The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Gated-Attention Architectures for Task-Oriented Language Grounding

Devendra Singh Chaplot
chaplot@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Kanthashree Mysore Sathyendra*
ksathyen@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Rama Kumar Pasumarthi*
rpasumar@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Dheeraj Rajagopal*
dheeraj@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Ruslan Salakhutdinov
rsalakhu@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Abstract

To perform tasks specified by natural language instructions, autonomous agents need to extract semantically meaningful representations of language and map it to visual elements and actions in the environment. This problem is called task-oriented language grounding. We propose an end-to-end trainable neural architecture for task-oriented language grounding in 3D environments which assumes no prior linguistic or perceptual knowledge and requires only raw pixels from the environment and the natural language instruction as input. The proposed model combines the image and text representations using a Gated-Attention mechanism and learns a policy to execute the natural language instruction using standard reinforcement and imitation learning methods. We show the effectiveness of the proposed model on unseen instructions as well as unseen maps, both quantitatively and qualitatively. We also introduce a novel environment based on a 3D game engine to simulate the challenges of task-oriented language grounding over a rich set of instructions and environment states.

Introduction

Artificial Intelligence (AI) systems are expected to perceive the environment and take actions to perform a certain task (Russell and Norvig 1995). Task-oriented language grounding refers to the process of extracting semantically meaningful representations of language by mapping it to visual elements and actions in the environment in order to perform the task specified by the instruction.

Consider the scenario shown in Figure 1, where an agent takes natural language instruction and pixel-level visual information as input to carry out the task in the real world. To accomplish this goal, the agent has to draw semantic correspondences between the visual and verbal modalities and learn a policy to perform the task. This problem poses several challenges: the agent has to learn to *recognize* objects in raw pixel input, *explore* the environment as the objects might be occluded or outside the field-of-view of the agent, *ground* each concept of the instruction in visual elements or actions in the environment, *reason* about the pragmatics of language based on the objects in the current environment (for example instructions with superlative tokens, such as 'Go to the largest object') and *navigate* to the correct object while avoiding incorrect ones.

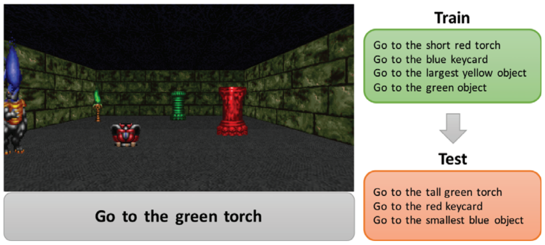
Go to the green torch
Train
Go to the short red torch
Go to the blue keycard
Go to the largest yellow object
Go to the green object
Test
Go to the tall green torch
Go to the red keycard
Go to the smallest blue object

Figure 1: An example of task-oriented language grounding in the 3D Doom environment with sample instructions. The test set consists of unseen instructions.

To tackle this problem, we propose an architecture that comprises of a *state processing module* that creates a joint representation of the instruction and the images observed by the agent, and a *policy learner* to predict the optimal action the agent has to take in that timestep. The state processing module consists of a novel Gated-Attention multimodal fusion mechanism, which is based on multiplicative interactions between both modalities (Dhingra et al. 2017; Wu et al. 2016).

The contributions of this paper are summarized as follows: 1) We propose an end-to-end trainable architecture that handles raw pixel-based input for task-oriented language grounding in a 3D environment and assumes no prior linguistic or perceptual knowledge. We show that the proposed model generalizes well to unseen instructions as well as unseen maps. 2) We develop a novel Gated-Attention mechanism for multimodal fusion of representations of verbal and visual modalities. We show that the gated-attention mechanism outperforms the baseline method of concatenating the representations using various policy learning methods. The visualization of the attention weights in the gated-attention unit shows that the model learns to associate attributes of

[*] Equal contribution

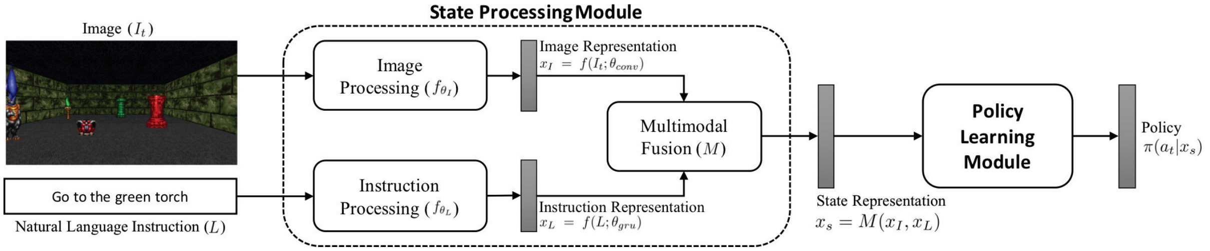
State Processing Module
Image (I_t)
Image Processing (f_{θ_I})
Image Representation
$x_I = f(I_t;\theta_{conv})$
Multimodal Fusion (M)
Go to the green torch
Natural Language Instruction (L)
Instruction Processing (f_{θ_L})
Instruction Representation
$x_L = f(L;\theta_{gru})$
State Representation
$x_s = M(x_I, x_L)$
Policy Learning Module
Policy
$\pi(a_t|x_s)$

Figure 2: The proposed model architecture to estimate the policy given the natural language instruction and the image showing the first-person view of the environment.

the object mentioned in the instruction with the visual representations learned by the model. 3) We introduce a new environment, built over ViZDoom (Kempka et al. 2016), for task-oriented language grounding with a rich set of actions, objects and their attributes. The environment provides a first-person view of the world state, and allows for simulating complex scenarios for tasks such as navigation. [1]

Related Work

Grounding Language in Robotics. In the context of grounding language in objects and their attributes, Guadarrama et al. (2014) present a method to ground open vocabulary to objects in the environment. Several works look at grounding concepts through human-robot interaction (Chao, Cakmak, and Thomaz 2011; Lemaignan et al. 2012). Other works in grounding include attempts to ground natural language instructions in haptic signals (Chu et al. 2013) and teaching robot to ground natural language using active learning (Kulick et al. 2013). Some of the work that aims to ground navigational instructions include (Guadarrama et al. 2013), (Bollini et al. 2013) and (Beetz et al. 2011), where the focus was to ground verbs like *go, follow, etc.* and spatial relations of verbs (Tellex et al. 2011; Fasola and Mataric 2013).

Mapping Instructions to Action Sequences. Chen and Mooney (2011) and Artzi and Zettlemoyer (2013) present methods based on semantic parsing to map navigational instructions to a sequence of actions. Mei, Bansal, and Walter (2015) look at neural mapping of instructions to sequence of actions, along with input from bag-of-word features extracted from the visual image. While these works focus on grounding navigational instructions to actions in the environment, we aim to ground visual attributes of objects such as shape, size and color.

Deep reinforcement learning using visual data. Prior work has explored using Deep Reinforcement learning approaches for playing FPS games (Lample and Chaplot 2016; Kempka et al. 2016; Kulkarni et al. 2016). The challenge here is to learn optimal policy for a variety of tasks, including navigation using raw visual pixel information. Chaplot et al. (2016) look at transfer learning between different tasks in the Doom Environment. In all these methods, the policy for each task is learned separately using a deep Q-Learning (Mnih et al. 2013). In contrast, we train a single network for multiple tasks/instructions. Zhu et al. (2016) look at target-driven visual navigation, given the image of the target object. We use the natural language instruction and do not have the visual image of the object. Yu, Zhang, and Xu (2017) look at learning to navigate in a 2D maze-like environment and execute commands, for both seen and *zero-shot* setting, where the combination of words are not seen before. Misra, Langford, and Artzi (2017) also look at mapping raw visual observations and text input to actions in a 2D Blocks environment. While these works also looks at executing a variety of instructions, they tackle only 2D environments.

Compared to the prior work, this paper aims to address grounding of natural language instruction in a challenging 3D setting involving raw-pixel input and partially observable envrionment, which poses additional challenges of perception, exploration and reasoning. Unlike many of the previous methods, our model assumes no prior linguistic or perceptual knowledge, and is trainable end-to-end.

Problem Formulation

We tackle the problem of task-oriented language grounding in the context of target-driven visual navigation conditioned on a natural language instruction, where the agent has to navigate to the object described in the instruction. Consider an agent interacting with an episodic environment $\mathcal{E}$. In the beginning of each episode, the agent receives a natural language instruction (L) which indicates the description of the target, a visual object in the environment. At each time step, the agent receives a raw pixel-level image of the first person view of the environment (I_t), and performs an action a_t. The episode terminates whenever the agent reaches any object or the number of time steps exceeds the maximum episode length. Let $s_t = \{I_t, L\}$ denote the state at each time step. The objective of the agent is to learn an optimal policy $\pi(a_t|s_t)$, which maps the observed states to actions, eventually leading to successful completion of the task. In this case, the task is to reach the correct object before the episode terminates. We consider two different learning approaches for language grounding using target-driven visual navigation: (1) Imitation Learning (Bagnell 2015): where the agent has access to an oracle which specifies the optimal action given any state in the environment; (2) Reinforcement Learning (Sutton and Barto 1998): where the agent receives a positive

[1] The code for the environment and the proposed model is available at https://github.com/devendrachaplot/DeepRL-Grounding

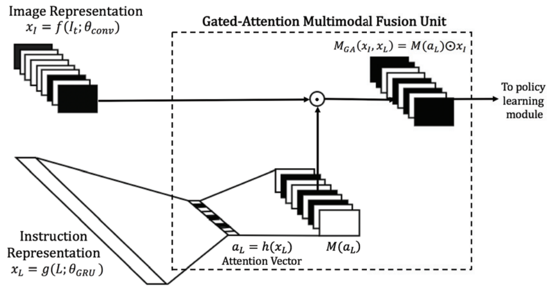
Image Representation
$x_I = f(I_t; \theta_{conv})$
Gated-Attention Multimodal Fusion Unit
$M_{GA}(x_I, x_L) = M(a_L) \odot x_I$
To policy learning module
Instruction Representation
$x_L = g(L; \theta_{GRU})$
$a_L = h(x_L)$
Attention Vector
$M(a_L)$

Figure 3: Gated-Attention unit architecture.

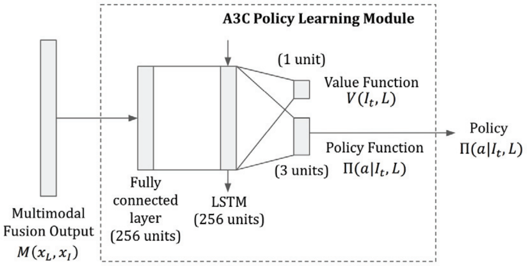

Figure 4: A3C policy model architecture.

reward when it reaches the target object and a negative reward when it reaches any other object.

Proposed Approach

We propose a novel architecture for task-oriented visual language grounding, which assumes no prior linguistic or perceptual knowledge and can be trained end-to-end. The proposed model is divided into two modules, state processing and policy learning, as shown in Figure 2.

State Processing Module: The state processing module takes the current state $s_t = \{I_t, L\}$ as the input and creates a joint representation for the image and the instruction. This joint representation is used by the policy learner to predict the optimal action to take at that timestep. It consists of a convolutional network (LeCun, Bengio, and others 1995) to process the image I_t, a Gated Recurrent Unit (GRU) (Cho et al. 2014) network to process the instruction L and a multimodal fusion unit that combines the representations of the instruction and the image. Let $x_I = f(I_t; \theta_{conv}) \in \mathcal{R}^{d \times H \times W}$ be the representation of the image, where θ_{conv} denote the parameters of the convolutional network, d denotes number of feature maps (intermediate representations) in the convolutional network output, while H and W denote the height and width of each feature map. Let $x_L = f(L; \theta_{gru})$ be the representation of the instruction, where θ_{gru} denotes the parameters of the GRU network. The multimodal fusion unit, $M(x_I, x_L)$ combines the image and instruction representations. Many prior methods combine the multimodal representations by concatenation (Mei, Bansal, and Walter 2015; Misra, Langford, and Artzi 2017). We develop a multimodal fusion unit, *Gated-Attention*, based on multiplicative interactions between instruction and image representation.

Concatenation: In this approach, the representations of the image and instruction are simply flattened and concatenated to create a joint state representation:

$$M_{concat}(x_I, x_L) = [\text{vec}(x_I); \text{vec}(x_L)],$$

where vec(.) denotes the flattening operation. The concatenation unit is used as a baseline for the proposed Gated-Attention unit as it is used by prior methods (Mei, Bansal, and Walter 2015; Misra, Langford, and Artzi 2017).

Gated-Attention: In the Gated-Attention unit, the instruction embedding is passed through a fully-connected linear layer with a sigmoid activation. The output dimension of this linear layer, d, is equal to the number of feature maps in the output of the convolutional network (first dimension of x_I). The output of this linear layer is called the attention vector $a_L = h(x_L) \in \mathcal{R}^d$, where $h(.)$ denotes the fully-connected layer with sigmoid activation. Each element of a_L is expanded to a $H \times W$ matrix. This results in a 3-dimensional matrix, $M(a_L) \in \mathcal{R}^{d \times H \times W}$ whose $(i, j, k)^{th}$ element is given by: $M_{a_L}[i, j, k] = a_L[i]$. This matrix is multiplied element-wise with the output of the convolutional network:

$$M_{GA}(x_I, x_L) = M(h(x_L)) \odot x_I = M(a_L) \odot x_I,$$

where $\odot$ denotes the Hadamard product (Horn 1990). The architecture of the Gated-Attention unit is shown in Figure 3. The whole unit is differentiable which makes the architecture end-to-end trainable.

The proposed Gated-Attention unit is inspired by the Gated-Attention Reader architecture for text comprehension (Dhingra et al. 2017). They integrate a multi-hop architecture with a Gated-attention mechanism, which is based on multiplicative interactions between the query embedding and the intermediate states of a recurrent neural network document reader. In contrast, we propose a Gated-Attention multimodal fusion unit which is based on multiplicative interactions between the instruction representation and the convolutional feature maps of the image representation. This architecture can be extended to any application of multimodal fusion of verbal and visual modalities.

The intuition behind Gated-Attention unit is that the trained convolutional feature maps detect different attributes of the objects in the frame, such as color and shape. The agent needs to attend to specific attributes of the objects based on the instruction. For example, depending on the whether the instruction is "Go to the green object", "Go to the pillar" or "Go to the green pillar" the agent needs to attend to objects which are 'green', objects which look like a 'pillar' or both. The Gated-Attention unit is designed to gate specific feature maps based on the attention vector from the instruction, a_L.

Policy Learning Module

The output of the multimodal fusion unit (M_{concat} or M_{GA}) is fed to the policy learning module. The architecture of the policy learning module is specific to the learning paradigm: (1) Imitation Learning or (2) Reinforcement Learning.

For imitation learning, we consider two algorithms, Behavioral Cloning (Bagnell 2015) and DAgger (Ross, Gordon, and

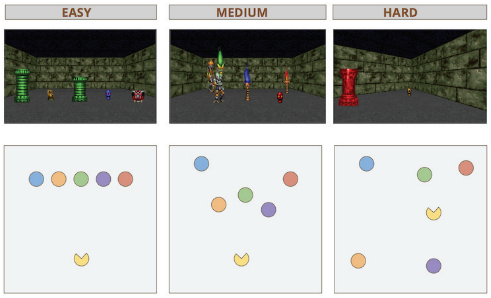

Figure 5: Sample starting states and bird's eye view of the map (not visible to the agent) showing agent and object locations in Easy, Medium and Hard modes.

Bagnell 2011). Both the algorithms require an oracle that can return an optimal action given the current state. The oracle is implemented by extracting agent and target object locations and orientations from the Doom game engine. Given any state, the oracle determines the optimal action as follows: The agent first reorients (using *turn_left, turn_right* actions) towards the target object. It moves forward (*move_forward* action), reorienting towards the target object if deviation of the agent's orientation is greater than the minimum turn angle supported by the environment.

For reinforcement learning, we use the Asynchronous Advantage Actor-Critic (A3C) algorithm (Mnih et al. 2016) which uses a deep neural network to learn the policy and value functions and runs multiple parallel threads to update the network parameters. We also use the entropy regularization for improved exploration as described by (Mnih et al. 2016). In addition, we use the Generalized Advantage Estimator (Schulman et al. 2015) to reduce the variance of the policy gradient updates.

The policy learning module for imitation learning contains a fully connected layer to estimate the policy function. The policy learning module for reinforcement learning using A3C (shown in Figure 4) consists of an LSTM layer, followed by fully connected layers to estimate the policy function as well as the value function. The LSTM layer is introduced so that the agent can have some memory of previous states. This is important as a reinforcement learning agent might explore states where all objects are not visible and need to remember the objects seen previously.

Environment

We create an environment for task-oriented language grounding, where the agent can execute a natural language instruction and obtain a positive reward on successful completion of the task. Our environment is built on top of the ViZDoom API (Kempka et al. 2016), based on Doom, a classic first person shooting game. It provides the raw visual information from a first-person perspective at every timestep. Each scenario in the environment comprises of an agent and a list of objects (a subset of ViZDoom objects) - one correct and rest incorrect in a customized map. The agent can interact with the environment by performing navigational actions such as turn left, turn right, move forward. Given an instruction "Go to the green torch", the task is considered successful if the agent is able to reach the *green torch* correctly. The customizable nature of the environment enables us to create scenarios with varying levels of difficulty which we believe leads to designing sophisticated learning algorithms to address the challenge of multi-task and zero-shot reinforcement learning.

An *instruction* is a combination of (action, attribute(s), object) triple. Each instruction can have more than one attribute but we limit the number of actions and objects to one each. The environment allows a variety of objects to be spawned at different locations in the map. The objects can have various visual attributes such as color, shape and size[2]. We provide a set of 70 manually generated instructions[1]. For each of these instructions, the environment allows for automatic creation of multiple episodes, each randomly created with its own set of correct object and incorrect objects. Although the number of instructions are limited, the combinations of correct and incorrect objects for each instruction allows us to create multiple settings for the same instruction. Each time an instruction is selected, the environment generates a random combination of incorrect objects and the correct object in randomized locations. One of the significant challenges posed for a learning algorithm is to understand that the same instruction can refer to different objects in the different episodes. For example, "Go to the red object" can refer to a red keycard in one episode, and a red torch in another episode. Similarly, "Go to the keycard" can refer to keycards of various colors in different episodes. Objects could also occlude each other, or might not even be present in the agent's field of view, or the map could be more complicated, making it difficult for the agent to make a decision based solely on the current input, stressing the need for efficient exploration.

Our environment also provides different modes with respect to spawning of objects each with varying difficulty levels (Figure 5): **Easy**: The agent is spawned at a fixed location. The candidate objects are spawned at five fixed locations along a single horizontal line along the field of view of the agent. **Medium**: The candidate objects are spawned in random locations, but the environment ensures that they are in the field of view of the agent. The agent is still spawned at a fixed location. **Hard**: The candidate objects and the agent are spawned at random locations and the objects may or may not be in the agents field of view in the initial configuration. The agent needs to explore the map to view all the objects.

Experimental Setup

We perform our experiments in all of the three environment difficulty modes, where we restrict the number of objects to 5 for each episode (one correct object, four incorrect objects and the agent are spawned for each episode). During training, the objects are spawned from a training set of 55 instructions, while 15 instructions pertaining to unseen *attribute-object* combinations are held out in a test set for zero-shot evaluation. During training, at the start of each episode, one of the train instructions is selected randomly. A correct target object is selected and 4 incorrect objects are selected at random. These objects are placed at random locations depending on

[2] See the list of objects and instructions at https://goo.gl/rPWlMy

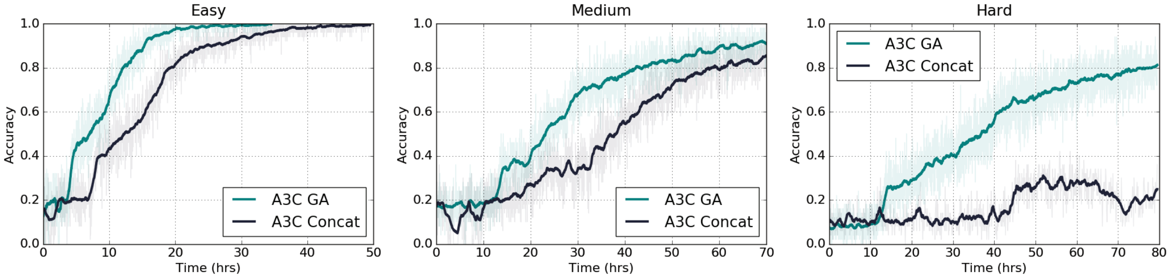

Figure 6: Comparison of the performance of the proposed Gated-Attention (GA) unit to the baseline Concatenation unit using Reinforcement learning algorithm, A3C for (a) easy, (b) medium and (c) hard environments.

the difficulty level of the environment. The episode terminates if the agent reaches any object or time step exceeds the maximum episode length ($T = 30$). The evaluation metric is the *accuracy* of the agent which is success rate of reaching the correct object before the episode terminates. We consider two scenarios for evaluation:
(1) **Multitask Generalization** (MT), where the agent is evaluated on unseen maps with instructions in the train set. Unseen maps comprise of unseen combination of objects placed at randomized locations. This scenario tests that the agent doesn't overfit to or memorize the training maps and can execute multiple instructions or tasks in unseen maps.
(2) **Zero-shot Task Generalization** (ZSL), where the agent is evaluated on unseen test instructions. This scenario tests whether the agent can generalize to new combinations of attribute-object pairs which are not seen during the training. The maps in this scenario are also unseen.

Baseline Approaches

Reinforcement Learning: We adapt (Misra, Langford, and Artzi 2017) as a reinforcement learning baseline in the proposed environment. Misra, Langford, and Artzi (2017) looks at jointly reasoning on linguistic and visual inputs for moving blocks in a 2D grid environment to execute an instruction. Their work uses raw features from the 2D grid, processed by a CNN, while the instruction is processed by an LSTM. Text and visual representations are combined using concatenation. The agent is trained using reinforcement learning and enhanced using distance based reward shaping. We do not use reward shaping as we would like the method to generalize to environments where the distance from the target is not available.

Imitation Learning: We adapt (Mei, Bansal, and Walter 2015) as an imitation learning baseline in the proposed environment. Mei, Bansal, and Walter (2015) map sequence of instructions to actions, treated as a sequence-to-sequence learning problem, with visual state input received by the decoder at each decode timestep. While they use a bag-of-visual words representation for visual state, we adapt the baseline to directly process raw pixels from the 3D environment using CNNs.

To ensure fairness in comparison, we use exact same architecture of CNNs (to process visual input), GRUs (to process textual instruction) and policy learning across baseline and proposed models. This reduces the reinforcement learning baseline to A3C algorithm with concatenation multimodal fusion (*A3C-Concat*), and imitation learning baseline to Behavioral Cloning with Concatenation (*BC-Concat*).

Hyper-parameters

The input to the neural network is the instruction and an RGB image of size 3x300x168. The first layer convolves the image with 128 filters of 8x8 kernel size with stride 4, followed by 64 filters of 4x4 kernel size with stride 2 and another 64 filters of 4x4 kernel size with stride 2. The architecture of the convolutional layers is adapted from previous work on playing deathmatches in Doom (Chaplot and Lample 2017). The input instruction is encoded through a Gated Recurrent Unit (GRU) (Chung et al. 2014) of size 256.

For the imitation learning approach, we run experiments with Behavioral Cloning (BC) and DAgger algorithms in an online fashion, which have data generation and policy update function per outer iteration. The policy learner for imitation learning comprises of a linear layer of size 512 which is fully-connected to 3 neurons to predict the policy function (i.e. probability of each action). In each data generation step, we sample state trajectories based on oracle's policy in BC and based on a mixture of oracle's policy and the currently learned policy in DAgger. The mixing of the policies is governed by an exploration coefficient, which has a linear decay from 1 to 0. For each state, we collect the optimal action given by the policy oracle. Then the policy is updated for 10 epochs over all the state-action pairs collected so far, using the RMSProp optimizer (Tieleman and Hinton 2012). Both methods use Huber loss (Huber 1964) between the estimated policy and the optimal policy given by the policy oracle.

For reinforcement learning, we run experiments with A3C algorithm. The policy learning module has a linear layer of size 256 followed by an LSTM layer of size 256 which encodes the history of state observations. The LSTM layer's output is fully-connected to a single neuron to predict the value function as well as three other neurons to predict the policy function. All the convolutional layers and fully-connected linear layers have ReLu activations (Nair and Hinton 2010). The A3C model was trained using Stochastic Gradient Descent (SGD) with a learning rate of 0.001. We used a discount factor of 0.99 for calculating expected rewards and run 16 parallel threads for each experiment.

Model		Parameters	Easy		Medium		Hard	
			MT	ZSL	MT	ZSL	MT	ZSL
Imitation Learning	BC Concat	5.21M	0.86	0.71	0.23	0.15	0.20	0.15
	BC GA	5.09M	**0.97**	0.81	0.30	0.23	**0.36**	0.29
	DAgger Concat	5.21M	0.92	0.73	0.45	0.23	0.19	0.13
	DAgger GA	5.09M	0.94	**0.85**	**0.55**	**0.40**	0.29	**0.30**
Reinforcement Learning	A3C Concat	3.44M	1.00	0.80	0.80	0.54	0.24	0.12
	A3C GA	3.39M	1.00	**0.81**	**0.89**	**0.75**	**0.83**	**0.73**

Table 1: The accuracy of all the models with Concatenation and Gated-Attention (GA) units. A3C Concat and BC Concat are the adapted versions of Misra, Langford, and Artzi (2017) and Mei, Bansal, and Walter (2015) respectively for the proposed environment. All the accuracy values are averaged over 100 episodes.

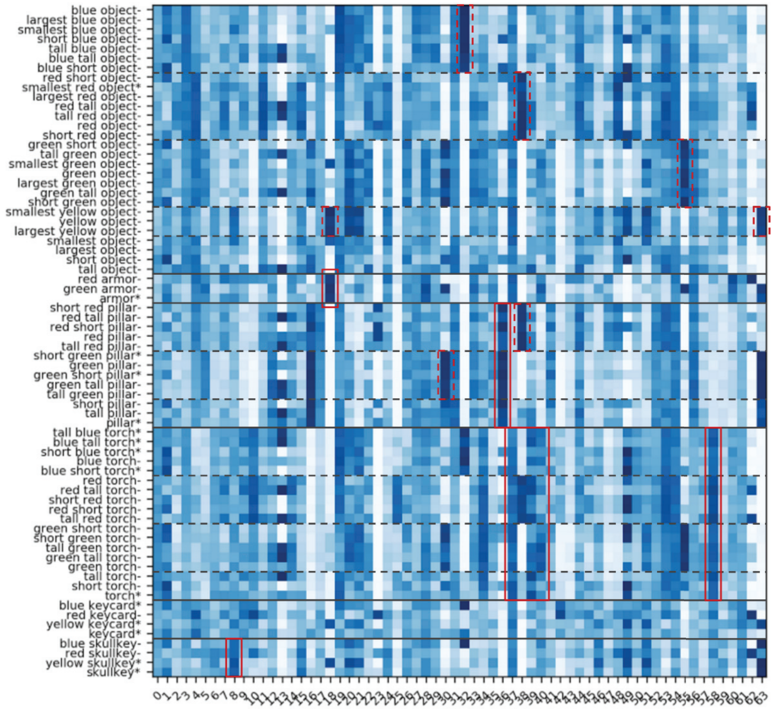

Figure 7: Heatmap of the values of the 64-dimensional attention vector for different instructions grouped by object type and sub-grouped by object color. The test instructions are marked by *. The red boxes indicate that certain dimensions of the attention vector get activated for particular attributes of the target object referred in the instruction.

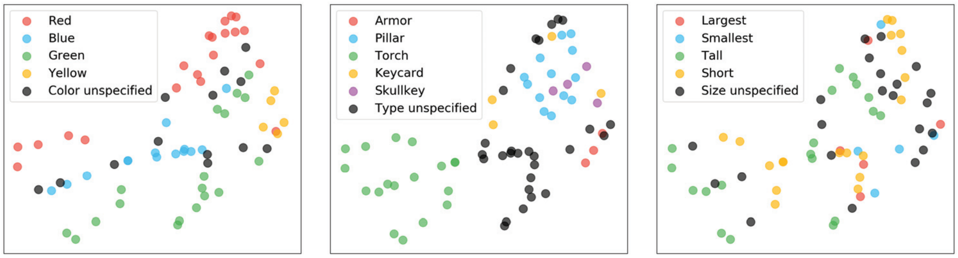

Figure 8: The t-SNE visualization of the attention vectors showing clusters based on object color, type and size.

Results & Discussions

For all the models described in section , the performance on both Multitask and Zero-shot Generalization is shown in Table 1. The performance of A3C models on Multitask Generalization during training is plotted in Figure 6.

Performance of GA models: We observe that models

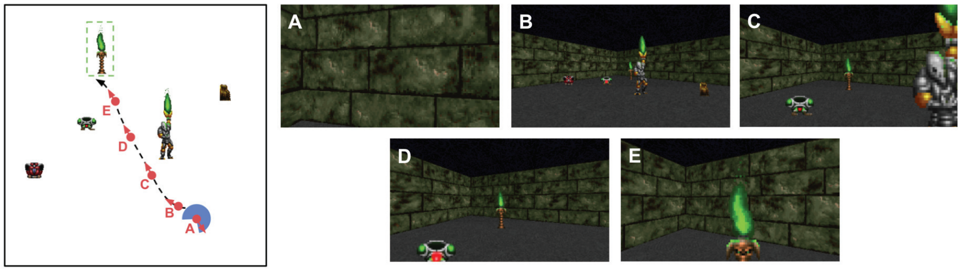

Figure 9: This figure shows an example of the A3C policy execution at different points for the instruction ‘Go to the short green torch’. *Left*: Navigation map of the agent, *Right*: frames at each point. **A:** Initial frame: None of the objects are visible. **B:** agent has turned so that objects are in the field of view. **C :** agent successfully avoids the tall green torch. **D :** agent moves towards the short green torch. **E :**agent reaches target.

with the Gated-Attention (GA) unit outperform models with the Concatenation unit for Multitask and Zero-Shot Generalization. From Figure 6 we observe that A3C models with GA units learn faster than Concat models and converge to higher levels of accuracy. In hard mode, GA achieves 83% accuracy on Multitask Generalization and 73% on Zero-Shot Generalization, whereas Concat achieves 24% and 12% respectively and fails to show any considerable performance. For Imitation Learning, we observe that GA models perform better than Concat, and that as the environment modes get harder, imitation learning does not perform very well as there is a need for exploration in medium and hard settings. In contrast, the inherent extensive exploration of the reinforcement learning algorithm makes the A3C model more robust to the agent's location and covers more state trajectories.

Policy Execution : Figure 9 shows a policy execution of the A3C model in the hard mode for the instruction *short green torch*. In this figure, we demonstrate the agent's ability to explore the environment and handle occlusion. In this example, none of the objects are in the field-of-view of the agent in the initial frame. The agent explores the environment (makes a 300 degree turn) and eventually navigates towards the target object. It has also learned to distinguish between a *short green torch* and *tall green torch* and to avoid the tall torch before reaching the short torch[3].

Analysis of Attention Maps: Figure 7 shows the heatmap for values of the attention vector for different instructions grouped by object type of the target object (additional attention maps are given in the supplementary material). As seen in the figure, dimension 18 corresponds to ‘armor’, dimensions 8 corresponds to the ‘skullkey’ and dimension 36 corresponds to the ‘pillar’. Also, note that there is no dimension which is high for all the instructions in the first group. This indicates that the model also recognizes that the word ‘object’ does not correspond to a particular object type, but rather refers to any object of that color (indicated by dotted red boxes in 7). These observations indicate that the model is learning to recognize the attributes of objects such as color and type, and specific feature maps are gated based on these attributes. Furthermore, the attention vector weights on the test instructions (marked by * in figure 7) also indicate that the Gated-Attention unit is also able to recognize attributes of the object in unseen instructions. We also visualize the t-SNE plots for the attention vectors based on attributes, color and object type as shown in Figure 8. The attention vectors for objects of red, blue, green, and yellow are present in clusters whereas those for instructions which do not mention the object's color are spread across and belong to the clusters corresponding to the object type. Similarly, objects of a particular type present themselves in clusters. The clusters indicate that the model is able to recognize object attributes as it learns similar attention vectors for objects with similar attributes.

Conclusion

In this paper we proposed an end-to-end architecture for task-oriented language grounding from raw pixels in a 3D environment, for both reinforcement learning and imitation learning. The architecture uses a novel multimodal fusion mechanism, *Gated-Attention*, which learns a joint state representation based on multiplicative interactions between instruction and image representation. We observe that the models (A3C for reinforcement learning and Behavioral Cloning/DAgger for imitation learning) which use the Gated-Attention unit outperform the models with concatenation units for both Multitask and Zero-Shot task generalization, across three modes of difficulty. The visualization of the attention weights for the Gated-Attention unit indicates that the agent learns to recognize objects, color attributes and size attributes.

Acknowledgements

We would like to thank Prof. Louis-Philippe Morency and Dr. Tadas Baltruaitis for their valuable comments and guidance throughout the development of this work. This work was partially supported by BAE grants ADeLAIDE FA8750-16C-0130-001 and ConTAIN HR0011-16-C-0136.

References

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

[3] Demo videos: https://goo.gl/rPWlMy

Bagnell, J. A. 2015. An invitation to imitation. Technical report, DTIC Document.

Beetz, M.; Klank, U.; Kresse, I.; Maldonado, A.; Mösenlechner, L.; Pangercic, D.; Rühr, T.; and Tenorth, M. 2011. Robotic roommates making pancakes. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, 529–536. IEEE.

Bollini, M.; Tellex, S.; Thompson, T.; Roy, N.; and Rus, D. 2013. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, 481–495. Springer.

Chao, C.; Cakmak, M.; and Thomaz, A. L. 2011. Towards grounding concepts for transfer in goal learning from demonstration. In *Development and Learning (ICDL), 2011 IEEE International Conference on*, volume 2, 1–6. IEEE.

Chaplot, D. S., and Lample, G. 2017. Arnold: An autonomous agent to play fps games. In *AAAI*, 5085–5086.

Chaplot, D. S.; Lample, G.; Sathyendra, K. M.; and Salakhutdinov, R. 2016. Transfer deep reinforcement learning in 3d environments: An empirical study. In *NIPS Deep Reinforcemente Leaning Workshop*.

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

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Chu, V.; McMahon, I.; Riano, L.; McDonald, C. G.; He, Q.; Perez-Tejada, J. M.; Arrigo, M.; Fitter, N.; Nappo, J. C.; and Darrell, T. 2013. Using robotic exploratory procedures to learn the meaning of haptic adjectives. In *Robotics and Automation (ICRA)*, 3048–3055.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dhingra, B.; Liu, H.; Yang, Z.; Cohen, W.; and Salakhutdinov, R. 2017. Gated-attention readers for text comprehension. *ACL*.

Fasola, J., and Mataric, M. J. 2013. Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots. In *Intelligent Robots and Systems (IROS)*, 143–150. IEEE.

Guadarrama, S.; Riano, L.; Golland, D.; Go, D.; Jia, Y.; Klein, D.; Abbeel, P.; Darrell, T.; et al. 2013. Grounding spatial relations for human-robot interaction. In *IROS*, 1640–1647. IEEE.

Guadarrama, S.; Rodner, E.; Saenko, K.; Zhang, N.; Farrell, R.; Donahue, J.; and Darrell, T. 2014. Open-vocabulary object retrieval. In *Robotics: science and systems*, volume 2, 6. Citeseer.

Horn, R. A. 1990. The hadamard product. In *Proc. Symp. Appl. Math*, volume 40, 87–169.

Huber, P. J. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics* 35(1):73–101.

Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; and Jaśkowski, W. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. *arXiv preprint arXiv:1605.02097*.

Kulick, J.; Toussaint, M.; Lang, T.; and Lopes, M. 2013. Active learning for teaching a robot grounded relational symbols. In *IJCAI*.

Kulkarni, T. D.; Saeedi, A.; Gautam, S.; and Gershman, S. J. 2016. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Lample, G., and Chaplot, D. S. 2016. Playing fps games with deep reinforcement learning. *arXiv preprint arXiv:1609.05521*.

LeCun, Y.; Bengio, Y.; et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10):1995.

Lemaignan, S.; Ros, R.; Sisbot, E. A.; Alami, R.; and Beetz, M. 2012. Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction. *International Journal of Social Robotics* 4(2):181–199.

Mei, H.; Bansal, M.; and Walter, M. R. 2015. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. *arXiv preprint arXiv:1506.04089*.

Misra, D. K.; Langford, J.; and Artzi, Y. 2017. Mapping instructions and visual observations to actions with reinforcement learning. *arXiv preprint arXiv:1704.08795*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 1928–1937.

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

Ross, S.; Gordon, G. J.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning.

Russell, S., and Norvig, P. 1995. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs* 25:27.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

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

Tellex, S. A.; Kollar, T. F.; Dickerson, S. R.; Walter, M. R.; Banerjee, A.; Teller, S.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2).

Wu, Y.; Zhang, S.; Zhang, Y.; Bengio, Y.; and Salakhutdinov, R. 2016. On multiplicative integration with recurrent neural networks. In *NIPS*.

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

Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J. J.; Gupta, A.; Fei-Fei, L.; and Farhadi, A. 2016. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *arXiv preprint arXiv:1609.05143*.