Attention Based Natural Language Grounding by Navigating Virtual Environment

Akilesh B,* Abhishek Sinha,* Mausoom Sarkar & Balaji Krishnamurthy
Adobe Systems Inc,
Noida, Uttar Pradesh, India
{akb,abhsinha,msarkar,kbalaji}@adobe.com

Abstract

In this work, we focus on the problem in which an agent learns to navigate to the target object in a 2D grid environment. The agent receives visual information through raw pixels and a natural language instruction telling what task needs to be achieved. We propose a simple architecture consisting of a novel attention mechanism for multimodal fusion of visual and textual modalities in order to ground natural language instructions in our environment. Our model does not have any prior information of both the visual and textual modalities and is end-to-end trainable. Our experimental results show that our attention mechanism outperforms the existing multimodal fusion mechanisms proposed in order to solve the above mentioned task. We demonstrate through the visualization of attention weights that our model learns to correlate attributes of the object referred in the instruction with visual representations and also show that the learnt textual representations are semantically meaningful as they follow vector arithmetic. We also show that our model generalizes effectively to unseen scenarios and exhibit *zero-shot* generalization capabilities. In order to simulate the above described challenges, we introduce a new 2D environment for an agent to jointly learn visual and textual modalities.

1 Introduction

Understanding of natural language instructions is an important aspect of an Artificial Intelligence (AI) system. In order to successfully accomplish tasks specified by natural language instructions, an agent has to extract representations of language that are semantically meaningful and ground it in perceptual elements and actions in the environment.

Consider a task in which an agent has to learn to navigate to a target object in a 2D grid environment. The agent receives visual information through raw pixels and a natural language instruction at the beginning of every episode which specifies the characteristics of the target object. The environment consists of several other objects as well as many obstacles which act as distractors. The agent sees an egocentric view of the environment in which the objects present outside a certain predefined radius are not visible. The challenges that the agent has to tackle here are manyfold: a) the agent has to develop the capability to recognize various objects, b) have some memory of objects seen in previous states while exploring the environment as the objects may occlude each other and/or may not be present in the agent's field of view c) ground the instruction in visual elements and actions in the environment and d) learn a policy to navigate to the target object by avoiding the obstacles and other non-target objects.

*These authors contributed equally

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

(a) Complete view of environment. (b) Egocentric view as seen by agent.

Figure 1: Example state of environment.

We tackle this problem by proposing an end-to-end trainable architecture that creates a combined representation of the image observed by the agent and the instruction it receives. We develop a novel attention mechanism for multimodal fusion of visual and textual modalities. In order to simulate the above described challenges, we introduce a new 2D environment for an agent to jointly learn visual and textual modalities. Finally, in order to enable the reproducibility of our research and foster further research in this direction, we open source the code for our environment as well as the models that we developed.

Section 7 contains a brief description of prior work and approaches proposed for grounding natural language instructions.

2 Environment

We create a new 2-D grid based environment in which the agent interacts with the environment and performs one of the four actions: *up*, *down*, *left* and *right*. Each scenario in the environment consists of an agent, a list of objects and a list of obstacles (walls) as shown in figure 1a. Every object and obstacle has a set of attributes associated with it like color, size etc. Environment is completely customizable as the grid size, number of objects, type of objects, number of obstacles along with their corresponding attributes can be specified in a configurable specification file which is in JSON format. At the beginning of every episode, the agent receives a natural language instruction (Eg: "Go to red apple") and obtains a positive reward (+1) on successful completion of the task. The agent gets a negative reward (-1) whenever it hits a wall. The agent also receives a small negative (-0.5) reward if it reaches any non-target object . In every episode, the environment is reset randomly, i.e., the agent, the target object, non-target objects are placed at random locations in the grid. Some example instructions are: *"Go to car"*, *"Go to green apple"*, *"Go to medium blue sofa"*, *"Go to south of blue bus"*, *"Go to bottom right corner"*, *"There is a pink chair. Go to it."*, *"There is a small blue bag. Go to it."*, *"There are multiple green tree. Go to smaller(or larger) one."*, *"There is a small green car and a medium yellow bus. Go to former(or latter)."*

The complete set of objects and instructions are provided in section 7.

3 Proposed Approach

The proposed model can be divided into three phases a) input processing phase b) multi-modal fusion phase and c) policy learning phase, as illustrated in figure 2

3.1 Input processing phase

The agent at each time step (t) receives a RGB image of the environment which is egocentric (E_t) along with an instruction (I) at the start of the episode. The image E_t is passed through several layers of convolutional neural network (CNN) to obtain a good representation of the image. The final representation of the image obtained in this phase is R_E and is of dimension $W \times H \times D$, where W, H are width and height of each feature map and D represents the number of feature maps.(Refer section 7.2 for exact details of network hyper-parameters). Every word in the instruction I is converted into an one hot vector and then passed through the Gated Recurrent Unit (GRU) Chung et al. [2014]. The complete list of these words are provided in section 7.5. The output of the GRU is then passed through multiple parallel (say n) Fully-Connected (FC) layers each of size 64 (equal to the number of feature maps in the final image representation) and reshaped into $1 \times 1 \times 64 \times 1$. This results in generation of vectors V_1, V_2 ...,V_n

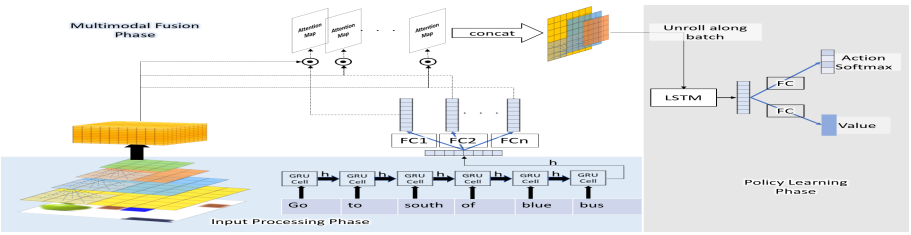


Figure 2: Our network architecture consisting of all three phases.

(a) Egocentric view as seen by agent. (b) Highlights the target objects (blue bag). (c) Highlights the non-target objects. (d) Highlights the non-target objects.

Figure 3: Visualization of attention maps for sentence: "There is a blue bag. Go to it."

3.2 Multimodal fusion phase

We propose a novel attention method for the multimodal fusion of R_E and V_j where $j \in \{1,2,\ldots,n\}$. Each V_j is used as a 1×1 filter to perform 2D convolution over feature volume R_E which is of dimension $7 \times 7 \times 64$ resulting in an attention map of dimension $7 \times 7 \times 1$. All these attention maps are then concatenated to result in a tensor of dimension $7 \times 7 \times n$ (say M_{attn} and the corresponding model $A3C_{attn}$). In our experiments we try with $n \in \{1,5,10\}$ (refer section 4 for comparison of performance). This attention mechanism draws inspiration from Xu et al. [2015]. M_{attn} is then fed as input to the policy learning phase. Through this fusion mechanism, we have discarded the unnecessary information and obtained a joint minimalistic representation (M_{attn}) of both the textual and visual modalities which symbolizes consciousness of an AI agent as highlighted in Bengio [2017]. We also consider a small variant of this fusion mechanism where we concatenate one of the attention maps with the visual representation to result in a tensor of dimension $7 \times 7 \times 65$ (say $M_{netattn}$ and the corresponding model $A3C_{netattn}$). We also compare our method with previously proposed multimodal fusion methods: a) Gated Attention unit Chaplot et al. [2017] (say $A3C_{hadamard}$) b) Concatenation unit Yu et al. [2017] (say $A3C_{concat}$). We do not mimic the complete architecture used by the authors in their works for our comparison, but just adopt the multimodal fusion approach proposed by them.

3.3 Policy learning phase

This phase receives as input, the output of the multimodal fusion phase. We adopt a reinforcement learning approach using the Asynchronous Advantage Actor-Critic (A3C) Mnih et al. [2016] algorithm. Refer section 7.2 for exact details of our architecture.

4 Experimental Setup

In all our experiments, we fix the grid size to be 10x10 with 32 objects and 8 obstacles. At the start of every episode, we randomly select between 3 to 6 objects from the set of 32 objects and a single instruction is randomly selected from a set of feasible instructions. We use *accuracy* of the agent and *mean reward* it obtains as the evaluation metrics. We consider two modes of evaluation:

1) **Unseen scenario generalization:** At test time, the agent is evaluated in unseen environment scenarios with instructions in the train set.

2) **Zero-shot generalization:** At test time, the agent is evaluated with unseen instructions that consists of new combinations of attribute-object pairs.

5 Results and Discussions

Table 1 depicts the performance of our attention mechanism ($A3C_{attn}$) as described in section 3.2 for various n values on unseen scenarios. The *accuracy* and *mean reward* values are averaged over 100 episodes. For *zero-shot* evaluation, we measure the agent's success rate on 19 instructions that were held out during training and these instructions consist of new combinations of attribute-object pairs not seen before by the agent during train phase (refer section 7.6 for these instructions). Figure 4a shows the *mean reward* obtained by our $A3C_{attn}$ model for different n values as the training progresses. We observed that $A3C_{attn}$ ($n = 5$) achieves the best *mean reward* of 0.95 on unseen scenarios and 0.8 on *zero-shot* evaluations. Figure 4b shows the comparison of our best performing $A3C_{attn}$ model in which all convolutional and FC layers have PReLU activations against the case in which all convolutional and FC layers have ReLU activations. Table 1 also portrays the performance of a variant of our attention method ($A3C_{netattn}$) as well as with other previously proposed multimodal fusion mechanism methods ($A3C_{hadamard}$ and $A3C_{concat}$) as described in section 3.2.

We also visualize the attention maps for our best performing $A3C_{attn}$ model. Figure 3 shows the visualization of attention maps for the case when the sentence is "There is a blue bag. Go to it." As shown in figures 3b, 3c and 3d, different attention maps focus on different regions in the environment and the agent uses a combination of these in order to learn the policy and successfully navigate to the target object while avoiding incorrect objects and obstacles. The attention map 3b focuses on the target objects. This attention map highlights both the smaller blue bag and the larger blue bag as the agent gets a positive reward if it correctly navigates to either of them. On the other hand, the attention maps 3c and 3d focus on the non-target objects (blue sofa and orange bag) and obstacles. Figure 5 provide another visualization of attention maps for the case when the instruction is "There are multiple apples. Go to larger one."

In order to understand the quality of instruction embedding learnt by the GRU, we do a two-dimensional Principal Component Analysis (PCA) projection of the original 16 dimension representation obtained when the input instruction is encoded through a GRU of size 16. As shown in figure 6a, *vector("Go to green apple") - vector("Go to apple")* is parallel to *vector("Go to green sofa") - vector("Go to sofa")*. In figures 6b and -6c, we notice similar relationships. These evidently highlight our model's ability to organize various concepts and learn the relationships between them implicitly. Additionally, we also illustrate that these instruction embeddings follow vector arithmetic. In order to elucidate this, we obtain the embeddings for the instruction *"Go to small green tree"* as follows: *vector("Go to tree") + (vector("Go to green apple") - vector("Go to apple")) + vector("Go to small red car") - vector("Go to red car")*. Using this, the agent learns to navigate successfully to the small green tree as depicted in 7. Figure 7a is the complete state of the environment consisting of small green tree, medium green tree, small red car and medium red car. Figure 7b shows the initial egocentric view of the agent. Although the larger green tree is visible to the agent in its initial view and the smaller green tree is not visible, the agent learns to avoid the larger green tree and navigate correctly to the smaller green tree.

Language grounding eventually means capturing the essence of words. In order to examine if the agent actually understands the words semantically, we train the agent by giving it instructions in both English and French language. At test time, to assess the instruction embedding, we ask the agent to translate English instructions to French and vice-versa. We find that after complete training the agent manages to translate 85% of the instructions. It is important to note that the translation was done in a completely unsupervised way, without explicitly giving parallel corpora to the agent. Refer section 7.2 for description of architecture used to perform this. In all these experiments, the agent's trajectory as it navigates to the target object are stored in the form of GIFs and are available `https://github.com/rl-lang-grounding/rl-lang-ground`.

6 Conclusion

In the paper we presented a novel and simple architecture to achieve grounding of natural language sentences via reinforcement learning. We show that retaining just the representation obtained after multimodal fusion phase (i.e. multiple attention maps) and discarding the visual features helps the agent achieve its goals. We justify this claim by visualization of the attention maps which reveal that they contain sufficient information needed for the agent to find the optimal policy. In order to

Table 1: Comparison of our model for different n values with other baseline methods on unseen scenario generalization.

Model	mean reward	accuracy
$A3C_{attn}$ $(n = 1)$	0.87	0.86
$A3C_{attn}$ $(n = 5)$	**0.95**	**0.92**
$A3C_{attn}$ $(n = 10)$	0.94	0.88
$A3C_{netattn}$	-2.8	0.1
$A3C_{hadamard}$	0.4	0.5
$A3C_{concat}$	-2.9	0.1

encourage further research in this direction we have also open sourced the code for our environment as well as the models that we developed.

References

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

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 *International Conference on Machine Learning*, pages 2048–2057, 2015.

Yoshua Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*, 2017.

Haonan Yu, Haichao Zhang, and Wei Xu. A deep compositional framework for human-like language acquisition in virtual environment. *arXiv preprint arXiv:1703.09831*, 2017.

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

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

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, pages 2772–2778, 2016.

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):529–533, 2015.

Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *AAAI*, pages 2140–2146, 2017.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

7 Appendix

7.1 Related work

The task of grounding natural language instructions has been well explored by researchers in different domains. Given a natural language instruction, Chen and Mooney [2011] attempt to learn a navigation policy that is optimal in a 2D maze-like setting by relying on a semantic parser. Mei et al. [2016] focus on neural mapping of navigational instructions to action sequences by representing the state of the world using bag-of-words visual features.

Deep reinforcement learning agents have been previously used to solve tasks in both 2D Mnih et al. [2015] and 3D Mnih et al. [2016] environments. More recently, these approaches have been used for playing first-person shooter games in Lample and Chaplot [2017] and Kempka et al. [2016]. These works focus on learning optimal policy for different tasks using only visual features, while our work involves the agent receiving natural language instruction in addition to visual state of the environment.

The authors in Yu et al. [2017] propose an end-to-end framework by which an agent learns to navigate in 2D maze-like environment (XWORLD) using natural language instructions. They simultaneously learn visual representations, syntax and semantics of natural language instruction as well as the navigation action to perform. The task of the agent is basically navigation plus Visual Question Answering (VQA) Antol et al. [2015]; the agent at every step either gets a navigation instruction or a question about the environment, and the output is either a navigation action or answer to the question posed. Chaplot et al. [2017] propose a Gated-Attention architecture for task oriented language grounding and evaluate their approach on a new environment built over VizDoom Kempka et al. [2016].

In our work, we present a simple and novel architecture in order to ground natural language instructions in a 2D grid environment. Our model does not have any prior information of both the visual and textual modalities and is end-to-end trainable without the requirement of external semantic parser. Through our multimodal fusion mechanism, we obtain a joint concise representation of both the visual and textual modalities which is sufficient for the agent to learn optimal policy. When compared to prior work, the scenarios we work with are more complex as our environment has a larger grid size, more number of objects present concurrently in it, increased complexity of natural language instructions (two sentence instructions). In addition, the new environment introduced by us is thread compatible.

7.2 Implementation details

The input to the input processing phase neural network is a RGB image of size 84x84x3 and an instruction. The input image is processed with a CNN that has four convolution layers. The first layer convolves the image with 32 filters of 5x5 kernel size with stride 2, followed by another 32 filters of 5x5 kernel size with stride 2. This is then followed by 64 filters of 4x4 kernel size with stride 1 and finally by another 64 filters of 3x3 kernel size with stride 2. The input instruction is encoded through a GRU of size 16. The encoded instruction is then passed through a FC layer of size 64. The visual mode is then combined with textual mode through various multi-modal fusion mechanisms: a) simple concatenation, b) our novel attention mechanism and its variants, c) Gated-Attention unit (Hadamard product). In case of our novel attention mechanism, the multiple attention maps are concatenated and passed through two other convolutional layers (each having 64 filters of 3x3 kernel size with stride 1) before passing it to the LSTM layer. All our experiments are performed with A3C algorithm. Our policy learning phase has a LSTM layer of size 32, followed by fully connected layer of size 4 to estimate the policy function as well as fully connected layer of size 1 to predict value function. The LSTM layer enables the agent to have some memory of previous states. This is crucial as the agent receives egocentric view in which all the objects may

not be present and hence need to remember the previously seen objects. The network parameters are shared for predicting both the policy function and the value function except the final fully connected layer. All the convolutional layers and FC layers have PReLU activations He et al. [2015]. We observed during our experimentation the importance of not suppressing the negative gradients, as the same architecture in which all convolutional and FC layers have ReLU activations Nair and Hinton [2010] performed poorly on our evaluation scenarios (Refer section 4 for this comparison). The A3C algorithm was trained using Adam optimizer Kingma and Ba [2014] with an annealing learning rate schedule starting with 0.0001 and reducing by a fraction of 0.9 after every 10000 steps. For each experiment, we run 32 parallel threads and we use a discount factor of 0.99 for calculating expected rewards. The gradients of each of these worker threads are clipped in order to prevent overly-large parameter updates which can destabilize the policy. As described in Mnih et al. [2016] we use entropy regularization for improved exploration. Further, in order to reduce the variance of the policy gradient updates, we use the Generalized Advantage Estimator Schulman et al. [2015]. For language translation, we attach a decoder branch consisting of two separate decoders one each for English and French, on top of the instruction embedding obtained through GRU. Based on whether the natural language instruction is English or French, reconstruction loss through the corresponding decoder is backpropagated to modify the weights of the GRU.

7.3 List of objects

Our 2D environment is completely customizable. The different objects, obstacles along with their corresponding attributes like color, size can be specified in a configurable specification file which is in JSON format. The images used to represent these objects are publicly available. The various objects used in our experiments are: *objects = {red apple, green apple, orange, green sofa, blue sofa, red car, green car, orange chair, pink chair, blue bus, yellow bus, blue bag, orange bag, green tree, yellow cup, red cup, yellow flower, pink flower}*. The size attribute of each of these objects can be *small* (1x1), *medium* (2x2) or *large* (4x4).

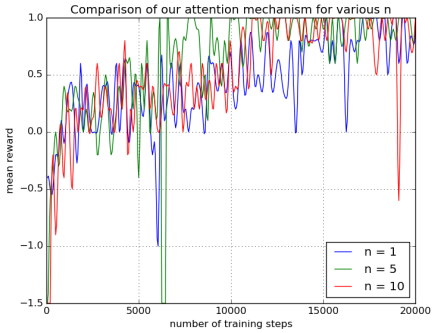
7.4 List of instructions

The various natural language instructions that specify the characteristics of the target object based on which the agent learns to navigate in our 2D grid environment are listed below.

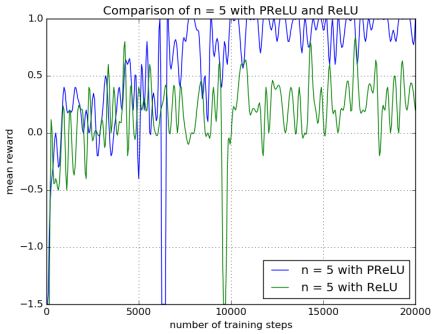
- *Go to [object]* - where [object] is one of the objects given in section 7.3, i.e. *apple*, *orange*, *sofa*, *car*, *chair*, *bus*, *bag* and *tree*.
- *Go to [color][object]* - where [color][object] pairs are present in *objects* list (section 7.3).
- *Go to [size][color][object]* - where [size] can be *small*, *medium* or *large* and [color][object] pairs are present in *objects* list (section 7.3).
- *Go to [direction] of [color][object]* - where [direction] can be *north*, *south*, *east* or *west* and [color][object] pairs are present in *objects* list (section 7.3).
- *Go to [top (or bottom)] [left (or right)] corner*.
- *There is a [color][object]. Go to it.* - where [color][object] pairs are present in *objects* list (section 7.3).
- *There is a [size][color][object]. Go to it.* - where [size] can be *small*, *medium* or *large* and [color][object] pairs are present in *objects* list (section 7.3).
- *There are multiple [color][object]. Go to smaller(or larger) one.* - where [color][object] pairs are present in *objects* list (section 7.3).
- *There is a [size1][color1][object1] and a [size2][color2][object2]. Go to former(or latter).* - where [size1], [size2] can be *small*, *medium* or *large* and [color1][object1], [color2][object2] pairs are present in *objects* list (section 7.3).

7.5 Vocabulary words

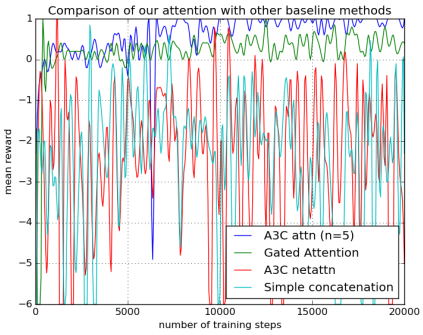
The different unique words that can be present in instruction are: *vocab = {apple, orange, sofa, car, chair, bus, bag, tree, flower, cup, go, to, of, there, is, and, are, a, it, red, green, orange, blue, pink, yellow, top, bottom, left, right, corner, north, south, west, east, small, medium, smaller, larger, one, multiple, former, latter}*



(a) Performance of $A3C_{attn}$ ($n = 1, 5, 10$).



(b) Performance of $A3C_{attn}$ ($n = 5$) with PReLU vs ReLU.



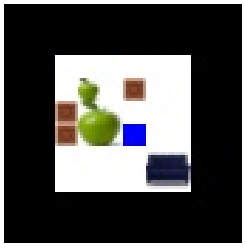
(c) Performance of $A3C_{attn}$ ($n = 5$) with other baseline methods.

Figure 4: Performance analysis of our attention mechanism and comparison with other baseline methods.

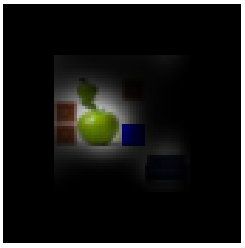
7.6 Zero shot generalization instructions

The list of unseen instructions used for evaluation under *zero-shot* generalization settings are: *instructions = {"Go to small red car", "Go to medium green apple", "Go to small orange", "Go to medium red cup", "There is a small red car. Go to it.", "There is a medium green apple. Go to it.", "There is a small orange. Go to it.", "There is a medium red cup. Go to it.", "Go to orange sofa", "Go to blue bus", Go to yellow flower", "There is a orange sofa. Go to it.", "There is a blue bus. Go to it.", "There is a yellow flower. Go to it.", "Go to north of orange chair.", "Go to south of blue bus.", "Go to west of green tree.", "Go to east of blue sofa.", "Go to bag."}*

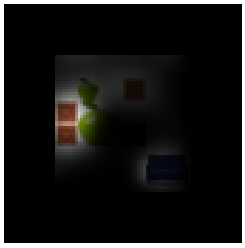
7.7 Figures



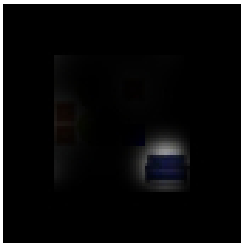
(a) Egocentric view as seen by agent.



(b) Highlights the target object (larger green apple).

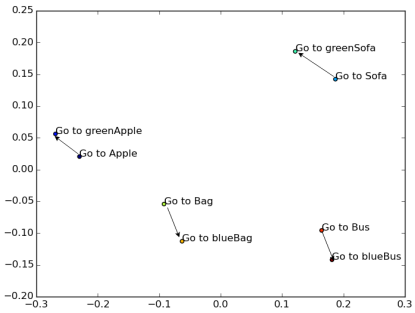


(c) Highlights the non-target objects.

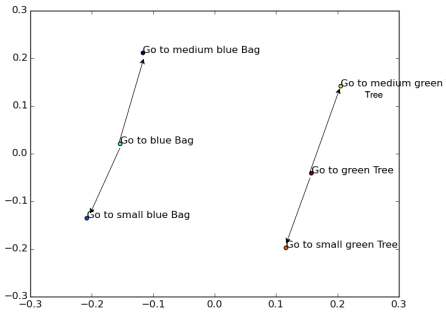


(d) Highlights the non-target objects.

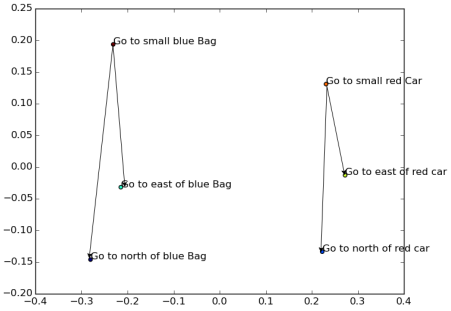
Figure 5: Visualization of attention maps for sentence: "There are multiple apples. Go to larger one."



(a) Instruction type - Go to [object] or Go to [color][object]

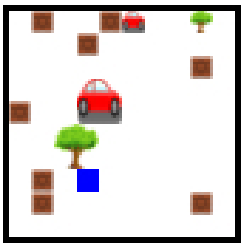


(b) Instruction type - Go to [color][object] or Go to [size][color][object]

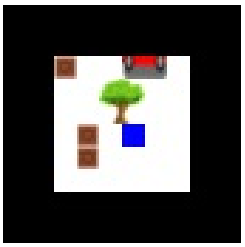


(c) Instruction type - Go to [direction] of [color][object] or Go to [size][color][object]

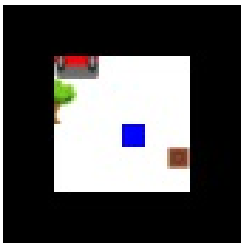
Figure 6: Two-dimensional PCA projection of instruction embedding learnt by the GRU.



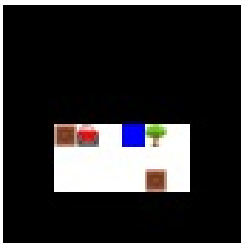
(a) Complete state of environment.



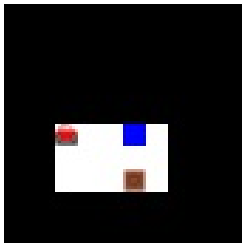
(b) Initial egocentric view as seen by agent.



(c) Intermediate state as it navigates to target.



(d) Agent is next to the target.



(e) Agent reaches the target.

Figure 7: Agent's trajectory as it navigates to the small green tree.