Superlative Learning in Semantic Goal Navigation

Damon George
georgdam@oregonstate.edu

Jacob Krantz
krantzja@oregonstate.edu

Joseph Valencia
valejose@oregonstate.edu

Abstract

Superlative reasoning, which requires identifying an instance from a set with the highest or lowest degree of some attribute, is often overlooked in the analysis of deep networks, despite the increased complexity of superlative tasks as compared to non-superlative tasks. We attempt to determine whether deep networks can learn superlative tasks and whether the increased complexity results in decreased performance. We first test a simple visual superlative task by teaching a CNN to identify the longest lines or largest polygons in an image. Our simple CNN learns this task easily. We further extend the problem to semantic goal navigation (SGN), where an agent must navigate an environment following language instructions. We extend the gym-miniworld virtual environment to support language grounding and superlative instructions, and train a Gated-Attention network using PPO on both superlative and non-superlative instructions. Unfortunately, results indicate that our agent was unable to learn SGN in this environment, eliminating our ability to analyze the performance of the network on superlative tasks.

1. Introduction

Superlatives are used to describe something as having the highest or lowest degree of some quality, and are an essential aspect of language and question answering. A simple superlative question could be "Who is the tallest basketball player?". Although the occasional deep learning paper incorporates superlative reasoning tasks, such as [1], no treatment has been given to distinguishing the performance of deep networks between superlative and non-superlative tasks in the visual and visual-linguistic domains. Non-superlative reasoning in this case could be any task defined without superlatives, such as "Who is player number 43?".

This paper specifically seeks to analyze the performance of deep networks on superlative and non-superlative visual tasks. We aim to determine whether deep networks can successfully learn superlatives, and if so, whether superlative tasks are more challenging than non-superlative tasks. Superlative tasks have an inherently higher difficulty than non-superlative tasks due to the need for the network to 1) identify specific instances in a set, 2) compare some attribute amongst those instances, and 3) select the instance with either the highest or lowest degree of the attribute in question. We suspect that deep networks perform worse on superlative tasks as compared to non-superlative tasks due to this difficulty and the inability of convolutional neural networks (CNN) to process precise absolute position information [9].

To answer these questions, we first analyze the performance of a simple CNN architecture on basic superlative tasks, such as identifying the tallest line or the largest polygon. Testing with simple lines and polygons shows that even a vanilla CNN has no difficulty in making predictions about basic superlative qualities.

To increase the difficulty of the tasks, we expand the problem to semantic goal navigation (SGN), in which an agent learns to navigate an environment in response to templated language instructions [1]. One popular simulated environment commonly used for this purpose is ViZDoom [6] based on the video game Doom, in which a sample instruction for the agent could be "Go to the green torch". However, due to installation issues with VizDoom, in this work we instead use the simpler *gym-miniworld* 3D simulator [2]. To analyze the performance of deep networks on superlative tasks in this environment, we first implement custom data generation and language grounding in the environment to allow us to train reinforcement agents with instructions split between superlative and non-superlative descriptors. Next, we implement our agent with a Gated-Attention network trained using proximal policy optimization [1, 12]. Finally, we train and test our agents using separate splits of superlative and non-superlative instructions in both easy and hard configurations of the environment. Due to challenges in hyperparameter tuning, strong SGN results in *gym-miniworld* were never achieved. Among our partial results, we observe that our agent achieved higher accuracy with non-superlative instructions than it did with superlative instructions despite equal training exposure.

2. Related Work

As stated, superlatives have received little focus in computer vision research, despite the focus they have received in the language processing community. A treatment of superlatives can be found in [10], in which the author describes the many computational challenges involved in answering superlative questions. Due to these challenges, it is possible that deep networks may perform poorly on superlative tasks.

Furthermore, Liu *et al.* have shown a failure of CNNs to translate between activated pixels in an image and their coordinates in the image [9]. The authors solve this problem by adding pixel coordinates as additional input channels to their CNN. However, this failure of the non-augmented CNN shows that deep networks may struggle with the positional instance identification and comparison that is required for answering superlative questions.

The Gated-Attention network paper from Chaplot *et al.* is the most notable prior attempt at learning superlatives in the visual domain [1]. The authors use deep reinforcement learning to train an agent to perform semantic goal navigation in the ViZDoom environment. They provide the agent with textual instructions containing specific objects, colors, and occasionally superlatives. Their network achieves an impressive 89% accuracy in medium difficulty and 83% accuracy in hard difficulty settings. However, the authors do not explicitly evaluate the network's difference in performance between the superlative and non-superlative tasks. Further, without controlling object generation in episodes, non-superlative reasoning can be used to solve superlative instructions. Such an instance would be to navigate to the tallest torch, but only one torch exists in the scene. In this paper, we implement the same Gated-Attention network as used by Chaplot *et al.* so we can compare the performance of our implementation in *gym-miniworld* to their results. Further work in multitask learning extends SGN approaches to include embodied question answering (EQA) [13]. We note that superlative reasoning in SGN may also be present in natural language-based EQA despite not being present in the existing EQA task definition [4]. This suggests that a more focused treatment of superlatives may benefit vision and language tasks in future research.

3. Learning Simple Visual Superlatives

We devise several toy problems as an initial test of our hypothesis that CNNs struggle to compare the spatial qualities of multiple objects. First, we generate a dataset of black and white images containing four lines of varying heights and train a simple CNN classifier to predict the tallest line. The class labels $i \in [0, 3]$ correspond to the line at the ith position from left to right. Next, we generate a similar image dataset of circles and regular polygons (triangles, squares, pentagons, and hexagons). The shapes vary in color and are placed against a black background. We train the network to predict the shape with the largest radius. The results on these two datasets are summarized in Table 1. The rapid convergence to high accuracy illustrates that CNNs are capable of predicting superlatives in highly constrained domains. This success motivates our exploration of the more complex task of semantic goal navigation.

Dataset	Image Dimension	Accuracy	Convergence
Lines	(24×24)	99.9	1
Shapes	(224×224)	95.0	5

Table 1: Results on toy datasets. Accuracy is on the validation set. Convergence refers to the number of epochs to $> 95\%$ accuracy.

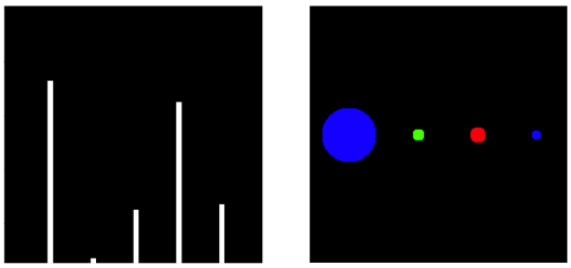


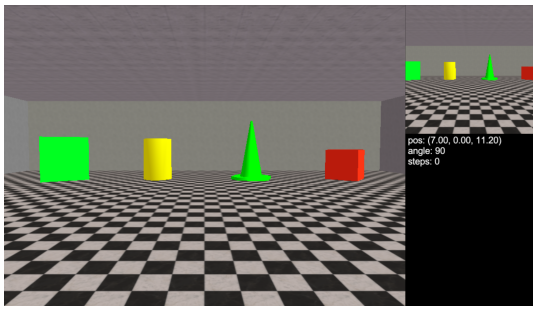
Figure 1: An example from each of our simple image datasets. Tallest line (left). Largest ball (right).

4. Task Definition: SGN-MiniWorld

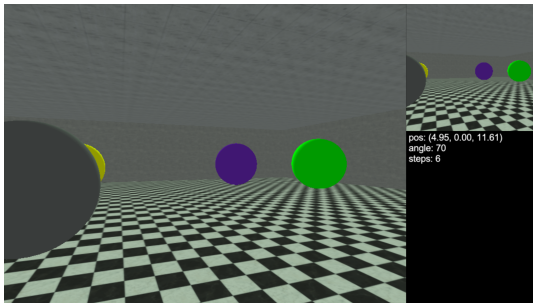
We develop a reinforcement learning environment called *sgn-miniworld* on top of the *gym-miniworld* GitHub repository [8]. This package provides a default scene consisting of a 3-dimensional simulated room, which we modify to approximate the original VizDoom environment from [1]. It comes with pre-existing object types and logic for ensuring that objects do not overlap within the scene. We constrain the placement and attributes of these objects and ensure that a given room configuration can be replicated repeatedly during training and testing.

Every room configuration includes a text instruction for the agent as the basis for the language grounding task. The grammar for generating instructions is shown below. Note that we have a total of 174 unique instructions, but each one can be associated with multiple room configurations:

S $\rightarrow$ go to the SUP COL OBJ
SUP $\rightarrow$ closest | farthest | tallest | shortest | ϵ
COL $\rightarrow$ red | green | blue | yellow | purple | gray | ϵ
OBJ $\rightarrow$ ball | box | barrel | cone | object



(a) Easy



(b) Hard

Figure 2: Difficulty Settings

We establish two difficulty settings for *sgn-miniworld*: (1) *Easy*, where each scene contains exactly four objects. All objects sit along a single line at a fixed distance in the room. When the instruction is based on a 'tallest' or 'shortest' superlative, the object heights are drawn from a regular grid on $[0.5, 2]$ with a step-size of 0.25 in order to guarantee a clear separation in heights. The agent spawns at the opposite side of the room facing the objects. (2) *Hard*, where the environment generates 4-6 objects with random heights in $[0.5, 2]$ in random positions within the room. The agent also spawns in a random position and orientation. Figure 2 shows example room configurations under both settings.

5. Methods

We develop a method for solving the *sgn-miniworld* task, adding neural network components that will help the agent successfully navigate its environment, particularly in response to superlative instructions. Such an agent needs spatial reasoning for observing the environment, temporal reasoning for remembering objects it has seen (particularly for the hard mode), and crucially, multi-modal reasoning to ground a linguistic (possibly superlative) instruction in the 3D environment. These reasoning systems build from our line and polygon visual experiments by introducing action in a 3D space.

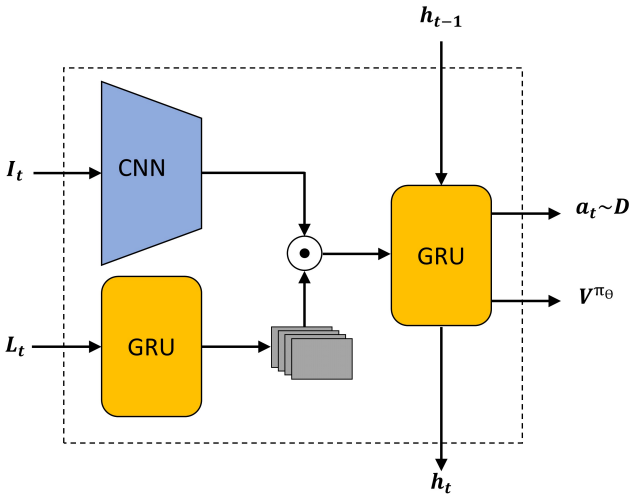


Figure 3: Network architecture that follows the same encoding procedure as [1]. Note that our action and value decoder is a GRU instead of an LSTM. At inference time we sample from the action distribution $\boldsymbol{D}$.

5.1. Architecture: Gated Attention

A natural network to use for the semantic goal navigation task is the Gated-Attention network developed by Chaplot *et al.*[1]. Since we are more concerned with the task and superlative analysis than developing new architectures in this work, we keep our version of this architecture fixed throughout the experiments. We expect the Gated-Attention network to be sufficient for *sgn-miniworld* because it achieves a high accuracy in the original SGN task.

The architecture itself is a straightforward multi-modal system. The image I at timestep t is encoded as $\mathcal{V}_t = f(I_t, \theta_{conv})$, where f performs three down-sampling convolutions. The instruction L is encoded using a GRU [3]: $\mathcal{I}_t = g(L, \theta_{GRU})$ followed by a linear layer activated with the sigmoid. The instruction encoding is expanded to $M_t(\mathcal{I}_t)$ and then used to attend over the image encoding via an element-wise multiplication: $O_t = \mathcal{V}_t \odot M_t(\mathcal{I}_t)$ where $\odot$ is the Hadamard product. The result of this multi-modal fusion is passed through a fully connected layer then a GRU to maintain temporal memory: $h_t = GRU(h_{t-1}, fc(O_t))$. This differs from the original architecture that uses an LSTM [5]. Finally, this newly computed hidden state is passed through two separate linear layers to produce a distribution over the action space and a value estimate. With an image observation space of $(224 \times 224 \times 3)$, our network, which we also call the policy, has around 1.7M parameters to optimize.

5.2. Training

We seek to find a set of parameters θ for our policy π that minimize the difference between π_θ and the optimal

π^*. To do so, we frame our environment as a Markov decision process (MDP) and apply proximal policy optimization [12]. PPO is a policy gradient method, and further, an actor critic method. The gradient of our policy π_θ is directly optimized (the actor) as well as the gradient of the advantage estimate function $A^{\pi_\theta}(s,a)$ (the critic). A commonly optimized actor-critic objective is:

$$L^{PG}(\theta) = \mathbb{E}_{\pi_\theta}\left[\log \pi_\theta(s,a)\hat{A}^{\pi_\theta}(s,a)\right]. \quad (1)$$

Alternatively, the PPO objective restricts the size of the gradient update by clipping the ratio of the current policy to the old:

$$L^{CLIP}(\theta) = \mathbb{E}_{\pi_\theta}\left[\min\left(r_t(\theta)A^{\pi_\theta}, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A^{\pi_\theta}\right)\right] \quad (2)$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}.$$

In this work, we apply the PPO update to the Gated-Attention network described in Section 5.1 and optimize using the Adam optimizer [7]. The core of our implementation comes from [8]. The advantage estimate is computed using generalized advantage estimation (GAE) which uses a λ-return with the state value function $V^{\pi_\theta}(s_t)$ [11]. The reward function $R(s,a)$ is an important component of the training regime. We define a reward that combines a timestep penalty, a progress-to-goal reward, and a success reward:

$$\begin{aligned} R(s,a) =& A \\ &+B(distance(s_{t-1}) - distance(s_t)) \\ &+C * \mathbb{1}(success) \end{aligned} \quad (3)$$

where we set $\{A : -0.01, B : 1.0, C : 10.0\}$ for a dense reward setting and $\{A : -0.01, B : 0.0, C : 10.0\}$ for a sparse reward setting.

6. Experiments

To evaluate the performance of our agent in superlative and non-superlative *sgn-miniworld*, we first trained the Gated-Attention network separately in both easy and hard modes of the environment. 3000 unique episodes were used for training, generated randomly from the 174 instructions. Each episode contained four objects, one of which was the target object, and each episode ended when the agent ran into any of the objects or reached the maximum number of steps. We ensure that superlative reasoning is required in each episode by placing multiple objects of the same type; for instance, we place a minimum of two boxes when the instruction includes "tallest box". This is not a guarantee in original SGN. For training, the instructions were split evenly between superlative and non-superlative commands. Both dense and sparse rewards were tested in various trials, as well as various values of forward step size, to determine the best training configuration of the network.

The trained networks were evaluated separately on 1000 random episodes of superlative instructions and 1000 random episodes of non-superlative instructions. The network parameters selected for testing were simply chosen as the parameters resulting in the highest success rate during training, which was computed using a running window 100 episodes long.

6.1. Results

The test results of our network are shown in Table 2 for both easy and hard difficulties. Unfortunately, it is clear that our network was unable to successfully learn to navigate in *sgn-miniworld*. Despite training the networks for multiple days and in multiple configurations, the success rate never reached above 30%. During training, success rates were consistently below 20% on average with a very high variance between episode success windows. However, we can still see in Table 2 that the success rate is indeed higher in easy mode, as we expected. Furthermore, the success rate is higher for non-superlative tasks in both modes of difficulty. However, the high variance in success rate during training means that there is little significance to the slight differences in success between superlative and non-superlative tasks. Table 2 also shows the test results in easy mode using sparse rewards, in which the agent only receives a reward on successful completion of an episode. The very low success rate in this mode indicates that a sparse reward setting is inefficient if not unsuitable for this task.

Mode	Steps	Nav Error	Success Rate
Easy-dense			
Superlative	73.38	4.04	22.7
Non-Superlative	76.85	4.17	23.3
Hard-dense			
Superlative	83.90	6.75	13.8
Non-Superlative	68.09	6.87	15.0
Easy-sparse			
Superlative	78.18	7.41	6.5
Non-Superlative	75.54	7.69	7.0

Table 2: Comparison of results for superlative vs non-superlative instructions. Navigation Error is in meters and Success Rate is a percentage. The different modes refer to the two levels of difficulty and two reward settings.

7. Discussion

In this work, we analyze the performance of deep networks on superlative and non-superlative tasks. We hypothesize that the higher complexity of superlative reasoning will result in poorer performance from deep networks as compared to non-superlative reasoning. In our initial testing of simple visual superlatives using lines and polygons, we see that even a simple CNN classifier can learn superlatives with almost perfect accuracy with only a few epochs. These results show that neural networks can learn simple superlative reasoning and suggests that our hypothesis could be wrong.

Expanding superlative reasoning to a more complex setting involving action language and vision, we consider semantic goal navigation and implement a reinforcement learning agent in the *gym-miniworld* environment, in the process adding the necessary architecture to the simulator in order to use templated language instructions. Despite training multiple agents in easy and hard modes with dense and sparse rewards, we unfortunately never achieve meaningful success in the SGN task. In all cases, higher accuracy is achieved on the non-superlative tasks, which hints at a successful confirmation of our hypothesis. However, success rate on test data never reached above 25%, and despite our attempts to produce meaningful analysis of our results, the low accuracy and high variance of our trained networks makes most analysis uninformative.

Clearly, there are many additional steps necessary to complete this work and achieve meaningful results in comparing the performance of deep networks on superlative and non-superlative tasks in the context of semantic goal navigation. From initial testing of our network, we know that our agent can learn to navigate the *gym-miniworld* environment – it performs the baseline goal of finding a single object without difficulty. However, it is possible that our implementation of Gated-Attention for language grounding is incorrect, or that the many hyper-parameters still require extensive tuning to achieve success in training. Further extensive testing and tuning of our network is therefore still quite necessary.

Another option for improving training is to use a pre-trained ResNet for the CNN. Although Chaplot *et. al*did not require this step, it has the potential to greatly speed up the initial training phase of the image encoder by giving the model a head start in distinguishing shapes. The 2D polygon dataset could also be incorporated as a pre-training task, particularly with more complex formulations like a variable amount of shapes and irregular shapes.

References

[1] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[2] M. Chevalier-Boisvert. gym-miniworld environment for openai gym. `https://github.com/maximecb/gym-miniworld`, 2018.

[3] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[4] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2054–2063, 2018.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, Sep 2016. IEEE. The best paper award.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[8] I. Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr`, 2018.

[9] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution, 2018.

[10] S. Scheible. Towards a computational treatment of superlatives. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 67–72, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

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

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] D. Singh Chaplot, L. Lee, R. Salakhutdinov, D. Parikh, and D. Batra. Embodied multimodal multitask learning. *arXiv preprint arXiv:1902.01385*, 2019.