WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Mamba Player: State Space Model in dynamic Video Games

Michał Dąbrowski, Wiktor Jędrasik, Kacper Mamiński

Bialystok University of Technology Białystok, Podlaskie, Poland

A B S T R A C T

This paper presents a structure of an AI model designed to interact with a variety of video games based on visual input. Our model builds upon the recent rise in popularity of State Space Models (SSMs) which have shown major improvements in interference speed and memory efficiency. In contrast to the previous researches, we were able to utilize a novel foundation model in conjunction with a higher refresh rate. Versatile structure of the system, relatively low hardware requirements and straight-forward dataset solutions show great potential to be expanded in the future.

Keywords: Mamba, state space models, video games

1. INTRODUCTION

Video games have been a subject of studies for AI researchers from their inception. Models are commonly trained on titles that share certain traits: the ability to interfere with memory and low system requirements, which facilitate the application of reinforcement learning. However, this limits us with the number of games we can choose from; the primary focus still persists on releases from before the 2000s. However, thanks to the recent progress made by foundation models, we can take an alternate route.

Models like Transformers, even though impressive as LLMs, with their language accuracy and coherence, require quadratic time complexity to calculate and substantial hardware to process and learn from data. These can be identified as the primary reasons why the revolution they initiated was less disruptive in video games AI compared to other fields.

Thankfully, researchers have identified these issues and, since then, searched for solutions that combine low computational usage with the attention mechanism that was their key feature. Growing popularity of State Space Models with Mamba as their representative have shown that there are numerous ways to achieve global understanding of the data, without the use of Transformers. Their basis can be found in both: classical state space models and combination of numerous already well known solutions like recurrent neural networks (RNNs) and convolutional neural networks (CNNs). As a result of putting strong emphasis on a high level of parallelism in calculations, hardware-aware algorithms and usage selection mechanism, it was to possible achieve linear scaling in sequence length during training, while maintaining strong performance.

2. Background

2.1 Previous researches

Most of the previous works, as stated before, use hand-crafted environments specifically tailored to their needs. Training AI agents by usage of behavioral cloning shifts focus to human-like interactions through natural for users input. Reasons for choosing `Counter-Strike 2' (CS2) as a staple for further research include:

- Possibility to verify progress made by new model by contrasting it with prior solutions.
- Presenting fast reactions in changing environment.
- Straight-forward responses to feedback provided by the game.
- Opportunity to standardize the results.
- Popularity and number of tools provided by the developers/fans.
- Easy way to build the dataset.

2.2 Environment

CS2 is a First Person Shooter which basic mechanics that are interchangeable with other titles. Model was trained and validated on 'Aim training' mode, as other variants require far larger datasets and training, long term planning, communication and access to audio input as well. This game mode provides player with an ability to improve their aiming skills through practice with static targets that change position upon being hit. Reloading is not required and player cannot be shoot by bots.

3. Dataset

Dataset consists of around 110 000 frames from 8 session of playing that adds up to 48 minutes of gameplay. Validation dataset contains 2465 frames from one recording. Every session is stored in a separate folder that contains spreadsheet that has file path to the frame, buttons pressed at the moment and the movement of mouse relative to the position from the previous image. This approach turned out to be helpful while coding the recorder, due to possibility of verifying the accuracy of the mouse movement. Every frame is saved in 1280x720 resolution as a trade-off between file size and further options to resize the image without losing much visual features. Every row represents one frame. There were also implemented 3 methods to select frames for training:

1. Raw-every frame that was recorded is processed by the model.
2. During the change in inputs - frame is processed if there is change in inputs in comparison with the previous frame. It can be expressed as:

$$\text{is frame valid}_i = \begin{cases} \text{True} & \text{if } |x_i| > 10 \text{ and } |y_i| > 10 \\ \text{False} & \text{otherwise} \end{cases}$$

3. During the movement with mouse - frame is processed if there was significant mouse movement. It can be expressed as:

$$\text{is frame valid}_i = \begin{cases} \text{True} & \text{if inputs}_i \neq \text{inputs}_{i-1} \\ \text{False} & \text{otherwise} \end{cases}$$

Filename	X	Y	LeftMouse	RightMouse	Buttons
/frames/1.jpg	0	4	False	True	['w','a']
/frames/2.jpg	23	41	False	False	['w','a']
/frames/3.jpg	-21	5	True	False	['w','key.ctrl_l']

Figure 1: An example of dataset rows.

4. Description of architecture

4.1 Choice of architecture

Task of playing dynamic video games require lightweight or simple model due to the time limitations of calculating one output, which for us meant time limit of 16 milliseconds per action. If we aim to achieve 60 FPS (Frames Per Second) on a consumer-level hardware our options are rather limited. Until recently most of the methods that fulfill these criteria originate from Convolutional Neural Networks (CNNs), which according to studies, number of use cases in which they are still relevant has shrunk in recent years. Up to now CNNs, thanks to their low compute time, have been the preferred option for efficient image classification, a field from which imitation learning derives from. Due to popularity of Transformers in recent years, performance recede into the background as striking accuracy becomes the primary point of focus. Simultaneously with their development State Space Models, such as Mamba, have started to shown impressive results with a magnitude lower requirements, thus we have choose them as the basis for the further studies**.**

4.2 Core components of the model

Given an image of size $\boldsymbol{BCHW}$, where $\boldsymbol{B}$ stands for batch size, $\boldsymbol{C}$ number of channels, $\boldsymbol{H}$ height, and $\boldsymbol{W}$ width, we rearrange it according to the formula:

$$BC(HP_1)(HP_2) \rightarrow B(HW)(CP_1P_2)$$

where $\boldsymbol{P_1}$ and $\boldsymbol{P_1}$ are patch sizes. After that, we apply layer normalization over the last dimension, pass it through a linear layer, and perform normalization again. Then the tensor enters stacked layers of Mamba, after which it is projected to dimensions representing states of buttons. Mamba layers are wrapper layers, based on Mamba and Mamba2 class from official repository. Binary conversion can be represented as:

$$out_i = \begin{cases} 1 & \text{if } x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

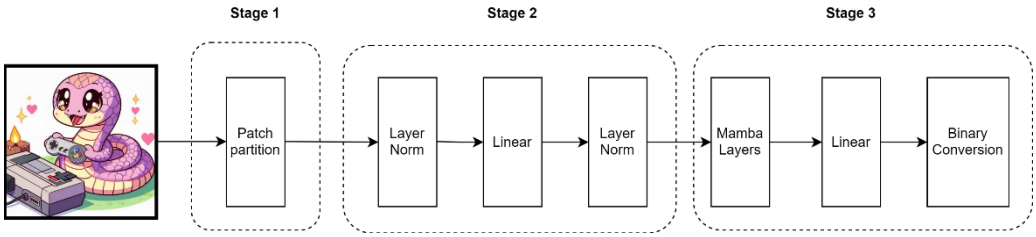

Figure 2: Model overview.

4.3 Hyperparameters

Hyperparameters of the model

To achieve reasonable performance on consument grade hardware, we have started with the default hyperparameters LLM of Mamba 130M, which was further modified by reducing the number of layers from 24 to 8. This reduced the number of parameters from 130M to 30M.

Hyperparameters of the image pre-processor

The starting image has 1280x720 resolution and 3 color channels. With accordance to the previous study, before passing the frame to the model, we are cropping the center area of the image resulting in 640x360 resolution. It is next resized to 360x360 to simplify the hyperparameters of the patch embedding layer.

Figure 3: Example of an image input before converting it to a tensor.

Hyperparameters of the mouse movement and key presses

Mouse movement is discretized into two lists of values, separate for x and y. These values are alike to these used in the study. Having to rely on multi-class classification, instead of multi-label one gave us the possibility to use CrossEntropyLoss for mouse movement, which have shown great results with Mamba. For the rest of buttons we are using Binary Cross Entropy With Logits Loss to simplify loss calculations for other buttons as the accuracy for them is not as important as for the mouse.

5. Training

Training was performed on a single RTX 3070 Ti for around 20 hours, during which 235559 steps were made. Due to low memory size of the GPU, batch size was set to 80. Automatic Mixed Precision was utilized for the training part to compensate for the 32-bit floating point that was used for the model. The time it took for one epoch varied significantly, but for the most time it revolved at around 1 minute per epoch. During the training the methods, by which frames were chosen, were switched with accordance to the results of the model.

6. Results

Batch Size	FPS
1	90.30
5	89.89
20	67.03
40	32.17
50	25.47
60	20.59
120	2.5

Figure 4: Frames Per Second during interference with changing Batch Size.

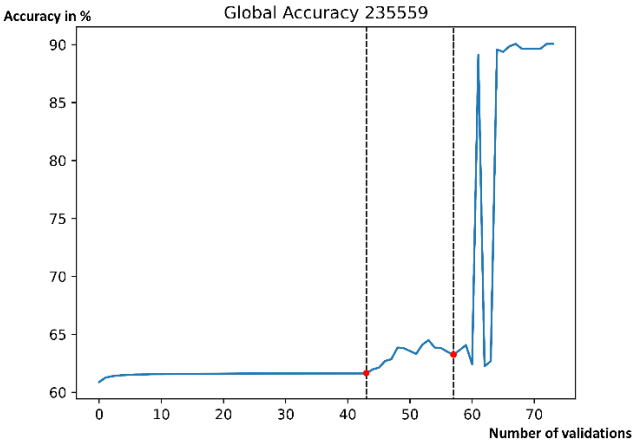

Figure 5: Accuracy with marked switching of methods.

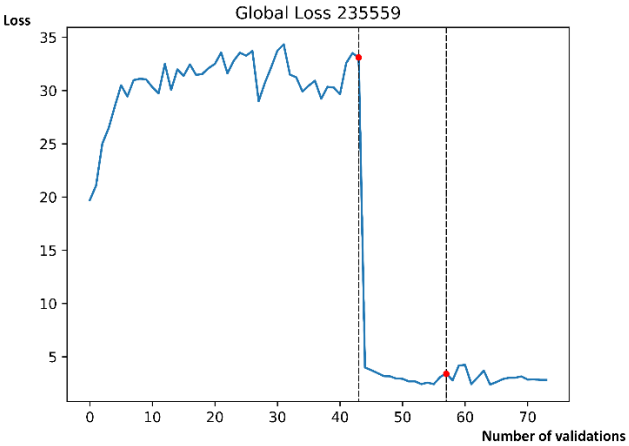

Figure 6: Loss with marked switching of methods.

While learning using all the available frames model was able to achieve around 62% accuracy. When switched to the second method of filtering, accuracy started to rise. AI reached best results when it switched to the third method, rising accuracy by 30% in short amount of time. We were able to achieve 90 FPS while using inferencing the model.

7. Analysis

When learning on a whole database, model is not able to reach acceptable results. Further fine-tuning on a selected number of frames, made the model predictions more unstable for a few next epochs, but in the end, caused to improve the performance. It may imply that the careful data selection for further stages of training is more important than the number of accumulated data at the beginning.

8. Evaluation in practice

Due to various issues we were not able to properly test the model in real environment. Issues that caused this were:

- Mamba is not able to run on Windows OS. We have chosen the Windows Subsystem for Linux (WSL) as an efficient way to use UNIX OS. It is more resourceful with memory than virtual machine and requires less setup than running UNIX OS as a bootable option.
- Python library used to record the screen is compatible only with Windows. Other alternatives showed significantly worse performance.

We have tried to mitigate these problems to a degree by utilizing virtual camera included in OBS software on a Windows machine to stream the screen using cam2web. Then, using a python script we were downloading these frames and performing further modifications on them.

Figure 7: Path the image had to travelwhen using the described method.

Latency and quality of the streamed images was acceptable, but the inconsistent nature of downloading photos, made this method obsolete. Other tried methods were:

- Saving the frame as an image and then reading the from within the WSL.
- Sending the most recent image using local server.

These have shown even larger inconsistency and latency made these methods under performing.

9. Conclusion

Foundation models show great potential when used as an AI agent to play video games. They appear to be a good choice especially when reinforced learning is not applicable. However, as they are still in it's initial phase, many things have to be improved, especially on the software side. With improved compatibility between the OS versions, we may see in the near future a growth in popularity of methods, hopefully narrowing the gap between human and AI perception of virtual entertainment.

10. Code

Code is available at: https://github.com/HaMikyu/MambaPlayer

References

1. C. Hu, Y. Zhao, Z. Wang, H. Du, and J. Liu, Games for artificial intelligence research: A review and perspectives, 2024. arXiv: 2304.13269 [cs.AI]. [On- line]. Available: https://arxiv.org/abs/2304.13269.
2. D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, General video game ai: A multi-track framework for evaluating agents, games and content generation algorithms, 2019. arXiv: 1802.10363 [cs.AI]. [On- line]. Available: https://arxiv.org/abs/1802.10363.
3. V. Mnih, K. Kavukcuoglu, D. Silver, et al., Playing atari with deep reinforcement learning, 2013. arXiv: 1312.5602 [cs.LG]. [Online]. Available: https:// arxiv.org/abs/1312.5602.
4. M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, Vizdoom: A doom-based ai research platform for visual reinforcement learning, 2016. arXiv: 1605.02097 [cs.LG]. [Online]. Available: https://arxiv.org/abs/ 1605.02097 (→ 2).

5. Vaswani, N. Shazeer, N. Parmar, et al., Anention is all you need, 2023. arXiv: 1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762.

6. Gu and T. Dao, “Mamba: Linear-time sequence modeling with selective state spaces”, arXiv preprint arXiv:2312.00752, 2023.

7. T. Dao and A. Gu, “Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality”, in International Conference on Machine Learning (ICML), 2024.

8. Gu, K. Goel, and C. Ré, Efficiently modeling long sequences with structured state spaces, 2022. arXiv: 2111.00396 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.00396.

9. T. Pearce and J. Zhu, Counter-strike deathmatch with large-scale behavioural cloning, 2021. arXiv: 2104.04258 [cs.AI]. [Online]. Available: https:// arxiv.org/abs/2104.04258.

10. L. Schäfer, L. Jones, A. Kanervisto, et al., Visual encoders for data-efficient imitation learning in modern video games, 2023. arXiv: 2312.02312 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2312.02312.

11. Chan, C. Ezell, M. Kaufmann, et al., Visibility into ai agents, 2024. arXiv: 2401.13138 [cs.CY]. [Online]. Available: https://arxiv.org/abs/2401.13138.

12. D. Huh and P. Mohapatra, Multi-agent reinforcement learning: A comprehensive survey, 2024. arXiv: 2312.10256 [cs.MA]. [Online]. Available: https://arxiv.org/abs/2312.10256