LEARNING MEMORY MECHANISMS FOR DECISION MAKING THROUGH DEMONSTRATION

William Yue[1], Bo Liu[1] & Peter Stone[1,2] *

ABSTRACT

In Partially Observable Markov Decision Processes, integrating an agent's history into memory poses a significant challenge for decision-making. Traditional imitation learning, relying on observation-action pairs for expert demonstrations, fails to capture the expert's memory mechanisms used in decision-making. To capture memory processes as demonstrations, we introduce the concept of **memory dependency pairs** (p, q) indicating that events at time p are recalled for decision-making at time q. We introduce **AttentionTuner** to leverage memory dependency pairs in Transformers and find significant improvements across several tasks compared to standard Transformers when evaluated on Memory Gym and the Long-term Memory Benchmark.

1 INTRODUCTION

Partially Observable Markov Decision Processes (POMDPs) offer a framework for modeling decision-making in environments where the agent's information is incomplete, a common situation in real-world scenarios such as a robot operating based on limited camera observations. Making effective decisions under such conditions necessitates incorporating the agent's history, which can be encoded through memory mechanisms like Recurrent Neural Networks (RNNs) Hausknecht & Stone (2017); Karkus et al. (2017) or self-attention architectures such as Transformers Esslinger et al. (2022). However, it's not always straightforward for a fully automated system to identify which points in history are crucial to remember for a particular decision. On the other hand, when humans learn, we are often taught not just the actions we need to take at the moment, but also which past events and memories should be recalled. For instance, a survival instructor might instruct students to recollect previously observed landmarks for navigating back to base camp or a coach could ask an athlete to recall a past encounter with an opponent when making their next move. With this motivation in mind, this study concentrates on learning memory mechanisms essential for decision-making in POMDP tasks via expert demonstrations, also known as imitation learning. Standard imitation learning methods, which involve experts providing observation-action pairs, are insufficient in POMDP tasks as they do not capture the memory processes experts employ during decision-making. To capture memory mechanisms through demonstration, we introduce the use of **memory dependency pairs** (p, q), where $p < q$, indicating that the observation at time p ought to be recalled for decision-making at time q. These memory dependency pairs can be integrated into the widely-used Transformers Vaswani et al. (2023) by applying a loss to the self-attention matrix to reinforce attention between tokens representing times p and q (Figure 1). The main contributions of this paper are as follows:

- We introduce **memory dependency pairs** to incorporate memory mechanisms into demonstrations for imitation learning in POMDPs and to improve long-term credit assignment
- We introduce **AttentionTuner**, a novel method for leveraging memory dependency pairs in self-attention architectures, and benchmark it against vanilla Transformers on Memory Gym Pleines et al. (2024) and LTMB (Long-term Memory Benchmark). Empirical analyses show that AttentionTuner significantly improves success rates on four tasks and aids the optimizer in consistently navigating the loss landscape towards solutions with better generalizability compared to those found by optimizing the vanilla Transformer. Ablations reveal that these improvements in learning can be attained with as few as 0.1% of demonstrations annotated.

2 BACKGROUND

This section defines the notation and framework for imitation learning in partially observable environments and provides a concise overview of Transformer architectures. This notation will be used to define AttentionTuner in Section 3.

*[1]The University of Texas at Austin. [2] Sony AI. Correspondence to: William Yue at william.yue@utexas.edu

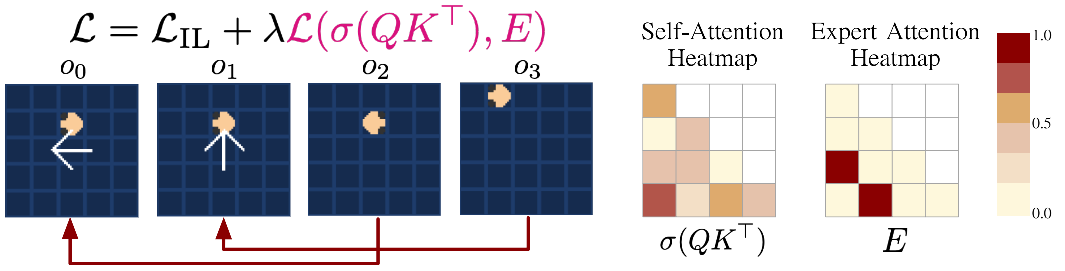


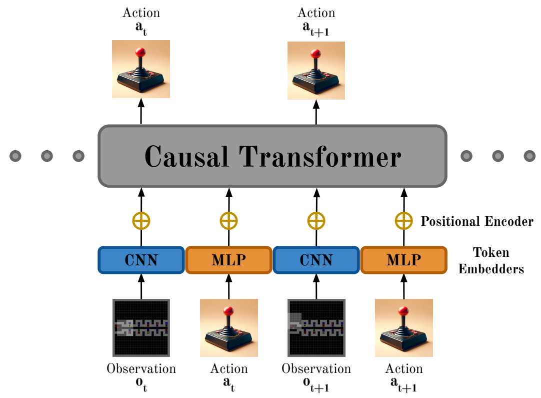
Figure 1: The red arrows indicate episodic memory dependencies labeled by an expert. The correct action to take in o_2 depends on o_0 and the correct action to take in o_3 depends on o_1. These memory dependency pairs are used to create the expert self-attention matrix $E \in \{0,1\}^{n\times n}$ where n is the length of the sequence and $E_{ij} = 1$ only if the expert has indicated that observation o_j should be recalled from memory at observation o_i, and $E_{ij} = 0$ otherwise. A binary cross entropy loss is taken between E and the learner's self-attention matrix $\sigma(QK^\top)$ to form the memory loss $\mathcal{L}(\sigma(QK^\top), E)$ that encourages the learner to learn the expert's memory mechanism. The memory loss is scaled by λ to match the magnitude of $\mathcal{L}_{\mathrm{IL}}$ and then added to form the final loss used during training.

2.1 Imitation Learning in Partially Observable Environments

Imitation learning algorithms aim to learn a policy π_θ parameterized by θ by imitating a set of expert demonstrations $D = \{\tau_i\}_{i=1\ldots M}$. Each demonstration trajectory τ_i is a sequence of observation-action pairs $(o_j, a_j), \quad j = 1\ldots|\tau_i|$, where $|\tau_i|$ denotes the trajectory length. These trajectories are generated from a Partially Observable Markov Decision Process (POMDP), which is characterized by the tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, T, O, \rho_0\rangle$. Here, $\mathcal{S}$ represents the state space, $\mathcal{O}$ the observation space, $\mathcal{A}$ the action space, $T : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$ the transition dynamics, $O : \mathcal{S}\times\mathcal{O} \to [0,1]$ the observation function, and ρ_0 the initial state distribution. In this study, we focus on behavioral cloning, where the objective is to minimize the negative log-likelihood loss function for a discrete action space:

$$\mathcal{L}_{\mathrm{IL}}(\theta) = -\mathbb{E}_{(o,y)\sim D}\left[\sum_a^{\mathcal{A}} \mathbb{1}_{y=a} \log(\pi(a \mid o))\right] \tag{1}$$

2.2 Transformers for Decision Making



Action $\mathbf{a}_t$
Action $\mathbf{a}_{t+1}$
Causal Transformer
Positional Encoder
Token Embedders
CNN
MLP
CNN
MLP
Observation $\mathbf{o}_t$
Action $\mathbf{a}_t$
Observation $\mathbf{o}_{t+1}$
Action $\mathbf{a}_{t+1}$

Figure 2: Architecture of the causal Transformer for sequential decision making modeling used in all AttentionTuner and vanilla Transformer experiments.

Transformers are a neural network architecture designed to process sequential data Vaswani et al. (2023). In the context of decision-making, trajectories can be modeled using the Transformer architecture, as illustrated in Figure 2. This approach is analogous to the methods used in Decision Transformers Chen et al. (2021) and Trajectory Transformers Janner et al. (2021). We represent a trajectory as $\tau = (o_1, a_1, o_2, a_2, \ldots, o_T, a_T)$ where o_i and a_i denote the observation and action, respectively, at timestep i. The Transformer predicts the action a_t for each observation o_t as detailed in Figure 2. These predictions are then utilized to compute the imitation learning training loss, as described in Equation 1. Although Transformers excel in numerous domains, they encounter specific challenges in POMDP memory tasks. These challenges lead to the difficult and unstable optimization of the behavioral cloning objective (Equation 1), an issue further explored in Section 4.2.

3 METHOD

We propose a novel approach to imitation learning by introducing **memory dependency pairs** within trajectories to address memory utilizations in decision-making. For each trajectory $\tau_i \in D$, we define $\mathcal{M}_i = \{(p_j, q_j)\}_{j=1...|\mathcal{M}_i|}$ as the set of memory dependency pairs, where each pair (p, q) indicates that observation o_p was recalled during the decision-making process for a_q. If the decision-making process for a_q depends on multiple past observations $o_{p_1}, \ldots, o_{p_n}$, then we can represent these dependencies with the pairs $(p_1, q), \ldots, (p_n, q)$. We extend the definition of an expert demonstration trajectory, initially described in Section 2.1, to $\tau_i = \left(\{(o_j, a_j)\}_{j=1...|\tau|}, \mathcal{M}_i\right)$, incorporating both observation-action pairs and memory dependencies. While in practice, a human would typically annotate memory dependency pairs (as shown in Appendix N), in the experiments reported in this paper, we instead use a computer program to automate the annotation of memory dependency pairs.

While in principle, memory dependency pairs could be used to enhance any memory-based learning architecture, in this paper we introduce **AttentionTuner**, which specifically leverages them to enhance Transformer-based architectures. For each trajectory τ_i, with length $n = |\tau_i|$, AttentionTuner constructs an expert self-attention matrix $E \in \{0, 1\}^{n \times n}$, detailed in Figure 1, where $E_{ij} = 1$ if $(j, i) \in \mathcal{M}_i$ and 0 otherwise. To encourage the Transformer to mimic the expert's memory mechanism, AttentionTuner applies a binary cross-entropy loss between the expert matrix E and the learner's self-attention matrix $A = \sigma(QK^\top) \in [0, 1]^{n \times n}$. The memory loss equation is $\mathcal{L}(A, E) = -\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} [E_{ij} \log(A_{ij}) + (1 - E_{ij}) \log(1 - A_{ij})]$. In AttentionTuner, this memory loss is applied to a single attention head within the first Transformer layer (Figure 6). The first layer is chosen because it is closest to the raw observation embeddings, making the application of memory dependency pairs more meaningful. Applying the loss to a single head allows other heads to learn additional memory mechanisms not captured by $\mathcal{M}_i$. Alternative applications of this memory loss are explored in Appendix E.

The memory loss is then scaled using a hyperparameter λ and combined with the imitation learning loss $\mathcal{L}_{\text{IL}}$ (defined in Equation 1) to form the final training loss $\mathcal{L} = \mathcal{L}_{\text{IL}} + \lambda \mathcal{L}(A, E)$. We set $\lambda = 10$ based on robust performance observed across various benchmark tasks, effectively balancing the magnitude of the memory loss $\mathcal{L}(A, E)$ with the imitation learning loss $\mathcal{L}_{\text{IL}}$. Comprehensive details of the model architecture, including the CNN and MLP embedders and the causal Transformer, are provided in Appendix D. Pseudocode for training AttentionTuner is provided in Appendix C Algorithm 1. The pseudocode for training vanilla Transformers is identical if the memory loss $\mathcal{L}_{\text{memory}}$ is removed by setting $\lambda = 0$.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluated AttentionTuner on the Memory Gym benchmark Pleines et al. (2023) and our newly introduced Long-term Memory Benchmark (LTMB), each featuring three procedurally generated POMDP tasks as illustrated in Figure 3. These tasks are specifically designed to necessitate and assess the use of memory mechanisms in agents. Four of the six tasks require memory dependencies on several past observations at a single timestep (shown in Appendix P Figure 9).

Baselines Due to the lack of a good dense reward function in our benchmark environments and in most real world decision making tasks, we do not directly compare our method against methods that learn memory mechanisms through reinforcement learning, as they are expected to perform poorly on these tasks. We believe AttentionTuner focuses on a new problem setting, and we are not aware of other methods for learning memory mechanisms through demonstrations. For this reason, we compare our method against a vanilla Transformer as a baseline.

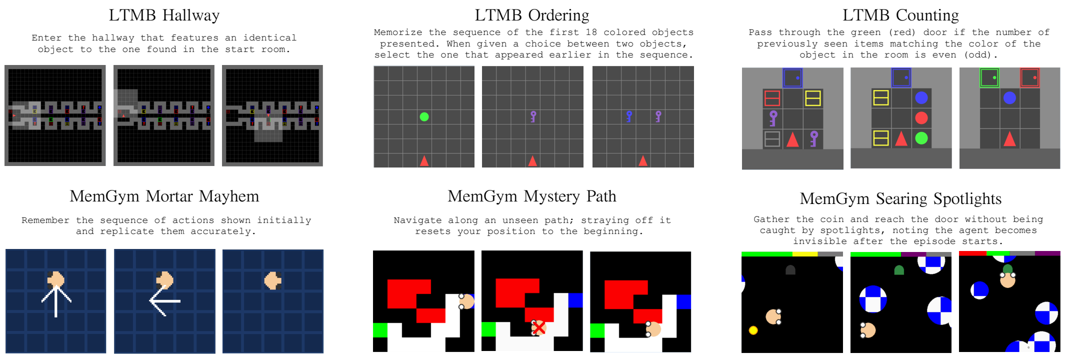

LTMB Hallway
Enter the hallway that features an identical object to the one found in the start room.
LTMB Ordering
Memorize the sequence of the first 18 colored objects presented. When given a choice between two objects, select the one that appeared earlier in the sequence.
LTMB Counting
Pass through the green (red) door if the number of previously seen items matching the color of the object in the room is even (odd).
MemGym Mortar Mayhem
Remember the sequence of actions shown initially and replicate them accurately.
MemGym Mystery Path
Navigate along an unseen path; straying off it resets your position to the beginning.
MemGym Searing Spotlights
Gather the coin and reach the door without being caught by spotlights, noting the agent becomes invisible after the episode starts.

Figure 3: Overview of LTMB and MemGym tasks. Detailed task descriptions are available in Appendix F.

4.2 Results and Analysis

Methods	Mortar Mayhem	Mystery Path	Searing Spotlights	Hallway	Ordering	Counting
Vanilla Transformer	20.8 ± 42.2	97.3 ± 0.7	62.2 ± 5.5	53.2 ± 28.3	59.4 ± 22.9	6.0 ± 0.7
AttentionTuner (ours)	$\mathbf{99.8} \pm 0.4$	$\mathbf{98.7} \pm 0.4$	$\mathbf{64.2} \pm 3.4$	$\mathbf{99.9} \pm 0.1$	$\mathbf{99.9} \pm 0.3$	$\mathbf{6.5} \pm 0.4$

Table 1: Average success rates and 90% confidence intervals for two different methods—Vanilla Transformer and AttentionTuner (our approach)—across various tasks in the Memory Gym and Long-term Memory Benchmark.

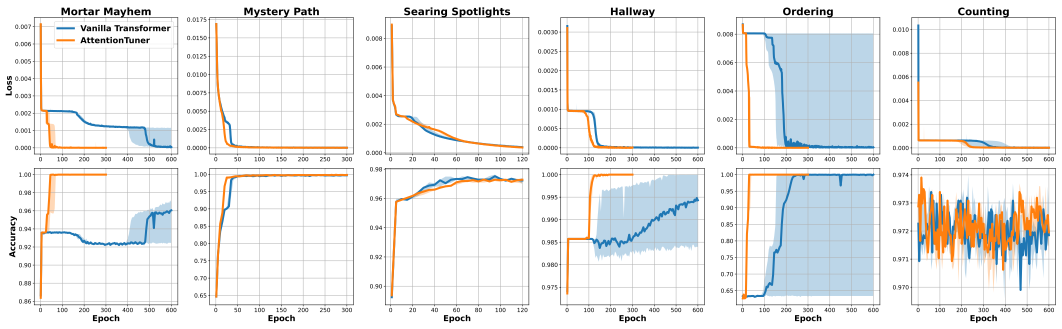
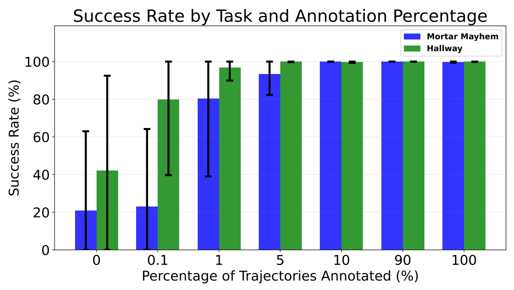


Figure 4: Median learning curves with interquartile range for Memory Gym and LTMB tasks are presented. The top row displays the training loss while the bottom row shows the test accuracy on action prediction (not success rate).

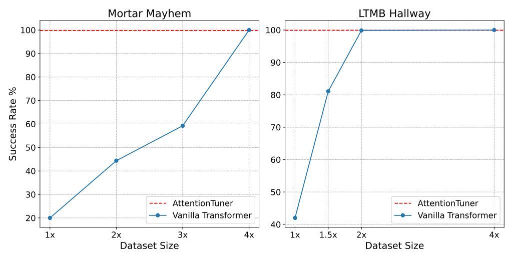
AttentionTuner achieves significantly better success rates. In Table 1, performance of AttentionTuner is compared with the vanilla Transformer on Memory Gym and LTMB tasks. We performed a Welch t-test (results in Appendix J Table 5) with a significance threshold of $\alpha = 0.05$ and found that AttentionTuner achieves significantly better success rates on Mortar Mayhem, Mystery Path, Hallway, and Ordering. The high variability observed in the vanilla Transformer's performance on Mortar Mayhem, Hallway, and Ordering tasks can be attributed to some runs achieving nearly perfect success rates, while others fall close to zero. This discrepancy is explained in the learning curves presented in Figure 4, indicating challenges in optimizing the behavioral cloning objective (Equation 1) for POMDP memory tasks with a large Transformer model. These challenges result in convergence to suboptimal local optima, as evidenced by flat segments in the training loss curves.

AttentionTuner aids the optimizer in navigating the loss landscape. A local optimum is apparent in the Hallway and Counting tasks, while less pronounced in Mystery Path and Searing Spotlights (Figure 4). Notably, Mortar Mayhem and Ordering present two local optima, posing additional challenges for the optimizer. A unique observation in the Ordering task is that AttentionTuner encounters a single local optimum, in contrast to the vanilla Transformer, which faces two. The intricacies of task design that lead to the formation of multiple or difficult-to-escape local optima are not fully understood, highlighting an area for future research. The interquartile ranges in Figure 4 suggest that AttentionTuner aids the optimizer in more efficiently and consistently escaping local optima, and potentially encountering fewer of them. For instance, in the Mortar Mayhem task, AttentionTuner enabled the optimizer to surpass both local optima in under 50 epochs with minimal variability across training seeds. In contrast, only a single training run of the vanilla Transformer overcame both local optima and did so with ~ 400 more training epochs. A similar pattern emerges in the Mystery Path, Hallway, Ordering, and Counting tasks where AttentionTuner consistently escapes from the local optima while the vanilla Transformer only escapes some of the time or escapes up to 200 epochs later than AttentionTuner (Figure 4). AttentionTuner's enhanced capability in traversing the loss landscape underscores its efficacy in facilitating long-term credit assignment and the learning of memory mechanisms.

4.3 Ablations



(a) Success rates and 90% confidence intervals for AttentionTuner training on 'Mortar Mayhem' and 'Hallway' tasks with missing annotations. Numerical values are presented in Appendix L Table 6.



(b) Success rates for vanilla Transformer with different training data sizes. The dotted red line represents AttentionTuner's success rate when training with a 1x dataset size. The number of demonstrations collected in the 1x datasets can be found in Appendix G.

Figure 5: Ablation Results

Imperfect Expert Annotations. Collecting comprehensive expert memory annotations for every demonstration trajectory in long complex decision-making tasks may not always be feasible. To explore the impact of this constraint, we conducted experiments varying the proportion of trajectories that included memory annotations, as detailed in Figure 5a. During training, the memory loss was only used on trajectories that included memory annotations. Trajectories without memory annotations were trained using only the imitation learning loss (Equation 1). The findings, presented in Figure 5a, reveal that having memory annotations for merely 10% of the trajectories achieves comparable results to having annotations for all trajectories. Performance degradation for AttentionTuner becomes noticeable when the proportion of annotated demonstration trajectories falls below 10%. Nonetheless, even with as few as 1% of trajectories annotated, AttentionTuner manages to achieve a relatively high average success rate. Moreover, having even just 0.1% of trajectories annotated still enables AttentionTuner to outperform the vanilla Transformer. Furthermore, human annotated trajectories could include errors. We found that AttentionTuner is robust against small perturbations of the memory dependency pair endpoints (Appendix M).

Value of Memory Dependency Pairs Compared to Additional Demonstrations. Figure 5b shows that it can take anywhere from 2 to 4 times more demonstrations to train a vanilla Transformer to achieve the same success rate as AttentionTuner. In our experiments, we found that it takes 2 to 3 times longer for a human to annotate memory dependency pairs than to just collect a demonstration (Appendix N). Considering that only 5-10% of the demonstrations need to be annotated (Section 4.3), we find that memory dependency pair annotations on a single demonstration is equivalent to providing 40 additional demonstrations on both the Mortar Mayhem and Hallway tasks. Annotating memory dependency pairs instead of collecting additional demonstrations results in a human labor time savings factor of 16 on Mortar Mayhem and 14 on Hallway. Appendix O details how these numbers are computed, though it is important to note that exact time savings can vary widely based on the task and the quality of the human annotator.

Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by the National Science Foundation (FAIN-2019844, NRT-2125858), the Office of Naval Research (N00014-18-2243), Army Research Office (W911NF-23-2-0004, W911NF-17-2-0181), DARPA (Cooperative Agreement HR00112520004 on Ad Hoc Teamwork), Lockheed Martin, and Good Systems, a research grand challenge at the University of Texas at Austin. The views and conclusions contained in this document are those of the authors alone. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

Riad Akrour, Marc Schoenauer, and Michèle Sebag. April: Active preference-learning based reinforcement learning, 2012.

Cameron Allen, Aaron Kirtland, Ruo Yu Tao, Sam Lobel, Daniel Scott, Nicholas Petrocelli, Omer Gottesman, Ronald Parr, Michael L Littman, and George Konidaris. Mitigating partial observability in sequential decision processes via the lambda discrepancy. *arXiv preprint arXiv:2407.07333*, 2024.

Alan Baddeley. Working memory. *Science*, 255(5044):556–559, 1992.

Yoshua Bengio and Paolo Frasconi. Credit assignment through time: Alternatives to backpropagation. *Advances in neural information processing systems*, 6, 1993.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Arkadii Bessonov, Alexey Staroverov, Huzhenyu Zhang, Alexey K Kovalev, Dmitry Yudin, and Aleksandr I Panov. Recurrent memory decision transformer. *arXiv preprint arXiv:2306.09459*, 2023.

Mikhail S. Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V. Sapunov. Memory transformer, 2021.

Carlos Celemin and Javier Ruiz del Solar. An interactive framework for learning continuous actions policies based on corrective feedback. *Journal of Intelligent & Robotic Systems*, pp. 1–21, 2019. URL `https://api.semanticscholar.org/CorpusID:115908814`.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, P. Abbeel, A. Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Neural Information Processing Systems*, 2021. URL `https://api.semanticscholar.org/CorpusID:235294299`.

Yuying Chen, Congcong Liu, Lei Tai, Ming Liu, and Bertram E. Shi. Gaze training by modulated dropout improves imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7756–7761, 2019. doi: 10.1109/IROS40897.2019.8967843.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

Nelson Cowan. What are the differences between long-term, short-term, and working memory? *Progress in brain research*, 169:323–338, 2008.

Kevin Esslinger, Robert Platt, and Christopher Amato. Deep transformer q-networks for partially observable reinforcement learning, 2022.

Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charlie Deck, Joel Z Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory, 2020.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches, 2023.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9: 1735–1780, 1997. URL `https://api.semanticscholar.org/CorpusID:1915014`.

Michael S Humphreys, John D Bain, and Ray Pike. Different ways to cue a coherent memory system: A theory for episodic, semantic, and procedural tasks. *Psychological Review*, 96(2):208, 1989.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):5223, 2019.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Neural Information Processing Systems*, 2021. URL `https://api.semanticscholar.org/CorpusID:235313679`.

Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets, 2015.

Jikun Kang, Romain Laroche, Xingdi Yuan, Adam Trischler, Xue Liu, and Jie Fu. Think before you act: Decision transformers with working memory. In *Forty-first International Conference on Machine Learning*, 2023.

Peter Karkus, David Hsu, and Wee Sun Lee. Qmdp-net: Deep learning for planning under partial observability, 2017.

Nan Rosemary Ke, Anirudh Goyal, Olexa Bilaniuk, Jonathan Binas, Michael C. Mozer, Chris Pal, and Yoshua Bengio. Sparse attentive backtracking: Temporal credit assignment through reminding, 2018.

W. Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *The Fifth International Conference on Knowledge Capture*, September 2009. URL `http://www.cs.utexas.edu/users/ai-lab?KCAP09-knox`.

Andrew Lampinen, Stephanie Chan, Andrea Banino, and Felix Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:28182–28195, 2021.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training, 2021.

Donald J Lewis. Psychobiology of active and inactive memory. *Psychological bulletin*, 86(5):1054, 1979.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners, 2024. URL `https://arxiv.org/abs/2407.14207`.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment, 2023.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Memory gym: Partially observable challenges to memory-based agents. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=jHc8dCx6DDr`.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Memory gym: Towards endless tasks to benchmark memory capabilities of agents, 2024.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011.

Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017. URL `https://api.semanticscholar.org/CorpusID:12226563`.

Akanksha Saran, Ruohan Zhang, Elaine Schaertl Short, and Scott Niekum. Efficiently guiding imitation learning algorithms with human gaze. *arXiv preprint arXiv:2002.12500*, 2020.

William Saunders, Girish Sastry, Andreas Stuhlmueller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention, 2017.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2 edition, 2018.

Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. University of Massachusetts Amherst, 1984.

Timon Ten Berge and René Van Hezewijk. Procedural and declarative knowledge: An evolutionary perspective. *Theory & Psychology*, 9(5):605–624, 1999.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *International Joint Conference on Artificial Intelligence*, 2018. URL `https://api.semanticscholar.org/CorpusID:23206414`.

Endel Tulving. Episodic and semantic memory. *Organization of memory*, pp. 381–403, 1972.

Endel Tulving. *Elements of Episodic Memory*. Oxford University Press, 1983.

Endel Tulving. Memory and consciousness. *Canadian Psychology/Psychologie canadienne*, 26(1):1, 1985.

Michael T Ullman. Contributions of memory circuits to language: The declarative/procedural model. *Cognition*, 92(1-2):231–270, 2004.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces, 2018.

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z. Leibo, Adam Santoro, Mevlana Gemici, Malcolm Reynolds, Tim Harley, Josh Abramson, Shakir Mohamed, Danilo Rezende, David Saxton, Adam Cain, Chloe Hillier, David Silver, Koray Kavukcuoglu, Matt Botvinick, Demis Hassabis, and Timothy Lillicrap. Unsupervised predictive memory in a goal-directed agent, 2018.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks, 2015.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Neural Information Processing Systems*, 2012. URL `https://api.semanticscholar.org/CorpusID:6019958`.

Christian Wirth, Johannes Fürnkranz, and Gerhard Neumann. Model-free preference-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers, 2022.

Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end simulated driving. In *AAAI Conference on Artificial Intelligence*, 2017. URL `https://api.semanticscholar.org/CorpusID:5929487`.

A RELATED WORK

This section surveys the literature on memory types, integration of human feedback in decision-making algorithms, challenges in long-term credit assignment, and the development of memory mechanisms in learning models.

A.1 TYPES OF MEMORY

Endel Tulving's 1972 research distinguishes between episodic memory, which stores information about specific events and their context, and semantic memory, a structured knowledge base for language and symbols Tulving (1972; 1983; 1985). Long-term memory preserves a wide array of information, from skills to life events, over lengthy durations. In contrast, short-term or working memory, crucial for tasks like language comprehension and problem-solving, holds information briefly and allows for its manipulation, such as in mental arithmetic Baddeley (1992); Cowan (2008). While other dichotomies exist in the study of memory, such as declarative versus procedural Humphreys et al. (1989); Ten Berge & Van Hezewijk (1999); Ullman (2004), and active versus inactive Lewis (1979), this work is primarily concerned with long-term episodic memory.

A.2 HUMAN FEEDBACK IN DECISION MAKING

Human feedback can be integrated into learning agents through various modalities. In reinforcement learning, scalar rewards may be assigned to individual states Knox & Stone (2009); Sutton & Barto (2018); Warnell et al. (2018) or preferences may be expressed between trajectory pairs, either online or offline Wilson et al. (2012); Akrour et al. (2012); Wirth et al. (2016); Sadigh et al. (2017); Lee et al. (2021); Stiennon et al. (2022); Christiano et al. (2023). In imitation learning, agents can learn from observation-action pairs, provided by humans in both online and offline contexts Ross et al. (2011); Zhang & Cho (2017); Saunders et al. (2017); Torabi et al. (2018). Additional feedback mechanisms include gaze tracking Chen et al. (2019); Saran et al. (2020), binary corrective signals Celemin & del Solar (2019), and human-provided trajectory outlines Gu et al. (2023). To the best of our knowledge, memory dependency pairs are the first modality through which humans can articulate their memory processes for decision-making.

A.3 LONG-TERM CREDIT ASSIGNMENT

The Long-Term Credit Assignment Problem highlights the difficulty of training agents to attribute consequences of actions over extended time frames Sutton (1984); Bengio & Frasconi (1993); Bengio et al. (1994). Humans can base decisions on events from years or even decades past. However, agents today struggle with credit assignment over even short horizons. This challenge primarily arises from vanishing or exploding gradients during backpropagation through time or the memory mechanism's inability to retain relevant information amidst noise Bengio & Frasconi (1993). Proposed solutions include adding skip connections to reduce backpropagation distances Ke et al. (2018); Hung et al. (2019) and using self-attention in transformers Vaswani et al. (2023), which allows direct gradient flow between relevant timesteps. However, self-attention has been shown to not improve long-term credit assignment nor fully exploit all available information in its context Liu et al. (2023); Ni et al. (2023). In this work, memory dependency pairs are shown to assist self-attention in long-term credit assignment.

A.4 LEARNING MEMORY MECHANISMS

RNNs were initially augmented with memory by incorporating hidden states and gating mechanisms, such as in Long Short-Term Memory (LSTM) networks Hochreiter & Schmidhuber (1997); Cho et al. (2014); Burtsev et al. (2021). Other approaches include integrating RNNs with differentiable memory that is key-addressable Graves et al. (2014); Weston et al. (2015); Graves et al. (2016); Wayne et al. (2018). Some researchers have also experimented with augmenting RNNs with stack-like memory modules Joulin & Mikolov (2015). Furthermore, combining LSTMs for short-term working memory with key-addressable memory for long-term episodic memory has been explored Fortunato et al. (2020). Another significant development is the integration of Transformers with differentiable memory that can be either key or content addressable Kang et al. (2023); Bessonov et al. (2023).

Allen et al. (2024) uses the difference in temporal difference and Monte Carlo value estimates to detect partial observability and improve memory retention for RNNs. Our work is the first to explore learning memory mechanisms through expert demonstrations.

B Limitations

This study introduces **memory dependency pairs** to enhance long-term credit assignment in POMDP tasks necessitating complex memory mechanisms. We proposed AttentionTuner, a method designed for self-attention architectures to effectively leverage memory dependency pairs. Empirical evaluation on Memory Gym and LTMB benchmarks demonstrate that AttentionTuner effectively mitigates the local optima challenges in memory tasks, either by accelerating escape from such optima or by circumventing them entirely (Figure 4). This optimization improvement significantly increases success metrics across various tasks compared to vanilla Transformers (Table 1). Notably, these learning benefits are achievable with minimal annotation, requiring as few as 0.1% of training trajectories to be annotated. This level of efficiency makes AttentionTuner a practical tool for real-world applications where learning complex memory mechanisms poses significant challenges.

Our approach primarily aims to enhance long-term episodic memory, thriving particularly when the expert attention matrix E exhibits sparsity. However, it encounters limitations in scenarios involving short-term or semantic memory, a challenge exemplified by the performance in the Searing Spotlights task (Table 1). To mitigate this limitation, incorporating a short-term memory mechanism, like frame stacking, could be an effective strategy to complement AttentionTuner's long-term episodic memory.

The practicality of pinpointing precise timesteps for memory dependency pairs becomes cumbersome with longer episode horizons. To address this issue, a plausible direction could involve the summarization of token sequences into more generalized and abstract representations such as done in HCAM Lampinen et al. (2021). These summary "token chunks" would allow for annotators to connect two events with natural language or approximate timesteps instead of having to connect two exact timesteps.

Another constraint stems from the Transformer model's mechanism of incorporating all preceding observations into its context, posing scalability challenges for tasks with extended horizons. Exploring hierarchical attention mechanisms or adopting a key-value cache system Wu et al. (2022) presents promising avenues. Memory dependency pairs could serve as valuable assets in these contexts, guiding the prioritization and retention of pivotal events within each hierarchical layer or assisting in the optimization of key-value cache retrieval and management strategies.

While the focus of this work has been on integrating memory dependency pairs within the Transformer architecture, memory dependency pairs are applicable to a variety of neural architectures. For instance, in RNNs, a reconstruction loss on hidden states could promote memory retention, while in key-addressable, differentiable memory systems, a loss could encourage accurate key additions and queries. State space models can be viewed as minimizing an online learning objective Liu et al. (2024), and therefore memory dependency pairs can be used to emphasize which tokens the model should prioritize for retention (like a re-weighted regret). These ideas are left as an exciting frontier for future research endeavors.

C Pseudocode

Algorithm 1 AttentionTuner

```
Require: Expert demonstrations $D = \{\tau_i\}_{i=1...M}$, each $\tau_i = \big(\{(o_j, a_j)\}_{j=1...|\tau|}, \mathcal{M}_i\big)$
Require: Hyperparameter for memory loss scaling $\lambda$
Require: Learning rate $\eta$
Require: Transformer model with CNN and MLP embedders, parameterized by $\theta$
 1: for all epochs do
 2:     for all $\tau_i \in D$ do
 3:         Extract observation-action pairs and memory dependency pairs: $\{(o_j, a_j)\}$, $\mathcal{M}_i$
 4:         $o_{\text{emb}} \leftarrow \text{CNN}(\{o_j\})$
 5:         $a_{\text{emb}} \leftarrow \text{MLP}(\{a_j\})$
 6:         input_seq $\leftarrow$ PositionalEncoding$(o_{\text{emb}}, a_{\text{emb}})$
 7:         $\{\hat{a}_j\}, A \leftarrow$ Transformer(input_seq) {$A$ is the self-attention matrix $\sigma(QK^\top)$ of the first
            attention head in the first Transformer layer}
 8:         Initialize $E \leftarrow \{0\}^{|\tau_i| \times |\tau_i|}$
 9:         for all $(p, q) \in \mathcal{M}_i$ do
10:             $E[q][p] = 1$
11:         end for
12:         $\mathcal{L}_{\text{memory}} \leftarrow \text{BinaryCrossEntropy}(A, E)$
13:         $\mathcal{L}_{\text{IL}} \leftarrow \text{NegativeLogLikelihood}(\{\hat{a}_j\}, \{a_j\})$
14:         $\mathcal{L} \leftarrow \mathcal{L}_{\text{IL}} + \lambda \cdot \mathcal{L}_{\text{memory}}$
15:         $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
16:     end for
17: end for
```

D Neural Network Architectures

Only the observation and action embedders differ in architecture between the two benchmarks. For both benchmarks, 4 Transformer layers were used with 2 self-attention heads per layer. Additionally, $d_{\text{model}} = 512$ for all experiments.

D.1 Causal Transformer

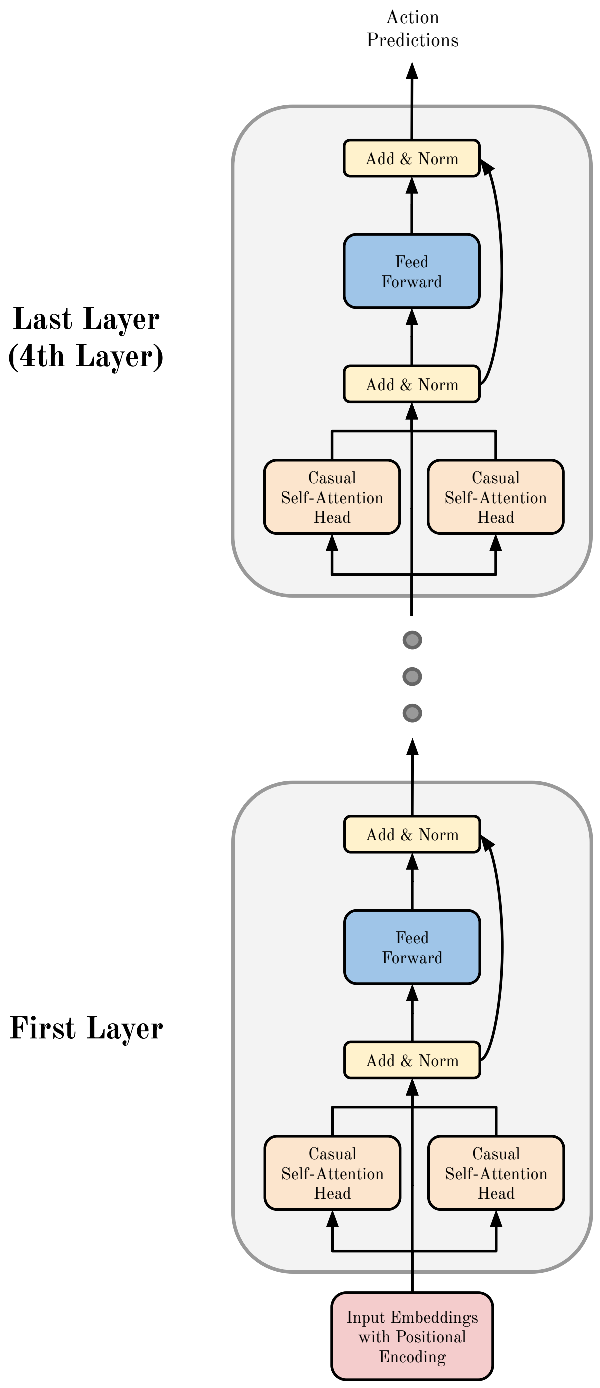

Action Predictions
Last Layer (4th Layer)
Add & Norm
Feed Forward
Add & Norm
Casual Self-Attention Head
Casual Self-Attention Head
First Layer
Add & Norm
Feed Forward
Add & Norm
Casual Self-Attention Head
Casual Self-Attention Head
Input Embeddings with Positional Encoding

Figure 6: Causal Transformer architecture with 4 layers and 2 self-attention heads per layer as used in all experiments. The memory loss was applied to a single self-attention head in the first Transformer layer.

D.2 Memory Gym

```
Transformer(
  (image_embedding): ImageEmbedding(
    (cnn): Sequential(
      (0): Conv2d(3, 32, kernel_size=(8, 8), stride=(4, 4))
      (1): ReLU()
      (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
      (3): ReLU()
      (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
      (5): ReLU()
    )
    (fc): Sequential(
      (0): Linear(in_features=3136, out_features=512, bias=True)
      (1): Tanh()
    )
  )
  (action_embedding): ActionEmbedding(
    (mlp): Sequential(
      (0): Linear(in_features=4, out_features=512, bias=True)
      (1): Tanh()
    )
  )
  (positional_encoding): PositionalEmbedding()
  (embedding_LN): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (transformer_layers): ModuleList(
    (0-3): 4 x TransformerLayer(
      (self_attention): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512,
    out_features=512, bias=True)
      )
      (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (feedforward): Sequential(
        (0): Linear(in_features=512, out_features=2048, bias=True)
        (1): ReLU()
        (2): Linear(in_features=2048, out_features=512, bias=True)
        (3): Dropout(p=0.1, inplace=False)
      )
      (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
    )
  )
  (output): Linear(in_features=512, out_features=4, bias=True)
)
```

D.3 LTMB

```
Transformer(
  (image_embedding): ImageEmbedding(
    (cnn): Sequential(
      (0): Conv2d(20, 40, kernel_size=(3, 3), stride=(1, 1), padding=(1,
    1))
      (1): ReLU()
      (2): Conv2d(40, 80, kernel_size=(3, 3), stride=(1, 1), padding=(1,
    1))
      (3): ReLU()
    )
    (fc): Sequential(
      (0): Linear(in_features=3920, out_features=512, bias=True)
      (1): Tanh()
    )
  )
  (action_embedding): ActionEmbedding(
    (mlp): Sequential(
      (0): Linear(in_features=7, out_features=512, bias=True)
      (1): Tanh()
    )
  )
  (positional_encoding): PositionalEmbedding()
  (embedding_LN): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (transformer_layers): ModuleList(
    (0-3): 4 x TransformerLayer(
      (self_attention): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=512,
    out_features=512, bias=True)
      )
      (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (feedforward): Sequential(
        (0): Linear(in_features=512, out_features=2048, bias=True)
        (1): ReLU()
        (2): Linear(in_features=2048, out_features=512, bias=True)
        (3): Dropout(p=0.1, inplace=False)
      )
      (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
    )
  )
  (output): Linear(in_features=512, out_features=7, bias=True)
)
```

E Application of the Memory Loss in Transformers

Tasks	No Layer No Heads	First Layer All Heads	Middle Layer All Heads	Last Layer All Heads	First Layer Single Heads	Middle Layer Single Heads	Last Layer Single Heads
Mortar Mayhem	20.8 ± 42.2	$\mathbf{100} \pm 0.1$	99.7 ± 0.6	99.9 ± 0.1	99.8 ± 0.4	100 ± 0.1	39.9 ± 52.1
Hallway	42 ± 50.5	99.9 ± 0.2	$\mathbf{100} \pm 0$	89.4 ± 22.5	99.9 ± 0.1	99.8 ± 0.3	99.12 ± 1.7

Table 2: Success rate and 90% confidence interval achieved on memory loss ablations

The memory loss in has to be applied to a self-attention head in the Transformer. We posited in Section 3 that applying the memory loss to the first Transformer layer would make the memory dependency pairs more meaningful as the self-attention mechanism attends over the raw observation embeddings. This hypothesis is supported by the results in Table 2, which demonstrate that implementations applying the memory loss to the first layer consistently yield near-perfect success rates, surpassing other configurations. While we also speculated that dedicating memory loss to a single attention head would permit the remaining heads to engage in other memory processes, this distinction was not markedly evident in our results. The probable explanation is that memory dependency pairs in these relatively simple tasks sufficiently encapsulate all necessary memory functions, diminishing the benefit of isolating the memory loss to a single head. Attempting to distribute memory dependency pairs among various heads (Appendix P Figure 10), especially in the context of Searing Spotlights with its large amount of memory dependency pairs, did not yield a notable improvement in performance.

F Environment Descriptions

F.1 Memory Gym

Memory Gym features three tasks—Mortar Mayhem, Mystery Path, and Searing Spotlights—set within a 7×7 gridworld. Agents receive 84×84 RGB image observations of the gridworld. For Mortar Mayhem and Mystery Path, the discrete action space includes: `move forward`, `turn left`, `turn right`, and `nop`. Searing Spotlights employs a multi-discrete action space, allowing movement in cardinal or ordinal directions plus a `nop` option.

Mortar Mayhem In this task, the agent memorizes and later executes a sequence of commands, indicated by arrows. An expert would annotate memory dependency pairs (p, q), with o_p representing the observation displaying the command and o_q the observation of its execution.

Mystery Path Agents navigate a gridworld with an invisible path, restarting from the beginning if they deviate. To progress, they must remember their path to the deviation point. An expert would annotate memory dependency pairs (p, q) where o_p is a cell adjacent to the cell at o_q.

Searing Spotlights Agents aim to reach a door in a 2D plane after collecting a key, initially visible but obscured after 6 timesteps by dimming lights. The agent has to make it to the key and door while avoiding "searing spotlights". An expert would annotate memory dependency pairs $(0, q), (1, q), \ldots, (q - 1, q)$ as agents must recall their starting position and all previous actions to deduce their current location.

F.2 Long-term Memory Benchmark

The Long-term Memory Benchmark (LTMB) comprises three tasks: Hallway, Ordering, and Counting, set in the Minigrid environment Chevalier-Boisvert et al. (2023). Agents navigate a gridworld, receiving standard Minigrid $7 \times 7 \times 3$ state-based observations. The discrete action space includes `move forward`, `turn left`, `turn right`, and `nop`.

Hallway Agents identify and enter a hallway with an object matching one in the start room. The agent's view is limited to the 7×7 grid ahead. The expert annotates $(1, q)$ where timestep 1 represents the initial object observation.

Ordering Agents memorize the sequence of the first 18 colored objects encountered. When choosing between two objects, the agent selects the one appearing earlier in the sequence. An expert would annotate memory dependency pairs (p, q), connecting the first observation o_p of an object to the query observation o_q.

Counting In this task, agents traverse gallery rooms, memorizing six objects each, and query rooms, deciding which door to pass based on the parity of a query object's previous appearances. Passing through the wrong door ends the episode. An expert would annotate (p, q) where o_p is a gallery room containing the query object and o_q is the query room.

G Environment Settings

Memory dependency pairs are annotated for every expert trajectory collected. In practice, memory dependency pairs would need to be annotated by the human demonstrator, for example via a graphical user interface of some sort. While it remains to be verified that this process can be made relatively seemless for human experts, for the purposes of this paper, we sidestep this human-computer interaction issue by simulating both the expert demonstrations and the annotation of memory dependency pairs.

G.1 Mortar Mayhem

Discrete action movements were used with a command count of 10. Settings were default, except where noted. A total of 4,000 expert trajectories were collected, each with 118 timesteps.

G.2 Mystery Path

The origin and goal were not shown, with other settings at default. Training involved 4,000 expert trajectories, averaging 43 timesteps, and reaching up to 128 timesteps.

G.3 Searing Spotlights

The agent was visible for 6 timesteps before the lights dimmed completely. A single coin was used to unlock the exit. Other settings were left at the default. A single coin unlocked the exit, and other settings remained default. Training included 40,000 expert trajectories, averaging 30 timesteps, with a maximum of 75.

G.4 HALLWAY

The environment length was set to 30. A total of 5,000 expert trajectories were collected, averaging 67 timesteps, and maxing at 145 timesteps.

G.5 ORDERING

The environment length was set to 50. A total of 5,000 expert trajectories were collected, each with a length of 68 timesteps.

G.6 COUNTING

The environment length was 20, with test room frequency at 30% and empty tile frequency at 10%. A total of 10,000 expert trajectories were collected, averaging 97 timesteps, with a maximum of 140.

H TRAINING HYPERPARAMETERS

The following hyperparameters were shared by both AttentionTuner and the vanilla Transformer across all experiments:

Hyperparameter	**Value**	**Brief Description**
batch size	64	number of samples in each training iteration
learning rate	10^{-4}	learning rate for gradient descent
optimization algorithm	Adam	optimization algorithm used
β_1	0.9	exponential decay rate for first moment estimates in Adam
β_2	0.999	exponential decay rate for second moment estimates in Adam
epsilon	10^{-8}	small constant for numerical stability
weight decay	0	weight regularization
λ	10	memory loss multiplier defined in Section 3 (only for AttentionTuner)

Table 3: Hyperparameters used in experiments along with their brief descriptions

Only the number of training epochs differed between methods and tasks.

Task	**AttentionTuner**	**Vanilla Transformer**
Mortar Mayhem	300	600
Mystery Path	300	300
Searing Spotlights	120	120
Hallway	300	600
Ordering	300	600
Counting	600	600

Table 4: Number of training epochs used for each task

All experiments in Table 2 (Section E) used 300 training epochs. All experiments in Figure 5a (Section 4.3) used 300 training epochs expect for vanilla Transformer runs (0%) and Mortar Mayhem 0.1% and 1% which used 600 epochs.

I RANDOM SEEDS

All experiments were run with random seeds 1 through 5 except for the following:

- 10 random seeds were used for vanilla Transformer on LTMB's Hallway task in Table 1.

- 15 random seeds were used for vanilla Transformer on LTMB's Ordering task in Table 1.

More training runs were used for these experiments due to their high variability.

J Welch t-test for Statistical Significance

We applied the Welch t-test to assess the statistical significance of the performance differences between AttentionTuner and the vanilla Transformer, as reported in Table 1.

Task	**p-value**
Mortar Mayhem	0.016
Mystery Path	0.012
Searing Spotlights	0.546
Hallway	0.014
Ordering	0.008
Counting	0.261

Table 5: Welch t-test p-values for Performance Comparison Across Tasks

K Average Learning Curves with 90% Confidence Intervals

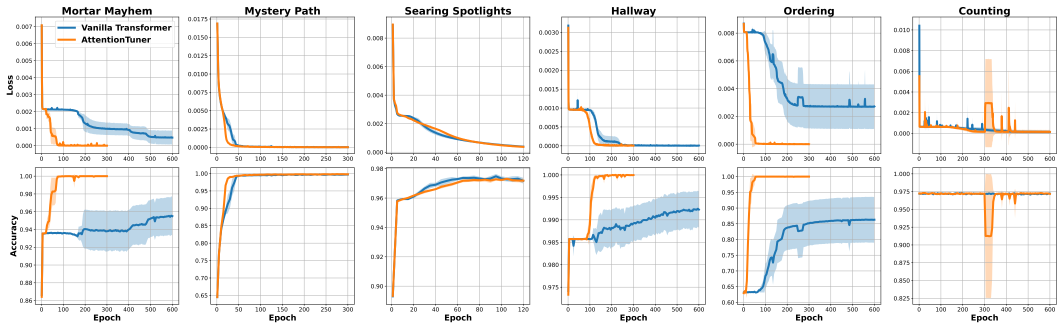


Figure 7: Mean learning curves with 90% confidence intervals for Memory Gym tasks.

L Missing Annotations Ablation Data

Table 6: Missing Annotations Ablations

Tasks	0%	0.1%	1%	5%	10%	90%	100%
Mortar Mayhem	20.8 ± 42.2	23 ± 41.2	80.3 ± 41.3	93.4 ± 11.1	100 ± 0	100 ± 0	99.8 ± 0.4
Hallway	42 ± 50.5	79.7 ± 40	96.7 ± 6.8	99.9 ± 0.2	99.8 ± 0.5	100 ± 0	99.9 ± 0.1

The table presents the success rates and their corresponding 90% confidence intervals for tasks under different levels of missing annotations. The percentages indicate the proportion of demonstration trajectories with fully annotated memory dependency pairs. Figure 5a illustrates these results in a bar plot.

M IMPRECISE ANNOTATIONS ABLATIONS DATA

Table 7: Partially Imprecise Annotations Ablations

TASKS	0	1	2	5	10	20
MORTAR MAYHEM	99.8 ± 0.4	99.22 ± 0.7	99.7 ± 0.5	100 ± 0.1	79.3 ± 38.2	21.2 ± 42
HALLWAY	99.9 ± 0.1	99.9 ± 0.1	82 ± 38.5	81.7 ± 38.8	88.4 ± 23.7	56.7 ± 45.6

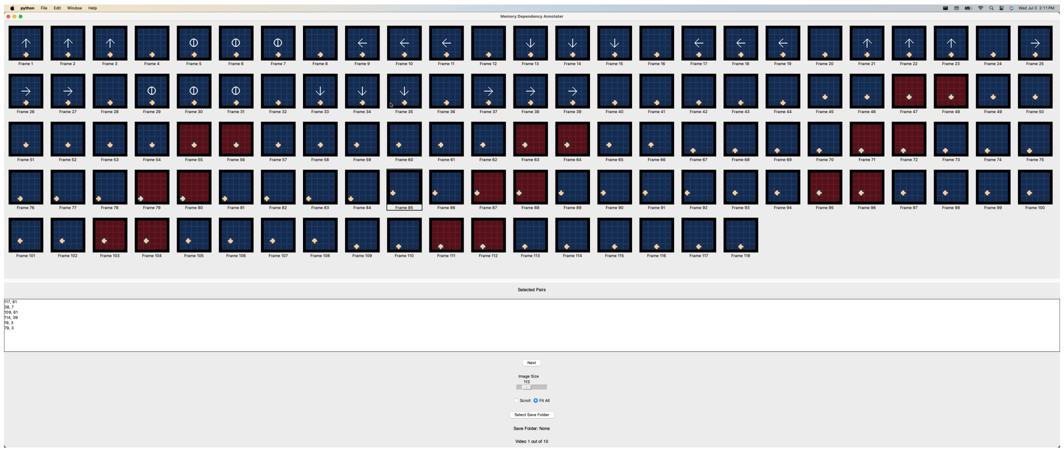
The table presents success rates and their corresponding 90% confidence intervals for tasks with varying levels of imprecise annotations. For each memory association pair (p, q), the recalled timestep p is perturbed by a delta Δ drawn from a normal distribution $\mathcal{N}(0, \sigma)$. The top row indicates the standard deviation σ.

Table 8: Imprecise Annotations Ablations

TASKS	0	0.5	0.75	1	1.5	2
MORTAR MAYHEM	99.8 ± 0.4	100 ± 0	99.4 ± 1	100 ± 0	57.5 ± 50.2	20 ± 42.6
HALLWAY	99.9 ± 0.1	100 ± 0	75.9 ± 36.4	20.3 ± 24.2	24.5 ± 36	3.5 ± 2.6

The table presents success rates and their corresponding 90% confidence intervals for tasks with varying levels of imprecise annotations. For each memory association pair (p, q), the recalled timestep p and the timestep of recall q are both perturbed by deltas Δ_p and Δ_q drawn from a normal distribution $\mathcal{N}(0, \sigma)$. The top row indicates the standard deviation σ.

N ANNOTATING MEMORY DEPENDENCY PAIRS



```
117, 61
38, 7
109, 61
114, 39
19, 3
79, 3
```

Image Size 113

Scroll [✓ Fit All]

Save Folder: None

Video 1 out of 10

Figure 8: Graphical User Interface used to collect memory dependency pairs.

Task	**Demonstration Collection Time**	**Annotation Time**
Mortar Mayhem	8 min 20 sec	21 min 6 sec
Hallway	2 min 9 sec	5 min 59 sec

Table 9: The time it took to collect and annotate 10 demonstrations is recorded.

Figure 8 shows the graphical user interface used to annotate demonstrations with memory dependency pairs. A single demonstration is loaded onto the GUI at a time. To annotate a memory dependency pair (p, q), the annotator clicks on frame p and then clicks on frame q. Table 9 compares the time it takes for humans to collect demonstrations versus annotating them.

O COMPUTING THE VALUE OF MEMORY DEPENDENCY PAIRS

O.1 MORTAR MAYHEM

There are 4000 demonstrations in the standard Mortar Mayhem training dataset (Appendix G). Only 400 of these demonstrations have to be annotated to achieve a 100% success rate (Figure 5a). It takes the vanilla Transformer 16,000 demonstrations to achieve a 100% success rate (Figure 5b). This means that each memory dependency pair annotation is worth $\frac{16000}{400} = 40$ additional demonstrations. According to Table 9, annotating a single demonstration takes $\frac{1266}{10} = 126.6$ seconds while collecting 40 additional demonstration takes $\frac{500}{10} * 40 = 2000$ seconds. This results in a human labor time savings factor of $\frac{2000}{126.6} = 15.7977883 \approx 16$.

O.2 HALLWAY

There are 5000 demonstrations in the standard Hallway training dataset (Appendix G). Only 250 of these demonstrations have to be annotated to achieve a 100% success rate (Figure 5a). It takes the vanilla Transformer 10,000 demonstrations to achieve a 100% success rate (Figure 5b). This means that each memory dependency pair annotation is worth $\frac{10000}{250} = 40$ additional demonstrations. According to Table 9, annotating a single demonstration takes $\frac{359}{10} = 35.9$ seconds while collecting 40 additional demonstrations takes $\frac{129}{10} * 40 = 516$ seconds. This results in a human labor time savings factor of $\frac{516}{35.9} = 14.3732591 \approx 14$.

P Expert Truth Attention Heatmaps

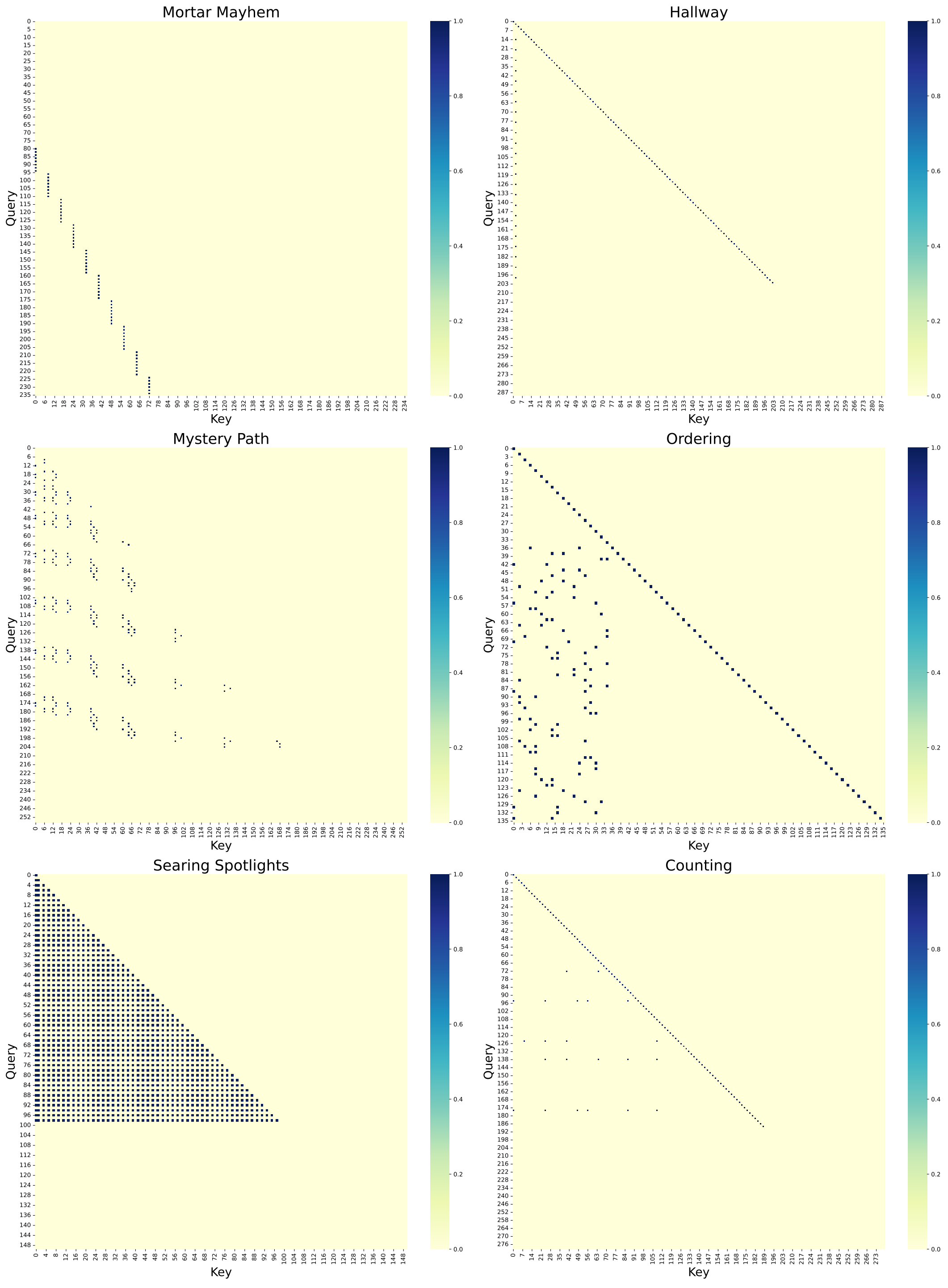


Figure 9: Sample expert attention heatmaps for E

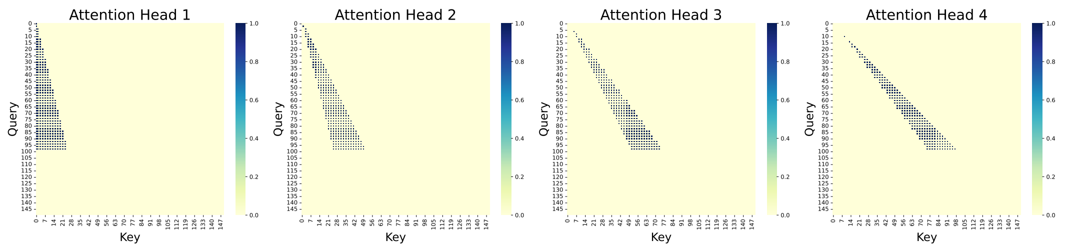


Figure 10: Sample expert attention heatmaps for split memory loss on Searing Spotlights. Notably, this experiment uniquely employs 4 self-attention heads, diverging from the typical configuration of 2 self-attention heads used in all other experiments.

Q Learned Heatmaps

Figures 11 and 12 illustrate all self-attention heads in the Transformer as learned by AttentionTuner and the vanilla Transformer after training on Mortar Mayhem. Figure 11 demonstrates that, with the application of memory loss to the initial head of the first layer, the correct memory mechanism was effectively acquired. Conversely, Figure 12 indicates that in the absence of memory loss, solely employing the behavioral cloning objective did not facilitate the acquisition of the precise memory mechanism. However, a partially accurate memory mechanism is observable in the first head of the second layer.

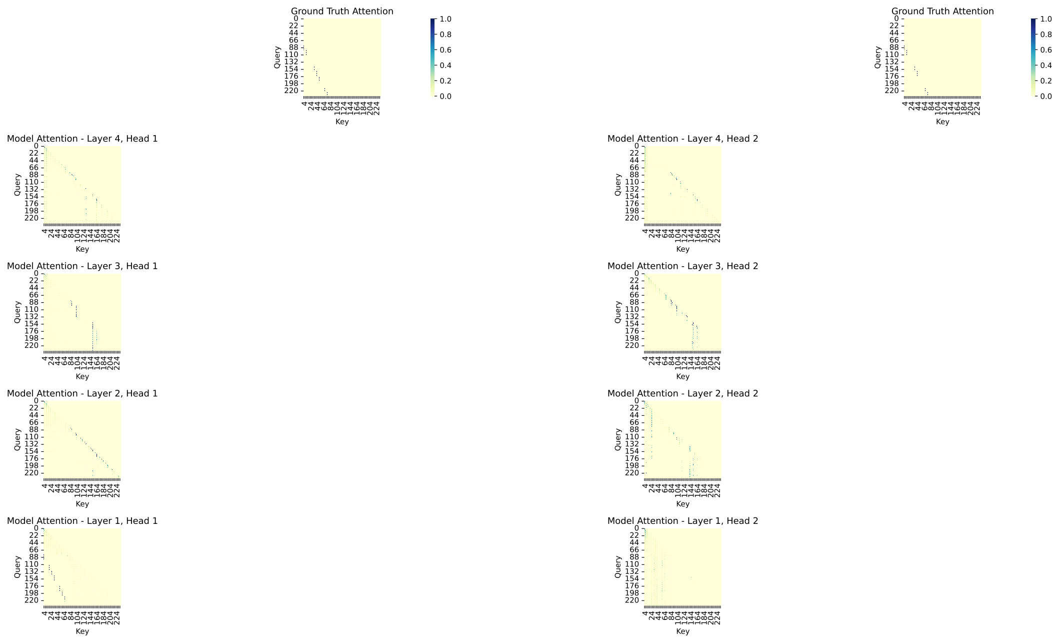


Figure 11: AttentionTuner's learned self-attention heatmap for all attention heads on Mortar Mayhem.

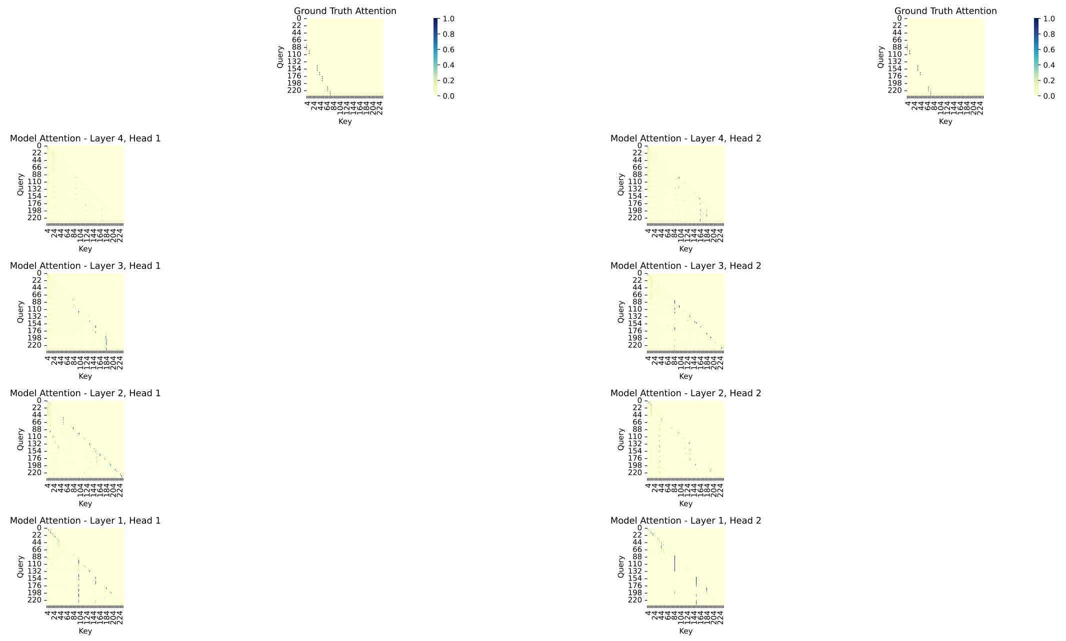


Figure 12: Vanilla Transformer's learned self-attention heatmap for all attention heads on Mortar Mayhem.

R Computational Resources

All experiments were conducted on Nvidia A40 and A100 GPUs with 40 or 80 GB of memory. The computational node featured two Intel Xeon Gold 6342 2.80GHz CPUs with 500 GB RAM. Experiment durations varied between 1 and 4.5 hours.