Mamba RL

April 2024

1 Introduction

The following work aims to examine the potential of using Mamba architecture for models aiming at solving RL tasks.

Decision Transformer (DT) [1] introduced a new approach for modeling MDP sequences in a way that allows using transformers architecture for solving RL tasks. Their model received state-of-the-art results on various common RL benchmarks, including the Atari video games which were a point of interest for many RL research works over the years [2].

The revolutionary work of DT opened a window for follow-up works taking inspiration from the formalization of RL tasks as long-context sequences. One such work is the Decision-S4 [3] which suggested replacing the transformer-based architecture with a state-space model, specifically the S4. The work proved potential for state-space models in RL tasks by demonstrating good results on tasks from openai-gym, especially when combining online learning with offline learning. However, it did not present much success on the popular Atari tasks, leaving space for more improvements in the integration of state-space models for RL.

Lately, a major improvement in state-space models was introduced when the new Mamba model [4] was published. The Mamba model allows state-space models to have data-dependent transition matrices, which arms the model with a much richer expressiveness compared to previous state-space models. [5] have even shown that it is possible to extract attention matrices from Mamba models, that capture more complex data relations than the regular well-known transformer.

In this work we want to test whether the increased power that Mamba brought to state-space models can lead to improved performance also in RL tasks. To do so, we test the model on Atari tasks.

The next chapter will present the setting of our experiments, followed by their results in Chapter 3.

2 Experiments setup

In our experiments we follow almost exactly the setup used in Decision Transformer [1], in order to get a fair comparison between a model using Mamba blocks and a model that uses Transformer blocks. We will now describe the experiments setup and highlight changes we have introduced compared to the work of DT.

2.1 Training Data

For training, we use the Atari DQN dataset, presented by [2].
This dataset is actually a combination of multiple datasets for different Atari games, each holds the trajectories received from a training process of a DQN agent on the relevant game. The trajectories were recorded with a frame skip of 4, and sticky actions (with 0.25 probability, the agent's previous action is executed instead of the current action), according to the guidelines presented in [6] which became a common standard for models trained and evaluated on Atari tasks. Each dataset holds 200 million frames, comprising 50 million transitions due to the frame skip.

Although [2] mentions it has been found that at least 5 million transitions are required at training time to receive good results with the models tested by it, in the DT paper only 500 thousand transitions were used for training.
In the first stage, we train also using 500 thousand transitions, in order to keep in line with the DT setup, but we also examine the effect of increasing the training set size, as will be shown in the results section.

The data is composed of sequences of return-to-go(A concept presented in the DT paper) $\times$ state $\times$ action tuples. At each batch we sample sequences of length K, to get a total of $3K$ tokens per sample in each batch. This limits the context length presented to the model. By default, we follow the DT best result and set $K = 30$, though we test the effect of various K values on performance, as presented in the results section.

2.2 Evaluation

For evaluation, we test our model on an online environment, using the Arcade Learning Environment (ALE) [7], mediated by Gymnasium [8]. It is worth noting that DT implemented the same evaluation process, but they used the atari-py package which is deprecated. ALE is the new officially supported package that replaced atari-py and has official support through Gymnasium.

After each epoch, we randomly generate 10 new game initialization (using Gymnasium) and let our model play each of the games. The score we report is the cumulative reward the model got along each game, averaged across the 10 game-iterations.

To this end, except of using Gymasium instead of atari-py we follow the same evaluation method used by Decision Transformer.

2.3 Model Architecture

We follow the DT architecture, which is an implementation of a mini GPT model, but we replace the self-attention blocks with Mamba blocks, exactly as those are presented in the Mamba paper [4]. We will shortly describe the architecture.

As mentioned above, our inputs are sequences of length $3K$ - each is composed of K transitions, described by tuples of <Return-to-go, State, Action>. The inputs are being embedded as follows:

- First, we calculate a special embedding for timestamp (which is going from 1 to K, i.e there is a unique timestamp for each transition). Here we used linear embedding.
- For both actions and Return-to-go inputs we use linear embedding, summed by the time-stamp encoding.
- For State inputs we use a small CNN as encoder, summed by the time-stamp encoding.

Than the inputs are flattened to create a $3K$ embedded sequence of the form

$$R_0, s_0, a_0, R_1, s_1, a_1, ..., R_k, s_k,$$

Where R stands for Return-to-go, s stands for state, and a stands for Action.

Next, the sequence goes through a dropout layer, followed by a series of Mamba Blocks, which are implemented exactly as in the Mamba paper.

Finally, we are interested in the model's predicted logits of the next action, i.e our model predicts the next action given the history sequence.

During training we optimize the model to predict the next action using Cross-Entropy loss.

3 Experiments

In this section we present experiments we have conducted and explain their results. The results are presented in detail in the following subsections, however for ease of readability we highlight here the main results:

- With the default hyper-parameters, DT seems to have superior performance compared to Mamba-based model (presented in section 3.1).
- The Mamba model seems to over-fit, and indeed changing hyper-parameters to reduce over-fitting improves the Mamba performance (presented in section 3.2).
- Same changes in hyper-parameters also improve performance of DT, and while the gap between the models reduces, DT keeps superiority over Mamba in our experiments , though it has higher training time with the same hyper-parameters (presented in section 3.3).

3.1 Mamba VS Transformer - Basic Comparison

As a first stage, we compare the Decision Trasformer architecture to a model with the same architecture and the same hyper-parameters, when we only change the self-attention blocks to Mamba blocks. For this we follow the hyperparameters suggested by the DT paper.

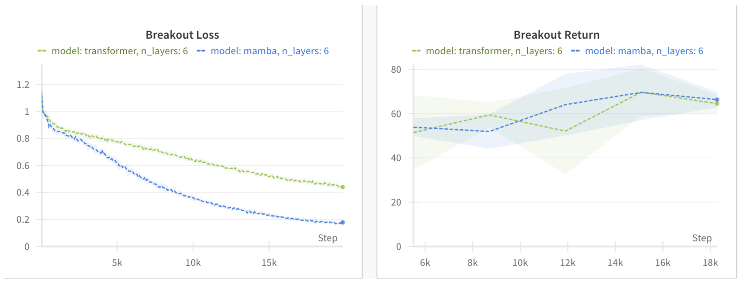
To get a valid baseline, we have evaluated each model using the implementation provided by the DT paper, and present these results in Appendix A. Nevertheless, few aspects of the DT paper implementation raised concerns for us, and led us to retest both models with a modified implementation.

First concern is that the DT paper implementation uses the deprecated atari-py package. This package is no longer supported by the authors of [7], and may raise erroneous results. Instead, the authors released a new official package, ale-py, which is also officially supported via the Gymnasium package [8] for RL tasks. Therefore, we build our implementation using Gymnasium.

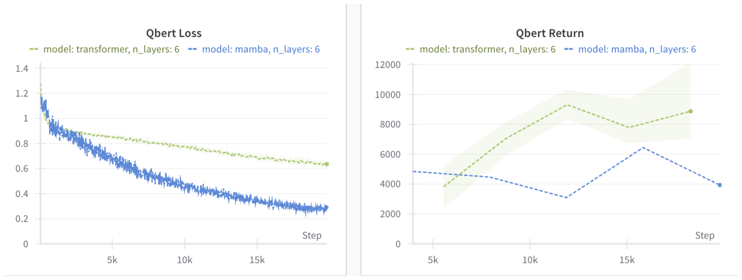
Another concern is that in the Decision Transformer paper the reported results are those of the best epoch. It is also worth mentioning that in their implementation, at each epoch they radomally generate totally new game initializations, such that the game initializations that the model is evaluated on at each epoch are different. This, together with the fact that they report best-epoch results, raises the concern of a "P-hacking" situation, where the reported results are chosen in a manner that does not genuinely reflect the model's average performance.
To overcome this, we have fixed the implementation so that at each epoch same game-initializations are evaluated. Also, to reflect more precisely the performance of both models, we report the results of the models at each epoch, and focus the comparison on the last epoch for both models.

Lastly, we note that following the recommendations of [6] for evaluation of models in the atari environment, it is a common approach in papers to use "sticky actions" (i.e previous action is taken with probability 0.25 instead of new predicted action). DT paper did not implement this, so we add sticky actions in our implementation.

Figure 1 summarizes the comparison between DT and a Mamba-based model on Breakout and Qbert games, following our suggested implementation.
The results show that DT gets better evaluation performance compared to the Mamba-based model on Qbert, while on Breakout the models have equal results. Nevertheless, in terms of training loss the Mamba model archives significantly better results, which leads to the hypothesis that with the default hyperparameters the Mamba model overfits. With that in mind, in the next section we conduct some parameter exploration to test wether it is possible to improve the Mamba evaluation performance by adjusting the hyper-parameters.



(a) Breakout



(b) Qbert

Figure 1: Comparison between DT and Mamba-based model on Breakout(a) and Qbert(b). For each model the training loss is presented along the training process, together with evaluation score along epochs.

More results of the basic comparison, specifically comparison using the DT implementation, and comparison without ”sticky actions”, can be found at Appendix A.

3.2 Exploring hyperparameters for Mamba model

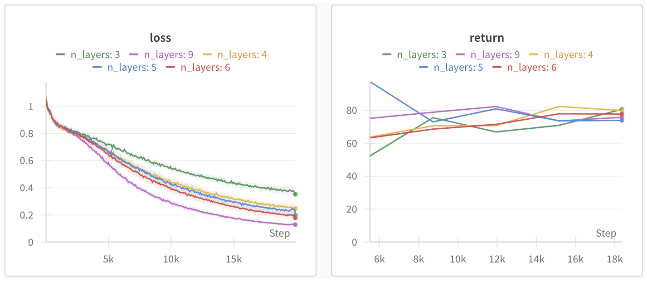
Following the results of the previous experiments, we would like to explore hyperparameters for the Mamba model. Keeping in mind the hypothesis that the default mamba model overfits, we would like to ”weaken” the Mamba model, in order to enforce some sort of regularization on it. In particular, we explore the following parameters:

- Number of layers
- Context length
- Dropout

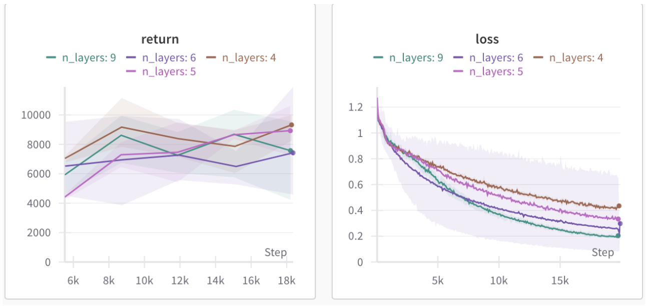
3.2.1 Numer of layers

The DT model uses 6 layers of self-attention . While it is common to replace each self-attention layer with two layers of Mamba (in order to keep same amount of parameters), we test also the effect of using less Mamba layers, in order to limit the expressiveness of the model, and hopefully avoid over-fitting.

Results from the Breakout and Qbert games are shown in Figure 2 and consistently lead to a conclusion that lower number of Mamba layerfs improve the evaluation performance of the model, reducing over-fitting.



(a) Breakout



(b) Qbert

Figure 2: Exploration of the effect of number of layers on the Mamba model. The number of layers at the default model was 6. (a) presents results for Breakout and (b) for Qbert.

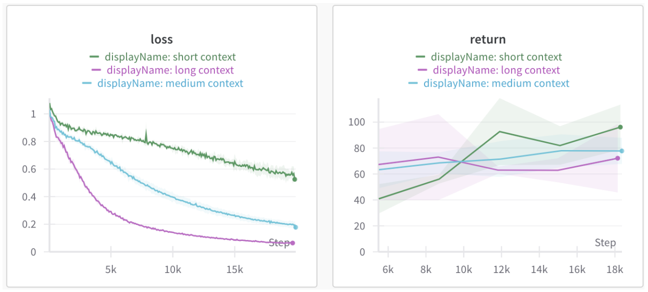
3.2.2 Context Length

Another parameter to consider when trying to ”weaken” the model is context length. We test three configurations of context length:

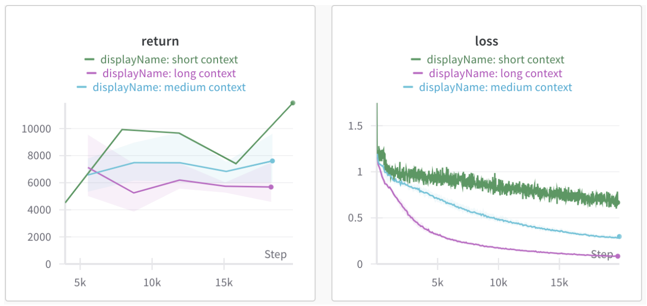
- Short context - 10 timesteps per context window.
- Medium context - 30 timesteps per context window, as suggested at the DT paper.
- Long context - 100 timesteps per context window.

Results for Breakout and Qbert games are shown in Figure 3, and consistently lead to a conclusion that lower context length improves evaluation performance and reduces overfitting.

It is worth noting that in RL tasks that run on MDPs (Markov Decision Process) a context window of size 1 should be, at least theoretically, enough to determine the optimal action. Nevertheless, in the DT paper they have shown that their model gains improvement from longer context size. One explanation may be that it helps the model approach the problem of ”future credit”, where an action leads to a reward only at the future. However, from our results it may be hypothesized that the Mamba model is better able to represent states of the MDP, which allows him to get good results from a shorter context, as expected in a setting of MDP.



(a) Breakout



(b) Qbert

Figure 3: Exploration of the effect of context length on the Mamba model. The context length at the default model was 6. (a) presented results for Breakout and (b) for Qbert.

3.2.3 Dropout

It is worth mentioning that in the DT there are two kinds of dropout - one in the self-attention blocks, and one in the global architecture. However, in Mamba blocks there is no dropout, therefore only the global dropout is relevant at this section.

In the DT paper a dropout of 0.1 is used. We test it against a dropout value of 0.2 on the Breakout game.

As can be seen in Figure 4, a larger dropout slightly improves evaluation performance and reduces over-fitting.

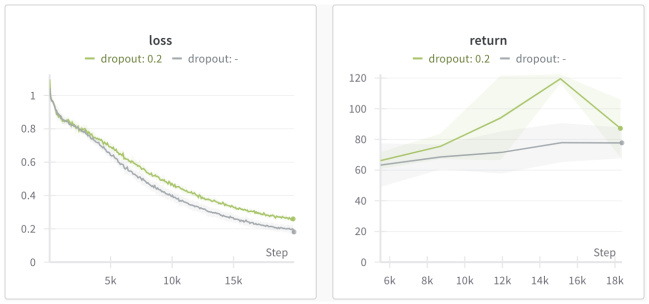


Figure 4: Exploration of the effect of dropout on the Mamba model in the Breakout game. The dropout at the default model was 0.1

3.2.4 Combination of parameters

Following the results of previous sections, we test the performance of the Mamba model when we change multiple parameters to reduce over-fitting of the model. One important point is that the changes we introduce to the model's parameters lead to a smaller model, with a faster training time. This allows us to increase the training data while paying by just a slight increase at training time.

We introduce the results in Figure 5. The comparison is between the following configurations:

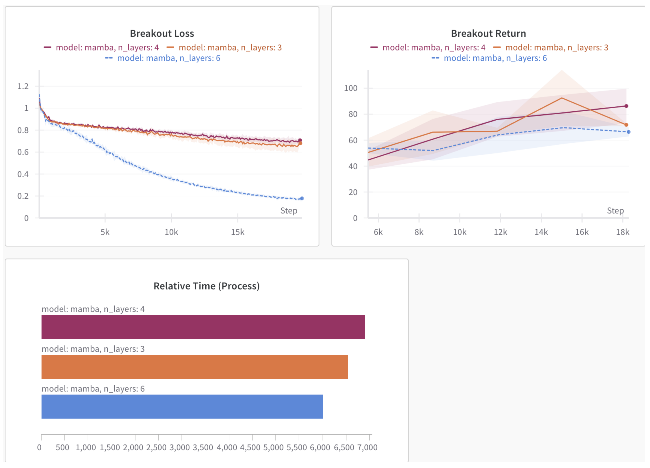
- 3 layers, 16 context length, 0.1 dropout, 10^6 data points, batch-size 256.
- 4 layers, 16 context length, 0.2 dropout, 10^6 data points, batch-size 256.
- (Default configuration) 6 layers, 30 context length, 0.1 dropout, $500,000$ data points, batch-size 128.

.

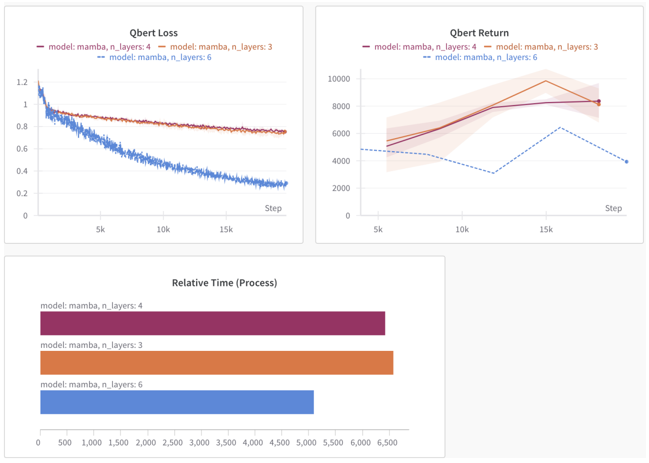
It is evident that the configurations that aim at reducing overfit indeed achieve better evaluation scores.

3.3 Revisiting comparison to DT

As the new configurations show potential for improvement for the Mamba model, it is a required step to test these configurations on DT as well. Figures 6 - 7 show the results of such tests, on both configurations presented in section 3.2.4. Although in the DT paper they present their configuration as optimal for DT, it seems that the configurations tested in this section improve the performance of DT. A possible explanation for that may be the fact that we re-implemented the evaluation process with Gymasium and with sticky-actions.



(a) Breakout



(b) Qbert

Figure 5: Comparison of different parameter configurations. These include:

- 3 layers, 16 context length, 0.1 dropout, 10^6 data points, batch-size 256.
- 4 layers, 16 context length, 0.2 dropout, 10^6 data points, batch-size 256.
- 6 layers, 30 context length, 0.1 dropout, $500,000$ data points, batch-size 128.

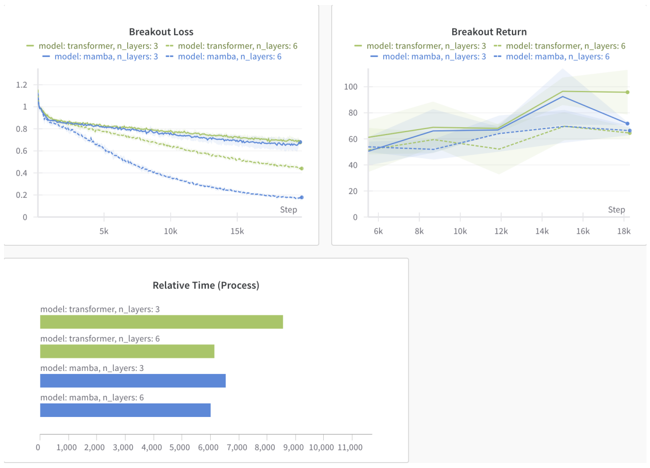
(a) presents results on Breakout game, while(b) is on Qbert.

It seems that the new configurations improve both models, and while the gap between the models evaluation scores seems to shrink, the DT model continues to present a moderate superiority on the Mamba model. It is yet worth mentioning that with the same hyper-parameters, the DT training time is higher compared to Mamba.

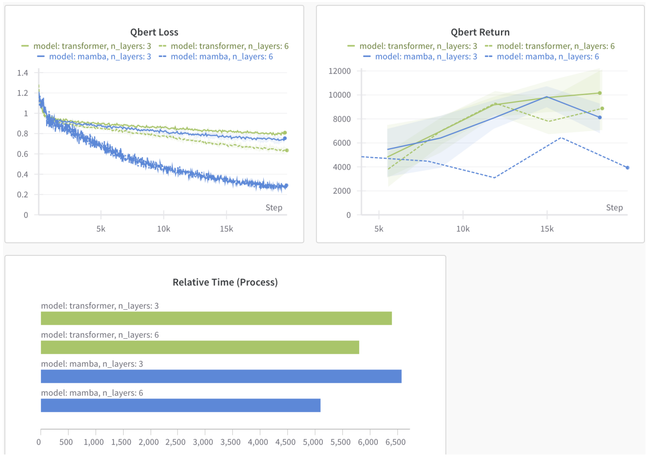
4 Conclusion

The results of our experiments show potential to use state-space models (and especially Mamba) for RL tasks, that do not fall significantly of the potential of attention-based model. Nevertheless, DT still presents superior results compared to Mamba in the Atari games, suggesting that more work has to be done to adjust Mamba for MDP sequences.

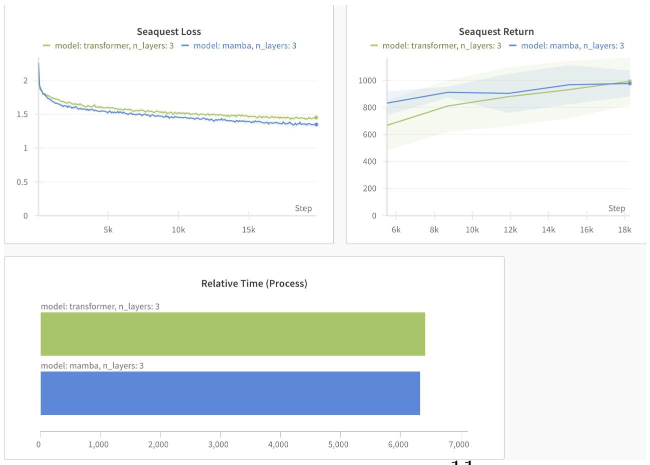
Besides further adjusting the hyper-parameters of the model, it may be interesting to suggest new architectures that leverage the characteristic of Mamba as a state-space model to better represent states of MDPs. These may include bi-directional learning for the Mamba model, to allow for future planning, and combining offline learning with online learning, as done in the Decision-S4 [3].



(a) Breakout

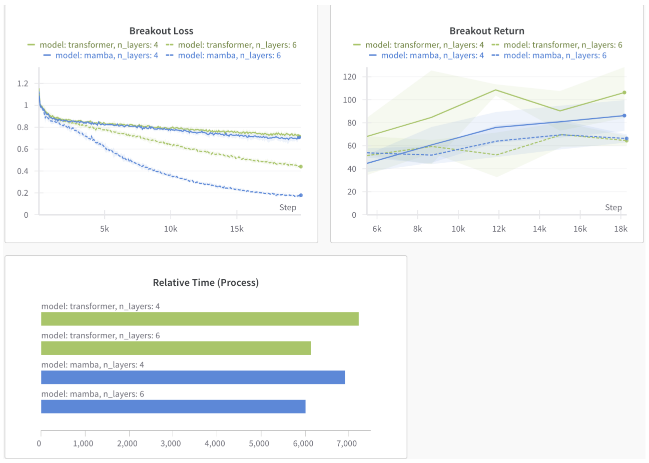


(b) Qbert

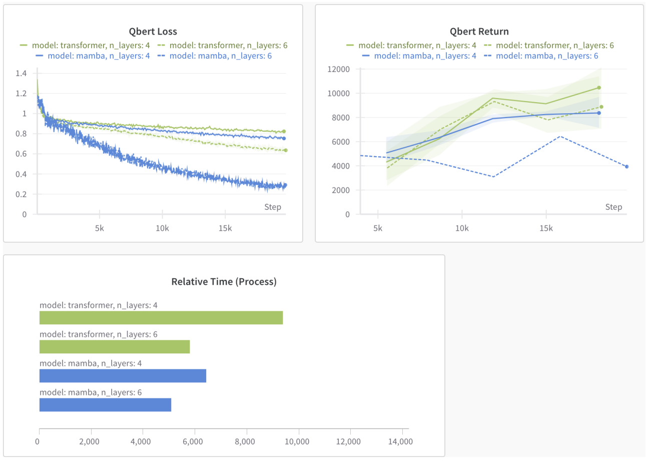


(c) Seaquest

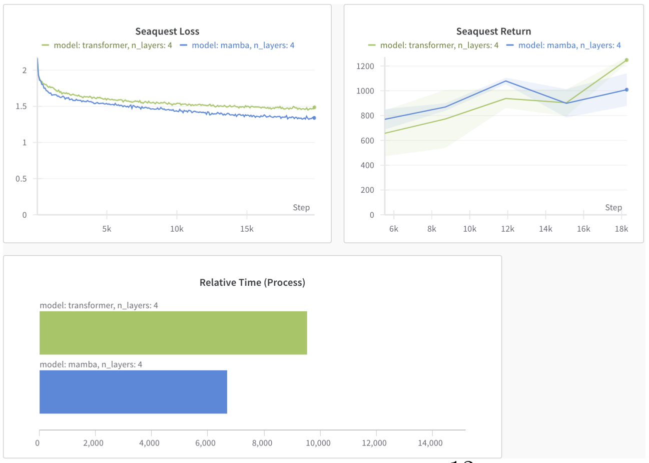
Figure 6: Comparing DT vs Mamba with a configuration of 3 layers - 16 context length - 0.1 dropout - 10^6 data points - batch size 256. (a) presented results for Breakout, (b) for Qbert, (c) for Seaquest.



(a) Breakout



(b) Qbert



(c) Seaquest

Figure 7: Comparing DT vs Mamba with a configuration of 4 layers - 16 context length - 0.2 dropout - 10^6 data points - batch size 256. (a) presented results for Breakout, (b) for Qbert, (c) for Seaquest.

References

[1] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097. Curran Associates, Inc., 2021.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 104–114. PMLR, 13–18 Jul 2020.

[3] Shmuel Bar David, Itamar Zimerman, Eliya Nachmani, and Lior Wolf. Decision s4: Efficient sequence-based RL via state spaces layers. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[5] Ameen Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of mamba models, 2024.

[6] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[7] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

[8] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

A Additional comparisons between DT and Mamba with default parameters

A.1 Comparison using the DT paper implementation

Table 1 presents the results when the evaluation is based on the DT paper's implementation, which uses the deprecated atari-py package. In the Decision Transformer paper the reported results are those of the best epoch of each evaluation iteration. It is also worth mentioning that in their implementation, at each epoch they radomally generate totally new game initialization, such that the game initializations that the model is evaluated on at each epoch are different. This, together with the fact that they report best epoch results, raises the concern of a "P-hacking" situation, where the reported results are chosen in a manner that does not genuinely reflect the model's average performance. Therefore, we add to our report also the results of the last epoch. In addition, to reflect more precisely the performance of both models, we add in figure 8 the results of the models at each epoch.

		Best Epoch		Last Epoch	
	DT reported	DT	MAMBA	DT	MAMBA
Breakout	267.5 ± 97.5	243.7 ± 52.1	$\mathbf{284.8 \pm 63}$	$\mathbf{184.4 \pm 26.7}$	178.6 ± 27.3
Qbert	15.1 ± 11.4	23.6 ± 10.6	$\mathbf{32.8 \pm 19.5}$	6.12 ± 3.6	$\mathbf{17.9 \pm 16.4}$
Seaquest	2.5 ± 0.4	$\mathbf{2.4 \pm 0.2}$	1.8 ± 0.3	$\mathbf{2.3 \pm 0.3}$	1.7 ± 0.2

Table 1: Basic comparison. The results in this table are received using the Decision Transformer evaluation implementation, which is based on the deprecated package atari-py. The results are normalized so that score 0 equals the expected score of a randomized model, while score 100 equals the expected score of a professional human player.

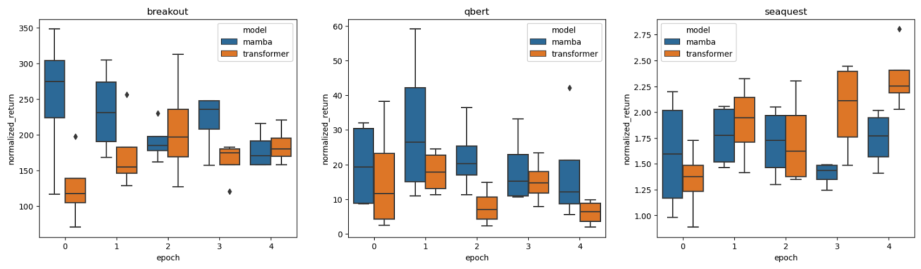


Figure 8: Scores at each epoch for three different games. At each epoch, each model was evaluated using 5 different seeds, so the boxplots present the distribution of the scores over these 5 seeds.

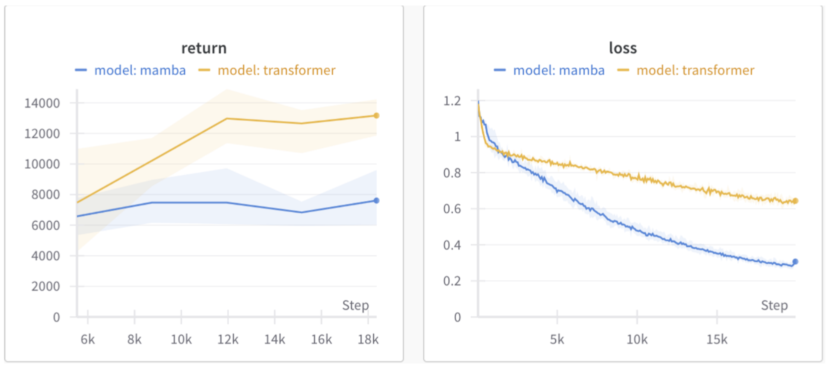
A.2 Comparison using Gymnasium, without ”sticky actions”

Two main issues raise some difficulties with the results above.
First, the fact that at each epoch a totally new set of game initializations are generated makes it hard to test the progress of the training process.
Second, the use of the deprecated atari-py package is less favorable compared to more updated packages such as Gymnasium.

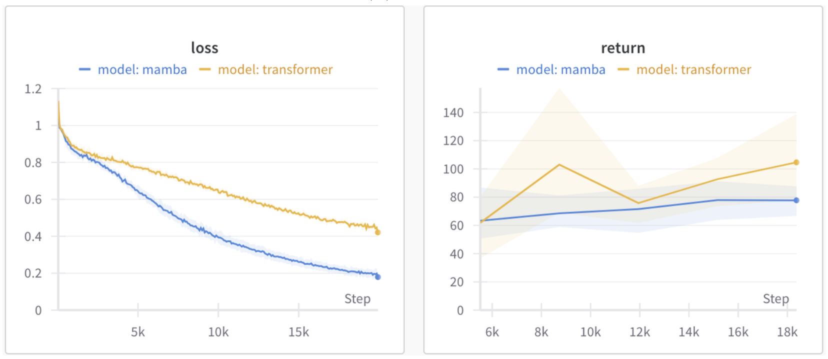
To overcome these issues we have re-implemented the evaluation process using Gymasium, making sure that at each epoch, as long as the same seeds are used, the same game initializations are received.

Figure 9 shows the results with Gymnasium. It shows both the training loss along the training steps and the evaluation scores along the epochs. The intervals presented in the figure present the distribution of the different seeds used during the evaluation. It is important to clarify that unlike the results presented in section 3.1 at the body of this document, here the evaluation does not include sticky-actions.

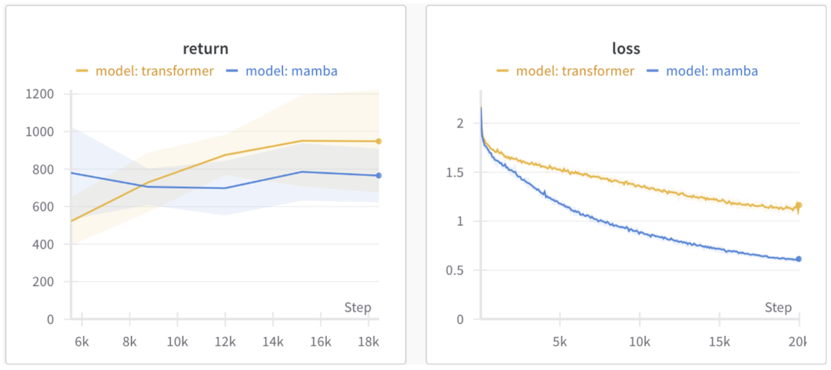
One result worth mentioning is that the results of both models, including the Decision Transformer, are better when evaluated using Gymnasium. Also, in Gymnassium it becomes notable that the Decision Transformer receives better evaluation results compared to the Mamba-based model that is trained with the same hyperparameters.



(a) Qbert



(b) Breakout



(c) Seaquest

Figure 9: Training losses and evaluation returns based on Gymnassium. In this figure the return values are not normalized.