relative robustness of t-RNetAttn and t-CNN under observational and textual noise in ISI, in terms of decrease in policy quality for goal navigation with RL.



(a) Observational noise **(b)** Textual noise

Figure 12: Relative robustness of t-RNetAttn and t-CNN under observational and textual noise in ShapeWorld, in terms of decrease in prediction accuracy.

in Figure 10(a) we observe that there is a larger magnitude of the weights on the top-right corner of the map. This implies that the proposed model successfully attends to the object that is specified by the instruction ("*a cyan pentagon*"). This observation shows that the proposed model still works well when the raw images are presented.

C.2 Robustness

ISI. To show that the proposed model is more robust to the noise, Figure 11 plots the relative decrease in policy quality for both t-CNN and t-RNetAttn as a function of the amount of observational or textual noise in ISI. We can see that our model (green line) suffers less from both types of noise (a drop of 40% vs. >80% for t-CNN on the observational noise with 10 unseen objects and a drop of 2.5% vs. >20% for t-CNN on the textual noise with 10 random words). This implies that the proposed approach is robust to the noise because of the relation network.

ShapeWorld. To show that the proposed model is more robust to the noise, Figure 12 plots the relative decrease in accuracy for both t-CNN and t-RNetAttn as a function of the amount of observational or textual noise in ShapeWorld. For the observational noise, instead of adding the unseen objects, we add noise patches in the input images. The element of each patch is sampling from Gaussian distribution with the mean being zero and variance being one. For the textual noise, we use the same procedure as the one in the PuddleWorld and ISI. We can see that our model (green line) suffers more from both types of noise. One possible reason for this is that in order to reduce the dimension of the observation map, we first perform a

	SW
	ACC ↑
t-RNet (ours)	**0.73**
t-RNetAttn (ours)	0.72

(a) Classification

	PW local			PW global			ISI	
	MSE ↓	PQ ↑	MD ↓	MSE ↓	PQ ↑	MD ↓	MSE ↓	MD ↓
t-RNet (ours)	**0.19**	**0.94**	**1.95**	**0.31**	**0.91**	**3.13**	0.16	3.66
t-RNetAttn (ours)	0.22	0.92	**1.95**	0.40	**0.91**	3.82	**0.15**	3.43

(b) Value map regression

	PW local		PW global		ISI	
	PQ ↑	MD ↓	PQ ↑	MD ↓	PQ ↑	MD ↓
t-RNet (ours)	**0.92**	2.41	**0.93**	**3.56**	**0.88**	**3.22**
t-RNetAttn (ours)	0.91	2.10	**0.93**	4.23	0.84	3.79

(c) Goal navigation with RL

Table 3: Performance on the test set with the three metrics (PQ: policy quality, MD: Manhattan distance, MSE: mean squared error) in PuddleWorld, ISI and ShapeWorld under both supervised and RL. The symbols ↑ and ↓ signify larger numbers are better and smaller numbers are better, respectively. The best values are bold. Note that in the case of supervised learning, we do not report the value of t-VGG since it is designed to solve the task in ShapeWorld (SW). The MSE of t-UVFA in the ISI is not reported since it was not reported in Janner et al. (2018).

convolution on the observation map. The resulting embeddings with a smaller width and height are the inputs to the relation module. Unlike directly using the embedding from the original observation map, this dimension-reduction approach creates a coarse representation of the observation map. This makes a relation module vulnerable to observational and textual noise. In contrast, t-CNN directly operates on the observation map. This makes t-CNN less vulnerable to the noise. One solution to increase the robustness of t-RNetAttn in ShapeWorld is to use an object detector to segment the objects from the map. This would allow t-RNetAttn directly to use the object information rather than the embeddings from the pixel. We leave the improvement of this as a future research direction.

C.3 Ablation Study

One question of the proposed model is that whether the improvement in performance is due to RNet or the attention. To this end, we remove the attention mechanism and simply take an average over all LSTM outputs as h. Figure 3 shows the result. We include the numbers reported in Table 1 for clarity.

C.4 Limitations of the Proposed Model

The robustness experiment in ShapeWorld in Section C.2 shows that the proposed model is vulnerable when the embedding comes from raw pixels instead of the entity itself. Another limitation is that the proposed model only computes representations in a 1-square neighborhood around each cell. This may be problematic when we ask it to resolve "*reach the cell **three above** the easternmost star*", where the goal is three blocks away from the star. One possible solution to this is that we can have another relation network that considers the entities in a 3-square neighborhood around each cell. This creates another relation map Z_3 (similar to Z_1) that captures the long dependency. Then we concatenate Z_3 with Z_1 and Z_2 and feed this tensor ($[Z_1; Z_2; Z_3] \in \mathbb{R}^{m \times n \times 2}$) into a convolutional layer to predict the value function. We leave the improvement of this as a future research direction.

D Border Impact

The proposed approach could be applied in many fields that are required to learn multi-modal representations while providing transparency of the model. For example, in the personalized robotic assistants setting where the agent aims to complete tasks specified by the instruction, the transparency in the model is critical to build trust between humans and AI and mitigate safety risks in making decisions. In addition, in the visual question answering (Antol et al., 2015) where it is required to have a visual understanding of the scene to answer many questions, the proposed model could enhance the performance while unveiling the inner work of the model. Moreover, in the Simultaneous Localization and Mapping (SLAM) (Durrant-Whyte and Bailey, 2006) system where the agent aims to construct the map of the environment and locate itself, the proposed model could fuse multi-sensor input data such as laser and ultrasonic sensors to learn a representation for generating the map.