Long Horizon Episodic Decision Making for Cognitively Inspired Robots

Shweta Singh, Vedant Ghatnekar, Sudaman Katti

***Abstract:** The Human decision-making process works by recollecting past sequences of observations and using them to decide the best possible action in the present. These past sequences of observations are stored in a derived form which only includes important information the brain thinks might be useful in the future, while forgetting the rest. we propose an architecture that tries to mimic the human brain and improve the memory efficiency of transformers by using a modified Transformer XL architecture which uses Automatic Chunking which only attends to the relevant chunks in the transformer block. On top of this, we use Forget Span which is technique to remove memories that do not contribute to learning. We also theorize the technique of Similarity based forgetting to remove repetitive memories. We test our model in various tasks that test the abilities required to perform well in a human-robot collaboration scenario.*

***Keywords:** Robotics, Machine Vision and Scene Understanding, Reasoning Under Uncertainty*

I. INTRODUCTION

Human cognition and decision-making works on reflection on only relevant parts of memory. We can recall specific past sequences of events in detail, without paying attention to everything in our memory (Chan et al., 2017, [1]) (Sols et al., 2017, [2]). Irrelevant and repetitive parts of memory are overlooked, preferring storage of a broader picture of events based on the importance of each event. Robotic agents should have similar cognition to function well in long horizon and multi-modal tasks like navigation or human-robot collaboration. The memory buffer should be concise, containing events that will be useful for decision making in the present and future while forgetting the rest (Nematzadeh et al., 2020, [3][8][9][10]). To emulate this in our architecture we propose the use of Automatic Chunking and Forget Span on the TransformerXL memory buffer. Automatic chunking helps by chunking the memory and only using the relevant chunks in the TransformerXL layers while ForgetSpan masks out unnecessary and repetitive elements from the memory creating a more concise memory buffer which improves memory efficiency and performance.

Manuscript received on 01 December 2023 | Revised Manuscript received on 10 December 2024 | Manuscript Accepted on 15 February 2024 | Manuscript published on 28 February 2024.

* Correspondence Author (s)

Shweta Singh, Department of Brain, Cognition and Computation Lab, IIIT, Hyderabad (Telangana), India. E-mail: shweta.singh@research.iiit.ac.in, ORCID ID: 0009-0003-7055-0251

Vedant Ghatnekar*, Department of Mechanical Engineering, MIT WPU, Pune (Maharashtra), India. E-mail: 1032190997@mitwpu.edu.in, ORCID ID: 0009-0006-4004-6646

Sudaman Katti, Department of Mechanical Engineering, MIT WPU, Pune (Maharashtra), India. E-mail: sudaman.katti19@vit.edu, ORCID ID: 0000-0002-9082-0103

We also test a preliminary version of Similarity Weight which decides whether a current observation should be stored in the memory by comparing it with the existing elements in the buffer.

II. ARCHITECTURE AND TESTING METHODOLOGY

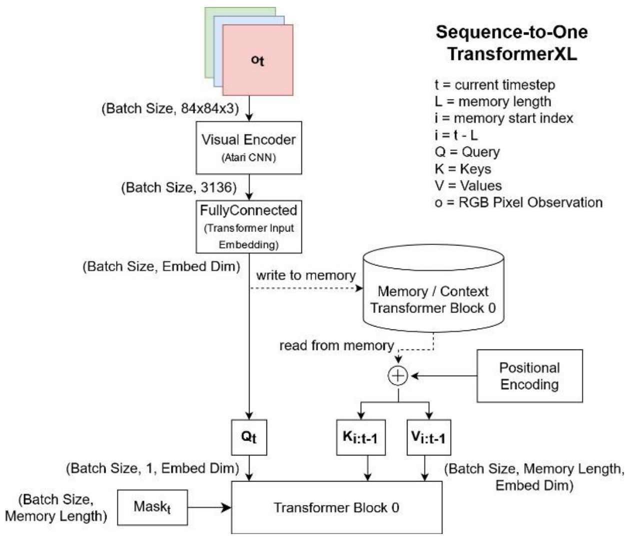

Sequence-to-One TransformerXL
t = current timestep
L = memory length
i = memory start index
i = t - L
Q = Query
K = Keys
V = Values
o = RGB Pixel Observation
ot
(Batch Size, 84x84x3)
Visual Encoder (Atari CNN)
(Batch Size, 3136)
FullyConnected (Transformer Input Embedding)
(Batch Size, Embed Dim)
write to memory
Memory / Context Transformer Block 0
read from memory
Positional Encoding
Qt
Ki:t-1
Vi:t-1
(Batch Size, 1, Embed Dim)
(Batch Size, Memory Length, Embed Dim)
(Batch Size, Memory Length)
Maskt
Transformer Block 0

Figure 1: Transformer XL Architecture

We use a Gated TransformerXL backbone (Pleines et al., 2023, [4]) which we modify to add Automatic Chunking, Forget Span and Similarity Weight. The main architecture of Gated TransformerXL consists of a cyclic memory buffer which stores a specified number of pre-processed observations. The input observations are first pre-processed by a 3 layered convolutional encoder. This encoded observation is stored in memory and also fed to the TransformerXL block as the query. The memory buffer is used to calculate the key and value in the TransformerXL block. The output of the transformer block is then used to create a categorical distribution over the action space, from which actions are sampled. PPO2 (proximal policy optimization) is used to in all models to perform consistent updates and to limit how far we can change the policy in each iteration using KL-divergence. The network policy learns to take appropriate actions based on the current observation and memory during training.

A. Automatic Chunking

In Automatic chunking, we insert our chunking algorithm in between the step where the memory buffer is passed to the Transformer XL block to calculate the key and value.

The memories are instead split into sequential chunks of constant size and each chuck is assigned a summary value which is calculated through mean pooling. The memory mask is taken into account while chunking so unfilled memory elements do not contribute to the summary value. Attention is performed on these mean values and the query to calculate the top-k chunks of memory that are relevant to the current scenario. These top-k memories are then combined and sent to the Transformer XL block to be used as the key and value in place of the whole memory buffer. This reduces the number of memories that need to be attended by the transformer block as well as provides more contextual memories. This allows the transformer to work more efficiently with lesser memories as well as provide a more accurate output in the current context. The chunk size and number of chunks used in our experiments have been detailed in the appendix.

B. ForgetSpan

In ForgetSpan, we use masking to remove memories from the memory buffer after a certain span of time. The model learns to decide the span each memory element will stay in the buffer using the logic below and removes memories that do not contribute to learning. This allows the transformer to learn from a more concise memory buffer, improving learning as well as reducing the memory requirement of the model.

We calculate a ForgetSpan $f_i \in [0, F]$ for every element in memory m_i

$$f_i = F\sigma(W^T m_i + B) \tag{1}$$

Here W and B are a trainable weight and bias, *sigma* is a sigmoid function for activation and F is the maximum span an element can stay in memory. W and B constitute a basic linear layer applied to the memories which learns to approximate an ideal function for calculating the span each memory element should stay in the memory as it trains.
We calculate the remaining span r_{ti} at every timestep t for the i^{th} memory element.

$$r_{ti} = f_i - (t - i) \tag{2}$$

When r_{ti} becomes negative, it means the element has to be forgotten and can be masked out of the memory buffer. We use a soft masking function that creates a smooth mask from 1 to 0 once the element has to be forgotten.

$$s_{ti} = max(0, min(1, 1 + r_{ti}/R)) \tag{3}$$

Where R is the ramp length of the ramp between 1 and 0. This allows f_i to receive a gradient to train as the masking function has a non-zero gradient between $[-R, 0]$. The parameters for ForgetSpan used in our experiments are detailed in the appendix.

C. SimilarityWeight

In SimilarityWeight we calculate the similarity between the current observation with all the elements currently in the memory buffer using cosine similarity. We then bin the similarity values into 10 bins and calculate the number of values in the top k bins. We use k=3 for our experiment in the Minigrid Task. This number is used to represent the similarity of the current element with the memory as it denotes that the number of memories the current observation is highly similar with. We use a threshold of 0.6 was used in our experiment which denotes that a memory element which is similar to more than 60 percent of the memory buffer will be removed. If the value of similarity is greater than the threshold, that means the memory is highly similar to the memory buffer and so it is not stored. If the value of similarity is lower than the threshold, the memory is stored. Where *similarity* is:

$$similarity = topkbins(cossimi(obs, memory)) \tag{4}$$

Where *topkbins* is the function to bin and choose the top-k highest populated bins. Cosine sim- ilarity is calculated using the torch.nn.CosineSimilarity function. SimilarityWeight is employed to remove new observations that are extremely similar to elements already in the memory thus creating a small memory with highly focused elements.

D. Testing Methodology

We test our model with various combinations of our proposed memory handling techniques on 5 different tasks. Each task is designed to test various abilities of the model such as memory, navigation, planning, robotic control and multi-modal deciphering. We believe these to be important abilities a robotic agent would require in achieving human-robot collaboration tasks in real world applications. We train the models on each task till they achieve a satisfactory performance and then modify the environment during testing to test the generalizability of the trained model. Training parameters used in each task as stated in Appendix A.

The Minigrid memory task was implemented using the Minigrid environment package (Chevalier- Boisvert et al., 2023, [5]). Unity MLAgents toolkit (Juliani et al., 2020, [6]) was used for implementing the Audio-Visual Instructions Task and Visual Corridor task. The Visual Instructions task was implemented using the Miniworld environment package (Chevalier-Boisvert et al., 2023, [5][11][12]), the Humanoid locomotion task was implemented using the Mujoco physics engine. We interface the environments with our models coded in PyTorch using the OpenAI Gym API (Brockman et al., 2016, [7]). The tasks are discussed in detail in the Section 3 below. Various combinations of Gated and Ungated TransformerXL, Automatic Chunking, ForgetSpan and SimilarityWeight were tested in all the tasks to see the effects on training performance.

III. RESULTS AND DISCUSSIONS

A. Minigrid Memory Task

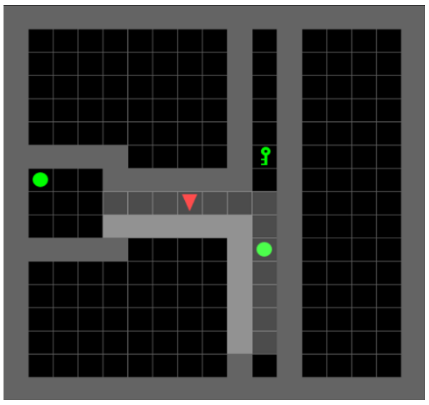


Figure 2: Minigrid Task

The goal of this task is to correctly remember the object seen in the initial room (on the left) and then navigate to the end of the corridor and touch the same object. The agent's observation space includes a 5x5 square image of the grid ahead of the agent. The action space is discrete with the actions Turn left, Turn right and Move forward. This task tests the agent's ability to remember the information at the start of the episode and use it effectively to reach the final goal. In Figure 3, we plot then average rewards across episodes for Baseline Gated TransformerXL, Gated TransformerXL with Automatic chunking, Gated TransformerXL with Automatic chunking and ForgetSpan and Automatic chunking with ForgetSpan and Similarity Weight.

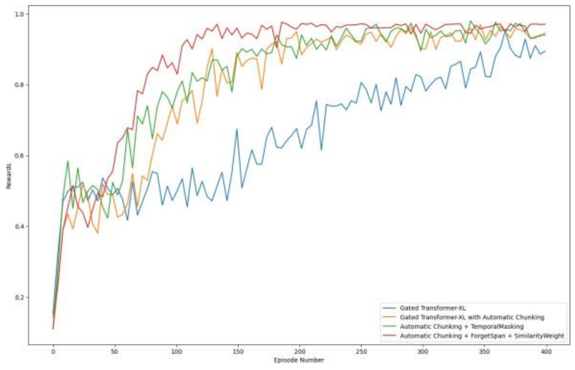


Figure 3: Minigrid Task training rewards

Automatic Chunking with ForgetSpan learns the task slightly faster than Gated TransformerXL with Automatic chunking which in turn trains faster than the baseline Gated TransformerXL. Automatic Chunking with ForgetSpan and SimilarityWeight gives the best results by training the fastest and with the highest reward. Automatic Chunking with ForgetSpan and SimilarityWeight gives the best results by pre-processing the memories and complimenting Automatic Chunking but the computational cost increase is significant as we have to calculate similarity of a new observation with the whole memory buffer every timestep.

B. Audio Visual Instructions Task

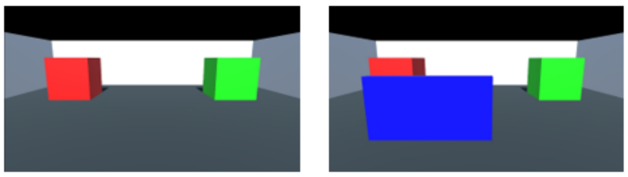


Figure 4: Audio-Visual Instructions Task with and without wall

In this task the agent gets one of two audio commands randomly at the start of each episode, either "red cube" or "green cube". The agent then has to navigate based on visual inputs to the specified cube. The observation space consists of audio spectrograms of size 41 X 42 X 1 along with visual observations of size 41 X 42 X 3. This task tests the agent's recollection as well as multi-modal instruction deciphering ability. The episode ends whenever the agent touches one of the objects. We tested three versions of this task:

1. Static Boxes with static reward (+10 for reaching the correct goal and -1 for not)
2. Moving Boxes with dynamic rewards based on the distance from the correct goal
3. Moving Boxes with dynamic rewards and a moving wall for partial observations

The Static version was used in the results shown in Figure 5 and Table 1 while we used the Moving versions to test the generalizability of the Automatic Chunking with ForgetSpan model in a more dynamic environment. Dynamic rewards were used to incentivize the correct goal and the positions of the boxes and wall were randomized at the start of every episode. The trained model was tested for 100 episodes and the results are shown in Table 2.

In Figure 5, we plot the average training rewards for Gated TransformerXL, Gated TransformerXL with Automatic chunking, Automatic chunking TransformerXL with ForgetSpan with ramp length 64 and 32. As we can see Automatic chunking with ForgetSpan with Ramp length 32 has the best performance with the highest rewards and fastest training. Increasing the Ramp length to 64 led to worsening performance. This is probably caused by the gradient used to train the ForgetSpan is more gradual leading to slower learning of the ForgetSpan layer. Automatic Chunking TransformerXL performed better than baseline TransformerXL while being better than ForgetSpan with ramp length 64 and worse than ForgetSpan with ramp length 32.

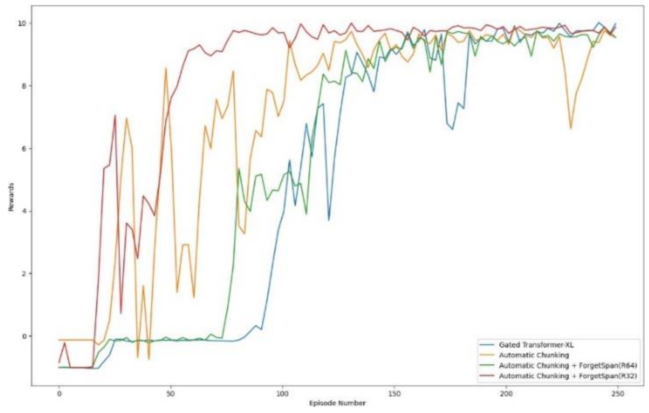


Figure 5: Audio-Visual Instructions Task Rewards

Table 1: Audio-Visual Instructions with Static Boxes Task Testing Results

Task	Success/Total episodes	Fail/Total episodes
Static Boxes with same positions	27/29	2/29
Static Boxes with changed positions	24/29	5/29
Static Boxes with Color changed	51/100	49/100

To test whether our model was generalizable and had learned a correct mapping between the color of the box and the audio instruction we tested the trained model on three scenarios:

1. Static boxes in the same positions as in training
2. Static boxes in different positions than in training
3. Static boxes in the same positions as in training but the green box color is changed to blue

This would test if the model had learned a proper mapping between audio and visual observations. Table 1 shows the number of episodes where the agent went to the correct goal. As we can see the model was able to go to the correct target with a success rate of 93% and 82% in the first two scenarios proving that it had learnt to navigate to the target correctly. The episodes where it missed the targets could be attributed to the agent travelling past the boxes and not being having them in its view. In scenario 3 the agent achieved a success rate of 51% when the audio ”red cube” was played. But it was observed that when the ”green cube” audio was played, the agent avoided going to the blue box and just roamed around the arena searching for a green box. This proves that it had learnt a correct mapping between the color of the target and the audio cue.

Table 2: Audio-Visual Instructions with Moving Boxes Task Testing Results

Task	Success/Total episodes	Fail/Total episodes
Moving Boxes without Wall	94/100	6/100
Moving Boxes with Wall	96/100	4/100

As can be seen in Table 2, the trained model was able to achieve a success rate of 94% and 96% in the Moving cubes version of the Audio-Visual Instructions Task with and without the wall respectively. These results prove that Automatic Chunking with ForgetSpan help achieve generalization by reaching the goal even when the environment is dynamic with changing goal positions, obstructions and partial observations. Automatic Chunking with ForgetSpan help the model to give more importance to the goals and thus is able to adapt in a dynamic goal scenario.

C. Visual Corridor Task with Variable Distractor

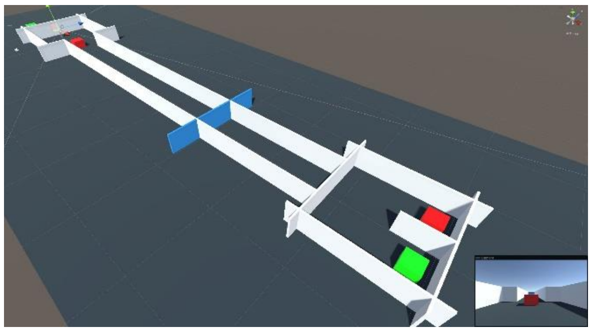


Figure 6: Visual Corridor Task environment in Unity

In this task the agent observes one of the two cubes either red or green in color at the start of each episode. The agent then has to navigate along a long corridor of variable length until it reaches the end at which time it is teleported to the final room where it has to go to the cube it saw at the start of the episode. The observation space consists of visual observations of size 40 X 40 X 3 and the position of the agent. This task tests the agent’s ability to recall information after a variable distractor phase. We only tested Automatic Chunking with ForgetSpan in this task as we wanted to test the forgetting of ForgetSpan as well as the chunk selection of Automatic Chunking in a more dynamic scenario.

In Figure 7 we can see that Automatic Chunking with ForgetSpan using ramp length 100 trained by 150 episodes and learnt to do the task even with the variable distractor phase. Automatic chunking without ForgetSpan took longer to train but reached the same final rewards.

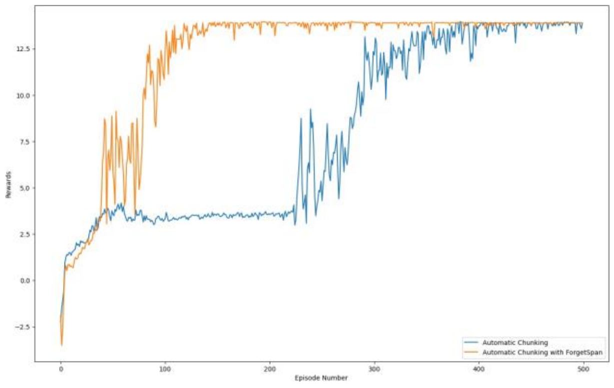


Figure 7: Visual Corridor Task Rewards

This shows that ForgetSpan improves training performance of Automatic Chunking significantly while also improving memory efficiency. To test whether our model was generalizable, during the test scenario we doubled the length of the variable distractor and tested for 30 episodes. Both models managed to reach the final goal for 30 out of 30 episodes as shown by the approximately 15 reward received by each of them every episode in Figure 8.

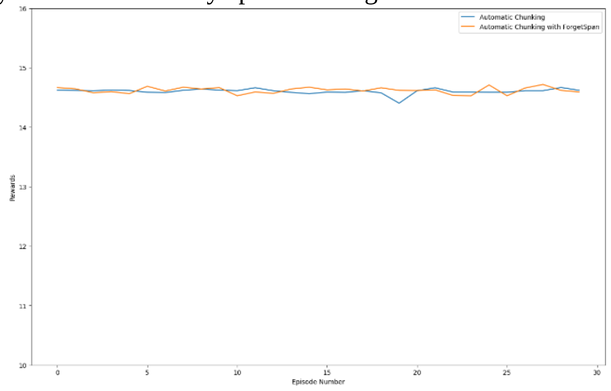


Figure 8: Visual Corridor Test Rewards

D. Visual Instructions Task

In this task the goal of the agent is to perceive a sign present in the environment which displays the name of a specific color. The agent then has to use its visual navigation abilities to navigate to the object with the specified color. The action space is discrete, with 3 actions, Turn right, Turn left and Go forward. RGB Visual observations of size 80 X 60 x 3 were used. This task tests the ability of the agent to remember the information seen at the start of the episode and correctly decipher to navigate to the final goal. Thus, this task tests the agent’s memory along with its visual navigation abilities.

In Figure 10, we plot the average worker rewards and average entropy (randomness) of the model across episodes respectively for Ungated Transformer XL, Gated Transformer XL, Ungated

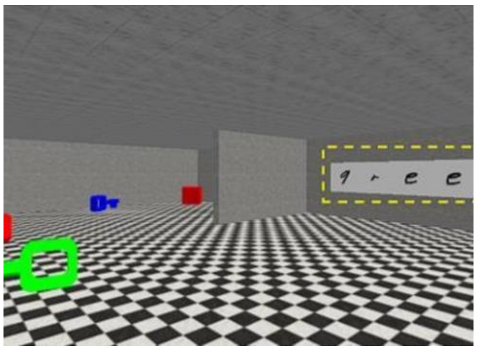
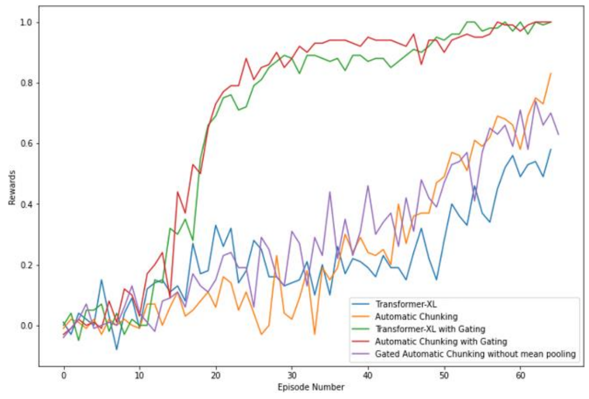
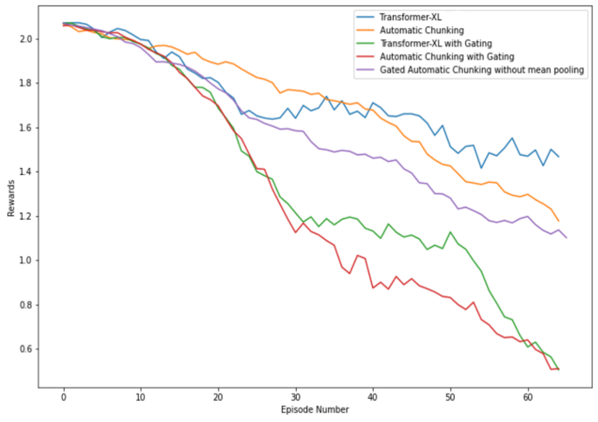


Figure 9: Visual Instructions Task



(a) Rewards



(b) Entropy

Figure 10: Visual Instructions Task Results

TransformerXL with Automatic chunking, Gated TransformerXL with Automatic chunking and Gated Automatic chunking without mean pooling. Gated TransformerXL with Automatic chunking had the highest rewards and lowest entropy during training. Decrease in entropy signifies successful training. Both gated algorithms showed superiority in terms of training time, randomness and rewards achieved. Gated Automatic chunking and Ungated Automatic chunking showed better results as compared to Gated TransformerXL and Ungated TransformerXL respectively. The purple line in corresponds to Gated Automatic chunking without use of mean pooling. In this case, direct averaging across chunks was done to calculate mean values. Although this method led to lower entropy values, the reward values were considerably lower during training. Only Gated automatic chunking with mean pooling managed to surpass Gated TransformerXL due to a more concise memory being used by the transformer.

E. Humanoid Task

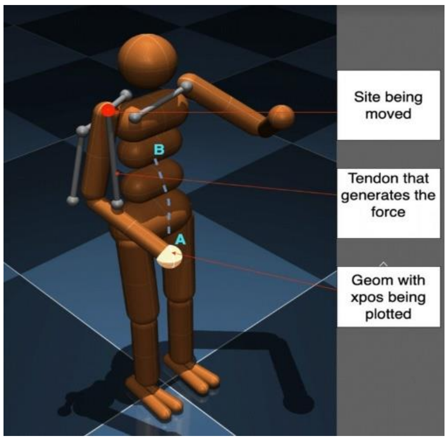
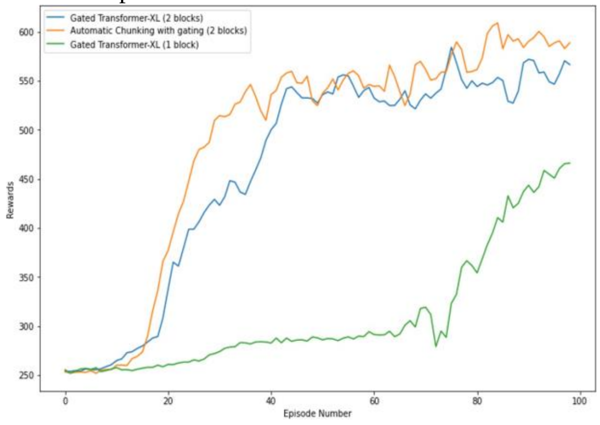


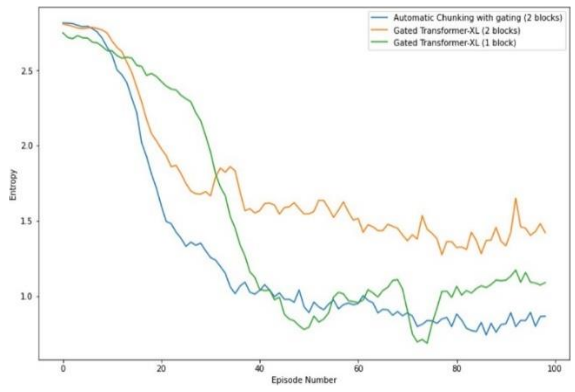
Figure 11: Humanoid Task

In this task, the main goal is to make the humanoid agent balance standing upright for as long as possible. The observation space used is 376 dimensional, consisting of the position values, angular values and the forces applied across the various joints. The reward is directly proportional to the amount of time spent by the agent in standing position. The action space for this task is 17 dimensional, which includes movement of all its body parts. This task tests agent's ability to perform effectively when dealing with huge action and observation spaces, generally found in real world robotic tasks. Training results for Gated Automatic chunking using 2 transformer blocks, Gated TransformerXL with 2 blocks and Gated TransformerXL with 1 block are plotted in Figure 12.

The gated algorithms with two transformer blocks proved to be better in terms of training rewards as compared to Gated TransformerXL with 1 transformer block. The entropy for Gated TransformerXL was lower with 1 transformer block however the fluctuations were also higher. Gated TransformerXL with Automatic chunking and two transformer blocks had the highest re- ward and lowest entropy (randomness in actions taken) during training. This shows that increasing the number of transformer blocks is beneficial when dealing with high dimensional action and observation spaces.



(a) Rewards



(b) Entropy

Figure 12: Humanoid Task Results

IV. CONCLUSION

Transformers with Automatic chunking and memory handling techniques like ForgetSpan and SimilarityWeight showed greater memory efficiency and performance over regular trans- formers models in memory, robot navigation, and multi-modal tasks. Automatic chunking im- proved the baseline TransformerXL by giving a more focused memory for the transformer block to attend to. ForgetSpan and SimilarityWeight showed good synergy with Automatic chunking, improving the training speed as well as the memory efficiency of the model by creating a concise memory with only relevant memories for the transformer architecture to work on. This work aims to improve the performance of Robotic agents in Human-Robot Collaboration tasks which are generally multi-modal, long horizon and dynamic in nature and would greatly benefit from human-like memory. Automatic Chunking, ForgetSpan and SimilarityWeight are a step towards emulating human-like cognition in robots.

DECLARATION STATEMENT

Funding	No, I did not receive.
Conflicts of Interest	No conflicts of interest to the best of my knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence.
Availability of Data and Material/ Data Access Statement	Not relevant.
Authors Contributions	All authors have equal participation in this article.

REFERENCES

1. Stephanie CY Chan, Marissa C Applegate, Neal W Morton, Sean M Polyn, and Kenneth A Norman (2017) 'Lingering representations of stimuli influence recall organization', Neuropsychologia, vol. 97, pp. 72–82, DOI: 10.1016/j.neuropsychologia.2017.01.029
2. Sols, I. et al. (2017) 'Event Boundaries Trigger Rapid Memory Reinstatement of the Prior Events to Promote Their Representation in Long-Term Memory', Current Biology, 27(22), pp. 3499-3504.e4. doi: 10.1016/j.cub.2017.09.057.
3. Aida Nematzadeh, Sebastian Ruder, and Dani Yogatama (2020) 'On memory in human and artificial language processing systems', ICLR 2020: In Bridging AI and Cognitive Science Workshop, 26 April-1 May. Available at: https://api.semanticscholar.org/CorpusID:221088218.
4. Pleines, M. et al. (2023) 'TransformerXL as Episodic Memory in Proximal Policy Optimization', GitHub Repository. Available at: https://github.com/MarcoMeter/episodic-transformer-memory-ppo.
5. Chevalier-Boisvert et al. (2023) 'Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks', CoRR, abs/2306.13831.
6. Juliani, A. et al. (2020) 'Unity: A general platform for intelligent agents', arXiv preprint arXiv:1809.02627. Available at: https://arxiv.org/pdf/1809.02627.pdf.
7. Brockman, G. et al. (2016) 'OpenAI Gym', arXiv Eprint arXiv:1606.01540. Available at: http://arxiv.org/abs/1606.01540.
8. Patil, Dr. K., & Kulkarni, Dr. M. S. (2019). Artificial Intelligence in Financial Services: Customer Chatbot Advisor Adoption. In International Journal of Innovative Technology and Exploring Engineering (Vol. 9, Issue 1, pp. 4296–4303). https://doi.org/10.35940/ijitee.a4928.119119
9. Hudaa, S., Setiyadi, D. B. P., Lydia, E. L., Shankar, K., Nguyen, P. T., Hashim, W., & Maseleno, A. (2019). Natural Language Processing utilization in Health care. In International Journal of Engineering and Advanced Technology (Vol. 8, Issue 6s2, pp. 1117–1120). https://doi.org/10.35940/ijeat.f1305.0886s219
10. Vatan, Sharma, A., & Goyal, S. (2019). Artificial Intelligence on the Move: A Revolutionary Technology. In International Journal of Recent Technology and Engineering (IJRTE) (Vol. 8, Issue 4, pp. 12112–12120). https://doi.org/10.35940/ijrte.d7293.118419
11. Mishra, S. (2022). A Comparative Analysis of Diabetes Prediction using Different Machine Learning Algorithms. In Indian Journal of Artificial Intelligence and Neural Networking (Vol. 2, Issue 5, pp. 1–7). https://doi.org/10.54105/ijainn.e1057.082522
12. P A, J., & N, A. (2022). Faceium–Face Tracking. In Indian Journal of Data Communication and Networking (Vol. 2, Issue 5, pp. 1–4 https://doi.org/10.54105/ijdcn.b3923.082522

AUTHORS PROFILES

Shweta Singh, I work as a Researcher in the Brain, Cognition & Computation Lab at IIIT Hyderabad. My focus lies in the interdisciplinary field merging cognitive science, artificial intelligence, and robotics. I specialize in crafting AI models influenced by cognitive science principles, aiming to create intelligent AI agents with cognitive abilities. My work revolves around developing AI that emulates human cognitive processes, paving the way for agents capable of understanding, learning, and adapting like humans. My goal is to bridge the gap between AI technology and human cognition, offering a pathway to more advanced, cognitively adept artificial intelligence.

Vedant Ghatnekar, I am a student from MIT-WPU, Pune studying BTech in Mechanical Engineering. My research interests include Cognitive Robotics, Human-Robot Collaboration, Memory in AI Agents, Reinforcement Learning and Deep Learning. I have been pursuing these interests in my various projects and internships to builds a deep and intuitive understanding of various concepts in these fields. I have published multiple papers on the reinforcement learning models and their memory architectures that I have worked on. My eventual career goals include building collaborative robots that help people in their day to day lives as well as improve the quality of life of elderly or differently abled people.

Sudaman Katti, I am a student at VIT, Pune studying BTech in Mechanical Engineering. My research interests include Memory based Reinforcement Learning, Transformers and Robotics. In my research work, reinforcement learning using proximal policy optimization was used for learning of various visual navigation, locomotion through limb manipulation and memory intensive tasks related to robotics. Unity development software was used for creation of various 3D environments and PyTorch was used for designing the algorithm. I am working towards improving my knowledge in this field and wish to work on Artificial Intelligence for robots in the future.

Appendix A. Parameters and Implementation Details

All the hyperparameters used while training the models in this research work are listed below.

Appendix A.1. PPO Parameters

- learning rate(initial): 3e-4 (decays consistently during training, final value is 3e-5)
- gamma: 0.995
- lambda: 0.95
- updates: 100
- epochs: 5
- n workers: 20
- n mini batch: 10

The above standard Proximal policy optimization parameters were chosen with extensive testing for the purpose of making sure that optimal behavior is learnt within 200-250 training episodes for the standard TransformerXL model. These parameters were kept the same across all the models used in this research work in order to obtain appropriate comparative results. All tasks made use of 20 workers and 10 mini batches in order to reduce training time.

Appendix A.2. Transformer Parameters

- embed dim: 250
- number of heads: 5
- memory length: 64
- positional encoding: True
- gating: True

With extensive testing, the above parameters were changed based on the task in order to speed up training and get stable results. However, the same values were taken during comparative study with different architectures. The embed dimension parameter specifies the common dimension to which the keys, queries and values will be converted to make the multilevel attention mechanism work. The number of heads parameter specifies the amount of transformer heads. For all tasks, the embed dimension was 250 and the number of heads were 5. Both positional encoding and layer normalization were set to true for all the tasks to ensure that proper and effective sequence processing is performed by the transformer. The memory length parameter specifies the amount of time step information stored in the memory buffer. Memory length for the humanoid loco- motion task was set to 300 and for the audio-visual navigation task it was set to 384. For the visual instructions task and minigrid task, it was set to 250 and for the visual corridor task was set to 500. In order to understand the effectiveness of a single automatic chunking mechanism operating on the entire memory, only a single transformer block was used for all the experiments except the humanoid locomotion task. The Gating parameter was used to decide whether a gating mechanism is implemented.

Appendix A.3. Memory Parameters

- n chunks: 3
- chunk size: 50
- max span: 250
- ramp length: 50

The chunk size and number of chunks denote the length and number of sequential events being selected during training. The number of chunks were set to 3 for all experiments. For the humanoid locomotion and audio-visual instructions task, the chunk size was set to 80. For the visual instructions and minigrid tasks the chunk size was set to 50 while for the visual corridor task the chunk size was set to 100. We decided the chunk size so that the summarized memory buffer size was approximately 60% of the complete memory buffer as this gave better results during testing. In all tasks the max span of ForgetSpan was kept to be the size of the memory buffer while ramp lengths were changed according to the task. While testing we concluded that lower ramp lengths gave better results.