INTERACTIVE GROUNDED LANGUAGE ACQUISITION AND GENERALIZATION IN 2D ENVIRONMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We build a virtual agent for learning language in a 2D maze-like environment. The agent sees surrounding environment images, listens to a virtual teacher, and takes actions to receive rewards. It interactively learns the teacher's language from scratch based on two language use cases: sentence-directed navigation and question answering. It learns simultaneously the visual representations of the environment, the language, and the action control. By disentangling language grounding from other computational routines and sharing a concept detection function between language grounding and prediction, the agent reliably extrapolates to interpret navigation commands that may contain not only new word combinations but also new words not appeared in training commands. These new words are transferred from question answering. This language ability is trained and evaluated on a population of over 1.6 million distinct sentences consisting of 119 object words, 8 color words, 9 spatial-relation words, and 50 grammatical words. The proposed model significantly outperforms four comparison methods on interpreting zero-shot sentences. We additionally demonstrate human-interpretable intermediate outputs of the model.

1 INTRODUCTION

Some empiricists argue that language may be learned based on its usage (Tomasello, 2003). Skinner (1957) suggests that the successful use of a word reinforces the understanding of its meaning as well as the probability of it being used again in the future. Bruner (1985) emphasizes the role of social interaction in helping a child develop the language, and posits the importance of the feedback and reinforcement from the parents during the process. This paper takes a positive view of the above behaviorism and tries to explore some of the ideas by instantiating them in a 2D virtual environment where *interactive* language acquisition happens. This interactive setting contrasts to a common learning setting in that instead of from static labeled data, language is learned from dynamic interactions with an environment.

Language acquisition can go beyond mapping language as input patterns to output labels for merely obtaining high rewards or accomplishing tasks. We take a step further to require the language to be *grounded* (Harnad, 1990). Specifically, we consult the paradigm of procedural semantics (Woods, 2007) which advocates that words, as abstract procedures, should be capable to pick out referents. We will attempt to explicitly link words to environment concepts instead of treating the whole model as a black box. Such capability also implies that depending on the interactions with an environment, words would have particular meanings in a particular context and some content words in the usual case might not even have meanings in our case. As a result, the goal of this paper is to acquire "in-context" word meanings regardless of their completeness in all scenarios.

It has been argued that a child's exposure to adult language provides inadequate evidence for language learning (Chomsky, 1991), but some induction mechanism might exist to bridge this gap (K. Landauer & T. Dumais, 1997). This characteristics is critical for any AI system to learn infinite sentences from finite training data. Such language generalization problem is specially addressed in our problem setting. After training, we want the agent to extrapolate to interpret *zero-shot* sentences that contain:

1) new combinations of previously seen words for the same use case, or
2) new words learned from other use cases by different network modules.

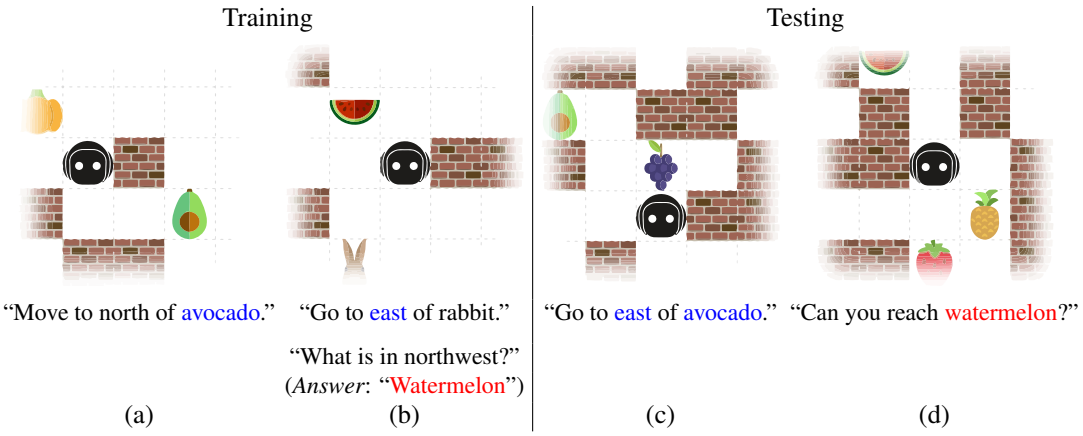


Figure 1: Illustration of the XWORLD environment and language use cases. (a) and (b): NAV and QA training. (c): ZS1 command contains a new word combination ("east" and "avocado") that never appears in NAV training. (d): ZS2 command contains a new word ("watermelon") that never appears in NAV training. This is only a conceptual illustration of the language generalization ability; in practice many training sessions are needed. The agent has to pass both test cases in this example. (Due to space limit, the maps are only partially shown.)

In the following, we will call the former ZS1 sentences and the latter ZS2 sentences. Note that so far the zero-shot problems, addressed by most recent work (Hermann et al., 2017; Chaplot et al., 2017) of interactive language learning, belong to the category of ZS1. In contrast, a reliable interpretation of ZS2 sentences, which is essentially a *transfer learning* (Pan & Yang, 2010) problem, will be a major contribution of this work.

We created a 2D maze-like environment called XWORLD (Figure 1), as a testbed for interactive grounded language acquisition and generalization. In this environment, a virtual agent has two language use cases: navigation (NAV) and question answering (QA). For NAV, the agent need to navigate to correct places indicated by language commands from a virtual teacher. For QA, the agent has to correctly generate single-word answers according to the teacher's questions. NAV tests the language comprehension ability while QA additionally tests the language prediction ability. They happen simultaneously: when the agent is navigating, the teacher might ask questions regarding its current interaction with the environment. Once the agent reaches the target or the time is up, the current *session* ends and a new one is randomly generated according to our configuration (Appendix A). The ZS2 sentences defined in our setting require word meanings to be transferred from answers to navigation commands, or more generally from language *prediction* to *grounding*. This is achieved by establishing an explicit link between grounding and prediction via a common concept detection function, which constitutes the major novelty of our model. With this transferring ability, the agent is able to navigate to a freshly taught object without the need of retraining the NAV pipeline.

It is worthwhile emphasizing that this seemingly "simple" environment in fact poses great challenge for language acquisition and generalization, because:

- *The state space is huge.* Even for a 7×7 map with 15 wall blocks and 5 objects selected from 119 distinct classes, there are already octillions (10^{27}) of possible different configurations, not to mention the intra-class variance of object instances (see Figure 11 in the Appendix). For two configurations that only differ in one block, their successful navigation paths could be completely different. This requires an accurate perception of the environment. Moreover, the configurations are constantly changing from session to session, and from training to testing. Particularly, the target changes across sessions regarding both its location and appearance.
- *The goal space implied by the language is huge.* For a vocabulary size as small as 185, the total number of distinct commands that can be said by the teacher confirming to our defined grammar is already over half a million. Two sentences that differ only in one word could imply completely different goals. This requires an accurate grounding of language.

- *The environment demands a strong language generalization ability from the agent.* The agent has to learn to interpret zero-shot sentences whose lengths might be up to 13 words. It has to "plug" the meaning of a new word or word combination into a familiar sentential context while trying to still make sense of the unfamiliar whole. Recent work (Hermann et al., 2017; Chaplot et al., 2017) addresses ZS1 (for short sentences with several words) but not ZS2 sentences, which is a key difference between the challenge of our learning problem and theirs.

We describe an end-to-end model for the agent to interactively acquire language from scratch and generalize to unfamiliar sentences. Here "scratch" means that the model does not hold any assumption on the semantics or syntax: each sentence is simply a sequence of tokens with each token being equally meaningless in the beginning of learning. This is unlike some early pioneering system (*e.g.*, SHRDLU (Winograd, 1972)) that hard coded the syntax or semantics to link language to a simulated world, which potentially has the scalability issue. The interaction has two aspects: one is with the teacher (*i.e.*, language and rewards) and the other is with the objects in the environment (*e.g.*, stepping on objects or hitting walls). The model takes as input RGB images, sentences, and rewards. It learns simultaneously the visual representations of the environment, the language, and the action control. We evaluate our model on randomly generated XWORLD maps with random agent positions, on a population of over 1.6 million distinct sentences consisting of 119 object words, 8 color words, 9 spatial-relation words, and 50 grammatical words. Detailed analysis of the trained model shows that the language is explicitly grounded such that words are capable to pick out referents in the environment. We specially test the generalization ability of the agent for handling zero-shot sentences. The average navigation success rates are 84.3% for ZS1 and 85.2% for ZS2 when only half of the commands is used for training, comparable to the rate of 90.5% in a normal language setting.

2 MODEL

Our model incorporates two objectives: the first is to maximize the cumulative reward of NAV and the second is to minimize the classification cost of QA. For the former, we follow the standard reinforcement learning (RL) paradigm: the agent has to learn the action at every step from reward signals. It employs the Actor-Critic (AC) algorithm (Sutton & Barto, 1998) to learn the control policy. For the latter, we adopt the standard supervised setting of Visual QA (Antol et al., 2015): the groundtruth answer is provided by the teacher during training. The training cost is formulated as the multiclass cross entropy.

2.1 MOTIVATION

The model takes two streams of inputs: images and sentences. The key is how to model the language grounding problem. That is, the agent has to link (either implicitly or explicitly) language concepts to environment entities in order for it to correctly take an action by understanding the command in the current visual context. A straightforward idea would be to encode the sentence s with an RNN and encode the perceived image e with a CNN, after which the two encoded representations are mixed together. Specifically, let the multimodal module be $\mathbf{M}$, the action module be $\mathbf{A}$, and the prediction module be $\mathbf{P}$, this idea can be formulated as:

$$\begin{array}{ll} \text{NAV:} & \mathbf{A}\big(\mathbf{M}(\text{RNN}(s), \text{CNN}(e))\big) \\ \text{QA:} & \mathbf{P}\big(\mathbf{M}(\text{RNN}(s), \text{CNN}(e))\big) \end{array} \tag{1}$$

Hermann et al. (2017); Chaplot et al. (2017); Misra et al. (2017) all employ similar paradigms as such. In their implementations, $\mathbf{M}$ is either vector concatenation or element-wise product. In this paradigm, for any particular word in the sentence, its actual interaction with the image could happen anywhere starting from $\mathbf{M}$ all the way to the end, right before a label is output. This is due to the fact that the RNN folds the string of words into a compact embedding which then goes through the subsequent blackbox computations. Therefore, language grounding and other computational routines are entangled. Because of this, we say that this paradigm has an *implicit* language grounding strategy. Such poses a great challenge for processing a ZS2 sentence because it is almost unpredictable about how a new word learned from language prediction would perform in the complex entanglement involved. Thus a careful inspection on the grounding process is needed.

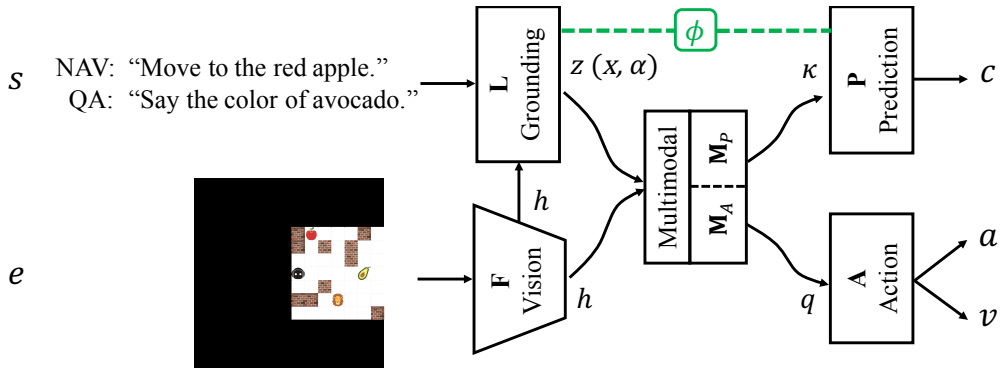


Figure 2: An overview of the model. We process e by always placing the agent at the center via zero padding. It facilitates the agent to learn navigation actions by reducing the variety of target representations. q is a compact representation that encodes the target x and the environment terrain (Appendix B). c, a, and v are the predicted answer, the navigation action, and the critic value for policy gradient, respectively.

2.2 Approach

The main idea of our approach is to *disentangle* language grounding from other computations in a model. This disentanglement makes it possible for us to explicitly define language grounding around a core function that is also used by language prediction. Specifically, both grounding and prediction are cast as concept detection problems, where each word (embedding) is treated as a detector. This opens up the possibility of transferring word meanings from the latter to the former. The overall architecture of our model is shown in Figure 2.

2.2.1 Explicit grounding

We begin with our definition of "grounding". Given a sentence s and an image representation h, we say that s is grounded in h as z if

1) h consists of M entities, and
2) $z \in \{0,1\}^M$ with each entry z_m representing a binary selection of the m-th entity of h.

Thus z is a combinatorial selection over h. An entity could be any subset of image features. For example, if h is a collection of object regions, then an entity could be one particular region. In our case, $h \in \mathbb{R}^{N \times D}$ is a spatially flattened feature cube (originally in 3D, now the 2D spatial domain collapsed into 1D for notation simplicity), where D is the number of channels and N is the number of feature vectors in the spatial domain. We adopt three definitions for an entity:

1) a feature vector at a particular image location,
2) a particular feature map along the channel dimension, and
3) a scalar feature at the intersection of a feature vector and a feature map.

Their grounding results are denoted as $\Phi_x(s,h) = x \in \{0,1\}^N$, $\Phi_\alpha(s,h) = \alpha \in \{0,1\}^D$, and $\Phi(s,h) = z \in \{0,1\}^{N \times D}$, respectively. We assume that Φ_x and Φ_α are independent:

$$\begin{aligned} \Phi(s,h) &= \Phi_x(s,h)\Phi_\alpha(s,h)^\intercal \\ z &= x\alpha^\intercal \end{aligned}$$

With this definition, we propose a new paradigm for our problem:

$$\begin{aligned} &\text{NAV:} \quad \mathbf{A}\big(\mathbf{M}_A(\Phi(s,h),h)\big) \\ &\text{QA:} \quad \mathbf{P}\big(\mathbf{M}_P(\Phi(s,h),h)\big) \end{aligned} \tag{2}$$

where $\mathbf{M}_A$, $\mathbf{M}_P$, $\mathbf{A}$, and $\mathbf{P}$ *all* rely on grounded results but *not* on other sentence representations. This *explicit* grounding strategy disentangles language grounding from subsequent computations such as obstacle detection, path planning, action making, and feature classification, which all should

be inherently language-independent routines. Also, it abstracts away irrelevant input signals from the execution of the tasks, which potentially improves the generalization to similar sensor inputs.

To make the model fully differentiable, in the following we relax the definition of grounding so that $x \in [0,1]^N$, $\alpha \in [0,1]^D$, and $z \in [0,1]^{N \times D}$. The attention map x is responsible for image spatial attention. The channel mask α is responsible for selecting image feature maps, and is thus assumed to be independent of the specific h, *i.e.*, $\Phi_\alpha(s,h) = \Phi_\alpha(s)$. The attention cube z is always an outer product of the two. Similar attention mechanism has only been explored partially so far. For example, Xu et al. (2015) employed only x for image captioning. Chaplot et al. (2017) employed only α for 3D navigation. Nevertheless, none attempted to establish a link between language grounding and prediction, which we will do next.

2.2.2 CONCEPT DETECTION

With language grounding being disentangled, now we relate it to language prediction. This relation is a common *concept detection* function. We assume that every word in a vocabulary, as a concept, is detectable against entities of type (1) defined in section 2.2.1. For meaningful detection of spatial-relation words that are irrelevant to image content, we incorporate parametric feature maps into h to learn spatial features. The concept detection operates by sliding over the spatial domain of the feature cube h, which can be written as a function ϕ:

$$\phi : h, \alpha, u \mapsto \chi$$

where $\chi \in \mathbb{R}^N$ is a detection score map and u is a word embedding vector. This function scores the embedding u against each feature vector of h, modulated by α that selects which feature maps to use for the scoring. Intuitively, each score on χ indicates the detection response of the feature vector in that location. A higher score represents a higher detection response.

While there are many potential forms for ϕ, we implement it as

$$\phi(h, \alpha, u) = h \cdot (\alpha \circ u) \tag{3}$$

where $\circ$ is the element-wise product. In order to do so, we have word embedding $u \in \mathbb{R}^D$ where D is equal to the number of channels of h.

2.2.3 PREDICTION BY CONCEPT DETECTION

Based on the detection function, $\mathbf{M}_P$ outputs a score vector $\kappa \in \mathbb{R}^K$ over the entire lexicon

$$\begin{aligned} \kappa_k &= \mathbf{M}_P(\Phi(s,h), h) \\ &= \Phi_x(s,h)^\intercal \phi(h, \Phi_\alpha(s), u_k) \\ &= x^\intercal \phi(h, \alpha, u_k) \\ &= x^\intercal \chi_k \end{aligned} \tag{4}$$

where u_k is the k-th entry of the word embedding table. The score κ_k is the result of weighting the scores on the map χ_k with x. It represents the correctness of the k-th lexical entry as the answer to the question s. To predict an answer

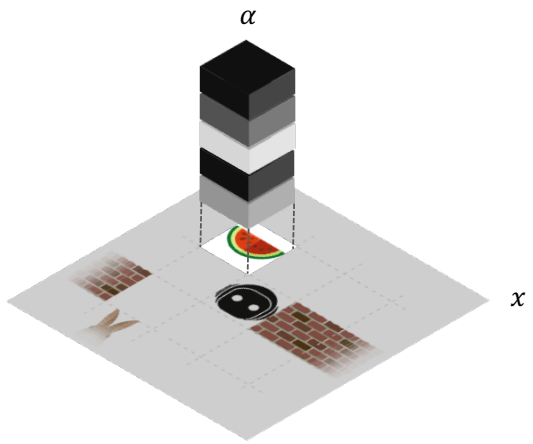
$$\mathbf{P}(\kappa) = \operatorname*{argmax}_k \big(\text{softmax}(\kappa)\big)$$

Note that the role of α in the prediction is to select which feature maps are relevant to the question s. Otherwise it would be confusing for the agent about what to predict (*e.g.*, whether to predict a color or an object name). By using α, we expect that different feature maps encode different image attributes (see an example in the caption of Figure 3). More analysis on α is done in section 4.6.

2.2.4 GROUNDING BY CONCEPT DETECTION

To define Φ, we define Φ_α and Φ_x separately given their independence. By assumption, the channel mask α is meant to be determined solely from s, namely, which features to use should only depend on the sentence itself, but not on the feature cube h. Thus it is computed as

$$\Phi_\alpha(s) = \text{MLP}\big(\text{RNN}(s)\big)$$



"What is the color of the object in the northeast?"

Figure 3: Illustration of the attention cube $z = x\alpha^\intercal$, where x attends to image regions and α selects feature maps. In this example, x is computed from "northeast". In order for the agent to correctly answer "red" (color) instead of "watermelon" (object name), α has to be computed from the sentence pattern "What ... color ...?".

where the RNN returns an average state of processing s, followed by an MLP with the Sigmoid activation. Basically, we assume no word-image interaction during the computation of α.

Then we want Φ_x to be built on the detection function ϕ. One can expect that Φ_x computes a series of score maps χ of individual words and merges them into x. Suppose that s consists of L words $\{w_l\}$ with $w_l = u_k$ for some k. Let $\tau(s)$ be a sequence of indices $\{l_i\}$ with $0 \leqslant l_i < L$. This sequence function τ decides which words of the sentence are selected and organized in what order. We define $\Phi_x(s, h)$ as

$$\begin{aligned} \Phi_x(s,h) &= \Upsilon\big(\phi(h, \mathbf{1}, w_{l_1}), \ldots, \phi(h, \mathbf{1}, w_{l_i}), \ldots, \phi(h, \mathbf{1}, w_{l_I})\big) \\ &= \Upsilon(\chi_{l_1}, \ldots, \chi_{l_i}, \ldots, \chi_{l_I}) \end{aligned} \tag{5}$$

where $\mathbf{1} \in \{0,1\}^D$ is a vector of ones, meaning that it selects all the feature maps for detecting w_{l_i}. Υ is an aggregation function that combines the sequence of score maps χ_{l_i} of individual words. As such, ϕ makes it possible to transfer new words from Eq. 4 to Eq. 5 during test time.

If we were provided with an oracle that outputs a parsing tree for any sentence, we could set τ and Υ according to the tree semantics. Neural Module Networks (NMNs) (Andreas et al., 2016a;b; Hu et al., 2017) rely on such tree for language grounding. They generate a network of modules where each module corresponds to a tree node. However, labeled trees are needed for training. Below we propose to learn τ and Υ based on word attention (Bahdanau et al., 2015) to bypass the need of labeled structured data.

We start with feeding a sentence $s = \{w_l\}$ of length L to a Bidirectional RNN (Schuster & Paliwal, 1997). It outputs a compact sentence embedding m and a sequence of L word context vectors $\overline{w}_l$. Each $\overline{w}_l$ summarizes the sentential pattern around the word. We then employ a meta controller called *interpreter*. For the i-th interpretation step, the interpreter computes the word attention as:

$$\tau^* \quad \begin{cases} \text{Word attention:} & o_l^i \propto \exp\left[S_{\cos}(p^{i-1}, \overline{w}_l)\right] \\ \text{Attended context:} & \overline{w}^i = \sum_l o_l^i \overline{w}_l \\ \text{Attended word:} & s^i = \sum_l o_l^i w_l \\ \text{Interpreter state:} & p^i = \text{GRU}(p^{i-1}, \overline{w}^i) \end{cases} \tag{6}$$

where $S_{\cos}$ is cosine similarity and GRU is the Gated Recurrent Unit (Cho et al., 2014). We use τ^* to represent an approximation of τ via soft word attention. We set p^0 to the sentence embedding m. After this, the attended word s^i is fed to the detection function ϕ. The interpreter aggregates its

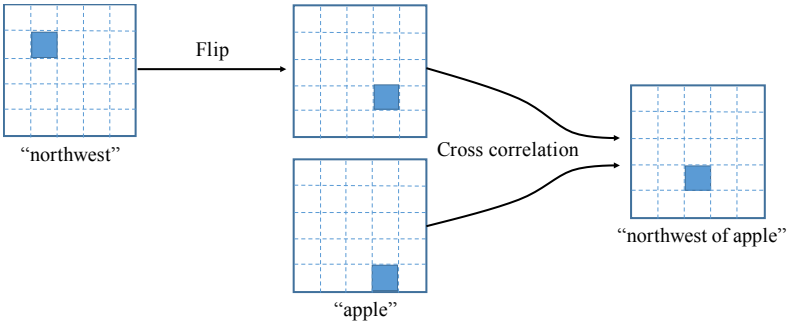


Figure 4: A symbolic example of the 2D convolution for attention map transform. 2D convolution can be decomposed into two steps: flipping and cross correlation. The attention map of "northwest" is treated as the kernel to translate that of "apple". Note that in practice, the attention is continuous and noisy, and the interpreter has to learn to find out the words (if any) to do this convolution.

score map by:

$$\Upsilon \quad \begin{cases} \text{Detection:} & y' = \text{softmax}\big(\phi(h, \mathbf{1}, s^i)\big) \\ \text{Map transform:} & x^i = y' * y^{i-1} \\ \text{Map update gate:} & \rho^i = \sigma(W_\rho p^i + b_\rho) \\ \text{Map update:} & y^i = \rho^i x^i + (1 - \rho^i) y^{i-1} \end{cases} \tag{7}$$

where $*$ denotes 2D convolution, σ is Sigmoid, and ρ^i is a scalar. W_ρ and b_ρ are parameters to be learned. Finally, the interpreter outputs x^I as $\Phi_x(s, h)$, where I is the predefined maximum step.

Note that in the above we formulate the map transform as a 2D convolution. This operation enables the agent to reason about spatial relations. Recall that each attention map x is ego-centric. When the agent needs to attend to a region specified by a spatial relation referring to an object, it can translate the object attention with the attention map of the spatial-relation word (Figure 4). For this reason, we set y^0 as a one-hot map where the map center is one, to represent the identity translation.

3 Related work

Our XWORLD environment is similar to the 2D block world MaseBase (Sukhbaatar et al., 2016). However, we emphasize the problem of grounded language acquisition and generalization, while they do not. There have been several 3D environments such as ViZDoom (Kempka et al., 2016), DeepMind Lab (Beattie et al., 2016), and Malmo (Johnson et al., 2016). Still, they are intended for visual perception and control, with less or no language.

Our problem setting draws inspirations from the AI roadmap delineated by Mikolov et al. (2015). Like theirs, we have a teacher in the environment that assigns tasks and rewards to the agent, potentially with a curriculum. Unlike their proposal of using only the linguistic channel, we have multiple perceptual modalities, the fusion of which is believed to be the basis of meaning (Kiela et al., 2016).

Contemporary to our work, several end-to-end systems also address language grounding problems in a simulated world with deep RL. Misra et al. (2017) maps instructions and visual observations to actions of manipulating blocks on a 2D plane. Hermann et al. (2017); Chaplot et al. (2017) learn to navigate in 3D under instructions, and both evaluate ZS1 generalization abilities. Despite short of the vision challenges in their environments, we have a much larger space of zero-shot sentences and additionally require ZS2 generalization which is in fact a transfer learning (Pan & Yang, 2010) problem.

Other recent work (Andreas et al., 2017; Oh et al., 2017) on zero-shot multitask learning treats language tokens as (parsed) labels for identifying skills. Their zero-shot settings are not intended for language understanding.

Our learning problem is related to some recent work on grounding language in images and videos (Yu & Siskind, 2013; Rohrbach et al., 2016; Gao et al., 2016). The NAV task is relevant

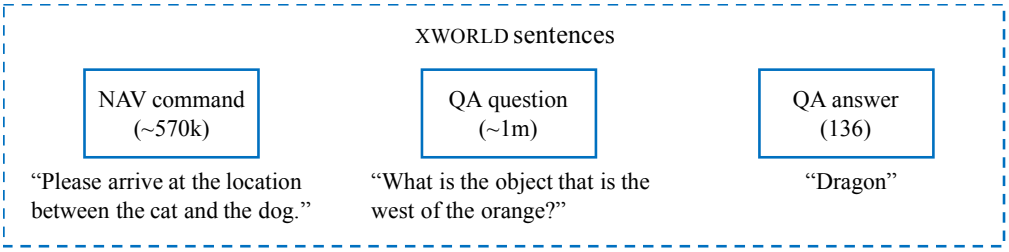
to robotics navigation under language command (Chen & Mooney, 2011; Tellex et al., 2011; Barrett et al., 2015). The QA task is relevant to image question answering (VQA) (Antol et al., 2015; Ren et al., 2015; Gao et al., 2015; Lu et al., 2016; Yang et al., 2016). The interactive setting of learning to accomplish tasks is similar to that of learning to play video games from pixels (Mnih et al., 2015).

4 EXPERIMENTS

We design a variety of experiments to evaluate the agent's language acquisition and generalization ability. Our model is first compared with several methods to demonstrate the challenges in XWORLD. We then evaluate the agent's language generalization ability in a zero-shot setting. We also visualize and analyze the intermediate outputs of the model and demonstrate interpretable results. Finally, we conclude this section with some preliminary thoughts on how to scale our model to a 3D world.

4.1 GENERAL SETUP

For all the experiments, both the sentences and the environment change from session to session, and from training to testing. The sentences are drawn confirming to the teacher's grammar. There are three types of language data: NAV command, QA question, and QA answer, which are illustrated in the figure below:



In total, there are ∼570k NAV commands, ∼1m QA questions, and 136 QA answers (all the content words including 119 objects, 9 spatial relations, and 8 colors). The environment configurations are randomly generated from octillions of possibilities of a 7×7 map, confirming to some high-level specifications such as the numbers of objects and wall blocks. Our model is evaluated on four types of navigation commands:

`nav_obj`:	Navigate to an object.
`nav_col_obj`:	Navigate to an object with a specific color.
`nav_nr_obj`:	Navigate to a location near an object.
`nav_bw_obj`:	Navigate to a location between two objects.

For the detailed experiment settings, we refer the reader to Appendix A.

4.2 COMPARISON METHODS

The four comparison methods are described below:

ContextualAttention **[CA]** A variant of our model that replaces the interpreter with a contextual attention model in **L**. The attention model uses a gated RNN to convert a sentence to a filter which is then convolved with the feature cube h to obtain the attention map x. The filter covers the 3×3 neighborhood of each feature vector in the spatial domain. The channel mask α is still computed as before. The rest of the model is unchanged.

StackedAttentionNet **[SAN]** An adaptation of Yang et al. (2016) which was originally proposed for VQA. We replace our interpreter with their stacked attention model in **L** to obtain the attention map x. The channel mask α is still computed as before. Instead of employing a pretrained CNN as they did, we train a CNN from scratch to accommodate to the XWORLD environment. The CNN is configured as the one employed by our model. The rest of our model is unchanged.

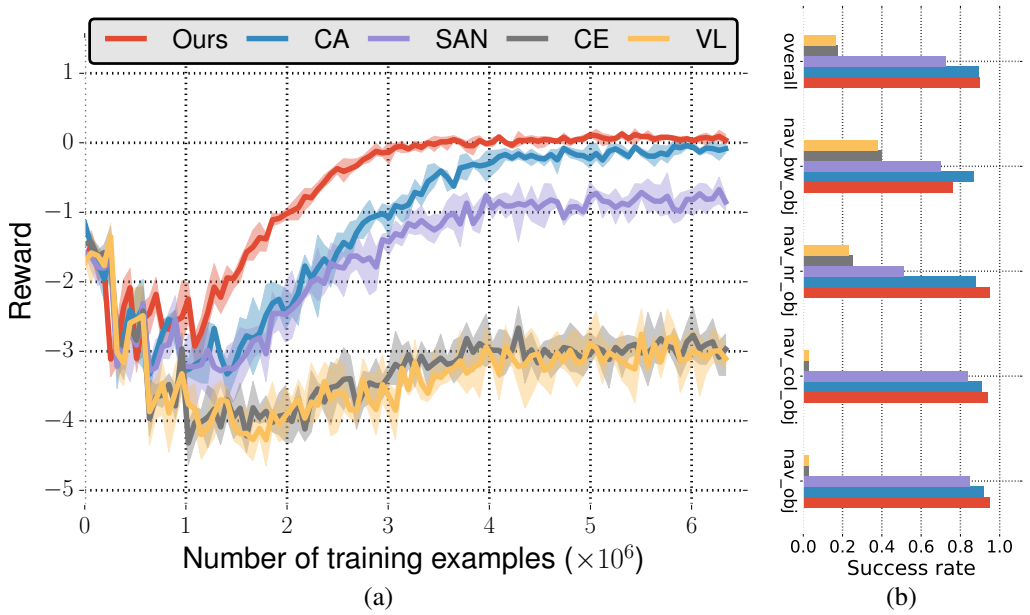


Figure 5: Basic evaluation in the normal language setting. (a) Training reward curves. The shown reward is the accumulated discounted reward per session, averaged every 8k training examples. The shaded area of each curve denotes the variance among 4 random initializations of the model parameters. (b) Average success rates of NAV during testing.

VIS-LSTM **[VL]** An adaptation of Ren et al. (2015) which was originally proposed for VQA. We flatten h and project it to the word embedding space $\mathbb{R}^D$. Then it is appended to the input sentence s as the first word. The augmented sentence goes through an LSTM whose last state is used for both NAV and QA (Figure 12 Appendix C).

ConcatEmbed **[CE]** An adaptation of Mao et al. (2015) which was originally proposed for image captioning. It instantiates **L** as a vanilla LSTM which outputs a sentence embedding. Then h is projected and concatenated with the embedding. The concatenated vector is used for both NAV and QA (Figure 13 Appendix C). This concatenation mechanism is also employed by Hermann et al. (2017); Misra et al. (2017).

In the following experiments, we train all five approaches (four comparison methods and our model) with a small learning rate of 1×10^{-5} and a batch size of 16, for a maximum of 200k mini-batches. Additional training details are described in Appendix B. After training, we test each approach on 50k sessions of navigation. We compute the average success rate of navigation where a success is defined as reaching the correct location before the time out of a session.

4.3 Basic Evaluation

In this experiment, the training and testing NAV commands are sampled from the same distribution over the entire space [1]. We call it the normal language setting.

The training reward curves and testing results are shown in Figure 5. **VL** and **CE** have quite low rewards without convergences. These two approaches do not use the spatial attention x, and thus always attend to an entire image with no focus. The region of a target location is tiny compared to the whole ego-centric image (a ratio of $1 : (7 \times 2 - 1)^2 = 1 : 169$). It is possible that this difficulty drives the models towards overfitting QA without learning useful representations for NAV. Both **CA** and **SAN** obtain rewards and success rates slightly worse than ours. This suggests that in a normal language setting, existing attention models are able to handle previously seen sentences. However,

[1] Although some test command as a whole might not be seen in training (*i.e.*, zero-shot) due to sampling, all the content words and their combinations (totaling a dozen of thousands) are highly likely to be exhausted by training. Thus we consider this experiment as a normal setting, compared to the zero-shot setting in section 4.4.

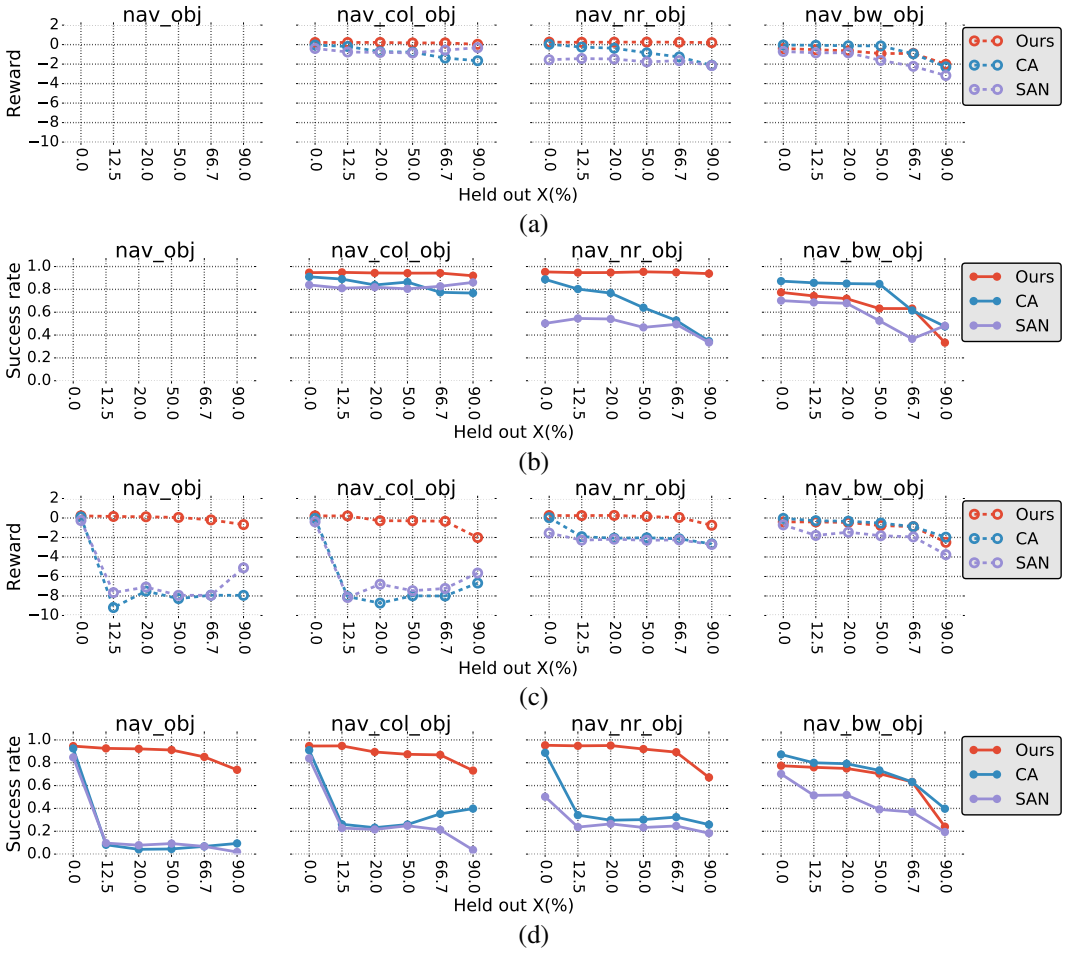


Figure 6: Test results of language generalization with a varying held-out portion of $X\%$, where $X = 0$ represents the basic evaluation in section 4.3. (a) and (b): Average reward and success rate per session for **ZS1**. (c) and (d): Average reward and success rate per session for **ZS2**. (The plots of `nav_obj` in (a) and (b) are empty because there is no ZS1 sentence of this type by definition.)

their language generalization abilities, especially on ZS2 sentences, are usually very weak. We will demonstrate this in the next section.

4.4 Language generalization

Our more important question is whether the agent has the ability of interpreting zero-shot sentences. For comparison, we use **CA** and **SAN** from the previous section, as they achieved good performance in the normal language setting. For a zero-shot setting, we setup two language conditions:

NewWordCombination **[ZS1]** Some word pairs are excluded from both the NAV commands and the QA questions. We consider three types of unordered combinations of the content words: (*object*, *spatial relation*), (*object*, *color*), and (*object*, *object*). We randomly hold out $X\%$ of word pairs during training and only use them in NAV testing.

NewWord **[ZS2]** Some single words are excluded from both the NAV commands and the QA questions but can appear in the QA answers. We randomly hold out $X\%$ of the object words during training and only use them in NAV testing.

We vary the value of $X\%$ (12.5%, 20.0%, 50.0%, 66.7%, and 90.0%) in both conditions to test how sensitive the generalization is to the amount of held-out data. The results are shown in Figure 6.

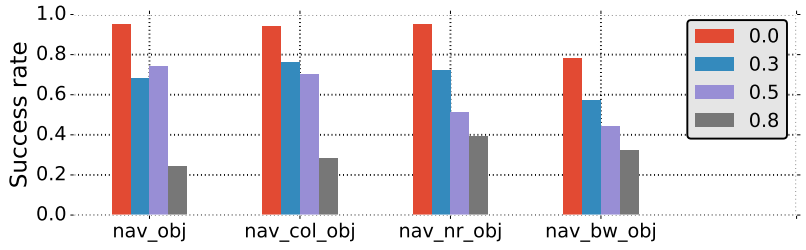


Figure 7: The navigation success rates of our model with the noise levels 0.0, 0.3, 0.5, and 0.8.

We draw two conclusions from the results. First, ZS1 sentences are much easier to interpret than ZS2 sentences. Neural networks seem to inherently have this capability to some extent. This is consistent with what have been observed in some previous work (Hermann et al., 2017; Chaplot et al., 2017) that addresses the generalization ability on new word combinations. Second, ZS2 sentences are difficult for **CA** and **SAN**. Even with a held-out portion as small as 12.5%, their success rates drop at most 80% compared to the normal language setting. In contrast, our model tends to maintain the same success rate until $X = 90.0$. Impressively, it achieves an average rate of 59.5% even when the number of new object words is 9 times that of seen object words!

Interestingly, we notice that `nav_bw_obj` is an outlier command type in which **CA** it is much less sensitive to the increase of X in the **ZS2** condition. Deep analysis reveals that for `nav_bw_obj`, **CA** learns to cheat by looking for the image region that contains the special pattern of object pairs in the image without having to recognize them first. This suggests that neural networks tend to exploit data usually in an unexpected way to achieve tasks if no constraints are imposed (Kottur et al., 2017).

To sum up, our model exhibits a strong generalization ability on both ZS1 and ZS2 sentences, the latter of which pose a great challenge for traditional language grounding models. Although we use a particular environment setup for evaluation in this work, the promising results imply the potential of scaling to an even larger vocabulary and grammar with a much larger language space.

4.5 Robustness to noise

We add noise to the test NAV commands to evaluate the robustness of our model. Given a sentence of length L, we insert $\lfloor \eta L \rfloor$ noise words in it at random locations, where η is the noise level selected as 0.3, 0.5, or 0.8. Each noise word has an embedding vector of which each entry is independently sampled from the standard normal distribution. Note that these noise words do not appear in training; they only appear in testing. We directly use our trained model from section 4.3 for testing. The navigation success rates under noise-added commands are shown in Figure 7. It can be seen that the model correctly executed the commands for a reasonable portion of test sessions despite the noise. This shows its potential of scaling to test scenarios where the commands are possibly not well formed.

4.6 Visualization and analysis

In this section, we visualize some intermediate results of our trained model.

Channel mask α. We inspect the channel mask $\Phi_\alpha(s)$ which allows the model to select certain feature maps from a feature cube h and predict an answer to the question s. We randomly sample 10k QA questions and compute α for each of them using the grounding module **L**. Because in our experiments each answer must be a content word, we can divide the 10k questions into 135 groups[2], where each group corresponds to a distinct content word. Then we compute an Euclidean distance matrix D where entry $D_{i,j}$ is the average distance between the α of a question from the i-th group and that from the j-th group (Figure 8). It is clear that there are three topics (object, color, and spatial relation) in the matrix. Distances within the same topic are much smaller than across topics. This demonstrates that with the channel mask, the model is able to look at different subsets of features

[2]The word "orange" is both a color word and an object word, which is why the number of groups is one less than 119 (object) + 9 (spatial relation) + 8 (color) = 136.

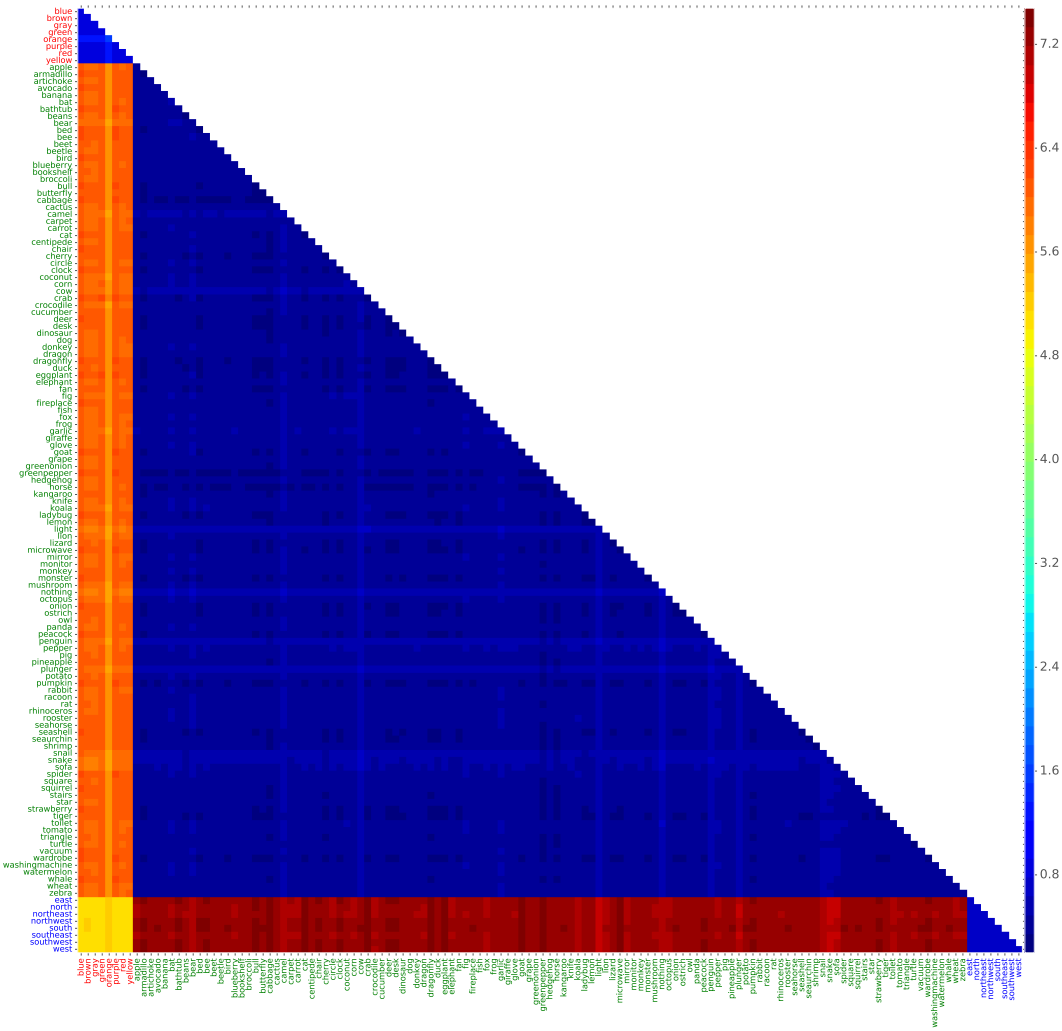


Figure 8: Euclidean distance matrix of the 135 question groups where each group corresponds to a labeled word. Each row (column) represents the sampled questions that have the labeled word as the answer. A matrix entry indicates the empirical expectation of the distance between the channel masks of the two question groups. The labels are arranged into three groups (*i.e.*, color, object, and spatial relation). Smaller distances indicate more similar channel masks. (Zoom in on the screen for a better view.)

for questions of different topics, while using the same subset of features for questions of the same topic. It also implies that the feature cube h is learned to organize feature maps according to image attributes.

Word context & attention. To intuitively demonstrate how the interpreter works, we visualize the word context vectors $\overline{w}_l$ in Eq. 6 for a total of 20k word locations l (10k from QA questions and the other 10k from NAV commands). Each word context vector is projected down to a space of 50 dimensions using PCA (Jolliffe, 1986), after which we use t-SNE (van der Maaten & Hinton, 2008; Ulyanov, 2016) to further reduce the dimensionality to 2. The t-SNE method uses a perplexity of 100 and a learning rate of 200, and runs for 1k iterations. The visualization of the 20k points is shown in Figure 9 (a). Recall that in Eq. 6 the word attention is computed by comparing the interpreter state p^{i-1} with word context vectors $\overline{w}_l$. In order to select content words to generate meaningful score maps via ϕ, the interpreter is expected to differentiate them from the remaining grammatical words based on the contexts. This hypothesis is supported by the above visualization in which the context vectors of content words (in blue, green, and red) and the those of grammatical words (in

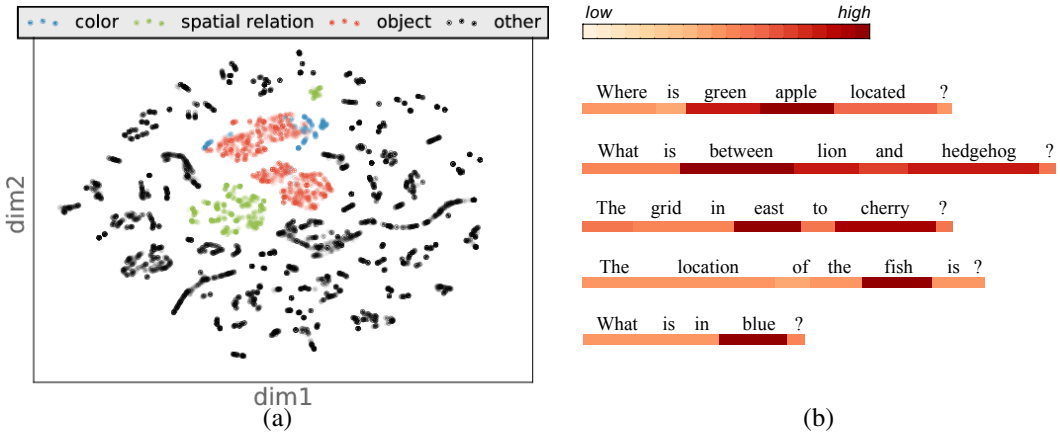


Figure 9: Visualizations of the computation of word attention. (a) The word context vector $\overline{w}_l$, and (b) the word attentions o_l of several example questions. Each attention vector, represented by a color bar, shows the attention accumulated over I interpretation steps.

black) are mostly separated. Figure 9 (b) shows some example questions whose word attentions are computed from the word context vectors. It can be seen that the content words can be successfully selected by the interpreter.

Attention map x. Finally, we visualize the computation of the attention map $\Phi_x(s, h)$. In each example, the intermediate results (x^i and ρ^i in Eq. 7) of three interpretation steps are shown. (Figure 10). The environment maps perfectly detect all possible entities and thus provide a guide for the agent to navigate successfully without hitting walls or confounding targets.

4.7 How does it adapt to 3D?

We discuss the possibility of adapting our model to build an agent with similar language abilities in a 3D world (*e.g.*, Beattie et al. (2016); Johnson et al. (2016)). This is our goal for the future, but here we would like to share some preliminary thoughts. Generally speaking, a 3D environment will mainly pose a greater challenge for visual perception but hardly for language understanding. The key element of our model is the attention cube z that is intended for an explicit grounding of language, which includes the channel mask Φ_α and the interpreter Φ_x. The channel mask only depends on the sentence itself, and thus is expected to work regardless of the world dimensionality. The interpreter depends on a sequence of score maps χ whose computation ϕ for now is defined as multiplying a word embedding with the feature cube (Eq. 3). A more sophisticated definition of ϕ will be needed to detect objects in a 3D environment. Additionally, the interpreter models spatial attention transform as a 2D convolution (Eq. 7). This assumption is no longer valid in 3D, and a better transformation method that accounts for perspective distortion is required. Lastly, the surrounding environment is only partially observable to a 3D agent. Working memory such as LSTM added to the action module **A** will be important for navigation in such case. Despite these changes to be made, we believe that the general explicit grounding strategy and the common detection function shared by language grounding and prediction have shed some light on the adaptation.

5 Conclusion

We have presented an end-to-end model for a virtual agent to acquire language from a 2D environment in an interactive and grounded manner, through the visual and linguistic perception channels. After learning, the agent is able to extrapolate to interpret zero-shot sentences that contain new word combinations or even new words. Such generalization ability is supported by an explicit grounding strategy that disentangles language grounding from subsequent language-independent computations. It also depends on sharing a detection function between language grounding and language prediction as the core computation. This function enables word meanings to transfer from prediction to grounding during test time. Promising language acquisition and generalization results have been

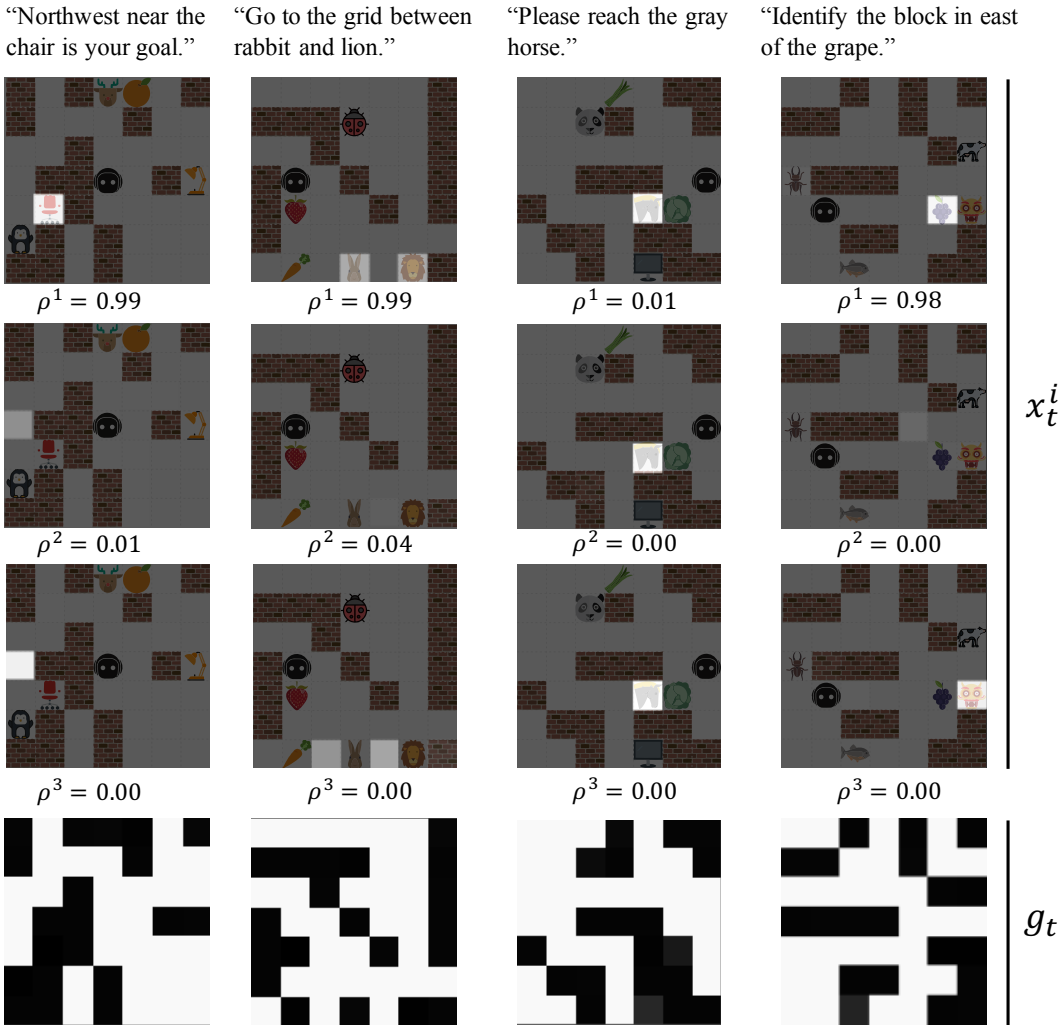


Figure 10: Example intermediate outputs of the interpreter. Each column shows three interpretation steps from top to bottom. Each step shows the current attention map x^i overlaid on the environment image e. The update gate ρ^i decides whether x^i is kept (1) or discarded (0) for the next step. The last attention map x^3 is the final output of the interpreter. The fourth row shows the learned environment maps indicating the environment terrain which helps the agent navigate (g in Appendix B). For a better visualization, the image regions corresponding to the zero paddings of the original ego-centric images are cropped.

shown in the 2D XWORLD environment. We hope that this work can shed some light on acquiring and generalizing language in a similar way in 3D environments.

REFERENCES

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, 2016a.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *ACL*, 2016b.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *ICML*, 2017.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Daniel Paul Barrett, Scott Alan Bronikowski, Haonan Yu, and Jeffrey Mark Siskind. Robot language learning, generation, and comprehension. *arXiv preprint arXiv:1508.06161*, 2015.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *CoRR*, abs/1612.03801, 2016.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

Jerome Bruner. Child's talk: Learning to use language. *Child Language Teaching and Therapy*, 1 (1), 1985.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *ACL Workshop on Language Grounding for Robotics*, 2017.

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

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Noam Chomsky. Linguistics and cognitive science: Problems and mysteries. In *The Chomskyan Turn*. 1991.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(Jul), 2011.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question. In *NIPS*, 2015.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Y Chai. Physical causality of action verbs in grounded language understanding. In *ACL*, 2016.

Stevan Harnad. The symbol grounding problem. *Physica D*, 42, 1990.

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

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. *arXiv preprint arXiv:1704.05526*, 2017.

M. Johnson, K. Hofmann, T. Hutton, and D. Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, 2016.

I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

Thomas K. Landauer and Susan T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104, 1997.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, 2016.

Douwe Kiela, Luana Bulat, Anita L. Vero, and Stephen Clark. Virtual embodiment: A scalable long-term strategy for artificial intelligence research. In *NIPS Workshop*, 2016.

Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. In *EMNLP*, 2017.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, 2016.

Junhua Mao, Xu Wei, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L Yuille. Learning like a child: Fast novel visual concept learning from sentence descriptions of images. In *ICCV*, 2015.

Tomas Mikolov, Armand Joulin, and Marco Baroni. A roadmap towards machine intelligence. *arXiv preprint arXiv:1511.08130*, 2015.

Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *EMNLP*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 2015.

Junhyuk Oh, Satinder P. Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *ICML*, 2017.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10), October 2010.

Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In *NIPS*, 2015.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. Grounding of textual phrases in images by reconstruction. In *ECCV*, 2016.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2016.

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11), 1997.

Burrhus Frederic Skinner. *Verbal behavior*. Copley Publishing Group, 1957.

Sainbayar Sukhbaatar, Arthur Szlam, Gabriel Synnaeve, Soumith Chintala, and Rob Fergus. Mazebase: A sandbox for learning from games. *arXiv preprint arXiv:1511.07401*, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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

Michael Tomasello. *Constructing a Language: A Usage-Based Theory of Language Acquisition*. Harvard University Press, 2003.

Dmitry Ulyanov. Muticore-tsne. `https://github.com/DmitryUlyanov/Muticore-TSNE`, 2016.

L.J.P van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9, 2008.

Terry Winograd. Understanding natural language. *Cognitive Psychology*, 3(1), 1972.

William A. Woods. Meaning and links. *AI Magazine*, 28(4), 2007.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *CVPR*, 2016.

Haonan Yu and Jeffrey Mark Siskind. Grounded language learning from video described with sentences. In *ACL*, 2013.

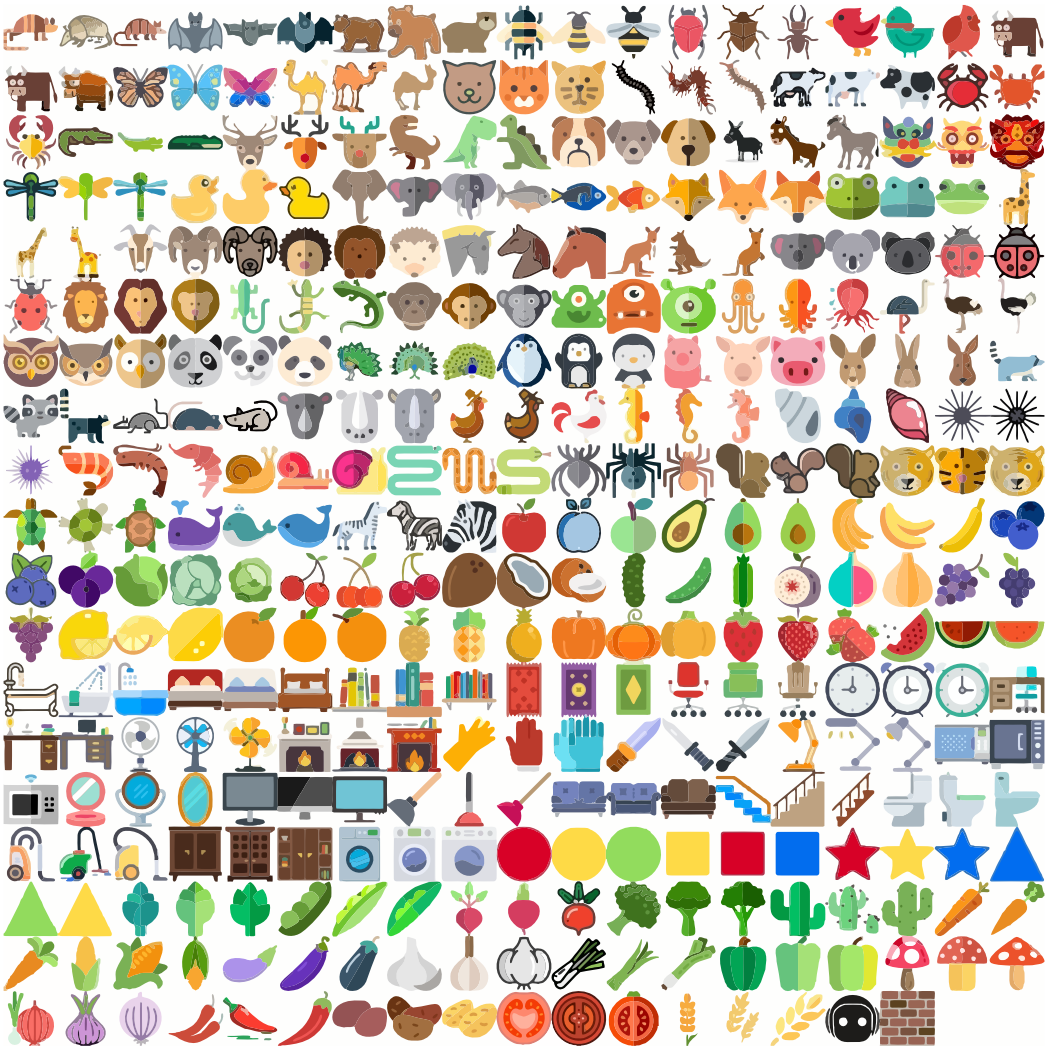


Figure 11: All the 357 object instances plus the agent (second-to-last) and the wall block (last).

APPENDICES

A XWORLD SETUP

The XWORLD environment is configured with 7×7 grid maps. On each map, the open space for the agent has a size ranging from 3×3 to 7×7. Smaller open spaces are setup for curriculum learning (Appendix D), but not for testing. To keep the size of the environment image fixed, we pad the map with wall blocks if needed. The agent has four navigation actions in total: `left`, `right`, `up`, and `down`. In each session,

I The maximum number of time steps is a constant computed as four times the map dimension ($7 \times 4 = 28$). That is, the agent only has 28 steps to reach a target before time out.

II The number of objects on the map ranges from 1 to 5.

III The number of wall blocks on the map ranges from 0 to 15.

IV The positive reward when the agent reaches the correct location is 1.0.

V The negative rewards for hitting walls and for stepping on non-target objects are -0.2 and -1.0, respectively.

Object	Spatial relation	Color	Other
apple, armadillo, artichoke, avocado, banana, bat, bathtub, beans, bear, bed, bee, beet, beetle, bird, blueberry, bookshelf, broccoli, bull, butterfly, cabbage, cactus, camel, carpet, carrot, cat, centipede, chair, cherry, circle, clock, coconut, corn, cow, crab, crocodile, cucumber, deer, desk, dinosaur, dog, donkey, dragon, dragonfly, duck, eggplant, elephant, fan, fig, fireplace, fish, fox, frog, garlic, giraffe, glove, goat, grape, greenonion, greenpepper, hedgehog, horse, kangaroo, knife, koala, ladybug, lemon, light, lion, lizard, microwave, mirror, monitor, monkey, monster, mushroom, octopus, onion, orange, ostrich, owl, panda, peacock, penguin, pepper, pig, pineapple, plunger, potato, pumpkin, rabbit, racoon, rat, rhinoceros, rooster, seahorse, seashell, seaurchin, shrimp, snail, snake, sofa, spider, square, squirrel, stairs, star, strawberry, tiger, toilet, tomato, triangle, turtle, vacuum, wardrobe, washingmachine, watermelon, whale, wheat, zebra	between, east, north, northeast, northwest, south, southeast, southwest, west	blue, brown, gray, green, orange, purple, red, yellow	?, ., and, block, by, can, color, could, destination, direction, does, find, go, goal, grid, have, identify, in, is, locate, located, location, me, move, name, navigate, near, nothing, object, of, on, one, please, property, reach, say, side, target, tell, the, thing, three, to, two, what, where, which, will, you, your

Table 1: The teacher's lexicon.

VI The time penalty of each step is -0.1.

The teacher has a vocabulary size of 185. There are 9 spatial relations, 8 colors, 119 distinct object classes, and 50 grammatical words. Every object class contains 3 different instances. All object instances are shown in Figure 11. Every time the environment is reset, a number of object classes are randomly sampled and an object instance is randomly sampled for each class. There are in total 16 types of sentences the teacher can speak, including 4 types of NAV commands and 12 types of QA questions. Each sentence type has several non-recursive templates, and corresponds to a concrete type of tasks the agent must learn to accomplish. In total there are 1,639,015 distinct sentences with 567,579 for NAV and 1,071,436 for QA. The sentence length varies between 2 and 13.

The object, spatial-relation, and color words of the teacher's language are listed in Table 1. These are the content words that can be grounded in XWORLD. All the others are grammatical words which serve to support sentence structures. Note that the differentiation between the content and the grammatical words is automatically learned by the agent according to the tasks. Every word is equally represented by an entry in the word embedding table.

The sentence types that the teacher can speak are listed in Table 2. Each type has a triggering condition about when the teacher says that type of sentences. Besides the conditions shown, an extra condition for NAV commands is that the target must be reachable from the current agent location. An extra condition for color-related questions is that the object color must be one of the eight defined colors. If at any time step there are multiple types triggered, we randomly sample one for NAV and another for QA. After a sentence type is sampled, we generate a sentence according to the corresponding sentence templates.

B IMPLEMENTATION DETAILS

The environment image e is a 156×156 ego-centric RGB image. The CNN in $\mathbf{F}$ has four convolutional layers: $(3, 3, 64), (2, 2, 64), (2, 2, 256), (1, 1, 256)$, where (x, y, z) represents a layer configuration of z kernels of size x applied at stride width y. All the four layers are ReLU activated. To enable the agent to reason about spatial-relation words (*e.g.*, "north"), we append an additional parametric cube to the convolutional output to obtain h. This parametric cube has the same number of channels with the CNN output, and it is initialized with zero mean and zero standard deviation.

The word embedding table is initialized with zero mean and unit standard deviation. All the gated RNNs (including the bidirectional RNN) in $\mathbf{L}$ have 128 units. All the layers in $\mathbf{L}$, unless otherwise stated, use Tanh as the activation function.

Sentence Type	Example	Triggering Condition
nav_obj	"Please go to the apple."	[C0] Beginning of a session. & [C1] The reference object has a unique name in the environment.
nav_col_obj	"Could you please move to the red apple?"	[C0] & [C2] There are multiple objects that either have the same name but different colors, or have different names but the same color.
nav_nr_obj	"The north of the apple is your destination."	[C0] & [C1]
nav_bw_obj	"Navigate to the grid between apple and banana please."	[C0] & [C3] Both reference objects have unique names in the environment and are separated by one block.
rec_col2obj	"What is the red object?"	[C4] There is only one object that has the specified color.
rec_obj2col	"What is the color of the apple?"	[C1]
rec_loc2obj	"Please tell the name of the object in the south."	[C5] The agent is near the reference object.
rec_obj2loc	"What is the location of the apple?"	[C1] & [C5]
rec_loc2col	"What color does the object in the east have?"	[C5]
rec_col2loc	"Where is the red object located?"	[C4] & [C5]
rec_loc_obj2obj	"Identify the object which is in the east of the apple."	[C1] & [C6] There is an object near the reference object.
rec_loc_obj2col	"What is the color of the east to the apple?"	[C1] & [C6]
rec_col_obj2loc	"Where is the red apple?"	[C2] & [C5]
rec_bw_obj2obj	"What is the object between apple and banana?"	[C7] Both reference objects have unique names in the environment and are separated by a block.
rec_bw_obj2loc	"Where is the object between apple and banana?"	[C7] & [C8] The agent is near the block which is between the two reference objects.
rec_bw_obj2col	"What is the color of the object between apple and banana?"	[C7]

Table 2: Descriptions of all the 16 sentence types of the teacher.

For NAV, x is used as the target to reach on the image plane. However, knowing this alone does not suffice. The agent also has to be aware of walls and possibly confounding targets (other objects) in the environment. Toward this end, $\mathbf{M}_A$ further computes an environment map $g = \sigma(hf)$ where $f \in \mathbb{R}^D$ is a parameter vector to be learned and σ is Sigmoid. We expect that g detects any blocks informative for navigation. Note that g is unrelated to the specific command; it solely depends on the current environment. After stacking x and g together, $\mathbf{M}_A$ feeds them to another CNN followed by an MLP to obtain q. The CNN has two convolutional layers $(3, 1, 64)$ and $(3, 1, 4)$, both with paddings of 1. It is followed by a three-layer MLP where each layer has 512 units and is ReLU activated.

The action module $\mathbf{A}$ contains a two-layer MLP of which the first layer has 512 ReLU activated units and the second layer is a Softmax whose output dimension is equal to the number of actions.

We use Adagrad (Duchi et al., 2011) with a learning rate of 10^{-5} for Stochastic Gradient Descent (SGD). The reward discount factor is set to 0.99. All the parameters have a default weight decay of $10^{-4} \times 16$. For each layer, its parameters have zero mean and a standard deviation of $1 \,/\, \sqrt{K}$, where K is the number of parameters of that layer. The agent has 1m exploration steps in total, and the exploration rate λ decreases linearly from 1 to 0.1. At each time step, the agent takes an action a with a probability of

$$\lambda \cdot \frac{1}{4} + (1 - \lambda) \cdot \pi(a)$$

where π is the current policy. We set the maximum interpretation step $I = 3$. The whole model is trained end to end.

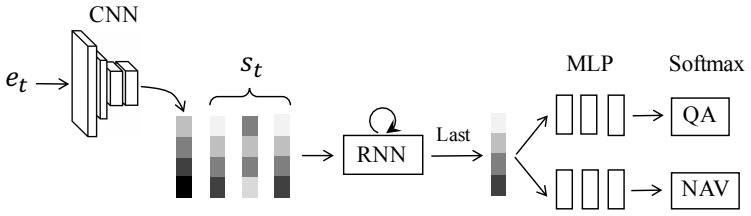


Figure 12: Overview of the baseline **VL**. The computations of NAV and QA only differ in the last MLPs.

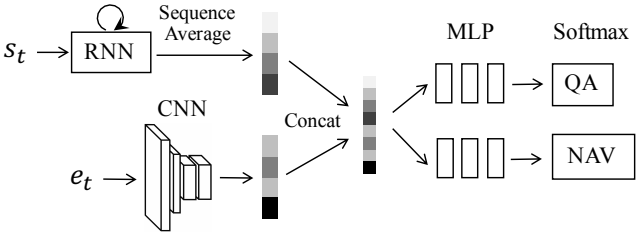


Figure 13: Overview of the baseline **CE**. The computations of NAV and QA only differ in the last MLPs.

C BASELINE DETAILS

Some additional implementation details of the baselines in section 4.3 are described below.

[CA] Its RNN has 512 units.

[VL] Its CNN has four convolutional layers $(3, 2, 64)$, $(3, 2, 64)$, $(3, 2, 128)$, and $(3, 1, 128)$. This is followed by an FC layer of size 512, which projects the feature cube to the word embedding space. The RNN has 512 units. For either QA or NAV, the RNN's last state goes through a three-layer MLP of which each layer has 512 units (Figure 12).

[CE] It has the same layer-size configuration with **VL** (Figure 13).

[SAN] Its RNN has 256 units. Following the original approach (Yang et al., 2016), we use two attention layers.

All the layers of the above baselines are ReLU activated.

D EXPERIENCE REPLAY AND CURRICULUM LEARNING

We employ Experience Replay (Mnih et al., 2015) for training both NAV and QA, for all the approaches. The environment inputs, rewards, and the actions taken by the agent in the most recent 10k time steps are stored in a replay buffer. During training, each time two mini-batches of the same size are sampled from the buffer, one for NAV and the other for QA. The mini-batch for NAV is sampled from the replay buffer using the Rank-based Sampler (Schaul et al., 2016) which has proven to increase the training efficiency by prioritizing rare experiences during sampling.

We exploit Curriculum Learning (Bengio et al., 2009) to gradually increase the environment complexity to help the agent learn. We increase the following quantities in proportional to $\min(1, G'/G)$, where G' is the number of sessions trained so far and G is the total number of curriculum sessions:

I The size of the open space on the environment map.

II The number of objects in the environment.

III The number of wall blocks.

IV The number of object classes that can be sampled from.

V The lengths of the NAV command and the QA question.

We found that this curriculum is important for an efficient learning. In the experiments, we set $G = 25$k during training while do *not* have any curriculum during test (*i.e.*, test with maximal difficulty).