Memory Based Reinforcement Learning with Transformers for Long Horizon Timescales and Continuous Action Spaces

Shweta Singh shwetasingh@rde.gov.in shweta.singh@research.iiit.ac.in, Sudaman Katti
sudaman.katti19@vit.edu

***Abstract*—** The most well known sequence models make use of complex recurrent neural networks in an encoder-decoder configuration. The model used in this research makes use of a transformer,which is based purely on self-attention mechanism, without relying on recurrence at all. More specifically, encoders and decoders which make use self attention and operate based on a memory are used. In this research work, results for various 3D visual and non-visual reinforcement learning tasks designed in Unity software were obtained. Convolutional neural networks, more specifically, nature CNN architecture is used for input processing in visual tasks and comparison with standard long short-term memory (LSTM) architecture is performed for both visual tasks based on CNNs and non-visual tasks based on coordinate inputs. This research work combines the transformer architecture with the proximal policy optimization technique used popularly in reinforcement learning for stability and better policy updates while training, especially for continuous action spaces, which are used in this research work. Certain tasks in this paper are long horizon tasks which carry on for a longer duration and require extensive use of memory based functionalities like storage of experiences and choosing of appropriate actions based on recall. The transformer, which makes use of memory and self-attention mechanism in an encoder-decoder configuration proved to have better performance when compared to LSTM in terms of exploration and rewards achieved. Such memory based architectures can be used extensively in the field of cognitive robotics and reinforcement learning.

***Keywords*—** Convolutional neural networks, Reinforcement learning, Self-Attention, Transformers, Unity

I. Introduction

In various sequence-to-sequence problems such as the neural machine translation, the popular approaches were based on the use of RNNs in an encoder-decoder fashion. However, these architectures have a great limitation when working with long sequences, their ability to retain information from the first elements gets lost when new elements are incorporated into the sequence. In the encoder, the hidden state in every step is associated with a certain element in the input sentence, usually based on how recent it is. Therefore, if the decoder only accesses the last hidden state of the decoder, it will lose the important information about the first elements of the sequence. Thus, to deal with this problem, a novel concept was introduced: the attention mechanism.

Instead of paying attention to the last state of the encoder as in the case of RNNs, in each step of the decoder. all the states of the encoder which are able to access information about all the elements of the input sequence are considered. This is the main working principle of the attention mechanism. This mechanism allows the decoder to assign greater weight or importance to a specific element of the input for each element of the output. Learning is done in every step to focus on the right element of the input to predict the next output element.

II. Methodology

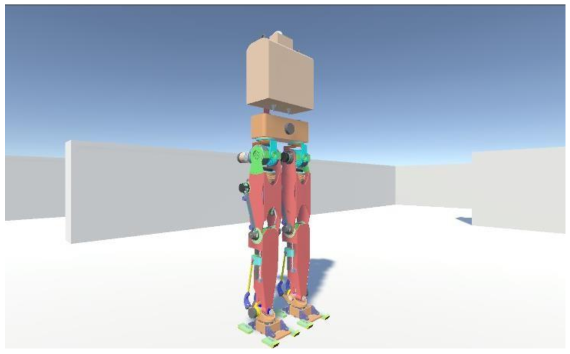


Fig.1 Biped Model

This research work makes use of the ML agents toolkit in unity for implementation of reinforcement learning algorithms through python. The environments for all the tasks were created in Unity. A model of a simple Biped robot which can transverse across planes was used as the agent for all the tasks. Continuous action spaces were used. Nature CNN architecture was used for input preprocessing in case of visual tasks.

A. Proximal Policy Optimisation- Proximal Policy Optimisation (PPO) is a fairly recent advancement in the field of Reinforcement Learning, which provides an upgrade on Trust Region Policy Optimization (TRPO). PPO aims to strike a balance between key factors like ease of implementation and tuning, sample complexity, sample efficiency and trying to compute updates at each step that minimizes the cost function

PPO was chosen for this research work as opposed to standard methods like DQN as it effective for continuous action spaces

B. Episodic scene Memory- The scene memory consists of all past observations per time step in an embedded form and the memory is updated at each time step. The decoder of the attention-based policy network makes use of the updated scene memory to compute a distribution over actions. the memory grows linearly with the episode length. Each observation is stored separately in the memory and aggregation of the information is only done when computing an action. As each received observation is embedded into low dimensional vectors, one can easily store hundreds of time steps on the hardware devices. While RNNs are restricted to a fixed-size state vector, which usually can only capture short-term dependencies.

C. Attention-based Policy Network-

The policy network π(a|o, M) makes use of the current observation and the scene memory to compute a distribution over the action space. Firstly, encoding of the memory by transforming each memory element in the context of all other elements is done. This step has the potential to capture the spatio-temporal dependencies in the environment. Then, an action is decoded according to the current observation, using the encoded memory as the context.

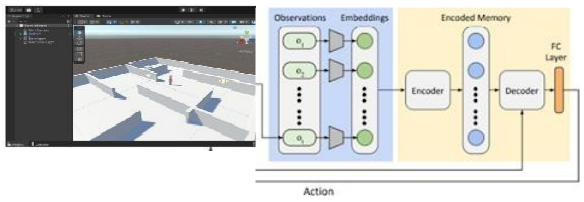


Figure 2. The Transformer architecture

Attention mechanism- The transformer model uses self-attention for encoding of the memory M. More specifically, M is used as both inputs to the attention block. This transforms each embedded observation by using its correlation with other past observations. This is because the three vectors U, K and V responsible for self attention are defined based on the inputs to the attention block (through weights), which in this case are M and M. Furthermore, a modified version of ResNet was used for better input processing.

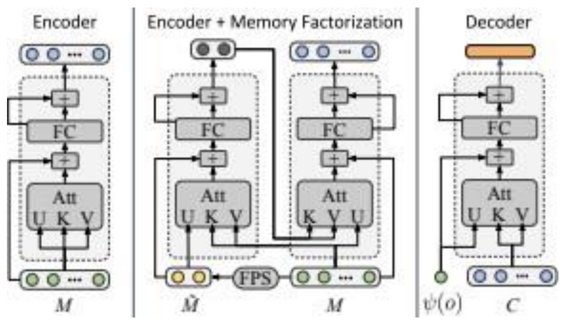


Figure 3. Attention mechanism

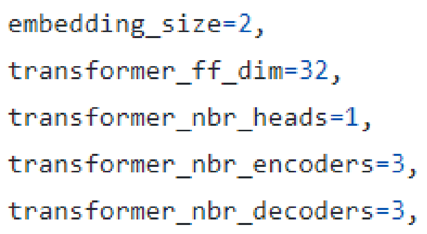
Encoder- The Transformer model makes use of self-attention to encode the memory M. More specifically, M is used as both inputs of the attention block. This transforms each embedded observation by using its relations to other past observations: Encoder (M) = AttBlock (M, M)

Decoder- The decoder is supposed to produce actions based on the current observation given the context C, which in this case, is the encoded memory. It applies similar machinery as the encoder, with the notable difference that the query in the attention layer is the embedding of the current observation ψ(o).

D. Implementation Details-

At each timestep, the observation space will consist of the (x, y, z) coordinate values of the agent in the environment. The action space here is continuous in nature with three possible actions- movement in X direction, movement in Z direction and rotation around Z axis (turning). Results with memory size of 1 and 128 were plotted and comparison was done with LSTM. Training in all 3 cases was done for 10000 timesteps.

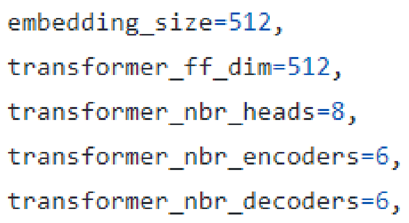
Transformer parameters for non visual task-



```
embedding_size=2,
transformer_ff_dim=32,
transformer_nbr_heads=1,
transformer_nbr_encoders=3,
transformer_nbr_decoders=3,
```

Figure 4. Transformer parameters for non visual task

Transformer parameters for visual task-



```
embedding_size=512,
transformer_ff_dim=512,
transformer_nbr_heads=8,
transformer_nbr_encoders=6,
transformer_nbr_decoders=6,
```

Figure 5. Transformer parameters for visual task

III. Results and Discussions

A. Maze Task-
The goal of the agent in this maze navigation environment is to navigate to the goal and get a reward of 2000. Every time this happens, the episode will end.

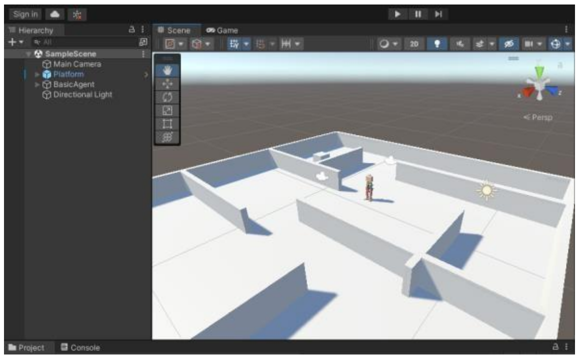


Figure 6. Maze Task Environment

The agent will also get a reward of 0.2 everytime it touches the wall right to the left of the goal, Using the Stable Baselines Package, a custom policy for a 3D environment which combines a transformer and memory architecture was created and implemented through the proximal policy optimisation algorithm in Stable Baselines. Comparison with LSTM was done.

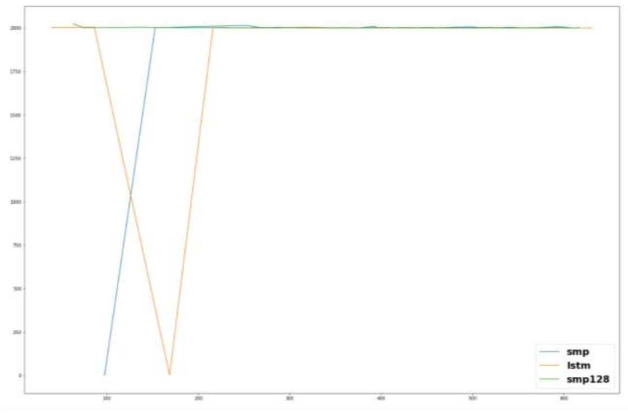


Figure 7. Comparison of rewards during training

The above graph represents the episodic rewards during training for transformer based memory and LSTM. The blue line corresponds to the transformer with memory size of 1 while the green line corresponds to the transformer with memory size of 128. The orange line corresponds to LSTM. Amongst the three, the transformer with a memory size of 128 managed to reach the goal and get the reward of 2000 more consistently throughout training. LSTM converged to a fixed path without trying to maximize the goal from the adjacent wall. This convergence was not observed in the case of the transformer. Overall, higher rewards were observed in the case of the transformer with memory architecture.

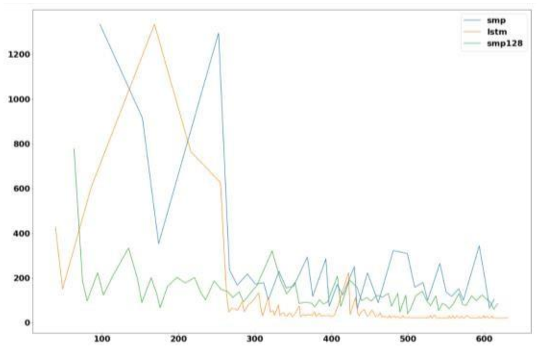


Figure 8. Comparison of episode lengths during training

The above graph represents the episode durations during training for Transformer and LSTM. Since the episode ends once the agent reaches the goal, this graph also represents the path length taken to reach the goal. The blue line corresponds to Transformer with memory size of 1 whereas the green line corresponds to Transformer with memory size of 128. The orange line corresponds to LSTM. Convergence to a fixed path was observed in case of LSTM. The transformer tried multiple variations of paths and tried to touch the adjacent wall multiple times as compared to LSTM in order to maximize the reward. As a result, convergence to a fixed path was not observed. However, these variations in path taken were considerably lower in the case of LSTM

B. Exploration Task-

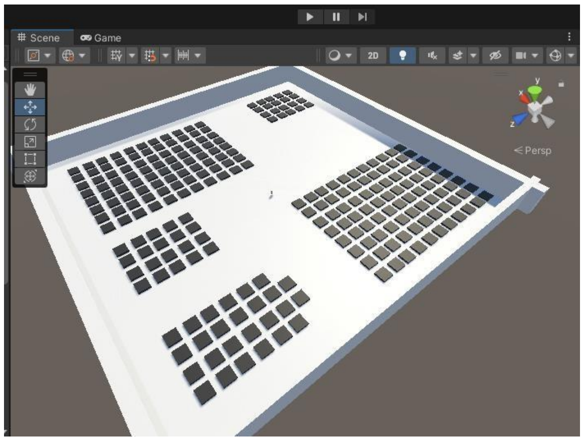


Figure 9. Exploration task environment

The main aim of this task is to move through the arena and explore as much of it as possible. These black tiles totalling at 242 are present throughout the arena and provide a reward of 1 when the agent touches them. This task tests the agent's ability to explore. Visual inputs of size 300 * 300 with 3 color channels (RGB) were used here and nature CNN architecture was used for input preprocessing. For both the transformer and

LSTM, training was done for 50000 timesteps and the amount of covered tiles were plotted.

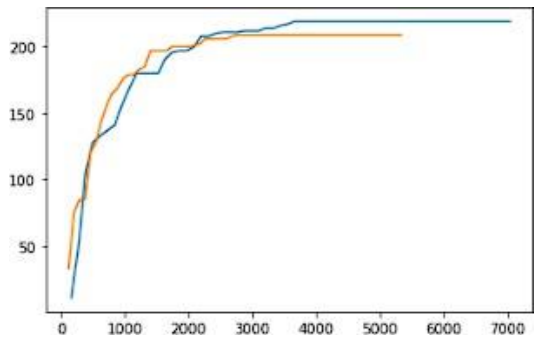


Figure 10. Comparison of LSTM and transformer

In the above figure, the orange line represents LSTM and the blue line represents the transformer. LSTM managed to cover 209 tiles whereas the transformer based memory architecture managed to cover 219 tiles.

- TABLE ONE - PERCENTAGE OF COVERAGE-

Algorithm	Percentage of covered tiles
Transformer based memory	90.4 % (219)
LSTM	86.3 % (209)

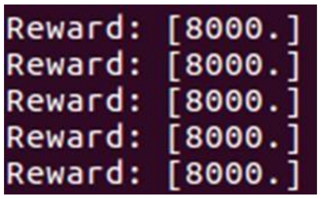
In the coverage task, the transformer agent performed higher exploration as compared to LSTM, thus the transformer memory had a more varied set of observations to update from.

C. The Long Horizon Search Task-

In this task, the main goal of the agent is to discover new objects. For this purpose, the agent makes use of 300 by 300 visual observations with 3 color channels (RGB). 4 categories of objects with different colors are used- cylinder, capsule, sphere and cube. The arena consists of 4 rooms and no room consists of all 4 categories. The agent gets a reward of 2000 every time it discovers a new category. Therefore, the maximum reward that can be achieved for an episode is 8000.

After 1 million training steps, since considerable decrease in entropy was observed, we decided to stop the training and move on to testing.

Test 1- The first test involved performing the search task in the same environment with all the objects being at the same place as in training. For this purpose, the trained SMT model was loaded first. 5 episodes were observed. The following were the results-



```
Reward:  [8000.]
Reward:  [8000.]
Reward:  [8000.]
Reward:  [8000.]
Reward:  [8000.]
```

Figure 11- Test 1 results

The camera view of the agent was also displayed throughout training as shown in the subsequent figure.

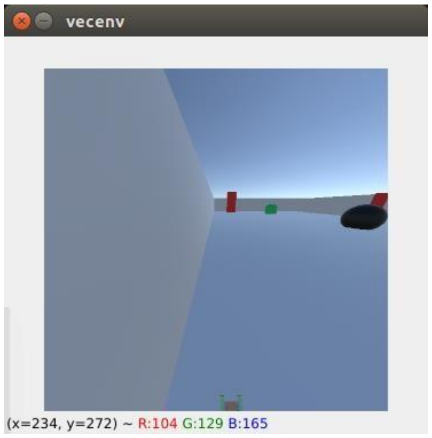
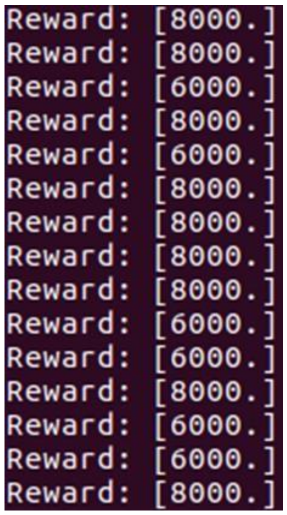


Figure 12- Agents camera view

The agent managed to get the reward of 8000 and thus discovered all 4 object classes for all 5 episodes. One more important characteristic observed was that once the agent got the object of a particular class, it avoided other objects of that class. In this case however, the movements were quick and following a trajectory.

Test 2- In this scenario, the arrangement was the same as during training with the exception that the position of two objects was changed slightly. The following the results-



```
Reward:  [8000.]
Reward:  [8000.]
Reward:  [6000.]
Reward:  [8000.]
Reward:  [6000.]
Reward:  [8000.]
Reward:  [8000.]
Reward:  [8000.]
Reward:  [8000.]
Reward:  [6000.]
Reward:  [6000.]
Reward:  [8000.]
Reward:  [6000.]
Reward:  [6000.]
Reward:  [8000.]
```

Figure 13- Test 2 results

The agent managed to get a reward of 8000 for 9 episodes and a reward of 6000 for 6 episodes. Since the visual observations, especially from a distance, were not that different from training in this case, the agent decided to stick to its training trajectory. The small variations in positions probably went unnoticed due to the agent looking the wrong way or some other reasons in some episodes resulting in 6000 reward. The agent in this case followed the fixed trajectory from training and only changed after not getting the anticipated trajector reward

Test 3-

In this scenario, the positioning and arrangement of objects for all 4 rooms was changed internally. Testing was done for 30 episodes. The agent managed to get a reward of 8000 for 26 out of 30 episodes, the reward was 6000 for the remaining 4 episodes

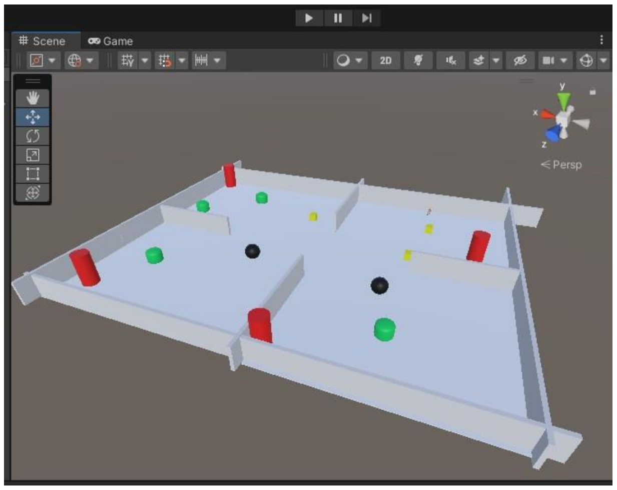


Figure 14. Original scenario for search task during training

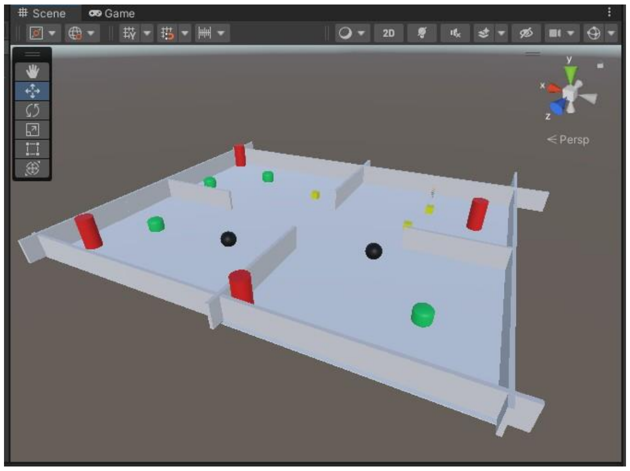


Figure 15. Scenario for search task during test 3

In this case, since the visual observations for all the rooms were considerably distinct, the agent decided to not go with its default training trajectory. The movements were not swift and a lot of time was spent in observation and alignment with respect to targets. However, the agent managed to successfully detect targets and get a reward of 8000 for 26 out of 30 episodes and a reward of 6000 for all other episodes.

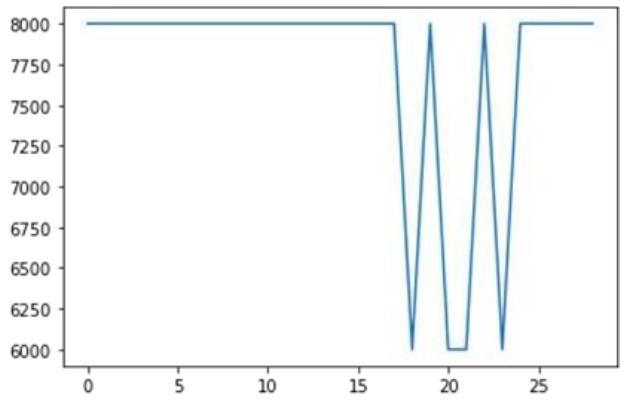


Figure 16. Test 3 results

Test 4-
In this scenario, the spawn location of the agent was changed, the arrangement of the objects in each room was changed drastically. The episode duration was doubled in this case.

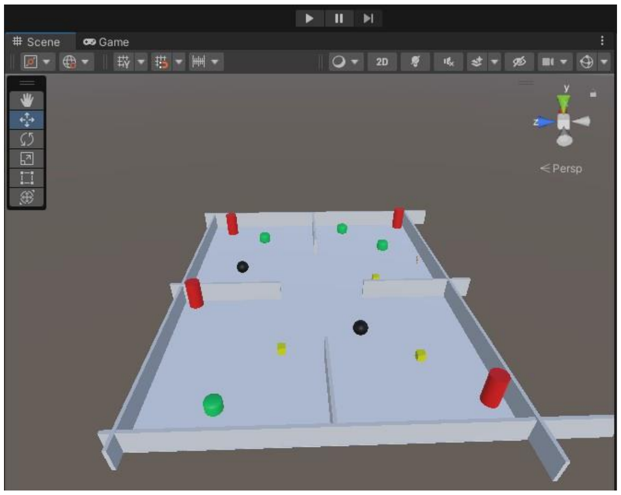


Figure 17. Scenario for search task during test 4

Swapping of objects between rooms was also done. Testing was done for 15 episodes.

Out of 15 episodes, the agent managed to get a reward of 8000 (discover all 4 target classes) for 10 episodes. The rest of the episodes had a reward of 6000.

D. The Long Horizon Multistage Task-

The multistage task consists of 3 phases out of which, the middle one is a distractor phase.

In the first phase the agent is supposed to go to any of the 4 capsules in front of it. After this, the agent is transported to the distractor phase wherein it gets a reward of 1 every time it touches a white cube, of which there are multiple throughout the phase.

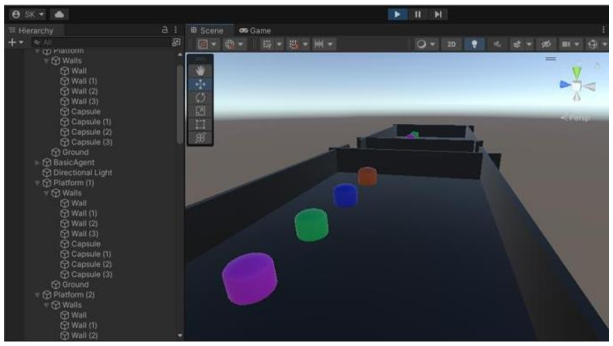


Figure 18- Multistage Task, 2nd phase

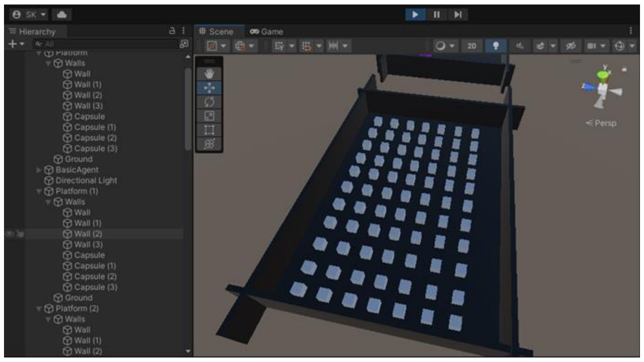


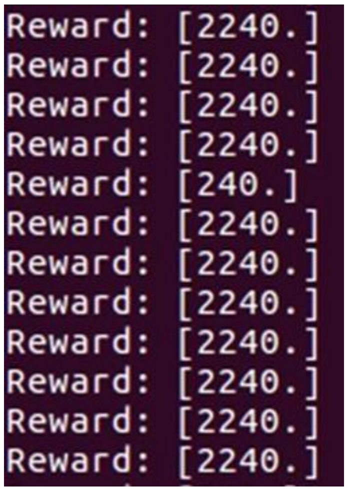
Figure 19. Multistage Task, distractor phase

Once the agent gets a reward of 45 in this phase, it is transported to the third phase in which it is supposed to recall the capsule it picked in phase one and go towards it purely based visual inputs processed by nature cnn architecture.

The distractor phase in this task acts as a mechanism which makes the agents forget its main goal and acts as a test of its memory, the agent managed to perform the task appropriately despite this forced forgetting.

Results-
The model, which works based on visual inputs with 3 color channels and nature cnn preprocessing was trained for 1 million steps with a continuous action space. Proximal policy approximation algorithm was used. The trained model was then loaded and tested for 12 episodes.

The arrangement of the objects was different in the first and third phase, the distractor phase tested the agent's ability to recall information across longer timesteps and take appropriate action



```
Reward: [2240.]
Reward: [2240.]
Reward: [2240.]
Reward: [2240.]
Reward: [240.]
Reward: [2240.]
Reward: [2240.]
Reward: [2240.]
Reward: [2240.]
Reward: [2240.]
Reward: [2240.]
Reward: [2240.]
Reward: [2240.]
```

Figure 20. Multistage Task Results

The agent correctly identified the object in phase 3 for 11 out of 12 episodes. Further research will be done for more dynamic testing conditions.

IV. Conclusion

- Even with changing spawn location and arrangement of objects in each room as compared to training during the search task, the transformer managed to discover all four classes successfully. This proves that transformer based memory in combination with proximal policy optimisation is generalizable.
- In the maze task, the transformer displayed higher levels of rewards as the tendency to touch the adjacent wall with small reward as many times as possible only grew with training, this was not observed in case of LSTM. The transformer also showed higher levels of exploration during training.
- In the exploration task, the transformer displayed a higher level of coverage compared to LSTM, the tendency to visit new tiles was observed to be higher in the case of the transformer.
- In the search task, the transformer showed an ability to generalize across different training and testing conditions
- The transformer based memory managed to avoid the rooms which only consisted of objects from already discovered classes.
- In the distractor task, despite the forced forgetting mechanism of the distractor phase, the transformer based memory managed to perform the identification perfectly.

Acknowledgment

This research was supported/partially supported by DRDO Dighi, Vishwakarma Institute of technology, Indian Institute of Information Technology

Hyderabad. We are thankful towards M.M Kuber, Dr. S E Talole and all our colleagues who provided expertise that greatly assisted the research.

REFERENCES

[1] K. Fang, A. Toshev, L. Fei-Fei, and S. Savarese, “Scene memory transformer for embodied agents in long-horizon tasks,” in CVPR,2019.

[2] “Learning to navigate in cities without a map,” in NeurIPS, 2018.

[3] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, “Gibson env: Real-world perception for embodied agents,” in CVPR, 2018.

[4] N. Savinov, A. Dosovitskiy, and V. Koltun, “Semi-parametric topological memory for navigation,” in ICLR, 2018.

[5] N. Tomatis, I. Nourbakhsh, and R. Siegwart, “Combining topological and metric: A natural integration for simultaneous localization and map building,” in Proc. European Workshop on Advanced Mobile Robots (Eurobot). ETH-Zurich, 2001. ¨

[6] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, and R. Hadsell,

[7] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, “Search on the replay buffer: Bridging planning and reinforcement learning,” in NeurIPS, 2019.

[8] Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell. Neural episodic control. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 2827–2836. JMLR.org, 2017.

[9] Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al. Imagination-augmented agents for deep reinforcement learning. In Advances in neural information processing systems, pages 5690–5701, 2017.

[10] A. Miyake and P. Shah. Models of working memory: Mechanisms of active maintenance and executive control. Cambridge University Press, 1999. doi: 10.1017/CBO9781139174909.