Home Assignment 6

Gavrish Boris

May 21, 2025

1 Task 2: Comparative Analysis

In this assignment, we investigate how different memory mechanisms affect the performance of a Decision Transformer model in a partially observable environment. The goal is to assess how memory—or its absence—impacts the model's ability to learn temporal dependencies and achieve high validation returns. We begin by testing the models with the baseline parameters provided in the problem statement.

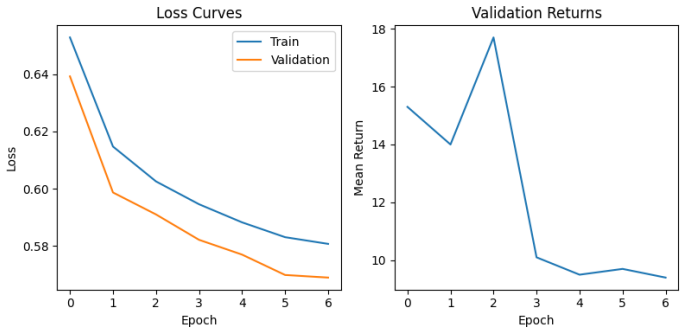

Figure 1: No Memory

The model without memory exhibits a decreasing training loss, but its validation returns are notably low, averaging 9.00 across 5 runs. This suggests that a memory-less model struggles to capture the temporal relationships critical for this task.

The Decision Transformer with GRU memory (DT+GRU) appears to overfit the training data, possibly due to the limited size of the dataset, leading to poor generalization.

In contrast, the Decision Transformer with LSTM memory (DT+LSTM) delivers the most consistent results, achieving an average validation return of 500, indicating robust performance.

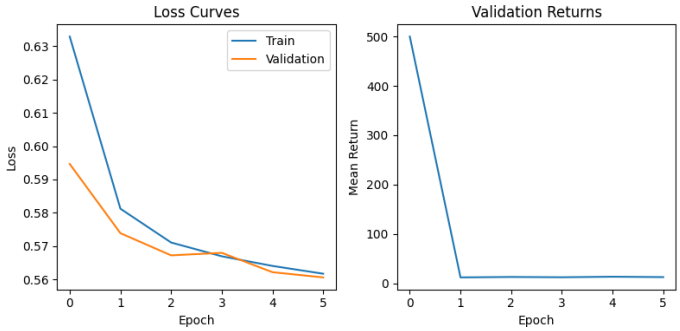

Figure 2: GRU Memory

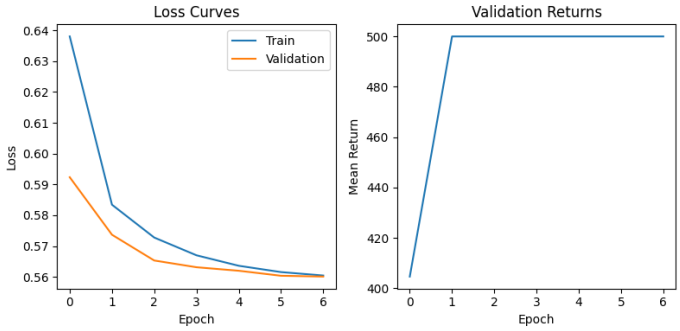

Figure 3: LSTM Memory

Next, we adjust the training parameters to explore their effects on model behavior.

1.1 Change in Context Length

We test how context length influences performance, hypothesizing that an overly long context might cause overfitting and degrade validation results.

1.1.1 Context Length = 10

With a context length reduced to 10 time steps, the GRU memory model converges successfully, suggesting that this shorter context suits its learning capacity.

The LSTM model continues to perform strongly, reaching a validation return of 500, confirming that a context length of 10 is adequate for effective training.

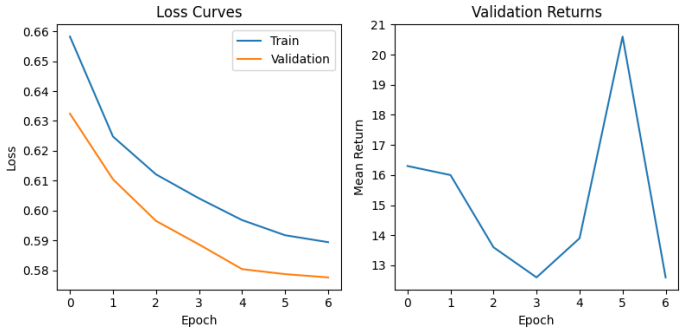

Figure 4: No Memory

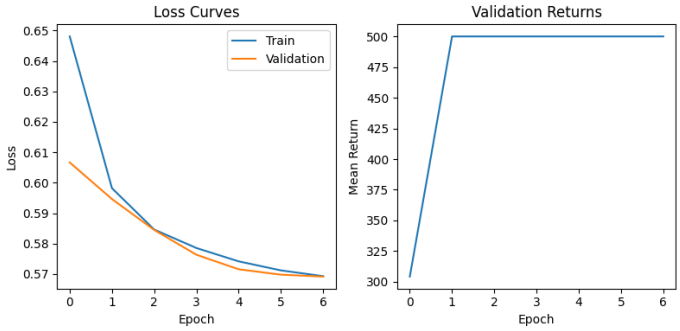

Figure 5: GRU Memory

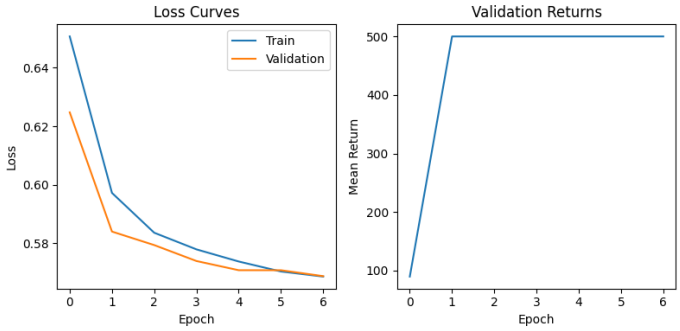

Figure 6: LSTM Memory

1.2 Change in Sample Size

To decrease computation time, we reduce the sample size to 200 trajectories and lower the number of training epochs.

The GRU model maintains its solid performance, while the LSTM model

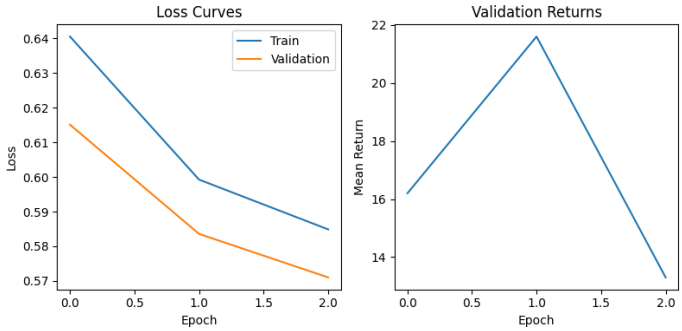

Figure 7: No Memory

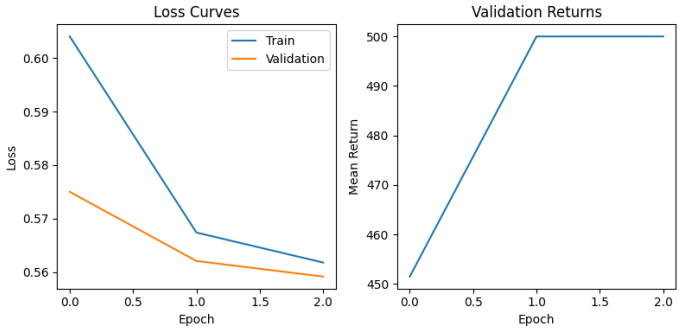

Figure 8: GRU Memory

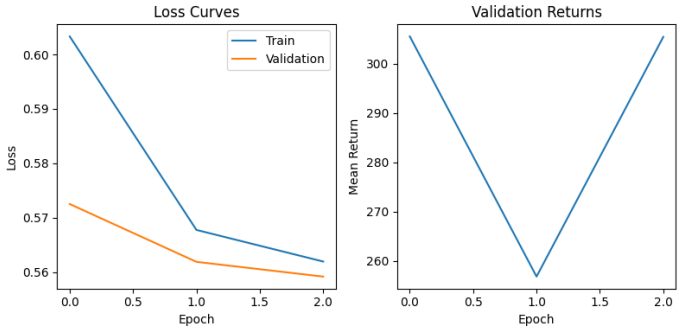

Figure 9: LSTM Memory

fails to converge, likely because the reduced epoch count limits its ability to learn effectively.

1.3 RTG Changes

We explore the impact of adjusting the target return-to-go (RTG) on the GRU memory model. Setting the target RTG to 300, the model terminates successfully, achieving an average return of 303.20 over 5 runs.

2 Task 3: Custom Memory Mechanism

We implement and evaluate several custom memory mechanisms inspired by reference articles. For optimal performance, we fix the context length at 10 and the sample size at 200, training each model for 5 epochs.

2.1 Fast Forgetful Memory (FFM)

The Fast Forgetful Memory (FFM) uses a complex-valued memory state to retain relevant past information while rapidly discarding outdated data through decay and context parameters. At each time step, it performs:

- Input gating via matrix multiplication and sigmoid activation
- Memory update with a decay factor
- Projection of the complex-valued state into the real domain

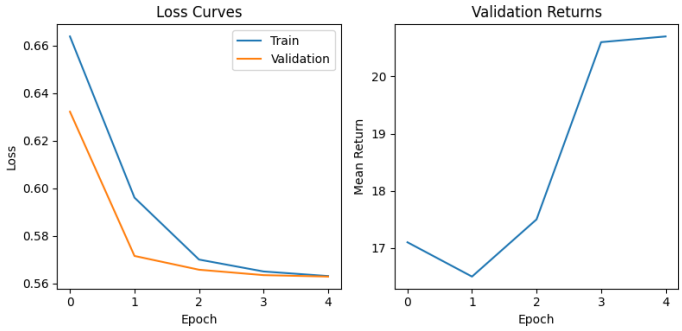

Figure 10: FFM Memory

Although the loss decreases, validation returns remain low, hinting that extended training might be required.

2.2 Stable Hadamard Memory (SHM)

The Stable Hadamard Memory (SHM) employs a memory matrix updated through calibration and Hadamard product operations for stability and structured retention. Its steps include:

- Computing calibration and update matrices
- Updating the memory state with the Hadamard product
- Performing a read operation using a query and the memory state

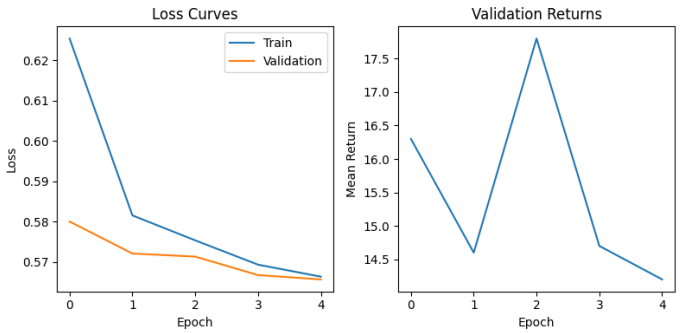

Figure 11: SHM Memory

Training results mirror those of FFM, showing little evidence of successful learning.

2.3 Associative Recurrent Memory Transformer (ARMT)

The Associative Recurrent Memory Transformer (ARMT) processes state embeddings in segments to model long-term dependencies via an associative memory mechanism. Its memory update involves:

- Generating key-value pairs
- Updating memory with DPFP-3 from associations
- Retrieving associations based on queries

ARMT achieves the lowest loss among the custom mechanisms and a validation return of 500, demonstrating its superior ability to learn the task efficiently.

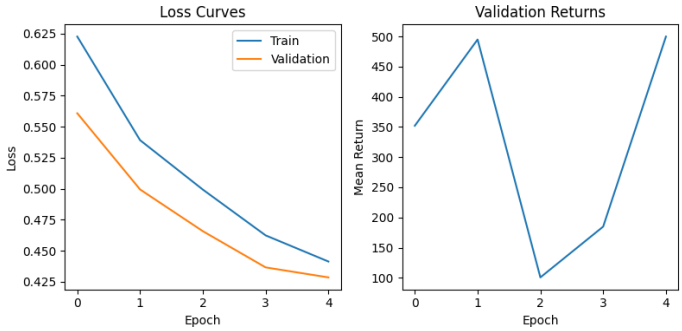

Figure 12: ARMT Memory

3 Key Takeaways

- For simpler tasks, a larger context length requires a correspondingly larger sample size to prevent overfitting.
- The target return-to-go (RTG) effectively regulates episode rewards.
- Among the custom memory mechanisms, ARMT stands out as the only one trainable within the given constraints.