RECAP Capstone Project: The Attention Neuron Is Not Truly Permutation Invariant

Chuen Yang Beh

June 2025

Abstract

The Attention Neuron [1] is a peculiar RL policy architecture which claims to create permutation-invariant representations of an agent's observations. In this week-long project, we observe that the architecture **can** conceptually be permutation invariant, but prove empirically that the implementation by Tang and Ha [1] is **not** permutation invariant in practice, due to the use of recurrent LSTM states in a permutation-sensitive manner. Fascinatingly, however, we observe that the architecture is **nonetheless resilient** to permutations of the observation vector. As such, we investigate how this architecture achieves this astonishing property through visualizations, ablations, and mathematical analysis on a toy environment (CartPoleSwingUpHarder [2]).

1 Introduction

The Attention Neuron architecture [1] appears to be motivated from a neuroscience perspective. The authors observe that humans are sometimes capable of orienting themselves despite "sensory substitutions" (e.g. riding a bike backwards instead of forwards). Justifying that this would imbue robustness and generalisability in agents' capabilities, the authors then propose the Attention Neuron architecture as a way to achieve this property.

While the authors' results [1] are extremely promising and reproducible, our experiments reveal that the implementation of the Attention Neuron architecture is not truly permutation invariant, due to the use of recurrent LSTM states in a permutation-sensitive manner.

2 Preliminary: The Attention Neuron

Drawing inspiration from the Set Transformer architecture [3], the Attention Neuron architecture claims to achieve permutation invariance by using cross-attention, where the query vector is a set of learned positional embeddings, the

key vector consists of observation and previous-action features, and the value vector is the observation vector itself.

2.1 Architecture Overview

A figure of the architecture is shown in Figure 1.

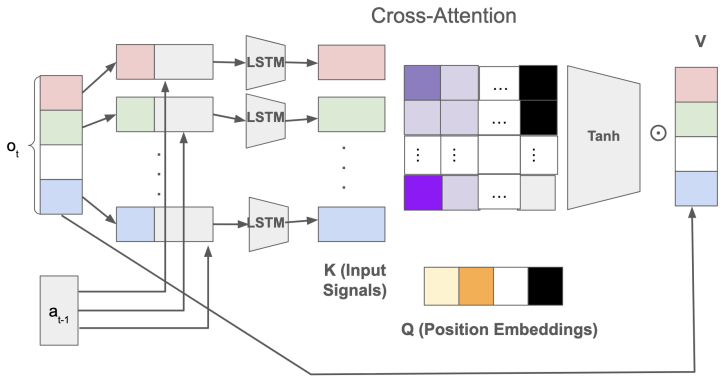


Figure 1: The Attention Neuron architecture from [1], redrawn for clarity.

At a high level, the architecture's forward pass can be described as follows (at each timestep t):

- The observation vector o_t at time t is broken up into d_{obs} *signals*, $\{o_{t,1}, o_{t,2}, \ldots, o_{t,d_{obs}}\}$.
- The signals are then independently concatenated with the previous action a_{t-1}, forming the *info signals* $I_{t,k} = (o_{t,k}) \oplus a_{t-1}$ with dimension $1 + d_{act}$ for $k = 1, 2, \ldots, d_{obs}$.
- The info signals (along with hidden LSTM state h_{t-1} and cell state c_{t-1}) are then passed (independently) through an LSTM Cell L, which computes new hidden states h_t of dimension d_{out} and cell states c_t of dimension d_{out}.
- The LSTM outputs and positional embeddings P are then mixed via bidirectional cross-attention, with the query vectors being the positional embeddings P of dim d_{out}, and the key vectors being the LSTM hidden state h_t. [1] This results in an attention matrix A_t of dimension $d_{out} \times d_{obs}$.
- Finally, the attention matrix A_t is multiplied with the observation vector o_t, resulting in a feature vector f_t of dimension d_{out}. This feature vector represents the "digested" observation vector, and is supposed to remain invariant to permutations of the observation vector.

[1] The non-linearity of the attention matrix, rather than being softmax, is a tanh function.

Mathematically speaking, the forward pass can be written as:

$$f_t(o_t, a_{t-1}) = \tanh\left(\underbrace{\tanh\left(P\left(\bigoplus_{i=1}^{d_{obs}} L(I_{t,i}, h_{t-1}, c_{t-1})\right)^T\right)}_{A_t} o_t\right)$$

where $\oplus$ denotes the concatenation operation (dimension implied), L is the LSTM function mapping, and A_t is the attention matrix.

2.2 Claimed Permutation Invariance of Attention Neuron

Some manual work and elementary linear algebra intuition shows that the claimed permutation invariance of the Attention Neuron architecture arises from the fact that whenever the observation o_t is permuted along its rows, the column space of A_t would theoretically be permuted the same way. Hence $A'_t o'_t = A_t o_t$.

This forms a basis for the authors' claim that the Attention Neuron architecture is permutation invariant [1].

Unfortunately, the authors do not provide a formal proof of this claim, and the decision to use an LSTM to model time dependencies in each info signal $I_{t,k}$ in fact renders the architecture **permutation sensitive**. We prove this empirically in the next section.

3 Experiments

3.1 Choice of Environment

We tested the Attention Neuron architecture on the CartPoleSwingUpHarder environment [2]. It is best thought of as an extremely difficult version of the CartPole environment [4, 5], as the cart not only needs to swing the pole up from a downwards-facing position, but also needs to balance the pole upright for an extended period of time.

This environment was primarily chosen since it was the simplest of all the environments used by the authors of the Attention Neuron paper [1]. With a low-dimensional observation space and no hidden layers in the policy architecture, it will be easier to visualize, compare, and contrast the effects of permutations on the Attention Neuron architecture.

3.2 Permutations Within Episodes

Considering that the authors tout the Attention Neuron architecture's robustness to permutations even within episodes [1], we decided it would be interesting to reproduce the authors' experiments in permuting the observation vector o_t multiple times within an episode.

3.3 Ablations

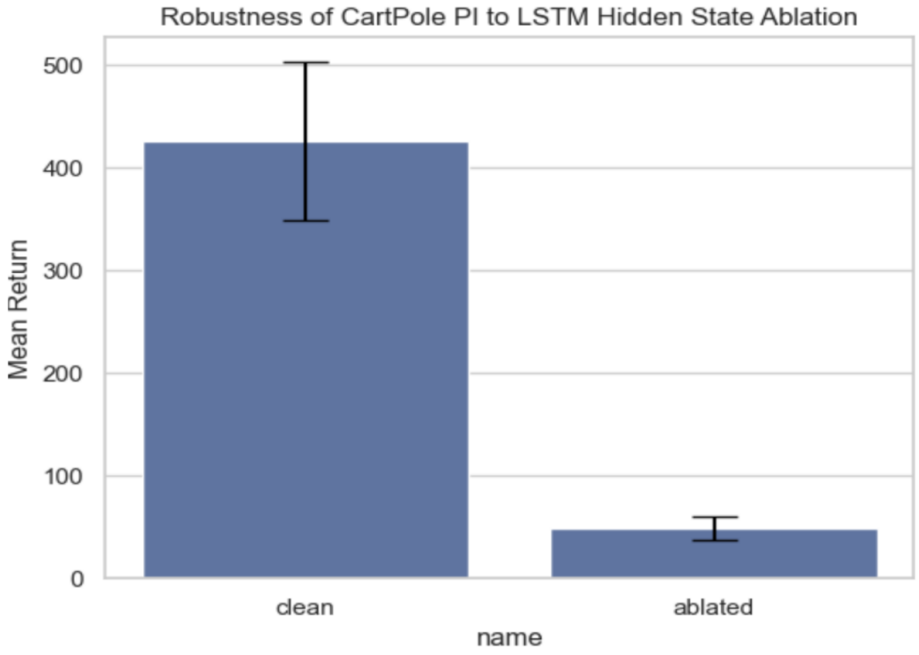
We performed the following ablations on the Attention Neuron architecture:

- **Zeroing the LSTM Recurrent State**: We zeroed the LSTM recurrent state at every time step.
- **Resetting the LSTM Recurrent State Every Permutation**: We reset the LSTM recurrent state to zero at whenever the observation vector was permuted.
- **Permuting the LSTM Recurrent State Every Permutation**: We permuted the LSTM recurrent state whenever the observation vector was permuted.

4 Results & Discussion

4.1 Zeroing the LSTM Recurrent State Causes Model Collapse

As shown in Figure 4.1, zeroing the LSTM recurrent state at every time step causes the agent's performance to collapse.



While it can be tempting to explain this as the LSTM's pivotal role in the model's success in CartPoleSwingUpHarder, we note that similar problems

have been solved before using non-recurrent architectures such as MLPs ([6]). Hence, it is likely that the policy was simply trained to rely heavily on the LSTM recurrent state to solve the task.

4.2 The Attention Neuron Is Robust Against Within-Episode Permutations

Just as the authors show in their paper [1], we find that the Attention Neuron architecture is robust to permutations of the observation vector even within an episode. Figure 2 shows the results of our experiments, where we permuted the observation vector at every time step within an episode.

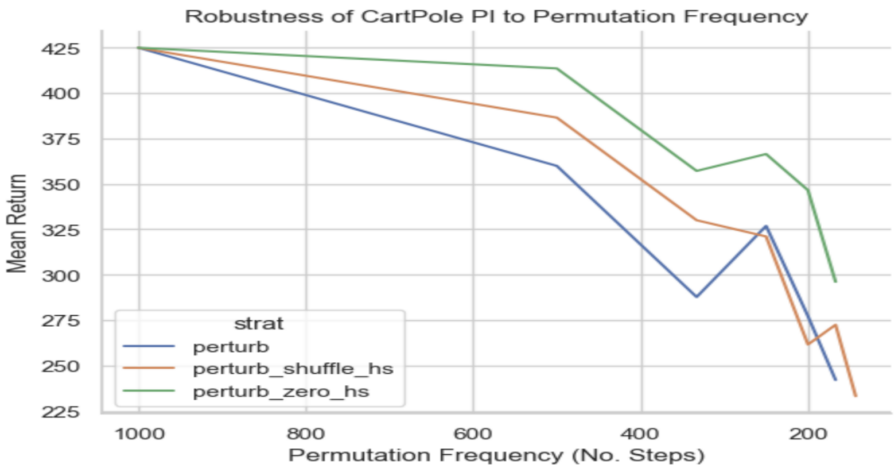


Figure 2: The Attention Neuron architecture is robust to permutations of the observation vector within an episode.

While the performance degrades somewhat, the agent's performance never truly collapses like in the previous section, as would be expected on the usual MLP architecture. In fact, it is possible to salvage the agent's performance somewhat through hidden state manipulation, as shown in Figure 2.

Unfortunately, permuting the LSTM recurrent state in the same way as the observation vector does not cause the agent's performance to fully recover. This leads us to believe that the Attention Neuron architecture may not be truly permutation invariant, for otherwise the recurrent state permutation would have been sufficient to recover the agent's performance FULLY.

4.3 The Attention Neuron is Not Truly Permutation Invariant

As shown in Figure 3, which is derived from randomly generated rollouts in CartPoleSwingUpHarder, the Attention Neuron architecture is indeed not truly permutation invariant, with one notable exception.

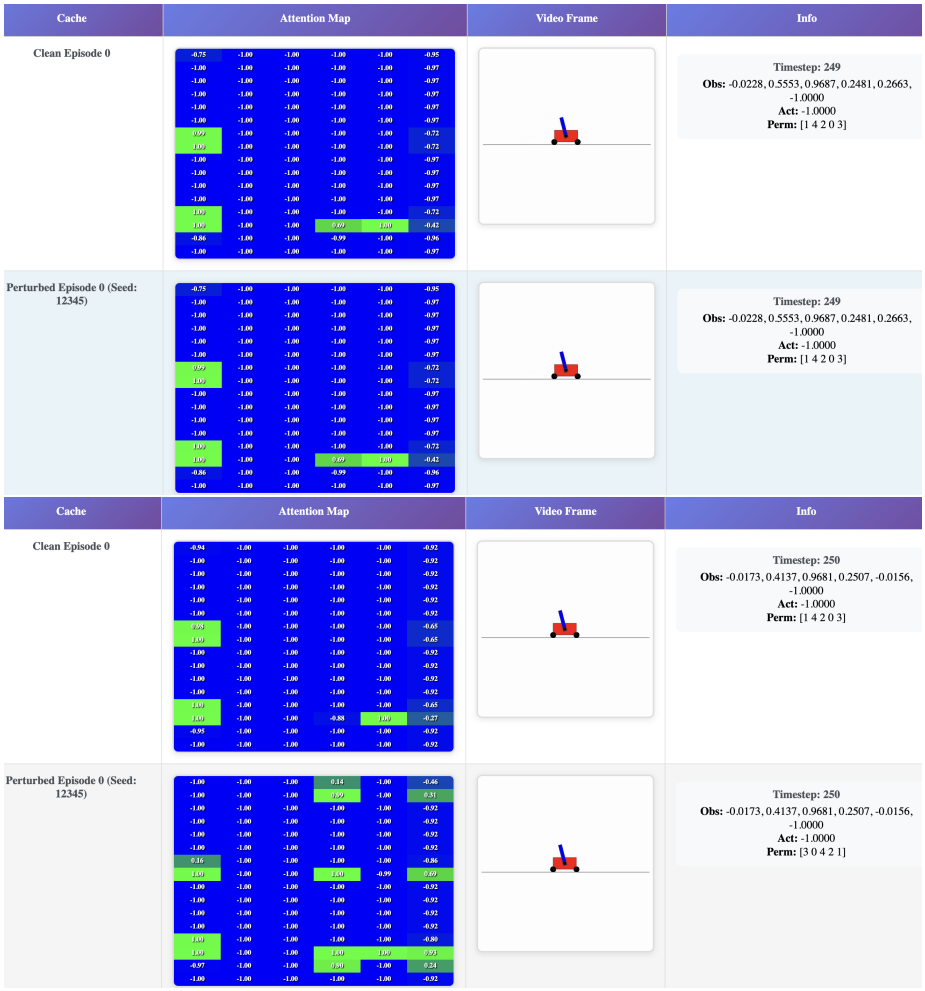


Figure 3: Notice how the attention masks (below) are no longer identical even after accounting for the permutation of the observation vector (above). This shows that the Attention Neuron architecture is not truly permutation invariant, as the LSTM recurrent state is not invariant to permutations.

If we ablate the LSTM recurrent state by zeroing it at every time step, the Attention Neuron architecture becomes permutation invariant, as shown in

Figure 4.

Figure 4: When the LSTM recurrent state is zeroed at every time step, the columns in the attention matrix have been permuted in the same way as the observation vector's rows. In this specific circumstance, we do see that the Attention Neuron architecture is permutation invariant.

This finding has some startling implications. For one, it shows that the Attention Neuron architecture can achieve robustness to permutation observations within episodes, even when its LSTM recurrent states have made the policy permutation sensitive overall.

This suggests that the architecture could have learned to rely on certain features of the input that are invariant to permutations, allowing it to maintain performance despite changes in the order of the input sequence.

5 Future Work

In future work, we would like to explore the following directions:

- **Testing on More Complex Environments**: We have thus far been focused on the mechanistic and architectural properties of the Attention Neuron. However, it would be interesting to see how the Attention Neuron architecture performs on more complex environments such as Ant, CarRacing, and Pong.
- **Examining Noisy Channels**: The authors [1] also state that the Attention Neuron architecture is robust to extraneous, noisy channels in the observation vector. A natural extension to our study is to analyze why this is the case, and whether the architecture can be made more robust to noisy channels.

- **Conducting More Ablations**: We have only conducted a few ablations on the Attention Neuron architecture. It would be interesting to conduct more ablations, such as ablations on the attention matrix, position embeddings, and information signals.
- **Optimizing Model Performance**: The authors [1] used an evolutionary strategy to optimize the performance of the Attention Neuron architecture *given a fixed compute budget*, by searching over multiple training instances simultaneously. However, this could mean that any individual model's potential is not fully realized, especially for more complex environments. Learning about the features which emerge over the course of training could also help us understand how the Attention Neuron architecture learns to be robust to permutations.

References

[1] Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning, 2021.

[2] C. Daniel Freeman, Luke Metz, and David Ha. Learning to predict without looking ahead: World models without forward prediction, 2019.

[3] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks, 2019.

[4] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024.

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

[6] Adithya Ramesh. Proximal policy optimization, 2022.