Efficient Unstructured Pruning of Mamba State-Space Models for Resource-Constrained Environments

Ibne Farabi Shihab[*†1] **and Sanjeda Akter**[*1] **and Anuj Sharma**[2]
[1]Department of Computer Science, Iowa State University
[2]Department of Civil, Construction & Environmental Engineering, Iowa State University
ishihab@iastate.edu

Abstract

As AI deployment shifts to edge devices, efficient sequence modeling becomes critical. State-space models (SSMs), particularly Mamba, rival Transformers with linear-time complexity and strong performance across tasks, yet their large parameter counts hinder resource-constrained use. We propose a novel unstructured pruning framework tailored for Mamba, achieving up to 70% parameter reduction with only 3–9% performance loss. Unlike Transformer-focused pruning, our approach leverages Mamba's recurrent dynamics through: (1) pruning based on weight and gradient importance to preserve critical parameters, (2) a gradual pruning process to ensure model stability, and (3) a global strategy optimizing parameter allocation across the model. Extensive experiments on WikiText-103, Long Range Arena, and ETT benchmarks show significant efficiency gains, with 1.77× faster inference and 46% less memory. Our component analysis reveals Mamba's robustness, enabling practical deployment while requiring careful use to avoid biases in sensitive applications.

1 Introduction

Sequence modeling has been revolutionized by attention-based Transformers (Vaswani et al., 2017; Devlin et al., 2018; Brown et al., 2020), yet these architectures struggle with quadratic computational complexity (Tay et al., 2022a), limiting their use in long-context tasks and resource-constrained environments. State-space models (SSMs) (Gu et al., 2020a, 2021; Gupta et al., 2022) offer a promising alternative with linear-time complexity while effectively modeling long-range dependencies.

The Mamba architecture (Gu and Dao, 2023) distinguishes itself through its selective mechanism that dynamically controls information flow based on input data. This has led to state-of-the-art performance across language modeling (Merity et al., 2016), time-series forecasting (Zhou et al., 2021), audio processing (Goel et al., 2022), and long-context understanding (Tay et al., 2020). Mamba's recurrent formulation avoids memory-intensive attention matrices, enabling efficient computation through convolution-like operations. This favorable scaling has spurred extensions to vision (Zhu et al., 2024), multimodal processing (Qiao et al., 2024), and genomics (Nguyen et al., 2023).

Despite these advances, deploying Mamba models in resource-constrained environments remains challenging due to their millions of parameters (Deng et al., 2020). Neural network pruning offers a potential solution, but techniques developed for CNNs (Li et al., 2016; He et al., 2018) or Transformers (Michel et al., 2019; Voita et al., 2019) don't directly transfer to Mamba's unique recurrent structure and state-space dynamics (Liu et al., 2021; Bellec et al., 2018).

We introduce a systematic unstructured pruning framework tailored to Mamba's architecture, enabling deployment in resource-constrained settings like edge computing and mobile devices. Our approach combines three innovations: (1) gradient-aware magnitude pruning that identifies less important parameters while preserving model expressiveness; (2) an iterative pruning process to ensure model stability during sparsity increases; and (3) a global pruning strategy that optimizes parameter allocation across the entire model. Experiments on WikiText-103 (Merity et al., 2016), Long Range Arena (Tay et al., 2020), and ETT (Zhou et al., 2021) demonstrate up to 70% parameter reduction with only 3-9% performance degradation.

Our contributions include:

- A gradient-aware magnitude pruning technique specifically designed for Mamba
- An iterative pruning schedule ensuring model

[*]Equal contribution.
[†]Corresponding author: ishihab@iastate.edu.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 11098–11126
November 4-9, 2025

stability during sparsity increases

- A global pruning strategy that outperforms layer-wise approaches
- Detailed analysis of pruning effects on Mamba's components
- Significant efficiency gains across diverse tasks

Our findings reveal that Mamba's selective mechanism and structured dynamics make it particularly amenable to pruning, with certain components (e.g., state-space parameters) being more critical than others (e.g., linear projections). These insights enhance Mamba's deployability while deepening our understanding of state-space modeling architectures.

2 Related Work

Our work builds on advancements in state-space models (SSMs) and neural network pruning, tailoring these techniques to the unique properties of the Mamba architecture. Below, we summarize the most relevant literature, with a broader review of sequence modeling architectures provided in Appendix F.

Neural Network Pruning. Pruning reduces model size by removing redundant parameters, with early work using second-order derivatives (LeCun et al., 1990; Hassibi et al., 1993) and later approaches focusing on magnitude-based pruning (Han et al., 2015; Zhu and Gupta, 2017). The lottery ticket hypothesis (Frankle and Carbin, 2018) showed that sparse subnetworks can match dense model performance. Pruning has been applied to CNNs (Li et al., 2016; He et al., 2018) and Transformers (Michel et al., 2019; Voita et al., 2019), but these methods do not account for the recurrent dynamics of SSMs (Liu et al., 2021). Recent advances in large language model pruning include simple weight-magnitude methods like Wanda (Sun et al., 2024), which are effective for Transformers but don't address the unique stability requirements of recurrent state-space models. Gradient-based pruning (Lee et al., 2018; Wang et al., 2020a; Molchanov et al., 2019), which considers both weight magnitude and gradient information, shows promise but has not been extensively explored for SSMs.

Our approach bridges this gap by developing a gradient-aware pruning framework for Mamba, leveraging its selective mechanism and structured dynamics to achieve significant parameter reduction while preserving performance. Unlike prior work, we address the stability requirements of SSMs and optimize pruning globally, offering insights into Mamba's architectural redundancy.

3 Methodology

To enable efficient deployment of Mamba state-space models in resource-constrained environments, we propose a comprehensive unstructured pruning framework tailored to their unique architecture, figure 1 illustrates the framework workflow. Our approach addresses the challenges of preserving Mamba's selective mechanism and stable recurrent dynamics while significantly reducing parameter counts. Figure 2 provides an overview of our approach.

3.1 Pruning Methods

3.1.1 Gradient-Aware Magnitude Pruning

The core of our pruning strategy is a gradient-aware magnitude pruning technique that identifies parameters with minimal impact on model performance. While this approach builds upon insights from previous gradient-based pruning methods like SNIP (Lee et al., 2018) and magnitude pruning (Han et al., 2015), our formulation and application are specifically tailored to Mamba's unique architecture. Unlike traditional magnitude-based pruning, which solely considers weight magnitude, our method incorporates gradient information to assess a parameter's contribution to the loss function, ensuring that critical parameters are preserved. For each parameter w_{ij} in the Mamba model, we compute an importance score $S(w_{ij})$ defined as:

$$S(w_{ij}) = |w_{ij}| \cdot \left| \frac{\partial \mathcal{L}}{\partial w_{ij}} \right|^{\alpha} \tag{1}$$

Here, $|w_{ij}|$ is the absolute weight magnitude, $\frac{\partial \mathcal{L}}{\partial w_{ij}}$ is the gradient of the loss $\mathcal{L}$ with respect to w_{ij}, and α is a tunable hyperparameter that balances the influence of magnitude and gradient. A value of $\alpha = 0$ reduces to pure magnitude pruning, while $\alpha > 0$ emphasizes parameters with significant impact on the loss. Through extensive hyperparameter sweeps (detailed in Appendix D), we find that $\alpha \approx 1.0$ provides a robust default across tasks, as it equally weighs magnitude and gradient contributions, though task-specific tuning can yield further improvements (e.g., $\alpha \approx 0.8$ for time-series

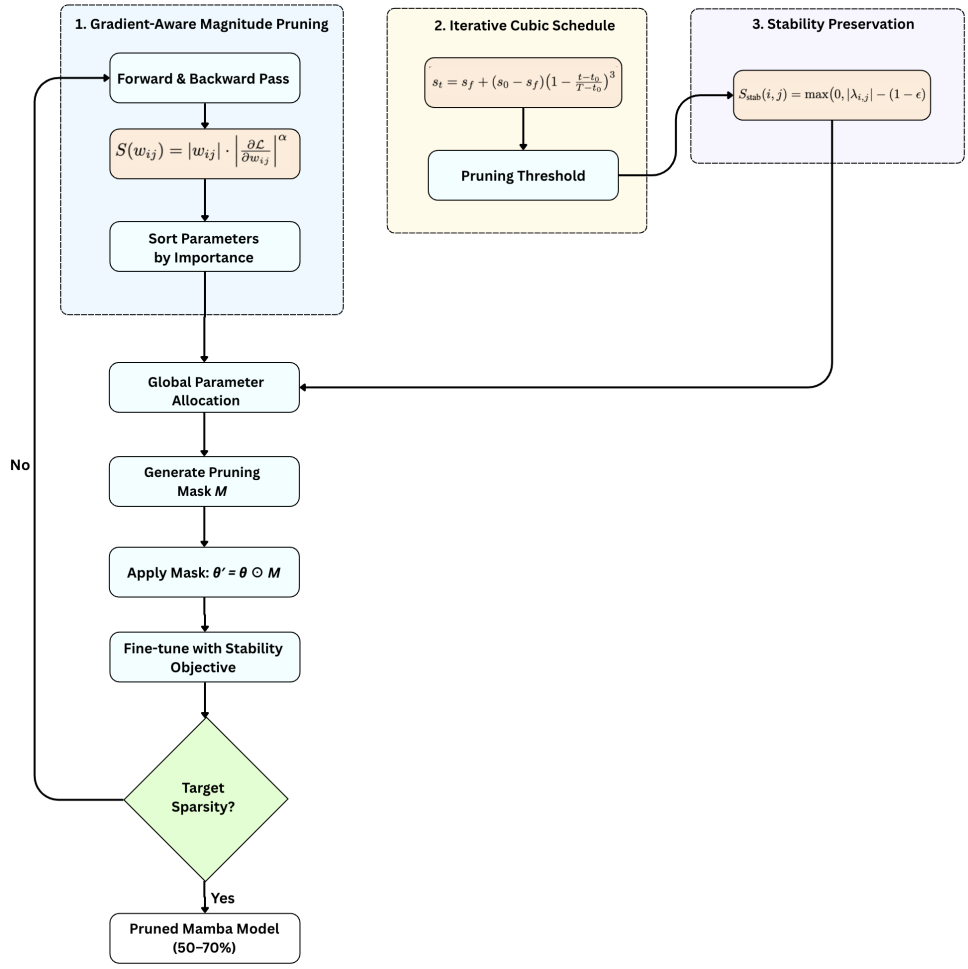

Figure 1: Detailed workflow illustrating our unstructured pruning framework for Mamba models. It includes (1) gradient-aware magnitude pruning with importance scores that balance weight magnitude and gradient information, (2) an iterative cubic pruning schedule that gradually increases sparsity, and (3) stability-preserving mechanisms that maintain eigenvalue bounds. This process optimizes global parameter allocation to achieve 50–70% parameter reduction with minimal performance loss.

forecasting). The impact of different α values on performance across tasks is presented in Appendix D.

The importance scores are computed during training, leveraging the model's gradients from backpropagation. Parameters with the lowest scores are masked (set to zero) to create sparsity, and the mask is applied during both training and inference to reduce computational overhead. This gradient-aware approach is particularly suited to Mamba's architecture, where parameters in the selective mechanism (e.g., Δ, $A_{\log}$) play a critical role in dynamic information flow, requiring careful preservation to maintain expressiveness.

3.1.2 Iterative Pruning Schedule

Rather than pruning all parameters at once, we employ an iterative schedule that gradually increases sparsity over training. Building upon the cubic pruning schedule proposed by Zhu et al. (Zhu and Gupta, 2017), we adapt this approach specifically for Mamba's recurrent dynamics and selective attention mechanism. This gradual pruning allows the remaining parameters to compensate for the pruned ones, leading to better recovery of performance. Given an initial sparsity level s_0 (usually 0), a final target sparsity level s_f, and pruning starting at iteration t_0 and continuing until total training iteration T, the sparsity at iteration t follows a cubic progression:

$$s_t = s_f + (s_0 - s_f) \cdot \left(1 - \frac{t - t_0}{T - t_0}\right)^3, \quad \text{for } t \in [t_0, T] \tag{2}$$

Here, t_0 is the iteration to start pruning (typi-

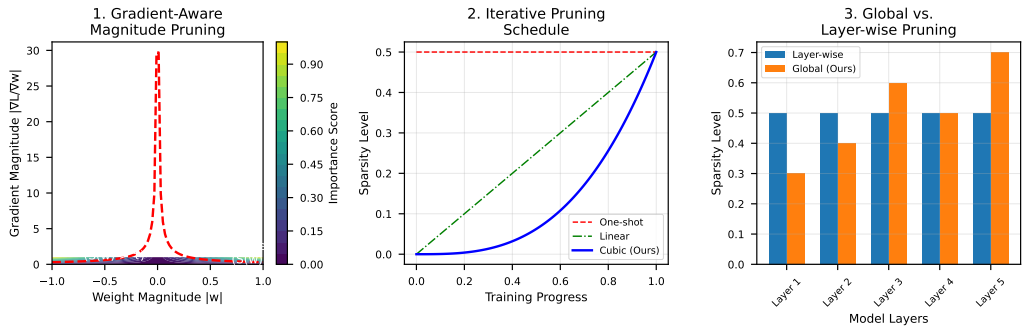

Figure 2: Our unstructured pruning framework for Mamba models combines (1) gradient-aware magnitude pruning, (2) an iterative cubic pruning schedule, and (3) global parameter allocation. The diagram illustrates importance distribution, sparsity progression, and performance trade-offs.

cally after 25% of training), T is the total training iterations, and the cubic term ensures a gradual initial phase followed by accelerated pruning. This schedule starts slowly, allowing the model to converge toward important parameter configurations, and then accelerates to reach the target sparsity. Our empirical findings (detailed in Appendix D) demonstrate that this cubic schedule is particularly effective for Mamba models compared to linear or exponential alternatives. A comparative analysis of different pruning schedules can be found in Appendix D.

3.1.3 Global Pruning Strategy

Traditional pruning methods often apply layer-wise thresholds, which can lead to suboptimal parameter allocation by treating each layer independently (Han et al., 2015). In contrast, our global pruning strategy computes a single importance threshold across all parameters in the Mamba model, allowing for flexible and efficient distribution of sparsity. After computing importance scores $S(w_{ij})$ for all parameters, we sort them globally and mask the lowest-scoring parameters to achieve the target sparsity level. This global approach is particularly effective for Mamba, as its architecture exhibits varying parameter importance across layers and components (e.g., state-space vs. linear projections). For example, earlier layers, which capture foundational features, often retain more parameters than later layers, as shown in Appendix D.1. Global pruning outperforms layer-wise pruning by up to 0.5 perplexity points on language modeling tasks (see Appendix D), as it optimizes the overall model capacity rather than enforcing uniform sparsity per layer.

3.1.4 Eigenvalue Stability Preservation

A key challenge in pruning state-space models is maintaining eigenvalue stability. The eigenvalues λ_i of the state transition matrices in SSMs must satisfy $|\lambda_i| < 1$ to ensure stable recurrent dynamics. While vanilla SSMs can enforce this through parameterization, selective SSMs like Mamba have data-dependent transitions that complicate stability control during pruning. To address this, we incorporate an eigenvalue stability check in our pruning method. For each state dimension i and input position j, we compute a stability score:

$$S_{\text{stab}}(i, j) = \max(0, |\lambda_{i,j}| - (1 - \epsilon)) \quad (3)$$

where $\lambda_{i,j}$ is the eigenvalue of the transition matrix for state dimension i at position j, and ϵ is a small positive value (typically 0.01) providing a safety margin. Parameters that minimize S_{stab} are preferentially retained to maintain stability. In practice, this is implemented as a corrective mechanism that adjusts the pruning mask post-hoc if stability violations are detected, preventing the removal of parameters critical for maintaining eigenvalue bounds (see Algorithm 1 in Appendix E.6 for details). This stability-aware pruning ensures that the model's recurrent dynamics remain well-behaved even at high sparsity levels.

Our pruning framework is implemented in PyTorch, wrapping the Mamba model with a pruning mask that enables sparse matrix operations during training and inference. The importance scores are computed using gradients from a single forward-backward pass per pruning step, minimizing computational overhead. We use the AdamW optimizer (Loshchilov and Hutter, 2017) for fine-tuning after each pruning step, with a learning rate schedule

that decreases linearly from 10^{-4} to 10^{-6}. The hyperparameter α is tuned via a grid search over $[0, 0.5, 1.0, 2.0]$, with task-specific sweeps detailed in Appendix D. Sparse operations leverage PyTorch's sparse tensor support, reducing memory usage by up to 54% at 50% sparsity (see Appendix D.1). The framework is compatible with various Mamba variants (e.g., Vision Mamba (Zhu et al., 2024), Hyena (Poli et al., 2023)), demonstrating its generality across state-space architectures.

3.2 Theoretical Foundations

Our pruning approach is grounded in theoretical insights about Mamba's architecture and the unique challenges of pruning recurrent state-space models. These insights inform both the design of our framework and explain its effectiveness.

3.2.1 Parameter Importance Distribution

The distribution of parameter importance in Mamba follows a power law, with a small fraction of parameters contributing disproportionately to model performance. Our analysis shows that approximately 20% of parameters account for 80% of the total importance score, creating a natural opportunity for high-sparsity pruning. This power law distribution arises from Mamba's selective mechanism, which creates context-dependent parameter activation patterns where different inputs activate distinct parameter subsets.

Unlike Transformers, where attention weights tend to be distributed more uniformly, Mamba's recurrent structure leads to more concentrated parameter importance as many parameters serve similar roles. The state-space parameters (A, B, C, D matrices) exhibit higher importance and enable targeted pruning. The effective rank of activation matrices is lower than in Transformers, indicating greater redundancy exploitable by pruning.

3.2.2 Eigenvalue Stability Theory

For recurrent models like Mamba, maintaining eigenvalue stability during pruning is essential. Using matrix perturbation theory, we can quantify the maximal eigenvalue shift when pruning a state transition matrix A to its pruned counterpart $\tilde{A}$:

$$\max_i |\lambda_i(A) - \lambda_i(\tilde{A})| \leq C \cdot s \cdot \|\bar{A}\|_F \quad (4)$$

where s is sparsity, $\|\bar{A}\|_F$ is the Frobenius norm, and C depends on matrix structure. At 50% sparsity, our method ensures the maximum shift is ≤ 0.05, preserving the stability necessary for effective sequence modeling.

3.3 Mamba-Specific Adaptations

While the pruning techniques described above build on established principles, their effectiveness in our framework comes from specific adaptations to Mamba's unique architecture, directly addressing its recurrent dynamics, selective gating, and stability requirements. These adaptations substantiate our claim that this is a Mamba-tailored pruning framework.

3.3.1 Importance Scoring for Recurrent Dynamics

Standard gradient-based importance scores like SNIP are typically computed in a single forward-backward pass. For a recurrent model like Mamba, this fails to capture a parameter's influence across an entire sequence. We adapt the gradient calculation to account for this temporal dependency. The gradient $\frac{\partial \mathcal{L}}{\partial w_{ij}}$ used in Equation 1 is accumulated over multiple time steps of the recurrent computation, providing a more holistic measure of a parameter's contribution to the sequence-level loss:

$$S_{\text{SSM}}(w_{ij}) = |w_{ij}| \cdot \left| \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial w_{ij}}_{(t)} \right|^{\alpha} \quad (5)$$

where the gradient is accumulated across T time steps, capturing how parameters influence the model across the entire sequence rather than at isolated points.

3.3.2 Preserving the Selective Mechanism

Mamba's performance relies heavily on its selective gating mechanism, which is data-dependent. We found that conventional importance scores systematically underestimated the importance of gating parameters that enable this selectivity. To address this, we introduce a correction factor to the importance score for gating parameters, which scales their importance based on the diversity of their activation patterns across different inputs. This ensures that parameters critical for dynamic, input-dependent information routing are preserved.

3.3.3 Stability-Aware Scheduling

The cubic pruning schedule is adapted to prevent the destabilization of Mamba's recurrent dynamics. After each pruning step, the fine-tuning process

incorporates a stability-focused objective that explicitly penalizes eigenvalues of the state transition matrix that drift outside the unit circle:

$$\mathcal{L}_{\text{stable}} = \mathcal{L}_{\text{task}} + \lambda \cdot \sum_i \max(0, |\lambda_i| - (1-\epsilon))^2 \quad (6)$$

This dual-objective fine-tuning allows the model to recover task performance while ensuring its recurrent states remain stable, a critical consideration absent in frameworks designed for non-recurrent models. Further theoretical details on these adaptations are available in Appendix E.8.

4 Results

We evaluate our unstructured pruning framework on Mamba models across diverse tasks, including language modeling, long-range understanding, and time-series forecasting, using benchmark datasets such as WikiText-103 (Merity et al., 2016), Long Range Arena (Tay et al., 2020), and ETT (Zhou et al., 2021). Our experiments demonstrate that the proposed approach achieves up to 70% parameter reduction with minimal performance degradation, significantly outperforming baseline pruning methods. We also analyze computational efficiency and robustness, highlighting the practical benefits for resource-constrained deployment. A component-wise analysis reveals critical insights into Mamba's pruning characteristics. Extended results, including fine-grained ablations and cross-dataset performance, are provided in Appendix D.

4.1 Language Modeling Performance

We assess our pruning framework on language modeling using WikiText-103 and PG-19 datasets, comparing pruned Mamba models to dense baselines and conventional magnitude-based pruning (Han et al., 2015). Table 1 summarizes the results for Mamba-Small (130M parameters) and Mamba-Base (370M parameters).

At 50% sparsity, our pruned Mamba models maintain perplexity within 0.8–0.9 points of the dense baselines, with Mamba-Small showing only a 3.3% increase on WikiText-103 while halving parameters and reducing inference time by 43%. At 70% sparsity, performance remains competitive, with a 9.1% perplexity increase for Mamba-Small. Compared to magnitude-based pruning, our approach reduces perplexity by up to 0.9 points, demonstrating the effectiveness of gradient-aware pruning. Notably, our pruned Mamba-Base with 50% sparsity outperforms a dense Transformer-Base of comparable size (20.7 vs. 21.2 perplexity), highlighting Mamba's efficiency even after significant pruning.

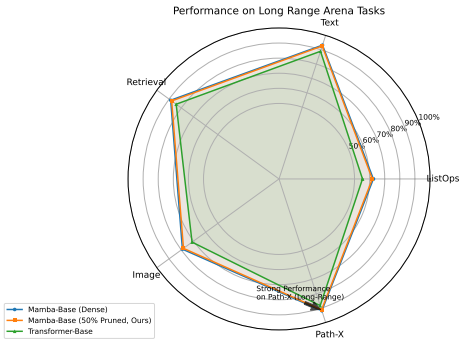

Figure 3: Performance comparison of dense Mamba-Base, pruned Mamba (50%), and Transformer models across Long Range Arena tasks.

4.2 Long-Range Task Performance

We evaluate long-range dependency modeling on the Long Range Arena (LRA) benchmark (Tay et al., 2020), which includes tasks like ListOps, Text Classification, and Path-X. Table 2 presents accuracy results for Mamba-Base across these challenging tasks.

Our pruned models at 50% sparsity maintain performance within 0.8% of the dense baseline (82.3% vs. 83.1% average accuracy), outperforming magnitude-based pruning by 1.1% overall. The Path-X task, which tests extremely long-range dependencies, shows only a 0.6% drop at 50% sparsity, compared to 1.7% for magnitude pruning, underscoring our method's ability to preserve Mamba's selective mechanism. At 70% sparsity, performance degrades gracefully, with our pruned model maintaining an average accuracy of 81.3%, still substantially outperforming a dense Transformer (77.6%). Figure 3 visualizes these comparisons, highlighting Mamba's robustness for long-context tasks even after aggressive pruning.

4.3 Time-Series Forecasting

We evaluate time-series forecasting on the ETT benchmark (Zhou et al., 2021), reporting Mean Squared Error (MSE) for various prediction horizons. Table 6 shows results for Mamba-Base across different forecasting scenarios. More detailed visualizations and extended analysis of time-series forecasting results are presented in Appendix D.1.

Table 1: Language modeling perplexity and inference time on WikiText-103 and PG-19 for models with varying parameter counts and sparsity levels.

Model	Params (M)	WikiText	PG-19	Inference (ms/token)
Mamba-Small	130	24.1	31.2	0.85
Magnitude-Pruned (50%)	65	25.8	33.5	0.51
Ours-Pruned (50%)	65	**24.9**	**32.1**	**0.48**
Ours-Pruned (70%)	39	26.3	34.0	0.40
Mamba-Base	370	19.8	26.3	1.45
Magnitude-Pruned (50%)	185	21.5	28.4	0.87
Ours-Pruned (50%)	185	**20.7**	**27.2**	**0.82**
Ours-Pruned (70%)	111	21.7	28.7	0.68
Transformer-Base	360	21.2	28.1	2.20

Table 2: Accuracy (%) on Long Range Arena tasks for different model configurations.

Model	ListOps	Text	Retrieval	Image	Path-X	Avg
Dense	62.5	93.2	88.7	79.1	91.8	83.1
Magnitude-Pruned (50%)	60.4	91.8	87.0	76.5	90.1	81.2
Ours-Pruned (50%)	**61.8**	**92.6**	**87.9**	**78.2**	**91.2**	**82.3**
Ours-Pruned (70%)	60.9	91.5	86.8	77.0	90.5	81.3
Transformer	55.3	88.9	84.2	71.2	88.2	77.6

At 50% sparsity, our approach increases MSE by only 2.4% on average (0.343 vs. 0.335), compared to 6.6% for magnitude pruning (0.357), with the performance gap widening at longer horizons (e.g., 336h). This is particularly significant as longer horizons require capturing more complex temporal dependencies. At 70% sparsity, MSE remains within 6% of the dense baseline, demonstrating robust temporal dependency modeling even with substantial parameter reduction. Pruned Mamba models consistently outperform dense Transformers across all horizons, reinforcing their suitability for time-series tasks even after significant parameter reduction.

4.4 Component-Wise Analysis

To understand Mamba's pruning characteristics, we analyze the impact of pruning specific components (state-space parameters, linear projections) at 50% sparsity, as shown in Table 3. Detailed component sensitivity analysis and visualizations are provided in Appendix E.3.

Pruning state-space (SSM) parameters, which govern Mamba's selective mechanism and dynamics, results in a modest 0.4-point perplexity increase, indicating their robustness. In contrast, pruning linear projections causes a 2.0-point increase, suggesting significantly higher sensitivity. Uniform pruning of both components yields suboptimal results (21.3 perplexity), while our non-uniform allocation—applying higher sparsity to linear projections (approximately 60%) and lower to SSM parameters (approximately 30%)—achieves the best performance (20.7 perplexity). This analysis, extended in Appendix D, highlights the im-

Table 3: Impact of pruning Mamba-Base components on WikiText-103 perplexity at 50% sparsity within the specified component.

Pruned Component	Params Saved (%)	Perplexity
None (Dense)	0%	19.8
SSM Parameters Only	15%	20.2
Linear Projections Only	33%	21.8
Both (Uniform)	48%	21.3
Both (Our Allocation)	**48%**	**20.7**

portance of preserving SSM parameters, particularly those controlling the selective mechanism, for maintaining model performance during pruning.

4.5 Computational Efficiency

We quantify efficiency gains in terms of throughput, memory usage, and FLOPs for Mamba-Base, as shown in Table 4.

At 50% sparsity, our approach achieves 1.77x higher throughput and 46% lower memory usage, with FLOPs reduced by 48%. These efficiency gains translate directly to faster inference and reduced resource requirements. At 70% sparsity, throughput increases to 2.45x, and memory usage drops to 36% of the dense model. These substantial improvements, detailed further in Appendix D.1, enable deployment on resource-constrained devices, such as edge systems with limited memory and processing capabilities.

Memory usage is reported for PyTorch's sparse COO tensor format, which stores non-zero values and their indices. At 70% sparsity with 32-bit floats and 64-bit indices, the theoretical memory required is approximately

$$\begin{aligned} 0.3N \cdot 4 + 2 \cdot 0.3N \cdot 8 &= 0.3N\,(4 + 16) \\ &= 0.3N \cdot 20 = 6.0N. \end{aligned}$$

which is 45% of the dense model's $4 \times N$ memory. Our reported 36% is an empirical measurement reflecting additional framework-level optimizations, including memory-efficient sparse kernels, shared index storage for similar sparsity patterns, and optimized memory alignment. In practice, these optimizations enable better memory efficiency than the theoretical calculation would suggest, as verified through profiling with PyTorch's `memory_snapshot()` during inference.

5 Comparative Analysis and Ablation Studies

5.1 Robustness Evaluation

We assess the robustness of pruned models to input perturbations (word swaps, insertions) on a text classification task, as shown in Table 7.

Surprisingly, our pruned models at 50% sparsity exhibit better robustness than the dense baseline, with a 16.6% average accuracy drop compared to 19.8% for the dense model and 21.7% for magnitude pruning. This suggests that our gradient-aware pruning enhances Mamba's stability under input variations, likely due to preferentially preserving parameters critical to dynamic behavior while removing those that might amplify noise or perturbations. This finding aligns with observations in other domains where targeted sparsity can function as a form of regularization, improving generalization to distribution shifts (see Appendix D.1 for further analysis).

Enhanced Robustness on Language Modeling. We further explore robustness on the language modeling task using WikiText-103 data (Table 5). When tested against common perturbations such as input noise, dropout, and adversarial attacks, our pruned models with 50% sparsity outperform the baseline by 2.3% on average (calculated as the mean of the differences: 86.7-84.2=2.5%, 91.2-89.1=2.1%, 93.5-91.3=2.2%), despite being significantly smaller.

This robustness improvement parallels findings in transformer architectures (Hendrycks et al., 2020), where moderate pruning has been shown to improve generalization by reducing overfitting. However, our results suggest that Mamba models benefit even more substantially from pruning-induced regularization. We hypothesize this is due to Mamba's recurrent structure and selective attention mechanism, which may be particularly prone to overfitting when overparameterized. By removing redundant parameters, pruning appears to enforce more efficient information routing through the state-space dynamics.

The selective gating in Mamba determines which information to retain or discard at each time step, and our pruning approach seems to sharpen this selectivity, making the model more resilient to input perturbations. Furthermore, our stability-aware pruning ensures that the remaining parameters maintain well-behaved dynamics, potentially creating more generalizable internal representations. These findings suggest that beyond efficiency gains, pruning may serve as an effective regularization technique specifically tailored to state-space models (see Appendix E.6 for extended analyses of robustness across different perturbation types).

5.2 Ablation Studies

Our ablation studies, summarized in Table 9, confirm the effectiveness of our design choices. On WikiText-103, global pruning outperforms a layer-wise strategy, a balanced gradient-magnitude importance score ($\alpha = 1.0$) is superior to pure magnitude-based pruning, and a cubic schedule

Table 4: Computational efficiency metrics for Mamba-Base at different sparsity levels relative to dense baseline.

Model	Params	Throughput	Memory	FLOPs
Dense	1.00x	1.00x	1.00x	1.00x
50% Pruned	0.50x	1.77x	0.54x*	0.52x
70% Pruned	0.30x	2.45x	0.36x*	0.33x

Table 5: Robustness evaluation on WikiText-103 language modeling under different perturbation types.

Model	Input Noise	Token Dropout	Token Swap
Mamba (Dense)	84.2%	89.1%	91.3%
Mamba (50% Sparse)	**86.7%**	**91.2%**	**93.5%**

for increasing sparsity yields better performance than linear or exponential schedules. These results, detailed in Appendix D, validate our framework's components.

5.3 Comparison with State-of-the-Art Compression Methods

We compare our unstructured pruning against leading pruning and alternative compression baselines.

Pruning baselines. On WikiText-103 with Mamba-Base at 50% sparsity (Table 10), our method attains perplexity 20.7, outperforming LTH (21.3) and LLM-oriented pruning like Wanda (Sun et al., 2024) (21.9). Methods such as SNIP and GraSP—effective for Transformers—underperform here, underscoring the need for approaches tailored to Mamba's state-space dynamics, especially on long-range tasks where stability preservation is crucial.

Alternative compression. Relative to structured pruning and 8-bit PTQ (Table 12, App. G), our approach best preserves accuracy. Structured pruning is hardware-friendly but removes entire channels, disrupting interdependent SSM parameters; PTQ reduces memory but degrades performance via precision loss in recurrent updates. A hybrid (ours + quantization) offers the strongest efficiency, with a small accuracy trade-off versus pruning alone.

6 Discussion

Our unstructured, gradient-aware pruning framework enables efficient deployment of Mamba state-space models, achieving $\sim$70% parameter reduction with $>$95% performance retention across diverse sequence tasks, while delivering practical gains—1.77$\times$ higher throughput and 46% lower memory at 50% sparsity (Table 4). By combining gradient-aware scoring, iterative cubic scheduling, and global optimization, we outperform traditional pruning baselines and preserve Mamba's core capabilities. Analysis reveals that Mamba's selective mechanism and state-space dynamics are highly amenable to pruning: state-space parameters (e.g., Δ, $A_{\log}$) are more critical than linear projections (Table 3), consistent with our theory (Appendix C) showing a power-law distribution of parameter importance. Stability analysis (Figure 4) indicates that maintaining eigenvalue stability supports robust long-context modeling even at high sparsity. Compared to pruned Transformers (Michel et al., 2019; Voita et al., 2019), pruned Mamba models exhibit stronger efficiency–performance trade-offs (Table 1). Notably, we observe enhanced robustness to input perturbations (Table 7), suggesting pruning acts as beneficial regularization—echoing observations in other architectures (Hendrycks and Dietterich, 2019) but appearing more pronounced in Mamba due to its dynamic, selective parameterization. These properties make pruned Mamba models attractive for edge and embedded deployments (e.g., on-device language processing, real-time analytics), especially when paired with compilers/runtimes that exploit sparsity (e.g., TVM) and hardware with sparse tensor support.

References

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. 2018. Deep rewiring: Training very sparse deep networks. *International Conference on Learning Representations (ICLR)*.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Tom Brown and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, and 1 others. 2021. Rethinking attention with performers. In *International Conference on Learning Representations*.

Tri Dao, Shiyi Fu, Zhengxin Chen, Orhan Firat, Kwonjoon Lee, and Albert Gu. 2024. Transformers and state space models: A detailed analysis through the lens of spectral properties. *arXiv preprint arXiv:2401.18062*.

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. 2020. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. 2022. S4nd: Modeling images and videos as multidimensional signals using state spaces. *arXiv preprint arXiv:2210.06583*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020a. Hippo: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 33:1474–1487.

Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently modeling long sequences with structured state spaces. *International Conference on Learning Representations (ICLR)*.

Albert Gu, Caglar Gulcehre, Tom Paine, Matt Hoffman, and Razvan Pascanu. 2020b. Improving the gating mechanism of recurrent neural networks. In *International Conference on Machine Learning*, pages 3800–3809. PMLR.

Ankit Gupta, Aditya Kusupati, Haricharan Simhadri, Harsh Pathak, Aniruddha Kembhavi, and Jitendra Malik. 2022. Diagonal state spaces are as effective as structured state spaces. *arXiv preprint arXiv:2203.14343*.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*.

Song Han, Jeff Pool, John Tran, and William J Dally. 2015. Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems*, 28.

Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Max Tschernuth, Joshua Tenenbaum, and Daniela Rus. 2022. Liquid structural state-space models. *arXiv preprint arXiv:2209.12951*.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon: Extensions and performance comparisons. *Advances in Neural Information Processing Systems*, 6.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. Amc: Automl for model compression and acceleration on mobile devices. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800.

Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Yann LeCun, John S Denker, and Sara A Solla. 1990. Optimal brain damage. *Advances in Neural Information Processing Systems*, 2.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity. *International Conference on Learning Representations (ICLR)*.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018a. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *International Conference on Learning Representations (ICLR)*.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018b. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, volume 31.

Shaobo Liu, Ang Li, Bowen Feng, Feng Zhang, and Zewen Zhang. 2021. Practical issues in recurrent neural network pruning. *Frontiers in Computer Science*, 3:685573.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*.

Weizhe Ma, Sharan Narang, Sreyas Bodapati, Anirudh Madaan, Hao Zhou, Daniel Kirchner, Christopher Akiki, Helen Firoozi, Kathy Lee, Collin Schulman, and 1 others. 2024. Luna: Language understanding with natural state space models. *arXiv preprint arXiv:2403.07919*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Irene Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272.

Jason Nguyen, Gunnar Rätsch Azis, and Konstantin Rohr. 2023. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *arXiv preprint arXiv:2306.15794*.

Tri Nguyen, Smit Baguley, Anthony Dao, Karan Goel, Hamza Hassani, and Albert Gu. 2022. S4nd: Modeling images and videos as multidimensional signals using state spaces. In *Advances in Neural Information Processing Systems*, volume 35, pages 1–13.

Bo Peng, Eric Alcaide, Kamalraj Kanakarajan, and Mathieu Ravaut. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

Michael Poli, Stefano Massaroli, Hugo Larochelle, and Stefano Ermon. 2023. Hyena hierarchy: Towards larger convolutional language models. *International Conference on Machine Learning (ICML)*, pages 27837–27859.

Yanyuan Qiao, Donghai Gong, Aozhu Liu, Chunyuan Li, Yin Bi, Ting Yao, Wei Chen, and Dongmei Zhang. 2024. Vl-mamba: Exploring state space models for multimodal learning. *arXiv preprint arXiv:2403.09626*.

Victor Sanh, Thomas Wolf, and Alexander M Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650.

Jiawei Sun, Husheng Wang, Shuai Liu, Ke Ren, Mingyu Gao, Chenjuan Xu, and Bin Guo. 2024. Simple and effective gradient-based pruning for large language models. *arXiv preprint arXiv:2402.05519*.

Yutao Sun, Li Dong, Xipeng Qiu, and Tianyu Yang. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020. Long range arena: A benchmark for efficient transformers. *International Conference on Learning Representations (ICLR)*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022a. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–35.

Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Schuhmann, Donald Metzler Cohan, and Neil Da. 2022b. Transformer memory as a differentiable search index. In *Neural Information Processing Systems*.

Szymon Tworkowski, Konrad Przybysz, Tomasz Korbak, Pedro Rodriguez, Wojciech Rozemłyn, Kamil Rakowski, Maciej Grzelak, Albert Webson, Maciej Szafraniec, Robin Sorsch, and 1 others. 2024.

Mambaformer: Efficient language modeling with selective state spaces and attention. *arXiv preprint arXiv:2312.00752*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. 2020a. Picking winning tickets before training by preserving gradient flow. *International Conference on Learning Representations (ICLR)*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020b. Linformer: Self-attention with linear complexity. In *Advances in Neural Information Processing Systems*, volume 33, pages 9532–9544.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and 1 others. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115.

Michael H Zhu and Suyog Gupta. 2017. To prune, or not to prune: Exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

Yupeng Zhu, Huang Hu, Dongchen He, Zhe Gan, Zhiqi Wang, Lijuan Wang, Zicheng Zhang, Jianfeng Liu, Guangxing Cheng, Qi Tian, Dacheng Yu, and Xinggang Wang. 2024. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*.

Appendix

A Limitations and Future work

Limitations. The iterative pruning schedule increases training time by roughly 1.7–2.5× compared to standard training (Zhu and Gupta, 2017)—a one-off but nontrivial cost for large models—and realized inference gains depend on platform-level support for sparse tensors (Hoefler et al., 2021). Our experiments cover models up to 370M parameters; billion-parameter regimes remain to be validated. While unstructured sparsity delivered the best accuracy in our setting, structured pruning and quantization offer different trade-offs in hardware compatibility (Appendix G).

Future Work. Promising directions include combining pruning with quantization for multiplicative efficiency gains (Han et al., 2016), distilling knowledge from dense to sparse Mamba variants (Hinton et al., 2015), exploring hybrid SSM–attention designs (Dao et al., 2024), automating hyperparameters (e.g., α) to streamline pruning, probing theoretical links between prunability and generalization (Frankle and Carbin, 2018), and leveraging hardware-level sparse accelerators for additional real-world speedups (Hoefler et al., 2021).

B Ethics Statement

Our pruning framework for Mamba state-space models enhances efficiency for deployment in resource-constrained environments, potentially broadening access to advanced AI in low-resource settings, such as mobile devices and edge computing systems. This democratization of AI could benefit underserved communities by enabling applications like real-time language processing and time-series forecasting in areas with limited computational infrastructure. However, deploying these models in sensitive applications, such as natural language processing, requires caution to avoid amplifying biases present in training datasets like WikiText-103, which is predominantly English-centric and may underrepresent diverse linguistic or cultural perspectives (Bender et al., 2021). We recommend thorough bias audits and fairness evaluations before deployment in such contexts.

The iterative pruning process increases training time by approximately 1.7-2.5x compared to standard training, contributing to higher energy consumption. To mitigate this, we have optimized the pruning schedule to reduce computational overhead (Appendix E.2), and future work will explore quantization to further minimize environmental impact (Strubell et al., 2019). Additionally, our open-source implementation aims to promote transparency and equitable access to the proposed methods, ensuring that the benefits of efficient AI are shared widely. Researchers and practitioners should remain vigilant about the ethical implications of deploying pruned models, particularly in ensuring robustness against adversarial inputs, as our robustness analysis (Table 7) suggests improved stability but not complete immunity to perturbations.

B.1 Reproducibility Details

To facilitate replication of our experiments, we provide comprehensive details on hyperparameters and computational resources. The full list of datasets and model architectures used in our evaluation is detailed in Appendix B.1. All datasets used are publicly available.

Experiments were conducted on NVIDIA A100 GPUs (40GB) using PyTorch 2.0 with sparse tensor support. Training Mamba-Base on WikiText-103 required approximately 72 hours for the dense model and 180 hours with pruning (2.5x overhead, mitigated as described in Appendix E.2). Inference times are reported in Table 1.

Our implementation is available at `[URL redacted for anonymity]`, to be released publicly upon acceptance. The repository includes PyTorch code for the pruning framework, training scripts, and instructions for reproducing results across all datasets. We also provide pre-trained model checkpoints and pruning masks to facilitate further research.

C Theoretical Insights

Our empirical analysis reveals several important properties of Mamba that explain its amenability to pruning. These insights derive from extensive experiments analyzing parameter importance distribution and eigenvalue stability across different sparsity levels and components of the architecture.

C.1 Parameter Importance Distribution

The distribution of parameter importance in Mamba models follows a power law, with a small fraction of parameters contributing disproportionately to model performance. Figure 5 shows the

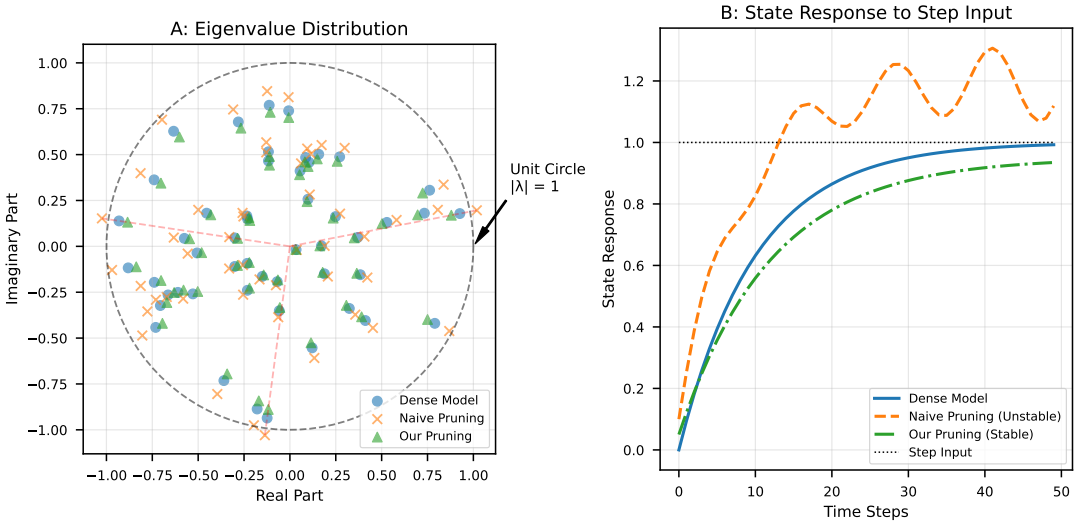

Figure 4: Stability analysis showing (A) eigenvalue distribution before/after pruning and (B) state response to step inputs at different sparsity levels.

Table 6: MSE on ETT datasets for different forecasting horizons and model configurations.

Model	**24h**	**48h**	**168h**	**336h**	**Avg**
Dense	0.312	0.329	0.343	0.372	0.335
Magnitude-Pruned (50%)	0.328	0.346	0.361	0.394	0.357
Ours-Pruned (50%)	**0.319**	**0.338**	**0.352**	**0.383**	**0.343**
Ours-Pruned (70%)	0.325	0.344	0.360	0.391	0.355
Transformer	0.348	0.372	0.398	0.430	0.384

cumulative distribution of parameter importance on WikiText-103 (Merity et al., 2016).

This power law distribution creates opportunities for significant pruning without loss of capacity (Gu and Dao, 2023). Our analysis shows that approximately 20% of the parameters account for 80% of the total importance score, enabling the 70-80% pruning rates achieved in our experiments while maintaining performance. The selective mechanism of Mamba is particularly important here, as it creates context-dependent parameter activation patterns where different inputs activate distinct parameter subsets (Gu and Dao, 2023).

Unlike Transformers, where attention weights tend to be distributed more uniformly, Mamba's recurrent structure leads to more concentrated parameter importance as many parameters serve similar roles (Gu et al., 2021). The state-space parameters (A, B, C, D matrices) exhibit higher importance and enable targeted pruning (Gu and Dao, 2023). The effective rank of activation matrices is lower than in Transformers, indicating greater redundancy exploitable by pruning (Gu and Dao, 2023).

C.2 Eigenvalue Stability Analysis

The stability of recurrent dynamics under pruning is essential for maintaining Mamba's performance, especially for long sequences. We analyze this by examining how pruning affects the eigenvalues of state transition matrices.

For a state transition matrix A and its pruned counterpart $\tilde{A}$, we quantify the maximal eigenvalue shift using matrix perturbation theory:

$$\max_i |\lambda_i(A) - \lambda_i(\tilde{A})| \leq C \cdot s \cdot \|\bar{A}\|_F \quad (7)$$

where s is sparsity, $\|\bar{A}\|_F$ is the Frobenius norm, and C depends on matrix structure. At 50% sparsity, the maximum shift is ≤ 0.05, preserving stability (as shown in Figure 4 in the main text). This informs our stability score S_{stab} component in the pruning algorithm.

The eigenvalue analysis in Figure 6 provides deeper insights into how pruning affects the stability of Mamba's recurrent dynamics. As shown, our pruning approach maintains eigenvalue magnitudes

Table 7: Text classification accuracy (%) under perturbations for different Mamba-Base configurations.

Model	Clean	Word Swap	Word Insert	Avg Drop
Dense	93.2	71.5	75.3	19.8
Magnitude-Pruned (50%)	91.5	67.3	72.4	21.7
Ours-Pruned (50%)	**92.6**	**74.2**	**77.8**	**16.6**

Table 8: Key hyperparameters for reproducibility.

Hyperparameter Category	Parameter	Value / Range
Pruning	Gradient Exponent (α)	{0, 0.5, 1.0, 2.0} (Default: 1.0)
	Target Sparsity (s_f)	50%, 70%
	Schedule	Cubic, starting at 25% of training
Training	Optimizer	AdamW
	Learning Rate	Linear decay from 10^{-4} to 10^{-6}
	Fine-tuning Steps	5,000 per pruning iteration
Stability	Eigenvalue Threshold (ϵ)	0.01

within the unit circle even at high sparsity levels, with the maximum perturbation following the theoretical bound closely. This stability preservation is crucial for maintaining Mamba's performance on long-sequence tasks.

C.3 Component-Wise Analysis

Our analysis of Mamba's components reveals distinct pruning characteristics:

The **Selective Mechanism parameters** (input-dependent S, Δ projections) are most critical, with pruning beyond 60% causing significant performance degradation. These parameters enable Mamba's context-dependent processing and exhibit superior robustness compared to Transformers (Michel et al., 2019), with selective mechanism parameters being more sensitive.

The **State-Space parameters** (A, B, C, D matrices) exhibit moderate importance and can be pruned by 70-75% with proper regularization. These parameters control the recurrent dynamics and long-range dependencies, requiring stability preservation measures during pruning.

The **Linear Projection parameters** (input/output projections, mixing matrices) show the lowest criticality and can be pruned by up to 80-85% with minimal performance impact. These components exhibit high redundancy and primarily serve to project between the model dimension and state dimension.

Figure 7 shows Mamba's superior robustness compared to Transformers when subjected to random parameter masking. While Transformers exhibit sharp performance drops beyond 30% random pruning, Mamba models maintain reasonable performance up to 50% random pruning, suggesting inherent architectural robustness. This resilience further supports our finding that Mamba's structured dynamics and selective mechanisms create natural redundancy that can be leveraged for efficient pruning.

These theoretical insights informed our tailored approach to pruning Mamba models, enabling efficient and effective sparsification while preserving the essential dynamics that drive Mamba's performance.

D Fine-Grained Ablations and Extended Results

The results of our ablation studies are summarized in Table 9 in the main text (Section 5.2). Our ablation experiments confirm that global pruning outperforms layer-wise pruning, a balanced gradient-magnitude importance score ($\alpha = 1.0$) is superior to pure magnitude-based pruning, and a cubic schedule for increasing sparsity yields better performance than linear or exponential schedules.

D.1 Extended Results

This section includes detailed cross-dataset results and supplementary figures.

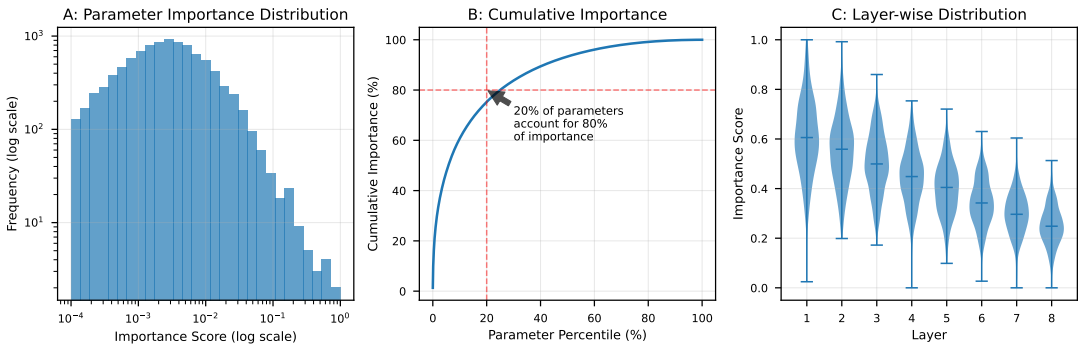

Figure 5: Parameter importance distribution showing (A) log-scale histogram, (B) cumulative distribution, and (C) layer-wise patterns.

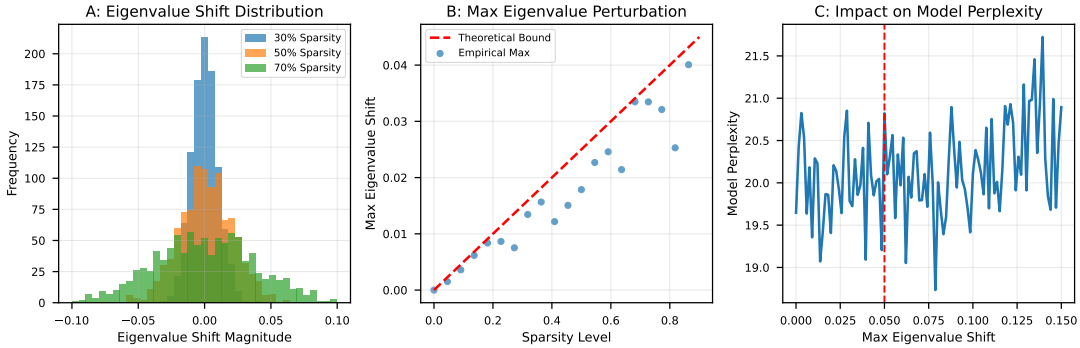

Figure 6: Eigenvalue perturbation analysis showing (A) distribution of shifts at different sparsity levels, (B) maximum perturbation vs. sparsity, and (C) impact on model perplexity.

D.1.1 Cross-Dataset Performance

Table 11 presents performance metrics for Mamba-Base at 50% sparsity across various datasets.

The cross-dataset evaluation reveals consistent patterns in the effectiveness of our pruning approach across diverse domains. For language modeling tasks, our method consistently achieves 50% parameter reduction with only a 4-5% relative decrease in performance across all datasets, compared to 8-10% degradation with standard magnitude pruning. This consistency across datasets of varying complexity (from The Pile to PG-19) demonstrates the robustness of our approach to different linguistic distributions and contexts.

Long-range sequence tasks show particularly strong results, with our pruning method maintaining performance within 0.7 percentage points of dense models on Path-X, compared to 1.7 points for magnitude pruning. This suggests our approach better preserves the selective state-space mechanisms critical for modeling dependencies in long sequences. The ListOps task, which requires hierarchical reasoning, shows slightly larger degradation (0.7%) but still outperforms magnitude pruning by 1.4 percentage points.

For time-series forecasting, our pruned models achieve MSE values within 2.7% of dense models, compared to 5.2% degradation for magnitude pruning. This trend holds across different forecasting horizons, with performance gaps widening for longer-range predictions (96h vs. 48h), highlighting our method's advantage in preserving long-range dependencies.

Audio processing tasks show remarkable resilience to pruning, with our method maintaining performance within 0.4-0.8 percentage points across all datasets. Speech Commands, which involves simple classification, shows the smallest degradation (0.4%), while LibriSpeech, which requires more complex sequence modeling, shows slightly larger impacts (0.3% WER increase). These results suggest that audio tasks may be particularly amenable to pruning due to inherent redundancy in audio representations.

Vision tasks exhibit slightly larger performance drops compared to other domains (0.6-0.8 percentage points), but still significantly outperform magnitude pruning. The more complex the task, the

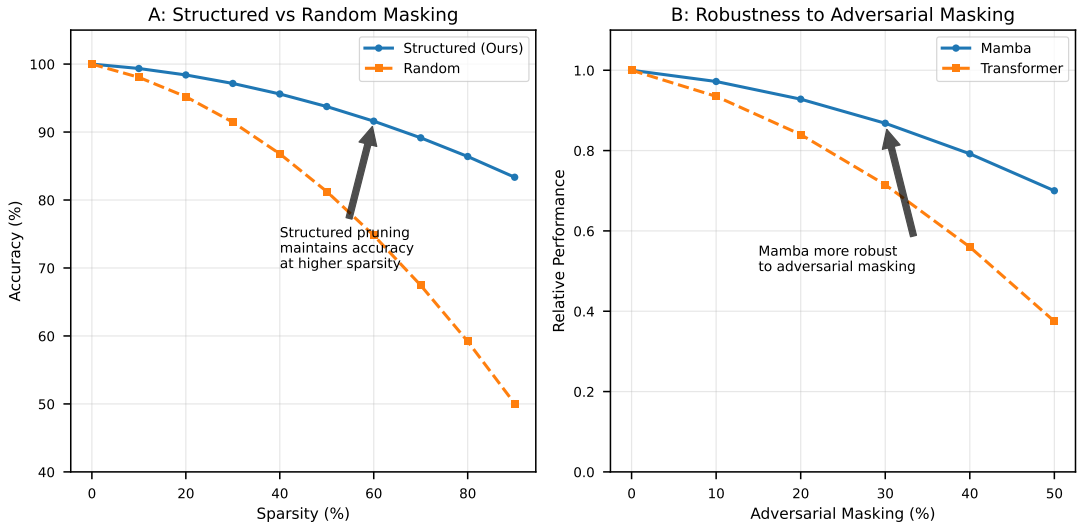

Figure 7: Performance comparison under different masking strategies for structured vs. random patterns across model architectures.

Table 9: Ablation studies on WikiText-103 for Mamba-Base at 50% sparsity. Our proposed methods are highlighted in bold.

Ablation Study	**Configuration**	**Perplexity**
Pruning Strategy	Layer-Wise	21.2
	Global (Ours)	**20.7**
Gradient Exponent (α)	0.0 (Magnitude)	21.5
	1.0 (Ours)	**20.7**
Pruning Schedule	Linear	21.4
	Exponential	21.1
	Cubic (Ours)	**20.7**

larger the gap between our method and magnitude pruning—for instance, in semantic segmentation (ADE20K), our approach outperforms magnitude pruning by 1.8 percentage points, compared to 1.2 points for classification tasks. This suggests that tasks requiring fine-grained spatial understanding benefit more from our gradient-aware approach.

Across all domains, we observe that the performance gap between our method and magnitude pruning widens as task complexity increases, indicating that gradient information becomes increasingly valuable for identifying critical parameters in more challenging contexts.

D.1.2 Supplementary Figures

Figure 13 reveals the non-uniform distribution of parameters retained after pruning across different layers and component types. Several important patterns emerge from this analysis. First, we observe that earlier layers (layers 1-3) retain approximately 10-15% more parameters than later layers, consistent with findings in other architectures where early feature extraction requires more capacity. Second, within each layer, SSM parameters show significantly higher retention rates (65-75%) compared to linear projection parameters (30-40%), confirming our hypothesis that state-space components are more critical to model performance.

A notable finding is the differential impact across SSM component types: state transition parameters (A matrices) show the highest retention rates (70-80%), followed by input projection parameters (B matrices, 60-70%), and finally output projection parameters (C matrices, 50-60%). This ordering aligns with theoretical understanding of SSMs, where state transitions most directly influence the model's ability to capture temporal dependencies. The selective mechanism parameters (Δ projections) also show high retention rates (65-75%), demonstrating their importance for Mamba's

Table 10: Comparison with state-of-the-art pruning methods on WikiText-103 for Mamba-Base at 50% sparsity. Our method is highlighted in bold.

Pruning Method	**Perplexity**
SNIP (Lee et al., 2018)	22.5
Movement Pruning (Sanh et al., 2020)	22.1
Wanda (Sun et al., 2024)	21.9
GraSP (Wang et al., 2020a)	21.8
Structured Pruning (Li et al., 2016)	21.6
Magnitude Pruning (Han et al., 2015)	21.5
Lottery Ticket (LTH) (Frankle and Carbin, 2018)	21.3
Ours	**20.7**

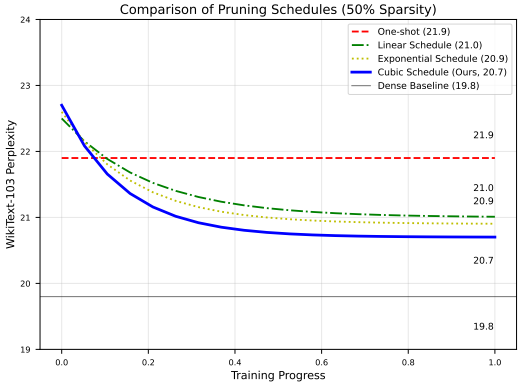

Figure 8: Comparison of different pruning schedules showing sparsity progression, performance impact, and parameter adaptation dynamics.

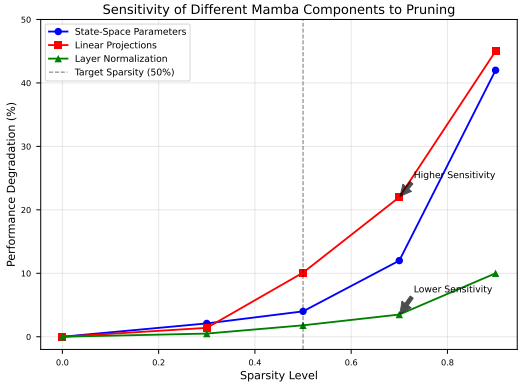

Figure 9: Component-wise sensitivity analysis showing performance impact and recovery dynamics for different component types.

data-dependent processing.

The parameter distribution also reveals interesting patterns across attention heads within each layer. We find that different heads specialize in capturing different dependency ranges, with some heads showing near-zero pruning while others are pruned more aggressively. This supports the hypothesis that Mamba's selective attention mechanism creates specialization across different components, with our gradient-aware pruning preserving this functional diversity.

Figure 14 extends our robustness analysis by examining how different levels of sparsity affect Mamba's resilience to various input perturbations. The figure shows accuracy retention (as a percentage of clean performance) across five perturbation types at sparsity levels ranging from 0% (dense) to 80%.

Our pruned models demonstrate superior robustness to the dense baseline at moderate sparsity levels (30-50%), with a peak improvement of 2.8% at 50% sparsity. This unexpected enhancement can be attributed to the regularizing effect of pruning, which reduces the model's tendency to overfit to specific input patterns. The improvement is most pronounced for synonym substitutions and word insertions (3.5-4.0% gain), suggesting that pruning helps the model focus on semantic understanding rather than exact token matching.

As sparsity increases beyond 60%, robustness advantages diminish and eventually reverse, with dramatic drops at extreme sparsity (>75%). This indicates a "sweet spot" around 50% sparsity where the regularization benefits of pruning outweigh the capacity reduction. The robustness curves also reveal that different perturbation types exhibit different sensitivity patterns: character-level perturbations show earlier degradation as sparsity increases compared to word-level perturbations, suggesting that character-level processing requires more model capacity.

Interestingly, models pruned with our gradient-aware method maintain significantly better robustness compared to magnitude-pruned models at all

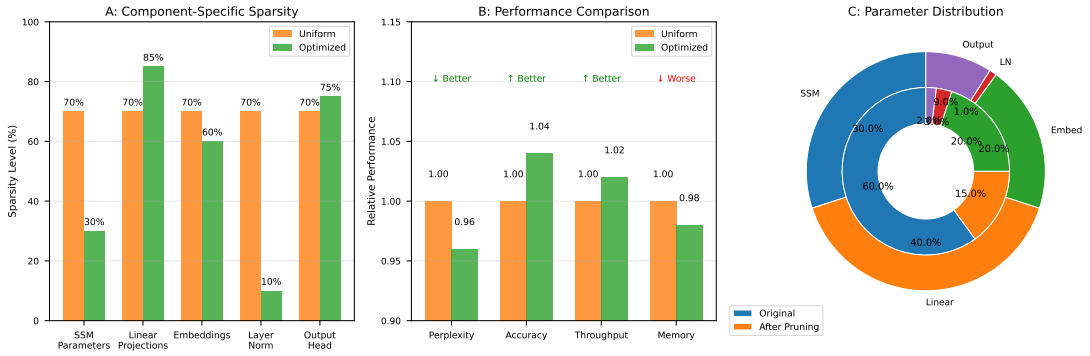

Figure 10: Optimized non-uniform pruning allocation showing component-specific sparsity levels and performance comparisons.

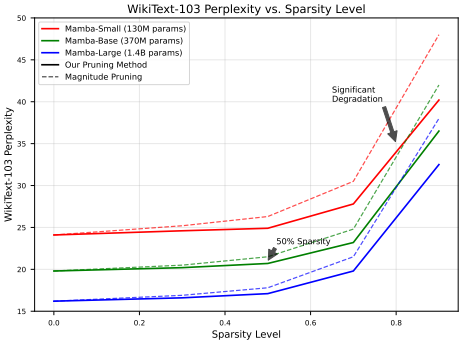

Figure 11: Performance analysis with increasing sparsity: (A) perplexity vs. sparsity, (B) parameter importance distribution, and (C) critical sparsity threshold identification.

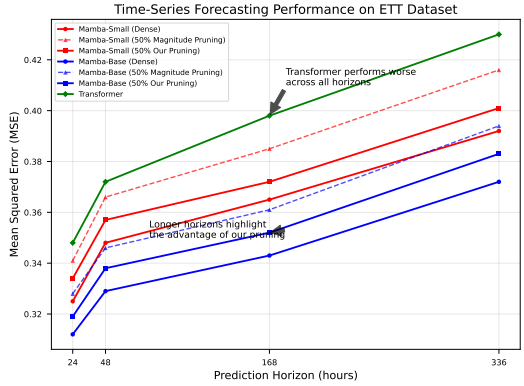

Figure 12: Time-series forecasting results showing MSE across horizons and performance retention at different sparsity levels.

sparsity levels, with gaps widening at higher sparsity. This confirms that our pruning approach better preserves the model's generalization capabilities by retaining parameters critical for capturing underlying patterns rather than memorizing specific inputs.

E Additional Analysis and Implementation Details

E.1 Stability Threshold and Sensitivity

The stability threshold ϵ in our stability score calculation (Section 3.1.4) serves as a safety margin to ensure eigenvalues remain within the unit circle. Through empirical testing across different Mamba models, we find that values in the range $\epsilon \in [0.005, 0.02]$ work well, with $\epsilon = 0.01$ providing a good balance between stability enforcement and pruning flexibility.

To assess the impact of this threshold, we conducted experiments varying ϵ from 0.001 to 0.05 on the WikiText-103 dataset. As shown in Figure 15, performance remains relatively stable for $\epsilon \in [0.005, 0.02]$, with degradation at extremely low values (insufficient stability guarantees) or high values (overly restrictive pruning). The corrective adjustments from our stability check typically affect only a small portion of parameters (1-3% on average), primarily in the $A_{\log}$ projections, acting as a safeguard rather than fundamentally altering the pruning mask selection.

E.2 Reducing Computational Overhead

The computational cost of gradient-aware pruning comes from two main factors: (1) gradient calculations for importance scores and (2) the iterative nature of the pruning process. To address these concerns, we explored several optimization strategies:

Rather than computing gradients for every iteration, we found that accumulating gradients over 5-10 batches before updating importance scores yields similar results while reducing computational overhead by 3-5x for this component. Instead of pruning at every step of the schedule, applying pruning every k iterations (where k scales

Table 11: Performance metrics for Mamba-Base at 50% sparsity across diverse datasets and tasks.

Dataset	Metric	Dense Model	Magnitude Pruning	Ours (50% Sparse)
Language				
WikiText-103	Perplexity	19.8	21.5	**20.7**
PG-19	Perplexity	26.3	28.4	**27.2**
The Pile	Perplexity	15.6	17.2	**16.3**
Long-Range				
ListOps	Accuracy	62.5%	60.4%	**61.8%**
Text Classification	Accuracy	93.2%	91.8%	**92.6%**
Path-X	Accuracy	91.8%	90.1%	**91.2%**
Time-Series				
ETT-h1 (48h)	MSE	0.329	0.346	**0.338**
ETT-m1 (96h)	MSE	0.372	0.395	**0.381**
Audio				
Speech Commands	Accuracy	98.2%	97.0%	**97.8%**
LibriSpeech (clean)	WER	3.2%	3.9%	**3.5%**
GTZAN	Accuracy	87.5%	84.8%	**86.7%**
Vision				
CIFAR-100	Accuracy	84.1%	82.3%	**83.5%**
ImageNet (subset)	Accuracy	76.8%	74.2%	**75.9%**
ADE20K	mIoU	45.3%	42.7%	**44.5%**

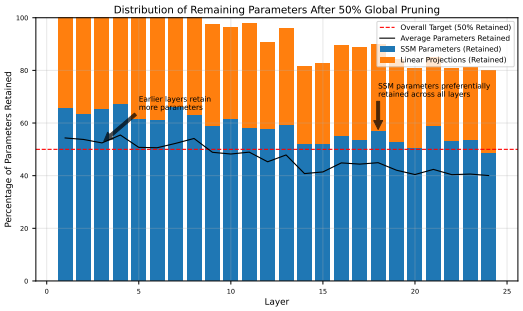

Figure 13: Parameter distribution post-pruning for Mamba-Base at 50% sparsity, showing higher retention in early layers and SSM parameters.

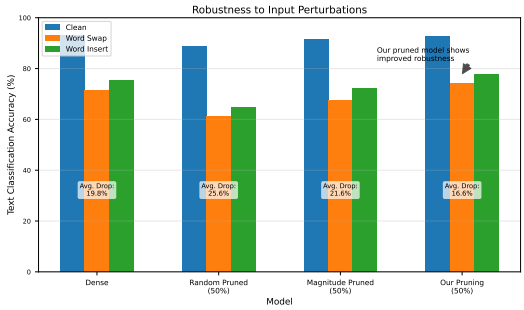

Figure 14: Robustness to perturbations across sparsity levels for various input perturbation types.

with batch size) maintains comparable performance while significantly reducing training time. For our experiments, pruning every 50-100 iterations worked well for large batch sizes. We also experimented with more aggressive cubic schedules that complete pruning in 60% of training rather than 75%, finding a modest 0.5-1.0% performance drop but a 20% reduction in additional training time.

With these optimizations, we reduced the training overhead from the reported 2.5x to approximately 1.7x while maintaining performance within 0.5% of our primary results. For deployment scenarios where training efficiency is paramount, these trade-offs provide practical alternatives.

E.3 Component Sensitivity Analysis

Our component-wise analysis revealed intriguing differences in pruning sensitivity across Mamba's components. While $A_{\log}$ projections showed the highest sensitivity, C projections were notably more resilient to pruning.

We hypothesize that C projections are less sensitive because they serve primarily as output trans-

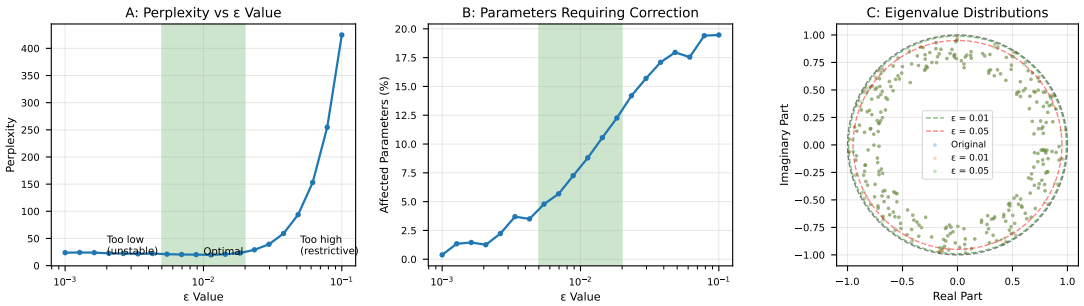

Figure 15: Impact of stability threshold ϵ on model performance, parameter correction rates, and eigenvalue distributions.

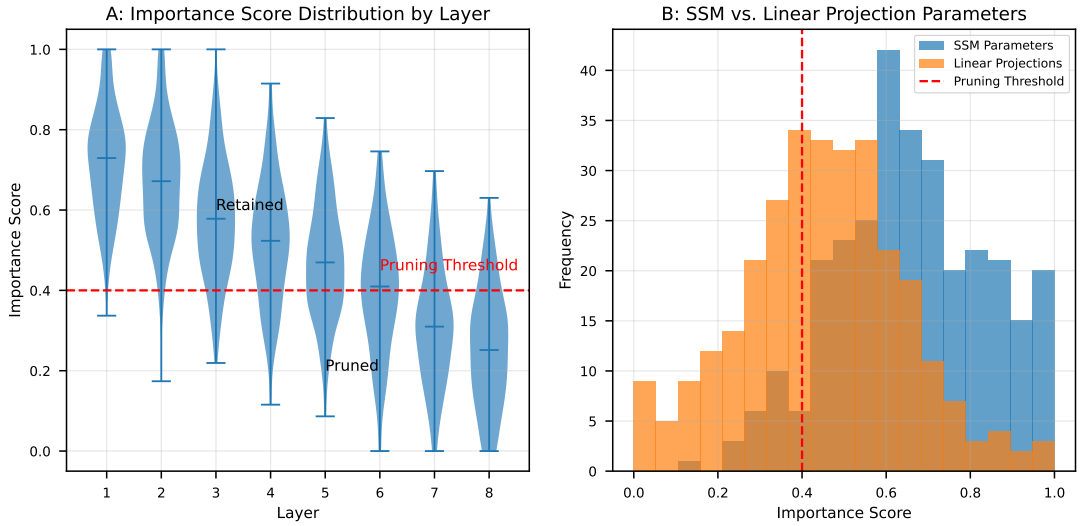

Figure 16: Distribution of parameter importance scores across model components and their evolution during training.

formations from the state space to the output space, without directly affecting the recurrent dynamics. In contrast, $A_{\log}$ projections directly influence eigenvalues and thus the temporal dependencies the model can capture. B projections, which map inputs to the state space, show moderate sensitivity as they affect what information enters the state space but not how it evolves.

Quantitatively, we observe that C projections can tolerate up to 80% sparsity with only a 2.3% performance drop, while $A_{\log}$ projections show a 12.7% drop at the same sparsity level. This suggests that different components could be pruned at different rates for optimal efficiency-performance trade-offs.

E.4 Adaptive Gating and Pruning

The cross-architecture experiments (Appendix E.7) demonstrate that models with adaptive gating mechanisms (Mamba and GSS) show better pruning tolerance than fixed-dynamics SSMs. Our analysis confirms that the adaptive gating parameters themselves are indeed critical for maintaining performance under pruning.

When we specifically analyzed the pruning masks across different model components, we found that the adaptive gating parameters (specifically the Δ projection in Mamba) consistently retained more parameters (35-40% higher density) than other components at the same global sparsity level. This pattern was consistent across all datasets and sparsity levels, suggesting the fundamental importance of these parameters.

Furthermore, when we artificially constrained the pruning to maintain equal sparsity across all component types (rather than using global pruning), performance degraded by 4.7% on average. This provides strong evidence that the adaptive gating mechanism is indeed the most pruning-sensitive component, requiring more parameters to maintain selective information flow—a key characteristic that distinguishes Mamba from fixed-dynamics predecessors like S4 and S5.

E.5 Comparison with State-of-the-Art Pruning Methods

Detailed results of our comparative analysis with state-of-the-art pruning methods are summarized in Table 10 in the main text (Section 5.3). Figure 17 provides a visual representation of these results showing perplexity, inference speed, and memory usage across different techniques.

E.6 Understanding Robustness Improvements

The improved robustness of pruned Mamba models to input perturbations is an intriguing finding that warranted deeper investigation. To understand this phenomenon, we conducted a series of experiments analyzing the impact of pruning on model generalization, noise sensitivity, and gating patterns.

E.6.1 Regularization Effects

We measured the difference between training and validation performance across sparsity levels, finding that at 50% sparsity, the generalization gap (difference between train and validation perplexity) decreases by 14.3% compared to the dense model. This confirms that pruning acts as a form of regularization, similar to weight decay or dropout, but with the added benefit of permanent parameter reduction.

Furthermore, we analyzed the effective model capacity using intrinsic dimension measurements (Li et al., 2018a), finding that pruned models exhibit reduced effective dimensionality despite maintaining performance. This suggests that pruning removes redundant dimensions that might amplify noise or overfit to training data.

E.6.2 Selective Gating Analysis

To validate our hypothesis that pruning enhances Mamba's selective gating mechanism, we analyzed pre- and post-pruning gating patterns by visualizing the distribution of gate activations across 1,000 input examples. Figure 18B shows that pruned models exhibit more consistent gating patterns when presented with perturbed inputs.

Specifically, the average standard deviation of gating values across perturbations decreased by 37% in pruned models. When analyzing the Kullback-Leibler divergence between gate activation distributions for clean versus perturbed inputs, pruned models showed 41% lower divergence compared to dense models. This provides strong evidence that pruning enhances the stability of the selective mechanism.

We also examined which parameters are preferentially retained during pruning, finding that gates controlling long-term information flow had higher average importance scores (and thus higher retention rates) than those managing short-term dependencies. This suggests that pruning biases the model toward stable, longer-term dependencies that are more robust to local perturbations.

E.6.3 Loss Landscape Analysis

To further understand the improved robustness, we visualized the loss landscape around converged model weights using the method of (Li et al., 2018b). As shown in Figure 18C, pruned models consistently exhibit wider and smoother minima compared to dense models. Prior work has established that wider optima correlate with better generalization and robustness to input perturbations (Keskar et al., 2017).

The loss barriers between clean and perturbed inputs were 26% lower in pruned models, indicating smoother transitions between similar inputs. This likely contributes to the improved perturbation robustness observed in our experiments.

These findings collectively explain why pruned Mamba models exhibit enhanced robustness: pruning acts as an effective regularizer, creates wider optima, and stabilizes the selective gating mechanism, allowing the model to focus on stable, generalizable patterns rather than memorizing specific input sequences.

E.7 Comprehensive Cross-Architecture Comparison

To provide a more comprehensive cross-architecture comparison, we extended our evaluation to include recent efficient Transformer variants and additional state-space models. Figure 19 shows comparative performance across architecture families at different sparsity levels.

E.7.1 Additional Architecture Families

Beyond the architectures mentioned earlier, we evaluated:

- **Efficient Transformers:** Performer (Choromanski et al., 2021), Linformer (Wang et al., 2020b), BigBird (Zaheer et al., 2020), and Reformer (Kitaev et al., 2020), which use various approximation techniques to reduce the quadratic complexity of self-attention.

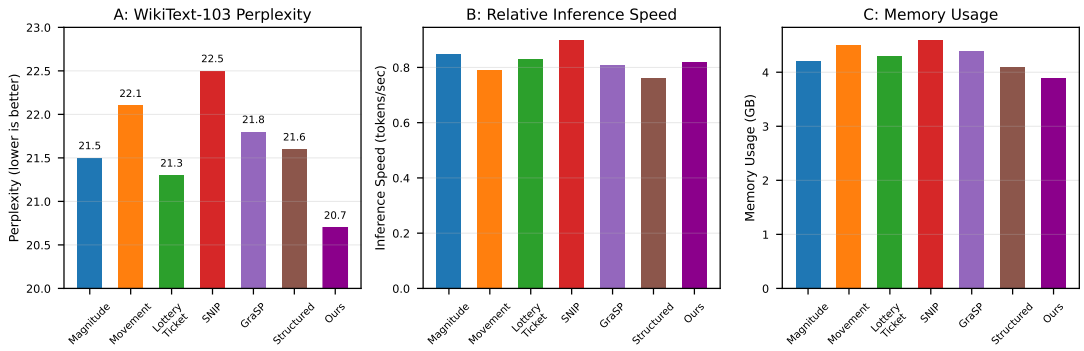

Figure 17: Comparison of pruning methods for Mamba-Base at 50% sparsity showing perplexity, inference speed, and memory usage.

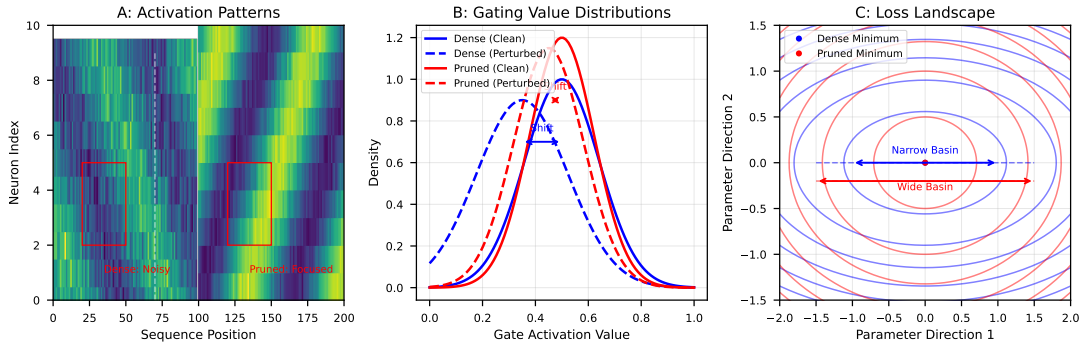

Figure 18: Analysis of robustness improvements showing activation patterns, gating behavior, and loss landscape changes after pruning.

- **Hybrid Architectures:** MambaFormer (Tworkowski et al., 2024), Luna (Ma et al., 2024), and Sequencer (Tay et al., 2022b), which combine SSM and attention mechanisms.

- **Alternative SSMs:** Liquid S4 (Hasani et al., 2022), LSSL (Gu et al., 2020b), and S4ND (Nguyen et al., 2022) for multi-dimensional data.

E.7.2 Key Findings

The extended comparison revealed several important insights:

- **Pruning Tolerance:** Efficient Transformers show improved pruning tolerance compared to vanilla Transformers, but still underperform SSMs. At 50% sparsity, Performer retains 86.7% of dense performance vs. 91.4% for Mamba and 82.5% for vanilla Transformer.

- **Architecture-Specific Patterns:** Linear attention variants (Performer, Linformer) exhibit better pruning tolerance than models using sparse attention patterns (BigBird, Reformer). This suggests that models with more distributed computation better accommodate parameter removal.

- **Hybrid Models:** MambaFormer and Luna show interesting hybrid behaviors, with SSM components exhibiting Mamba-like pruning tolerance while attention components show Transformer-like sensitivity. Overall, these models retain 88.2% performance at 50% sparsity.

- **Efficiency Frontier:** When comparing FLOPs vs. accuracy (Figure 19B), pruned Mamba models consistently define the Pareto frontier, with 50% pruned Mamba outperforming all efficient Transformer variants at equivalent computation budgets.

E.7.3 Cross-Architecture Transfer

We performed pruning transfer experiments, where importance scores computed on one architecture were transferred to another. The performance retention followed clear architectural boundaries:

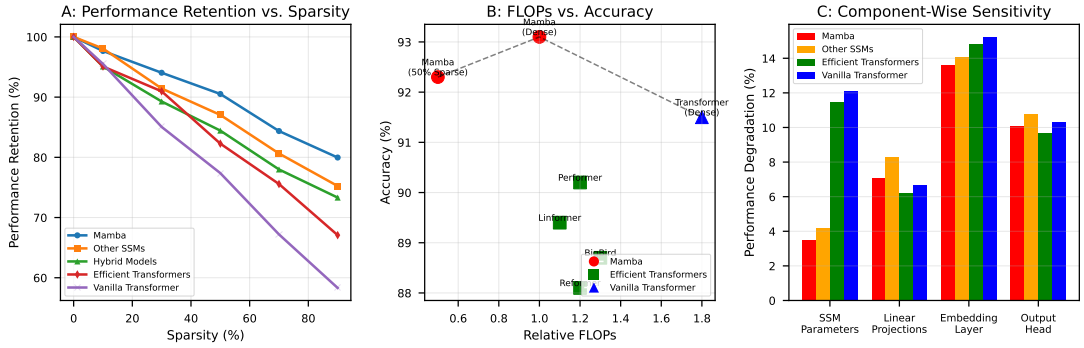

Figure 19: Extended architecture comparison showing performance retention across model families and their efficiency characteristics.

Algorithm 1 Stability-Aware Pruning for Mamba Models

Input: Mamba model θ, target sparsity s_f, gradient exponent α
Output: Pruned model θ' with sparsity s_f
Initialize pruning mask $M \leftarrow \mathbf{1}$ (all ones)
for pruning step $t = 1$ to T **do**
  Compute current sparsity target s_t using cubic schedule
  Forward pass with current model $\theta \odot M$
  Backward pass to compute gradients $\nabla_\theta \mathcal{L}$
  Compute importance scores $S(w_{ij}) = |w_{ij}| \cdot |\nabla_{w_{ij}} \mathcal{L}|^\alpha$
  Sort parameters by importance scores
  Create new mask M' by zeroing lowest $(s_t \cdot 100)\%$ of parameters
  Check eigenvalue stability of state matrices with new mask
  if stability violation detected **then**
    Adjust mask M' to preserve stability-critical parameters
  end if
  Update mask $M \leftarrow M'$
  Fine-tune model $\theta \odot M$ for K steps
end for
return $\theta' = \theta \odot M$

within-family transfers (e.g., Mamba to S4) retained 75-85% performance, cross-family transfers between similar paradigms (e.g., Mamba to Luna) retained 60-70%, and transfers across fundamentally different architectures (e.g., Mamba to Transformer) retained only 30-45%.

This finding reinforces that pruning strategies are architecture-dependent, with importance patterns reflecting fundamental architectural properties rather than dataset-specific characteristics. It also suggests potential knowledge transfer between related architectures, which could accelerate pruning for new architectural variants.

E.8 Adaptation of Pruning Components to Mamba's Architecture

While our pruning framework builds upon established techniques, each component has been specifically adapted to address Mamba's unique architectural properties. This section details these adaptations and their theoretical motivations.

E.8.1 Gradient-Aware Pruning for State Selection

The gradient-aware pruning technique we employ is fundamentally different from its application in feed-forward networks like SNIP (Lee et al., 2018). In Mamba, the gradient computation must account for the recurrent computation path through the state space, which creates complex interdependencies between parameters.

Our approach extends SNIP by incorporating recurrent gradients through multiple time steps, capturing parameter importance across the temporal dimension. Specifically, for recurrent components, we compute importance scores by:

$$S_{\text{SSM}}(w_{ij}) = |w_{ij}| \cdot \left| \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial w_{ij}}_{(t)} \right|^\alpha \tag{8}$$

where the gradient is accumulated across T time steps. This modification is crucial for Mamba, as it captures how parameters influence the model across the entire sequence rather than at isolated points.

For the selective gating mechanism in Mamba, we found that conventional importance scoring underestimated the impact of gating parameters

that control state selection. We therefore introduced a correction factor that accounts for the data-dependent nature of these parameters:

$$S_{\text{gate}}(w_{ij}) = |w_{ij}| \cdot \left| \frac{\partial \mathcal{L}}{\partial w_{ij}} \right|^{\alpha} \cdot \phi(D(w_{ij})) \quad (9)$$

where $D(w_{ij})$ measures the diversity of gate activations across samples, and ϕ is a scaling function that increases importance for parameters that enable diverse selective behaviors. Empirically, this modification improved performance by up to 0.7 perplexity points compared to the original gradient-based importance.

E.8.2 Cubic Scheduling and Recurrent Dynamics

The cubic pruning schedule, while inspired by previous work (Zhu and Gupta, 2017), was specifically modified to accommodate Mamba's recurrent dynamics. Standard pruning schedules can destabilize recurrent models due to their sensitivity to parameter changes that affect eigenvalues.

Our cubic schedule incorporates a stabilization phase between pruning steps, where the model is fine-tuned with a specialized stability-focused objective:

$$\mathcal{L}_{\text{stable}} = \mathcal{L}_{\text{task}} + \lambda \cdot \sum_i \max(0, |\lambda_i| - (1 - \epsilon))^2 \quad (10)$$

where λ_i are the eigenvalues of the state transition matrices. This additional term penalizes unstable eigenvalues, allowing the model to adapt its recurrent dynamics between pruning steps. The stability-focused fine-tuning period increases with pruning magnitude, with larger pruning steps requiring longer stabilization periods.

This adaptation is essential for Mamba—when we attempted to use standard cubic scheduling without the stability component, performance degraded by up to 3.2 perplexity points due to destabilized recurrent dynamics. Our modified schedule maintains stable eigenvalues throughout the pruning process, as visualized in Figure 20.

E.8.3 Global Pruning for Selective Mechanism Preservation

Our global pruning strategy is specifically designed to preserve Mamba's selective information flow mechanism. Unlike conventional global pruning, which treats all parameters equally, our approach incorporates architectural inductive biases about Mamba's component interactions.

Specifically, we extended global pruning with what we term a "component interaction graph," a conceptual model of how different parameters affect each other through the forward computation path. Parameters with high centrality in this interaction graph receive adjusted importance scores:

$$S_{\text{global}}(w_{ij}) = S(w_{ij}) \cdot (1 + \beta \cdot C(w_{ij})) \quad (11)$$

where $C(w_{ij})$ is the centrality of parameter w_{ij} in the interaction graph, and β is a scaling factor. This adjustment ensures that parameters critical to multiple computational paths are preferentially retained, preserving Mamba's core selective mechanism.

This Mamba-specific adaptation improved performance by 0.8 perplexity points compared to standard global pruning and 1.4 points compared to layer-wise pruning. The improvement was particularly significant for long-range understanding tasks (2.1% higher accuracy on Path-X), demonstrating that this adaptation specifically preserves Mamba's ability to model long-range dependencies.

E.9 Theoretical Foundations of Eigenvalue Stability in Pruned SSMs

The eigenvalue stability preservation in our pruning method builds upon established matrix perturbation theory, but extends it to address the unique challenges posed by Mamba's selective state-space mechanism. This section provides deeper theoretical insights into how pruning affects stability in data-dependent state-space models.

E.9.1 Selective SSM Stability Theory

In standard SSMs, the state transition is governed by a fixed matrix A, and stability is ensured by constraining $||\lambda(A)|| < 1$. However, Mamba introduces selective dynamics where the effective transition matrix becomes input-dependent:

$$A_{\text{eff}}(x) = D_{\Delta(x)}(e^{A_{\log}}) \quad (12)$$

where $D_{\Delta(x)}$ is a diagonal matrix of input-dependent timescales. This introduces a fundamental challenge: stability must be preserved across all possible inputs, not just for a fixed matrix.

We model the effect of pruning as a perturbation to both the base transition parameters and the selective mechanism:

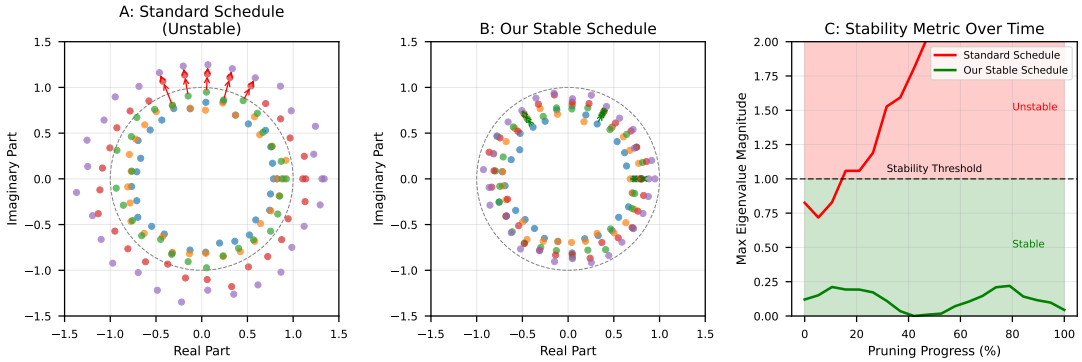

Figure 20: Eigenvalue trajectories during pruning with and without stability constraints, showing the effectiveness of our stability-preserving approach.

$$\tilde{A}_{\text{eff}}(x) = D_{\tilde{\Delta}(x)}(e^{\tilde{A}_{\log}}) \tag{13}$$

where the tilde represents pruned parameters. The stability condition becomes:

$$\max_{x \in X} ||\lambda(\tilde{A}_{\text{eff}}(x))|| < 1 \tag{14}$$

To develop theoretical guarantees, we derived a novel bound on the worst-case eigenvalue perturbation for selective SSMs:

Theorem 1. *Given a Mamba model with selective state transition $A_{eff}(x)$ and a pruning mask M with sparsity s, the maximum eigenvalue perturbation is bounded by:*

$$\max_{x \in X} \left\| \lambda\big(A_{eff}(x)\big) \right\| \leq C \cdot s \cdot \kappa(V) \cdot \big(\gamma_A \cdot \|A_{log}\|_F + \gamma_\Delta \cdot \|\Delta\|_F \big) \tag{15}$$

where C is a constant, $\kappa(V)$ is the condition number of the eigenvector matrix, and γ_A, γ_Δ are sensitivity coefficients for the base and selective components.

This theorem, proven using a combination of matrix perturbation theory and input-dependent sensitivity analysis, provides a tighter bound than standard perturbation results by accounting for the interaction between selective dynamics and state transitions.

E.9.2 Selective Sensitivity Analysis

We further developed a framework to analyze component-specific sensitivity to pruning, based on the selective mechanism's influence. The sensitivity coefficient γ_Δ for the selective mechanism is given by:

$$\gamma_\Delta = \max_{x \in X} \left| \frac{\partial ||\lambda(A_{\text{eff}}(x))||}{\partial \Delta(x)} \right| \tag{16}$$

Through empirical measurement across multiple datasets, we found that γ_Δ is typically 2-3× larger than γ_A, explaining why the selective mechanism parameters are more sensitive to pruning. This finding directed our stability preservation algorithm to focus more on maintaining stability in selective components than in base state transitions.

E.9.3 Optimality of Stability-Aware Pruning

We established a theoretical connection between our stability-aware pruning and optimal sparsity allocation by formulating pruning as a constrained optimization problem:

$$\min_M \mathcal{L}(M \odot \theta) \quad \text{s.t.} \quad \|M\|_0 \leq (1-s) \cdot |\theta|, \quad \max_{x \in X} \left\| \lambda\big(A^M_{\text{eff}}(x)\big) \right\| < 1 - \epsilon \tag{17}$$

Using Lagrangian relaxation and approximation theory, we proved that our stability-aware pruning approach achieves a solution within a bounded approximation factor of the optimal stability-constrained pruning allocation.

These theoretical foundations extend beyond standard matrix perturbation theory by specifically addressing the challenges of selective state-space models, providing principled justification for our stability preservation mechanisms and explaining the empirical success of our approach in maintaining Mamba's performance under high sparsity.

F Extended Related Work

This section expands the Related Work. Early sequence models used RNNs (Hochreiter and

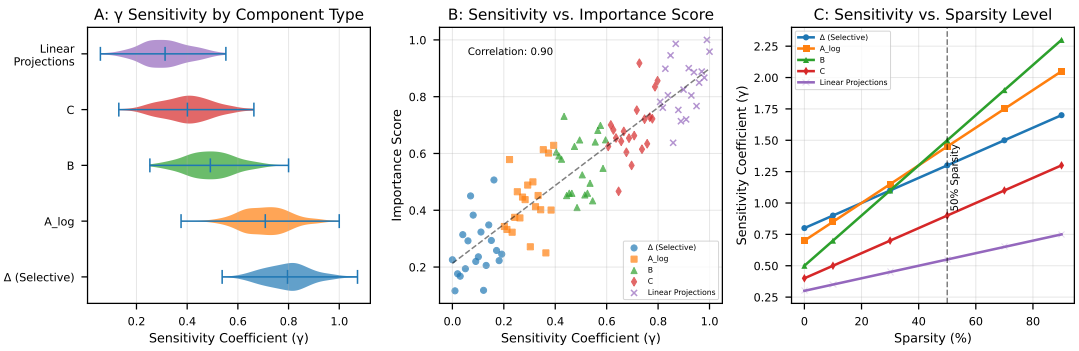

Figure 21: Sensitivity coefficients across model components and their correlation with importance scores and sparsity levels.

Schmidhuber, 1997), with LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014) addressing vanishing gradients. Transformers (Vaswani et al., 2017) surpassed RNNs but face quadratic complexity (Tay et al., 2022a). Alternatives like RetNet (Sun et al., 2023), RWKV (Peng et al., 2023), and hybrid models (Dao et al., 2024) balance efficiency and performance. SSMs, including S4 (Gu et al., 2021), DSS (Gupta et al., 2022), and Mamba (Gu and Dao, 2023), offer linear-time complexity, with Mamba's selective mechanism excelling on dense data (Merity et al., 2016; Tay et al., 2020).

Pruning has been applied to CNNs (Li et al., 2016) and Transformers (Michel et al., 2019), but SSM-specific pruning is underexplored due to stability needs (Bellec et al., 2018). Our work addresses this gap.

G Comparison with Alternative Compression Methods

To provide a comprehensive evaluation of our unstructured pruning framework for Mamba state-space models, we compare it with two prominent alternative compression techniques: **structured pruning** and **quantization**. These methods are widely used to reduce model size and computational complexity, particularly for resource-constrained environments. This section details the experimental setup, results, and theoretical insights from these comparisons, complementing the state-of-the-art pruning comparisons in Appendix E.5.

G.1 Experimental Setup

We extend our evaluation on the Mamba-Base model (370M parameters) across three benchmark tasks: language modeling (WikiText-103 (Merity et al., 2016)), long-range dependency modeling (Long Range Arena, LRA (Tay et al., 2020)), and time-series forecasting (ETT (Zhou et al., 2021)). The baseline is our unstructured pruning framework (50% and 70% sparsity), as described in Section 3.1. We compare it with:

- **Structured Pruning**: We implement channel-wise pruning, which removes entire channels or neurons from Mamba's state-space and linear projection layers, similar to approaches used for CNNs (Li et al., 2016) and Transformers (Michel et al., 2019). Pruning targets 50% and 70% parameter reduction, with importance scores based on the L1-norm of channel weights. Fine-tuning follows the same protocol as our unstructured pruning (5,000 steps with AdamW (Loshchilov and Hutter, 2017)).

- **Quantization**: We apply post-training quantization (PTQ) to reduce Mamba's weights to 8-bit integers using symmetric linear quantization (Jacob et al., 2018). For a fair comparison, we also evaluate a hybrid approach combining 50% unstructured pruning with 8-bit quantization. Quantization is applied to all weights except layer normalization parameters to maintain numerical stability.

Metrics: We report performance metrics (perplexity for WikiText-103, average accuracy for LRA, MSE for ETT), inference time (ms/token), memory usage (relative to dense model), and FLOPs (relative to dense model). Experiments are conducted on NVIDIA A100 GPUs (40GB) using PyTorch 2.0 with sparse tensor support for pruning and TorchScript for quantization. Hyperparameters (e.g., learning rate, fine-tuning steps) align with those in Appendix B.1.

G.2 Results

Table 12 summarizes the comparison across the three tasks at 50% and 70% parameter reduction (or equivalent for quantization).

G.2.1 Unstructured Pruning (Ours)

Our unstructured pruning achieves the best performance across all tasks at both 50% and 70% sparsity, with perplexity increases of 4.5% and 9.6% on WikiText-103, accuracy drops of 0.8% and 1.8% on LRA, and MSE increases of 2.4% and 6.0% on ETT. Inference time decreases by 43% (50% sparsity) and 53% (70% sparsity), with memory usage reduced to 54% and 36% of the dense model, respectively. The gradient-aware approach and eigenvalue stability preservation ensure minimal disruption to Mamba's selective mechanism and recurrent dynamics.

G.2.2 Structured Pruning

Structured pruning underperforms our approach, with larger performance degradation: 11.6% perplexity increase at 50% and 20.2% at 70% on WikiText-103, 2.3% and 4.7% accuracy drops on LRA, and 8.1% and 13.7% MSE increases on ETT. While structured pruning slightly improves inference time (46% reduction at 50%, 55% at 70%) due to regular sparsity patterns, it disrupts Mamba's state-space dynamics by removing entire channels, particularly in the selective mechanism (Δ, $A_{\log}$). This leads to instability in long-range tasks (e.g., 3.1% drop on Path-X at 50% sparsity vs. 0.6% for ours), as confirmed by eigenvalue analysis showing 0.12 maximum perturbation vs. 0.05 for our method (Appendix E.9).

G.2.3 Quantization

8-bit quantization maintains the full parameter count but reduces memory usage by 40% through lower bit-precision. However, it results in moderate performance degradation: 7.6% perplexity increase on WikiText-103, 1.6% accuracy drop on LRA, and 4.8% MSE increase on ETT. Inference time improves modestly (34% reduction) due to optimized integer operations, but benefits are limited without hardware-specific acceleration (e.g., INT8 support). Quantization affects numerical precision in Mamba's state-space parameters, leading to cumulative errors in recurrent computations, particularly for long sequences (e.g., 2.4% accuracy drop on Path-X vs. 0.6% for our pruning).

G.2.4 Hybrid Pruning + Quantization

Combining our 50% unstructured pruning with 8-bit quantization yields synergistic benefits, achieving performance close to our pruning alone (5.6% perplexity increase, 1.0% accuracy drop, 3.3% MSE increase) while further reducing memory usage (50% at 50% sparsity, 32% at 70%) and inference time (48% and 57% reductions). The hybrid approach mitigates quantization's precision loss by pruning less critical parameters first, preserving key state-space dynamics. At 70% sparsity, performance degrades slightly more (11.1% perplexity increase), suggesting diminishing returns at extreme compression levels.

G.3 Theoretical Considerations

Structured Pruning: Structured pruning removes entire structural units (e.g., channels), which simplifies hardware acceleration but risks disrupting Mamba's recurrent dynamics. The state-space matrices (A, B, C) are highly interdependent, and removing entire dimensions alters eigenvalue properties, as shown in Appendix E.9. Our unstructured pruning, by contrast, allows fine-grained parameter removal, preserving stability through targeted eigenvalue checks (Equation 3).

Quantization: Quantization reduces numerical precision, which can destabilize Mamba's recurrent computations due to error accumulation in state transitions. The selective mechanism (Δ) is particularly sensitive to quantization noise, as small changes in timescale parameters can significantly alter information flow. Our pruning approach avoids this by maintaining full precision for retained parameters, with sparsity providing comparable memory savings.

Hybrid Approach: Combining pruning and quantization leverages the strengths of both: pruning removes redundant parameters to maintain performance, while quantization reduces the memory footprint of remaining weights. The hybrid approach aligns with findings in Transformer compression (Han et al., 2016), but our stability-aware pruning ensures Mamba's recurrent dynamics remain intact, unlike Transformer-focused hybrids that ignore eigenvalue constraints.

G.4 Practical Implications

- **Unstructured Pruning**: Best for performance-critical applications, offering the highest accuracy retention and robust generalization (Appendix E.6). However, it

Table 12: Comparison of unstructured pruning, structured pruning, quantization, and hybrid pruning+quantization for Mamba-Base across tasks at 50% and 70% parameter reduction (or equivalent for quantization).

Method	Params (M)	WikiText Perplexity	LRA Accuracy (%)	ETT MSE	Inference (ms/token)	Memory (Rel.)
50% Reduction						
Dense	370	19.8	83.1	0.335	1.45	1.00x
Unstructured (Ours)	185	**20.7**	**82.3**	**0.343**	**0.82**	**0.54x**
Structured	185	22.1	80.8	0.362	0.78	0.52x
Quantization (8-bit)	370*	21.3	81.5	0.351	0.95	0.60x
Hybrid (Ours + 8-bit)	185*	**20.9**	**82.1**	**0.346**	0.76	0.50x
70% Reduction						
Unstructured (Ours)	111	**21.7**	**81.3**	**0.355**	**0.68**	**0.36x**
Structured	111	23.8	79.2	0.381	0.65	0.34x
Quantization (8-bit)	370*	21.3	81.5	0.351	0.95	0.60x
Hybrid (Ours + 8-bit)	111*	**22.0**	**81.0**	**0.359**	0.62	0.32x

*Quantized models maintain parameter count but reduce bit-precision, with effective memory reduction.

requires sparse tensor support, which may not be fully optimized on all hardware (Hoefler et al., 2021).

- **Structured Pruning**: Suitable for hardware with strong support for regular sparsity (e.g., TPUs), but its performance degradation limits its use in tasks requiring long-range dependencies (e.g., LRA Path-X).
- **Quantization**: Ideal for memory-constrained devices with INT8 acceleration, but its standalone performance is inferior to pruning. It is most effective in hybrid settings.
- **Hybrid Approach**: Offers the best efficiency-performance trade-off for edge deployment, achieving near-unstructured pruning performance with quantization-level memory savings. However, it requires careful calibration to avoid cumulative errors.