State Space Models for Improved Offline RL

Tara Rezaei
tarark@mit.edu

Oam Patel
oam@mit.edu

Abstract

There have been several recent claimed advances in sequence modeling architectures [3, 2], but so far these have only been applied in a supervised or semi-supervised training pipeline. We aim to benchmark these architectures on several offline RL environments in a similar vein to the original Decision Transformers line of work [1]. We compare and investigate these architectural advances.

1 Introduction

There have been recent advances in the area of State Space Models[3] for sequence modeling. We're interested in investigating how to formulate certain RL problems as sequence modeling problems to take advantage of these advances. For example, there is the Decision Transformer to generate sequences of actions associated with reward[1]. There has been previous work on leveraging information from future states and rewards for improving policy updates [6] as well as work on leveraging them for better credit assignment in sparse reward regimes [5]. We hypothesize that recent SSM models that don't use full sequence attention may have better performance in some regimes. We look investigate this claim in the offline RL regime and compare the two methods in terms of performance, inference time speed, memory usage and training loss.

2 Methods

2.1 Set up

This project focuses on three distinct environments from the OpenAI Gym: Walker, Half- Cheetah, and Hopper. For the simulator, we use the OpenAI Gym environment for all three as it provides a clean interface that we have already utilized through the class homework. The success criterion in each game is determined by the 'done' condition outputted by the simluator. We will track this over training and across our different methods along with the average reward obtained across trajectories.

2.1.1 State and Action Spaces

For all three of the environments we consider, the state space includes positions, velocities, and joint angles, encapsulating the physical status of the agent. The action space consists of continuous control signals to the agents' joints. Full details taken from the official gym documentation are included in Table 3.

	Hopper	Half-Cheetah	Walker
Action	Box(-1, 1, (3,), f32)	Box(-1, 1, (6,), f32)	Box(-1, 1, (6,), f32)
Observation / State	(11,)	(17,)	(17,)

Table 1: Description of Action and Observation Spaces

6.484 Computational Sensorimotor Learning Final Project (Spring 2022).

2.1.2 Reward

The Walker:

$$\begin{aligned}\text{healthy\_reward} &= \text{fixed\_healthy\_reward}\\ \text{forward\_reward} &= \text{forward\_reward\_weight} \times \frac{(x_t - x_{t-1})}{dt}\\ \text{ctrl\_cost} &= \text{ctrl\_cost\_weight} \times \sum a^2\\ \text{reward} &= \text{healthy\_reward} + \text{forward\_reward} - \text{ctrl\_cost}\end{aligned}$$

where:

- x_t is the x-coordinate after taking action a at time t.
- x_{t-1} is the x-coordinate before taking action a at time t.
- dt is the time between actions, which is dependent on the frame skip parameter ($dt = 0.008$).

The Half-Cheetah:

$$\begin{aligned}\text{forward\_reward} &= \text{forward\_reward\_weight} \times \frac{(x_t - x_{t-1})}{dt}\\ \text{ctrl\_cost} &= \text{ctrl\_cost\_weight} \times \sum a^2\\ \text{reward} &= \text{forward\_reward} - \text{ctrl\_cost}\end{aligned}$$

where:

- x_t is the x-coordinate after taking action a at time t.
- x_{t-1} is the x-coordinate before taking action a at time t.
- dt is the time between actions, which is dependent on the frame skip parameter ($dt = 0.05$).

The Hopper:

$$\begin{aligned}\text{healthy\_reward} &= \text{fixed\_healthy\_reward}\\ \text{forward\_reward} &= \text{forward\_reward\_weight} \times \frac{(x_t - x_{t-1})}{dt}\\ \text{ctrl\_cost} &= \text{ctrl\_cost\_weight} \times \sum a^2\\ \text{reward} &= \text{healthy\_reward} + \text{forward\_reward} - \text{ctrl\_cost}\end{aligned}$$

where:

- x_t is the x-coordinate after taking action a at time t.
- x_{t-1} is the x-coordinate before taking action a at time t.
- dt is the time between actions, which is dependent on the frame skip parameter ($dt = 0.008$).

2.2 Experiments

First, we replicate results from the original decision transformers paper. This gives us a platform on which to base our experiments and a baseline to compare to that would also take into account, specific implementation details such as hardware set up, parameters, etc.

In the next step, we test our hypothesis and use a state space sequence model. We evaluate this methods on the three proposed environments against the decision transformer and we compare the performance gain we get from them as well as loss, inference time and memory management.

3 Results

3.1 Mean reward

Mean reward comparison for a mamba model with 900k trainable parameters vs a transformer model with 1.2M trainable parameters. The numbers are averaged over 500 runs.

	Hopper	Half-Cheetah	Walker
DT	4.5 ± 1.1	3.1 ± 1.0	4.9 ± 1.1
SSM	4.7 ± 1.2	2.9 ± 1.2	4.3 ± 1.3

Table 2: Mean reward of 500 episodes for both models

From the results, we observe that we get similar performance with both models when we condition on high reward. The slight variations of both models fall within one standard deviation of the mean and there exists no consistent pattern of one performing better than the other.

3.2 Training loss

In figure 1 and 2, we see the comparison of the training loss vs training steps of the base sequence model for the hopper environment. We see that both models converge with a similar rate in both tasks and plateau around the same number. We take this to mean that both achieve similar performance in sequence prediction.

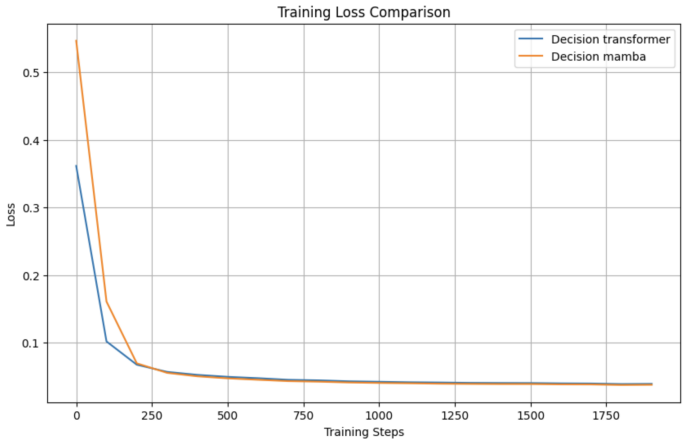

Figure 1: Traning loss vs training steps for the walker environment

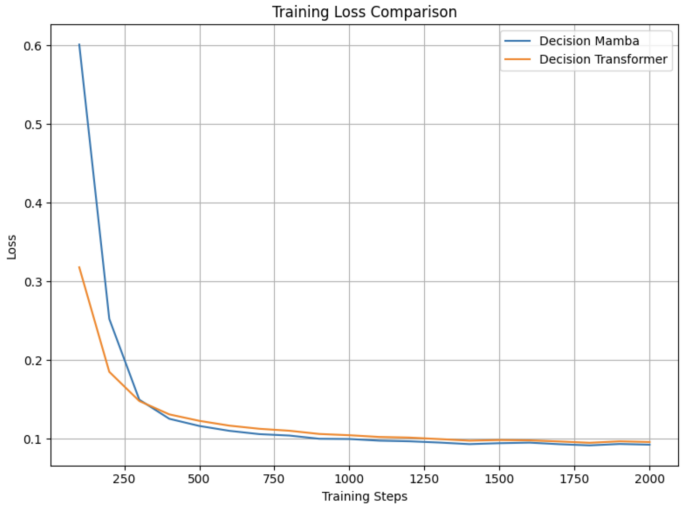

Figure 2: Traning loss vs training steps for the hopper environment

3.3 Reward

In figrue 3, we see the average reward received by the agent in both set ups, the steps correspond to the agent taking actions for the walker environment. [h]

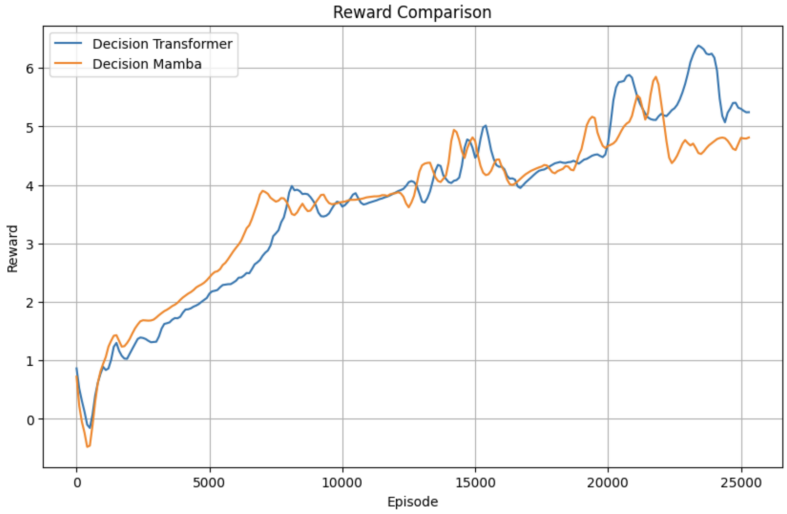

Figure 3: Traning loss vs training steps for the hopper environment

3.4 Runtime and Memory

We compare inference runtime and memory usage of both models. The numbers are found in the following. It is worth noting that our experiments were limited and not meant to leverage the full efficiency capacity of the SSM.

	Runtime	Memory
DT	27s	3.1GB
Mamba	25s	2.9GB

Table 3: Mean reward of 500 episodes for both models

4 Conclusion

We compared and analyzed the use of State Space sequence models as a substitute for transformer models in the offline RL regime. We found that they perform similarly and there is no obvious gain in terms of performance. We acknowledge that our experiments alone, are not enough to draw this conclusion and that there are memory advantages to using State Space Models for sequence prediction that is out of scope of our project but an exciting venue for future work.

5 Contributions

Oam did initial experiments replicating the decision transformer results and getting Mamba running with the Huggingface trainer. Tara did most experiments involving Mamba on the environments, runtime and memory comparison. We both contributed to framing of the project, method, and writing.

We used Huggingface to handle some training infrastructure. Huggingface also helpfully includes implementations of transformers, decision transformers, and the mamba architecture. We wrote a decision transformer wrapper class around the mamba architecture and then ran our experiments. We had to do lots of hyperparameter checks to get it working.

Our github repository CSLproject

References

[1] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021.

[2] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

[3] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022.