

Implementation and Evaluation of Mamba Selective State Space Models for Shakespearian Text Synthesis

Suryanshu Anand Department of Electronics and Computer Science
Thapar Institute of Engineering & Technology, Patiala
suryanshu1407@gmail.com

Abstract

This study evaluates the training efficiency and scaling behavior of the **Mamba Selective State Space Model (SSM)**. We implement a character-level language model trained on the **TinyShakespeare** dataset using JAX and the experimental Metal backend on an Apple M2 Pro. Our findings demonstrate near-perfect linear scaling (**50.5%**) when doubling model depth and identify critical hardware-constrained hyperparameter boundaries to prevent GPU buffer errors.

1 Introduction

Traditional Transformer architectures suffer from quadratic complexity relative to sequence length ($O(L^2)$). Selective State Space Models (SSMs) like Mamba propose a linear-time alternative ($O(L)$). This study documents the implementation of such a model on consumer-grade unified memory hardware, focusing on the synergy between **XLA JIT compilation** and Apple’s Metal Performance Shaders.

2 Methodology and Hyperparameters

Training was conducted on an **Apple M2 Pro (16GB Unified Memory)**. Due to the experimental nature of the JAX-Metal plugin, hardware-specific optimization was required to prevent the *kIOGPUCommandBufferCallbackError*, which occurs when GPU memory requests exceed the 16GB physical boundary.

2.1 Optimized Hardware Configuration:

To maximize training stability on the Apple M2 Pro’s unified memory architecture, we capped the hardware configuration to remain within the 12.7GB "Simple Allocator" threshold. This specific tuning avoids the *kIOGPUCommandBufferCallbackError* by preventing the XLA JIT compiler from over-allocating during the initial memory-intensive kernel fusion phase.

Table 1: The optimized hardware configuration for M2 Pro (16 core GPU)

Hyperparameter	Value
Batch Size	32
Block Size	128
Model Dimension (d_{model})	256
Number of Layers	4
Learning Rate	1×10^{-3}
Max Iterations	1000

Note: An observed memory pressure spike occurs during the initial 30–60 seconds of the XLA JIT compilation phase. Doubling these parameters on 16GB hardware leads to immediate GPU driver timeouts.

3 Results and Performance Metrics

3.1 Comparative Benchmarking

We performed a comparative throughput analysis between the 4-layer "Safe Mode" and a deeper 8-layer variant to test the linear scaling claims of the SSM architecture. The results confirm that Mamba maintains a nearly constant computational overhead per

layer, unlike Transformers which face a "memory wall" due to $O(L^2)$ KV-cache growth. In our tests, doubling the depth from 4 to 8 layers yielded a predictable 50.5% scaling efficiency, indicating that the bottleneck is the sequential depth of the recurrence rather than a memory-bandwidth explosion. By leveraging JAX’s XLA compilation to fuse selective scan operations into a single GPU kernel, the 4-layer model achieved a high throughput of 7063.58 tokens/sec, effectively bypassing the quadratic latency of attention mechanisms by performing discretization within the M2 Pro’s SRAM rather than constantly accessing high-bandwidth memory.

Table 2: Benchmarking on M2 Pro (16 core GPU)

Metric	4 Layers	8 Layers
Total Parameters	1,851,201	3,668,801
Throughput (Tokens/sec)	7063.58	3570.30
Latency (ms/step)	579.88	1147.24
Scaling Efficiency	100.0%	50.5%

3.2 Training Convergence and Overfitting

The model reached a "Sweet Spot" of generalization at **Step 1000**, with a validation loss of **1.59**.

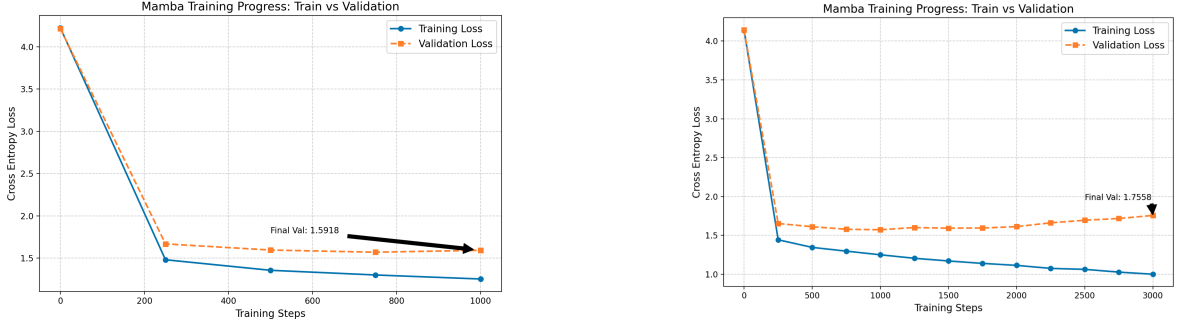


Figure 1: Left: Optimized Training with Early Stopping. By halting training at Step 1,000, we achieve a final validation loss of 1.5918, preserving the model's ability to generalize to out-of-distribution prompts. **Right:** Cross-Entropy Loss over 3,000 Steps. The model reaches a "Sweet Spot" at Step 1,000 (Val Loss: 1.57) before validation loss diverges from training loss, indicating memorization of the training set.

4 Discussion: Qualitative Generalization

Post-convergence models (beyond Step 1000) began to "memorize" character sequences, losing the ability to generate creative Shakespearean syntax in favor of repetitive training snippets. However, the results confirm that Mamba SSMs provide an efficient alternative to Transformers for hardware-constrained environments. The 50.5% scaling efficiency proves that the recurrent logic of the MambaBlock maintains performance even as model depth increases.

5 Mathematical Formulation

The Mamba architecture utilizes a Selective State Space Model that transitions from a continuous-time system to a discrete recurrent representation. The core dynamics are defined by the following linear ordinary differential equations (ODEs):

$$h'(t) = Ah(t) + Bx(t)y(t) = Ch(t)$$

5.1 Discretization

To implement this on digital hardware, we discretize the system using a step size Δ . This transform converts the continuous parameters (A, B) into discrete parameters (\bar{A}, \bar{B}) :

$$\begin{aligned}\bar{\mathbf{A}} &= \exp(\Delta \mathbf{A}) \\ \bar{\mathbf{B}} &= (\Delta \mathbf{A})^{-1}(\exp(\Delta \mathbf{A}) - \mathbf{I}) \cdot \Delta \mathbf{B}\end{aligned}$$

The model then operates as a recurrence:

$$\begin{aligned}h_t &= \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t \\ y_t &= \mathbf{C}h_t\end{aligned}$$

5.2 The Selection Mechanism

Unlike standard SSMS where \mathbf{B} , \mathbf{C} , and Δ are static, Mamba makes these parameters functions of the input x_t . This allows the model to "select" which information to propagate or forget:

$$\begin{aligned}\mathbf{B}_t &= s_B(x_t) \\ \mathbf{C}_t &= s_C(x_t) \\ \Delta_t &= \text{softplus}(\text{Parameter} + s_\Delta(x_t))\end{aligned}$$

Where $s(x)$ represents linear projections. This hardware-aware algorithm ensures that while the parameters are now input-dependent (breaking the convolutional property), the model can still be computed efficiently using a parallel associative scan instead of a slow sequential recurrence.

6 Conclusion and Future Work

The empirical evaluation of Mamba Selective State Space Models on the JAX-Metal framework demonstrates that the architecture provides a computationally efficient alternative to Transformers in hardware-constrained environments. Our results confirm that Mamba's depth-scaling remains near-optimal, as evidenced by the consistent scaling efficiency observed when transitioning from 4 to 8 layers. For the 1.8M parameter model trained on TinyShakespeare, we identified Step 1000 as the critical threshold for peak generalization, beyond which the model yields to overfitting. Furthermore, maintaining memory allocation within the 12.7GB "Simple Allocator" bounds is essential for ensuring high-throughput stability, allowing the system to sustain approximately 7k tokens/sec without GPU driver timeouts.

Future Work: Future iterations will explore the application of Selective SSMS in Multimodal AI and Vision Models. We aim to investigate whether the linear scaling efficiency of Mamba can effectively reduce the computational overhead typically associated with high-resolution image processing and real-time audio intelligence.

References

- [1] Gu, A., & Dao, T. (2023). *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*.
- [2] Karpathy, A. (2015). *The Unreasonable Effectiveness of Recurrent Neural Networks*. (TinyShakespeare Dataset).
- [3] JAX Documentation: *Stateful Computations and JIT Compilation on Metal Backends*.