An Empirical Study on Normalization in Mamba

Anonymous Authors[1]

Abstract

Mamba is a novel architecture designed to efficiently handle long sequence modeling by leveraging selective structured state space models (SSM). However, as with many deep architectures, the training stability of Mamba presents a significant challenge. Normalization plays a critical role in addressing the training instability and efficiency of deep neural networks. This paper provides a formal representation of normalization in Mamba and systematically investigates the impact of normalization type and position on the Mamba in sequence modeling and image classification tasks. Our exploration indicates that the choice of the appropriate normalization significantly improves model performance, and normalization applied after the SSM layer is more effective than when applied before the SSM layer. These findings motivate us to explore the optimal normalization combination for the Mamba, aiming to achieve effectiveness across various application scenarios. We found that combining normalization methods can further enhance model performance. We provide practical recommendations for selecting the appropriate normalization techniques when designing the Mamba architecture and substantiate these through extensive experiments. Finally, we give an intuitive explanation of the role of normalization in Mamba training.

1. Introduction

Mamba (Gu & Dao, 2023) achieves linear computational complexity and comparable performance in long-sequence modeling tasks by introducing the selective State Space Model(SSM) and hardware-aware parallel algorithms (Gu et al., 2021), which is increasingly gaining attention. However, it faces challenges during training, where larger models are more prone to instability, causing the model to diverge (Dao & Gu, 2024).

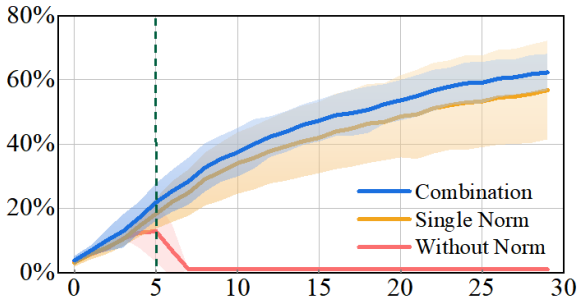

Figure 1. The accuracy of VMamba using different normalization strategies on the ImageNet dataset. The red line represents average performance without normalization, the yellow line shows with single normalization, and the blue line represents using our proposed combination normalization strategy. For more definition details about single and combination normalization, refer to Section 3.

To address this issue, several studies (Ma et al., 2024; Tang et al., 2024; Liu et al., 2024) have introduced different normalization strategies into the Mamba. These researches (Chen et al., 2024c; Zhou et al., 2024; Fan et al., 2024; Bai et al., 2024a) effort employ various normalization methods respectively to help alleviate training divergence. Although various normalization techniques have been tailored for specific domain tasks in the variants of the Mamba (Gu & Dao, 2023; Bai et al., 2024a; Liu et al., 2024; Gong et al., 2024; Ting et al., 2024), sufficient evidence to justify why such approaches are necessary has not been provided. Furthermore, whether there are universal patterns for utilizing normalization in Mamba and how to leverage normalization techniques better to optimize Mamba remains unknown.

To this end, we systematically research the impact of normalization *type* and *position* on the performance and stability of Mamba in sequence modeling and vision tasks. *Position* indicates where the normalization is applied relative to SSM in Mamba, while *type* represents which normalization method is chosen before and after the SSM. First, we provide a generalized landscape for normalization research in Mamba. Then, we categorize the normalization strategies used in Mamba variants based on *position* and *type*. Finally, we propose a *combination* normalization strategy that achieves better performance and stability, as shown in Figure 1.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

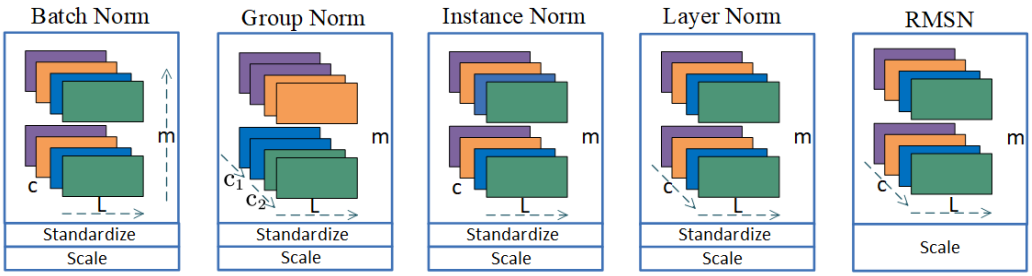

Figure 2. **Normalization methods for sequence:** Each subfigure displays the dimensional information of a feature map, where m represents the batch axis, d the channel axis, L the sequence length axis. The dashed arrows indicate that the mean and variance are computed by aggregating the values across these dimensions.

We conducted extensive experiments for *position*, *type* and *combination* scenarios in long sequence analysis and image classification tasks. For *type*, we found that selecting the appropriate normalization type is crucial for improving model performance, with Layer Normalization(LN) (Ba et al., 2016) and Group Normalization(GN) (Wu & He, 2018) generally performing best across both tasks. For *position*, we discovered that applying normalization after the SSM layer has a more significant impact on the model than applying it before the SSM layer. In the *combination* experiments, our proposed combination normalization approach consistently identified combinations that outperform single-type normalization techniques.

Building on this, we conducted an in-depth investigation into the root causes of training instability in the Mamba architecture. Our study reveals that in deeper network layers, the divergence in data distribution after the SSM layer becomes significantly more pronounced when normalization methods are not applied, leading to substantially higher weight norms in deeper layers compared to shallower ones. Selecting appropriate normalization methods after the SSM layer effectively regularizes the data distribution, preserves scale invariance of weight norms across layers of varying depths, and enhances the model's robustness.

Our main contributions are summarized as follows:

- Framework for Normalization in Mamba: We provide a formal representation of normalization in Mamba and found that the choice of normalization methods significantly impacts performance. Furthermore, applying normalization after the SSM layer consistently enhances model performance compared to applying it before the SSM layer.
- Normalization Combination: We proposed that specific combinations of different normalization techniques yield improved results. The L2 norm distribution of the weight matrices in Mamba provides an explanatory perspective for this phenomenon.
- We first identify the optimal normalization choice after the SSM layer, then apply an adaptive strategy for normalization before the SSM layer. We validate the performance of this normalization strategy across multiple datasets. Finally, we provide an intuitive explanation of the role of normalization in the Mamba architecture.

2. Background and Related Work

2.1. Mamba

Mamba introduces a selective mechanism that allows the parameters of SSM to be dynamically adjusted based on the input tokens. SSM uses first-order differential equations to map the input function $x_t \in \mathbb{R}^{M\times L}$ to the output function $y_t \in \mathbb{R}^{M\times L}$ through hidden state $h_t \in \mathbb{R}^{N\times L}$, defined as $h_t = \boldsymbol{A}h_{t-1} + \boldsymbol{B}x_t, \quad y_t = \boldsymbol{C}h_t + \boldsymbol{D}x_t$. Here state transition matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, input matrix $\boldsymbol{B} \in \mathbb{R}^{N\times M}$, output matrix $\boldsymbol{C} \in \mathbb{R}^{M\times N}$and forward channel transition matrix $\boldsymbol{D} \in \mathbb{R}^{M\times M}$. The variables N and M refer to the input and hidden state feature dimensions, respectively. Continuous parameters $\boldsymbol{A}, \boldsymbol{B}$ can be discretized by the zero-order hold method with the sampling interval Δ. Details of the process can be found in (Gu et al., 2021).

Apart from handling one-dimensional sequences, it can also effectively process two-dimensional (2D) image data. For example, in VMamba (Liu et al., 2024), the image is first divided into $h \times w$ patches, and then these patches are converted into a sequence of features with positional encoding added. Then, the same subsequent processing method is applied. As the sequence length increases, Mamba maintains linear computational complexity and comparable performance, making it a strong competitor to Transformer (Vaswani et al., 2023) and other architectures (Hochreiter & Schmidhuber, 1997; Zaremba et al., 2015). However, like other complex architectures, Mamba faces instability issues during training. Several normalization methods have been incorporated into the Mamba architecture to alleviate this issue.

2.2. Normalization

Normalization techniques are crucial for improving the stability and performance of model training. (Wu & He, 2018; Huang et al., 2023a). By adjusting the distribution of feature values, normalization can accelerate model convergence. They have been widely applied to sequential data (Dosovitskiy et al., 2021; Gu & Dao, 2023). The fundamental operation of normalization methods involves subtracting the mean and dividing by the standard deviation, $\mathrm{N}(x) = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \cdot \gamma + \beta$. Here $E[x]$ represents the expectation of the input x, $\mathrm{Var}[x]$ represents the variance of the input x, and ϵ is a small constant added to prevent division by zero. γ and β are the scale and shift parameter, respectively.

The key distinction of sequence normalization techniques lies in the domain over which statistical quantities are computed, as shown in Figure 2. Specifically, Batch Normalization(BN) (Wang et al., 2022) normalizes data across batch samples within the same channel, effectively accelerating training convergence. However, its reliability diminishes with small batch sizes due to unstable statistical estimates. In contrast, Instance Normalization(IN) (Ulyanov et al., 2017) processes spatial dimensions exclusively within individual channels of a single sample, eliminating inter-sample variations but potentially blurring cross-channel feature relationships. Layer Normalization(LN) (Ba et al., 2016) computes statistics across all channels of a single instance, making it suitable for variable-length sequences and small-batch scenarios, though it overlooks inter-channel statistical disparities. Group Normalization(GN) (Wu & He, 2018) partitions channels into predefined groups for localized normalization, balancing channel dependency and efficiency, albeit introducing sensitivity to manual group configuration. Diverging from these approaches, Root Mean Square Normalization(RMSN) (Zhang & Sennrich, 2019) relies solely on the root mean square, bypassing the zero-mean assumption to enhance robustness for asymmetric data distributions. However, this simplification risks information loss by ignoring first-order moments.

2.3. Normalization in Mamba

Different normalization techniques are employed in Mamba. Some methods aim to stabilize feature distributions before entering the SSM module. For instance, in sequence tasks, LN is employed in FMamba (Ma et al., 2024), MLSA4Rec (Su & Huang, 2024), and Zamba (Glorioso et al., 2024), while RMSN is applied in DiM-SUM (Phung et al., 2024), Quamba (Chiang et al., 2024), bi-CrossMamba (Wu et al., 2024), Mamba-PTQ (Pierro & Abreu, 2024), and CMAMBA (Zeng et al., 2024). BN and IN are also used in self-supervised (Liang et al., 2024) and MC-SEMamba (Ting et al., 2024), respectively. In vision tasks, LN is the most commonly adopted method, appearing in models such as RetinexMamba (Bai et al., 2024b), CU-Mamba (Deng & Gu, 2024), and RSMamba (Chen et al., 2024b), while RMSN is used in FST-Mamba (Wei et al., 2024) to improve feature stability.

Unlike the above methods that apply normalization before SSM effectively to mitigate input feature distribution shifts, some approaches focus on adjusting feature distributions after the SSM module to optimize subsequent processing. In sequence tasks, RMSN is used in DIFFIMP (Gao et al., 2024), IN is adopted by Bi-Mamba (Tang et al., 2024), and GN is applied in Mamba2 (Bai et al., 2024a). In vision tasks, LN is employed in IRSRMamba (Huang et al., 2024), while GN is used in MambaHSI (Li et al., 2024). These approaches fine-tune the feature distributions processed by the SSM module, aligning them with the requirements of downstream tasks.

In short, applying normalization in the Mamba architecture exhibits diversity. However, the selection of normalization methods and *combination* strategies remains an open question. Therefore, it is necessary to conduct a systematic investigation into the normalization methods used in Mamba, which could provide valuable directions and references for future Mamba framework designs.

3. Methodology

In this section, we first provide a formal representation of this framework for studying normalization patterns in Mamba, as illustrated in Figure 4. Within our proposed framework, previous works can be categorized into two patterns based on *position* and *type*, as shown in Figure 3. Furthermore, we introduce a new Combination pattern to extend these approaches.

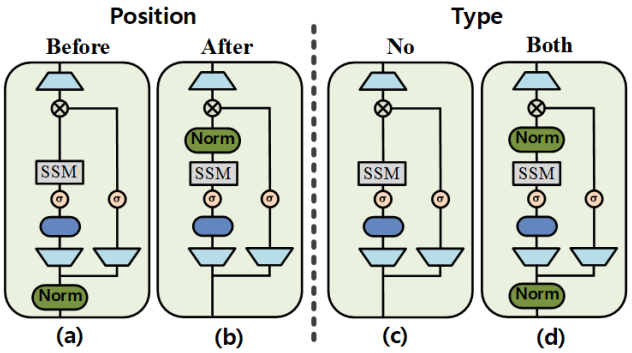

Figure 3. The Position Pattern can be divided into two forms: (a) normalization before SSM and (b) normalization after SSM. The *type* Pattern can be divided into two forms: (c) without normalization and (d) single normalization before and after SSM.

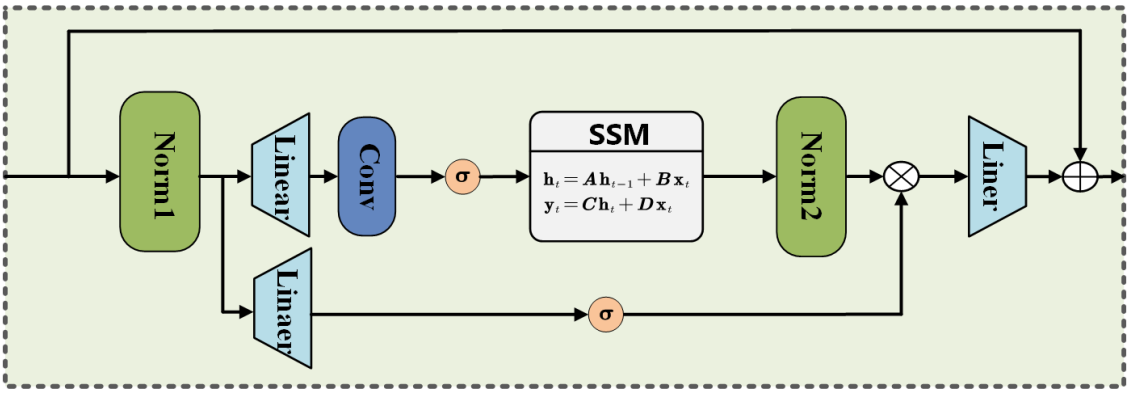

Figure 4. Framework for analyzing normalization in Mamba. This figure also provides a detailed explanation of the combination pattern.

3.1. Formulation of Our Framework

As shown in Figure 4, let N_1 represent the normalization layer, which normalizes the input x, followed by a series of transformations in the main branch :

$$f = N2(SSM(Act(Con(Lin(N1(x)))))). \quad (1)$$

Where $N1$ is the first Normalization, Lin is the Linear projection, Con is the Depth-Wise convolution, Act is the SiLU activation, SSM is the selective structured state space models, $N2$ is the second normalization. Meanwhile, in parallel, the normalized input $N1(x)$ is passed through another Linear projection and SiLU activation:

$$p = Act(Lin(N1(x))). \quad (2)$$

Then, the outputs of the main and parallel branches are combined via element-wise multiplication $u = f \otimes p$. Finally, the combined result is passed through a linear projection, and the original input x is added as a residual connection:

$$y = Lin(f \otimes p) \oplus x. \quad (3)$$

Where $\otimes$ is the element-wise multiplication, $\oplus$ is the element-wise addition. The normalization methods used in Mamba can be studied and categorized based on their positions and types.

3.2. Normalization Position and Type in Mamba

Normalization Position

The position of normalization layers relative to the SSM module significantly impacts the feature distribution at different positions in Mamba. These positions can be summarized into two cases: normalization before SSM and normalization after SSM.

Normalization Before SSM: Normalization layers placed before SSM stabilize the input features by reducing shifts in their distribution, which can be formulated as:

$$f = SSM(Act(Con(Lin(N1(x))))). \quad (4)$$

Where $N1$ represents the normalization applied before the SSM module. This position ensures SSM operates on well-conditioned inputs, minimizing potential disruptions caused by unstable feature distributions.

Normalization After SSM: Normalization layers placed after SSM refine the feature distribution generated by SSM. This can be expressed as:

$$f = N2(SSM(Act(Con(Lin(x))))). \quad (5)$$

Where $N2$ represents the normalization applied after the SSM module. This configuration aligns processed features with the requirements of downstream tasks.

Within this framework, some methods represent specific instances within our framework. For example, $N1 = RMSN$ is applied before the SSM in DiMSUM, Quamba, bi-CrossMamba, Mamba-PTQ, CMAMBA and FST-Mamba (Phung et al., 2024; Chiang et al., 2024; Wu et al., 2024; Pierro & Abreu, 2024; Zeng et al., 2024; Wei et al., 2024), while $N2 = RMSN$ is used after the SSM in DIFFIMP (Gao et al., 2024).

Normalization Type

To investigate the impact of normalization types in Mamba, we employ the same normalization method both before and after the SSM as follows:

$$f = N(SSM(Act(Con(Lin(N(x)))))). \quad (6)$$

This method prevent the normalization position from interfering with the SSM and adjusts feature distributions both before and after SSM

It is also evident that some of the current methods represent specific cases. For example, MambaDC, Bi-Mamba, TiM4Rec, ChangeMamba, MiM-ISTD, and Fusion-Mamba (Chen et al., 2024c; Zhou et al., 2024; Fan et al., 2024; Chen et al., 2024a;d; Dong et al., 2024) apply $N = LN$, while BMAMBA2 (Bai et al., 2024a) adopts $N = GN$. We primarily discuss five commonly used normalization methods: BN, GN, IN, LN, and RMSN.

Different normalization methods adjust the values of feature maps from various perspectives, prompting us to explore further whether combining different normalization techniques could further enhance model performance.

3.3. Normalization Combinations

Normalization combinations involve using different normalization techniques at different positions around the SSM module. By leveraging the strengths of various methods, this approach adjusts feature distributions more effectively. Combining normalizations before and after SSM can be expressed as:

$$f = N2(SSM(Act(Con(Lin(N1(x)))))). \quad (7)$$

Where $N1$, $N2$ represents different normalizations applied at each position, taking advantage of different techniques to optimize feature processing at various stages.

Our research aims to identify the optimal normalization settings by adjusting the *type*, *position*, and *combination* of normalization techniques. By exploring the optimal normalization strategy for the Mamba architecture, it is possible to mitigate the limitations of individual methods, improve parameter updates, and enhance convergence stability. These findings provide a practical guideline for designing robust normalization strategies in the Mamba architecture. Next, we will explore these through experiments.

4. Experiments

In this section, we evaluate the impact of normalization *type*, *position*, and *combination* on the performance of Mamba in long-sequence modeling and vision classification tasks. In Section 4.2, we validate the varying effects of different normalization methods on Mamba. In Section 4.3, we explore the more critical issue of the optimal placement of normalization. Finally, in Section 4.4, we discuss the further enhancement in model performance achieved by combining different normalization approaches. This analysis allows us to gain a deeper understanding of normalization configurations and provides insights into the practical design of the Mamba architecture.

4.1. Datasets

For long sequence modeling, we use the Breakfast dataset (Kuehne et al.). Breakfast is a large-scale dataset designed to evaluate models on long sequence modeling and activity segmentation. It consists of 1,712 videos recorded in 18 different kitchens, involving 52 participants performing 10 distinct actions related to breakfast preparation, such as making tea, frying eggs, and preparing toast. Each video is annotated with frame-level action labels, with sequences often comprising multiple nested actions. The dataset spans over four million frames, making it highly challenging for models to handle long temporal dependencies effectively.

For vision classification tasks, we use the ImageNet-100 dataset (Krizhevsky et al., 2012). ImageNet-100 is a randomly selected subset of the ImageNet-1k dataset from the 2012 Large Scale Visual Recognition Challenge. It contains 100 categories, covering various objects and scenes, ensuring diversity in vision tasks. The training set includes 1300 images per category, while the validation set contains 50 images per category, totaling 135,000 images.

4.2. Normalization Types

The performance of various normalization types across both sequence and visual experiments, as shown in Figure 5.

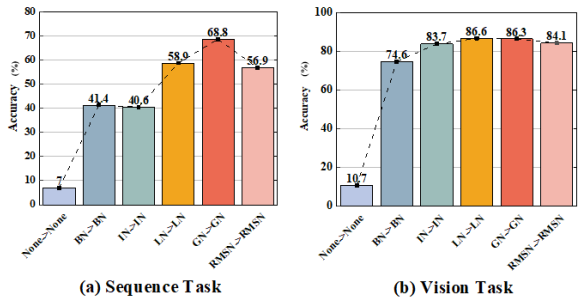

Figure 5. Performance of different type normalization methods on sequence analysis and image classification datasets.

The experimental results demonstrate that different types of normalization have varying impacts on model performance. In the sequence modeling task, applying $N = GN$ improved the performance of Mamba from the baseline of 7.0% (no normalization) to 68.8%. LN also significantly enhanced performance, reaching 58.9%, making it the next-best performer. In the image classification task, applying $N = LN$ increased the baseline accuracy from 10.7% to 86.6%. GN closely followed with an accuracy of 86.3%.

Both GN and LN achieved strong results in sequence analysis and vision tasks. This validates the rationale behind the VMamba(Liu et al., 2024) architecture's initial adoption of the LN normalization type. Additionally, the outstanding performance of GN supports the decision made by the Mamba2(Dao & Gu, 2024) architecture to choose GN after

SSM.

4.3. Normalization Positions

The impact of normalization position on the performance of Mamba in sequence analysis and vision classification tasks is shown in Figure 6.

We observe that, for both tasks, applying normalization after SSM generally yields better results. Specifically, in the sequence analysis task, applying GN after SSM improves accuracy by 49.6% compared to applying GN before SSM. Similarly, in the vision classification task, applying GN after SSM improves accuracy by 20.7% compared to applying GN before SSM. We conclude that the impact of normalization after the SSM layer is more significant than normalization before the SSM layer. We think this is because the output feature distribution of SSM becomes huge and unstable due to dynamic parameterization. Applying GN after SSM directly normalizes these dynamic features, eliminating distribution shifts and enhancing subsequent layers' training stability. In contrast, applying GN before SSM cannot effectively constrain the output distribution of SSM, making it difficult for downstream modules to learn effectively. Therefore, when studying appropriate normalization in Mamba, greater attention should be given to the normalization applied after the SSM layer.

Table 1. Performance of sequence modeling and image classification tasks with different normalizations applied before and after SSM.

Normalization Method	Accuracy (%)	Normalization Method	Accuracy (%)
BN→SSM→None	28.4	None→SSM→BN	28.4
IN→SSM→None	10.9	None→SSM→IN	7.0
LN→SSM→None	57.1	None→SSM→LN	59.1
RMSN→SSM→None	58.7	None→SSM→RMSN	60.5
GN→SSM→None	20.5	None→SSM→GN	70.1

Normalization Method	Accuracy1 (%)	Normalization Method	Accuracy1 (%)
BN→SSM→None	20.5	None→SSM→BN	67.8
IN→SSM→None	70.2	None→SSM→IN	83.8
LN→SSM→None	86.5	None→SSM→LN	86.7
RMSN→SSM→None	86.3	None→SSM→RMSN	84.2
GN→SSM→None	66.1	None→SSM→GN	86.8

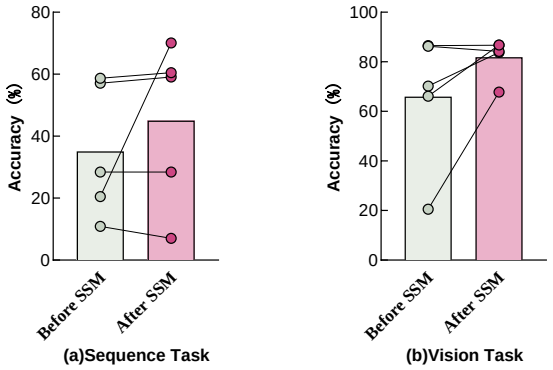

Figure 6. The bar chart represents the average of all data points. The lines connecting the data points reflect the numerical relationship in model performance results of the normalization before and after SSM.

4.4. Normalization Combinations

Now, we conduct experiments to explore the effect of combined normalization patterns on Mamba. Compared to single-type pattern normalization, we first verify that an appropriate combination always leads to improved model performance. Next, we investigate how to select an effective combination method to meet the requirements of Mamba across different task scenarios.

We fixed the normalization after the SSM layer and compared the results of the single *type* pattern normalization with those of the *Combination* pattern normalization, which yielded optimal performance, as shown in Figure 7.

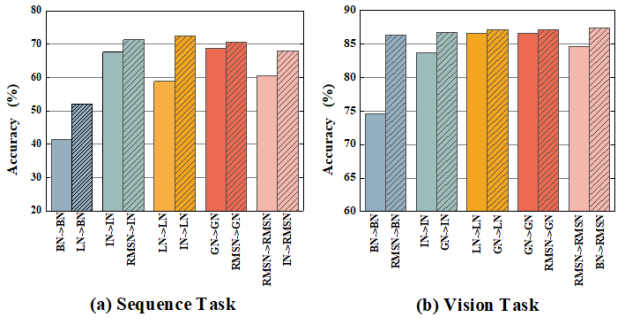

Figure 7. This figure compares the results of using single normalization (*type*) and best normalization combinations (*combination*). No padding represents the results with a single *type* pattern, while padding indicates the results with *Combination* pattern.

We found that, in both sequence analysis and image classification tasks, there exists an optimal combination of normalization methods (*Combination*) that significantly improves model performance compared to using a single normalization method (*type*). For example, in sequence tasks, using the single BN→BN normalization results in an accuracy of only 41.4%, whereas the LN→BN combination strategy improves the accuracy to 52.1%. In vision tasks, using the single BN→BN normalization results in a classification accuracy of only 74.6%, while the RMSN→BN combination strategy increases the accuracy to 87.3%. This suggests that our exploration of how to find the optimal normalization combination is meaningful.

Building on the findings from Section 4.3, we first need to determine the best normalization method after the SSM layer in the *combination* scenario to identify the optimal normalization combination. To this end, we conducted experiments in which the normalization after the SSM layer was fixed (as shown on the x-axis). In contrast, the normalization before the SSM layer was varied in combination.

For example, with GN on the x-axis, we varied the normalization types before the SSM layer, setting them to BN, IN, RMSN, and LN, respectively. Multiple experiments were conducted, and the results are presented as box plots, as shown in Figure 8.

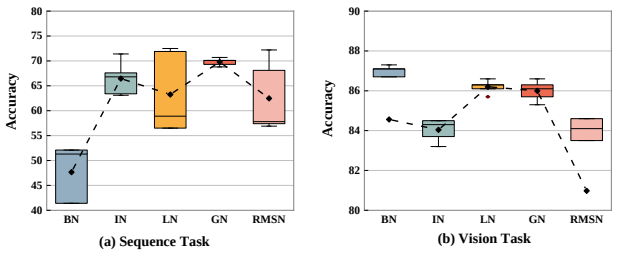

Figure 8. The exploration of the optimal normalization after the SSM layer in the combination scenario. The x-axis represents the normalization applied after the SSM, and the y-axis represents the average performance of the other four normalizations before SSM.

The normalization with higher accuracy and narrower box indicate that it should be considere the optimal choice to be applied after the SSM layer. In the sequence modeling task, we found that the average performance of GN was the best, and the distribution of different normalization combinations was the most concentrated, indicating that placing GN after SSM is the optimal choice for sequence analysis tasks. In the image classification task, we found that the average performance of LN was the best, and the distribution of different normalization combinations was the most concentrated, suggesting that placing LN after SSM is the optimal choice for image classification tasks. This will provide important guidance for future normalization choices when applying Mamba. Therefore, in sequence tasks, we fix the normalization after the SSM layer to GN, while in vision tasks, we fix the normalization after the SSM layer to LN.

The experimental results highlight the critical role of normalization techniques and their positions in neural network architectures. Applying normalization after the SSM module is generally more beneficial in sequence modeling and image classification tasks. Moreover, combining specific different normalization methods before and after SSM can significantly enhance model performance. However, how should we select the appropriate normalization before SSM? We will fix the normalization method after the SSM (e.g., GN for sequence modeling and LN for vision tasks) and then automatically vary the normalization method before the SSM during training. For more details about NormVary strategy, you can refer to D

Table 2. Comparison of test set accuracy on the LRA Benchmark. The results are averaged over three random seeds and reported with the standard deviation.

Experiment	Original	Our Method
Listops	37.90±0.23 %	40.27±0.15 %
Imdb	81.27±0.75 %	83.71±0.37 %
Cifar-100	58.33±0.22 %	62.20±0.28 %
Path-64	83.26±0.21 %	84.02±0.34 %

4.5. Validation Experiment

To validate our proposal, we compared test accuracy on the LRA Benchmark. The results are averaged over three random seeds and reported with the standard deviation. The experimental results are shown in Table 2.

For sequence tasks, the original Mamba normalization configuration is RMSN→SSM→None, while for vision tasks, the original Mamba normalization configuration is LN→SSM→LN. For sequence tasks, we propose the normalization configuration NormVary→SSM→GN, and for vision tasks, our proposed normalization configuration is NormVary→SSM→LN. The results for both sequence and vision tasks are shown in Table 2. It can be observed that our proposed method outperforms the original model in terms of experimental results, thereby validating the effectiveness of our proposed solution.

4.6. Intuitive Explanation

Finally, we provide an intuitive explanation of normalization's role in stabilizing the Mamba model's training. We found that in the Mamba architecture, without normalization, the output layer after the SSM consistently produces infinite values, causing the training to fail. Specifically, we observed that the lack of normalization in the Mamba layers leads to some elements in the input tensors growing extremely large as the network depth increases. However, when normalization is applied both before and after, the training proceeds smoothly without failure. We use the L2 norm B to describe the overall scale of the elements in the tensor. For the definition of the L2 norm, please refer to B. Using the Listops sequence analysis dataset as an example, the results are shown in Figure 9.

We found that when there is no normalization in Mamba, the L2 norm of the output tensors from each Mamba Block layer exhibits exponential growth as the number of layers increases. Normalization applied before the SSM layer does not mitigate this phenomenon, whereas normalization applied after the SSM layer helps maintain good scale invariance across the outputs of different layers. When normalization is applied both before and after the SSM layer, the scale disparity between layers is further reduced, and the distribution of the elements becomes more uniform.

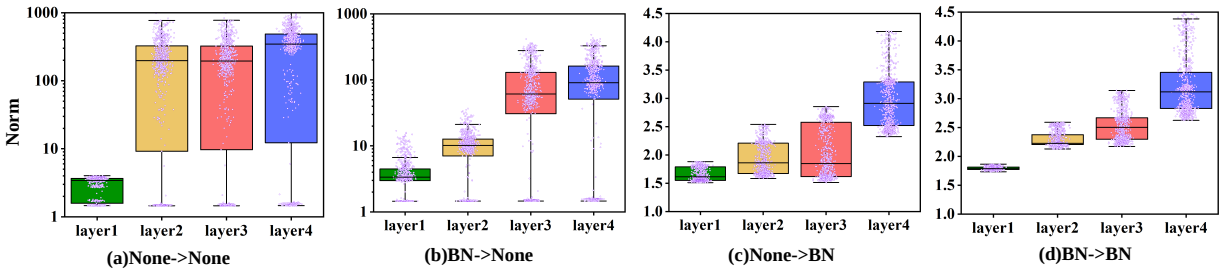

Figure 9. Plot of the L2 norm of the output tensors from each Mamba layer as a function of the layer number (using BN as an example). Each layer represents one Mamba Block, with the y-axis scaled logarithmically (log10). The points in the plot represent the elements in the tensor.

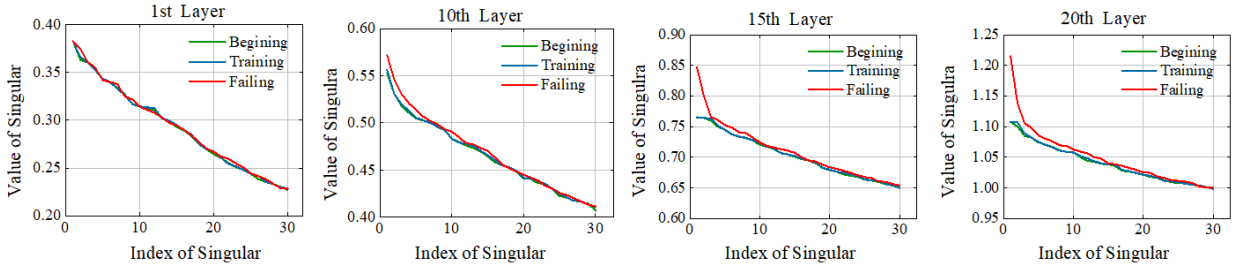

Figure 10. Singular values of the weights (output layer) in different Mamba layers. The green, blue, and red lines represent the states during the early training phase, normal training, and training failure, respectively.

Furthermore, we considered the impact of weights on the model's decision-making process and conducted an in-depth analysis of the singular values of the output mapping layer in the absence of normalization to handle the extensive number of weights. The results are shown in Figure 10.

A relatively uniform distribution of singular values can be observed across all layers during stable training. However, as the model approaches a state where infinite values occur, the variance of the singular values across different layers significantly increases. Some weights gain more importance in the model's decision-making process, leading to sharper model outputs (Lin et al., 2024). Specifically, as the number of layers increases, the singular values of the deeper Mamba Blocks show the most significant growth, suggesting that the model's decisions may heavily rely on certain weights in the deeper layers, thereby increasing their sensitivity.

5. Conclusions and Future Work

In this study, we comprehensively investigated the types, positions, and combinations of normalization layers within the Mamba architecture. Our findings reveal that applying proper normalization after the SSM Modules enhances training stability by mitigating large variations in weight norms. Moreover, compared to using a single normalization type, the combination of normalization techniques not only stabilizes the training process but also significantly improves model performance.

We also propose a strategy for combining normalization layers, where we first fix the important normalization after the SSM layer in the Mamba architecture and apply an adaptive approach for the normalization before the SSM layer. Finally, we provide indirect evidence of the role of normalization in Mamba to facilitate the exploration of stable training for deep Mamba architectures. This intuition provides valuable insights into choosing appropriate normalization methods for robust training of large-scale neural networks. Future research will focus on extending this intuition to more complex models and tasks, aiming to refine normalization strategies for further improvements in efficiency and robustness.

Impact Statement

This paper presents work whose goal is to advance the field of Deep Learning. There are many potential social consequences of our work, none which feel must be specifically highlighted here.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Bai, J., Fang, Y., Wang, J., and Zhang, X. A two-stage band-split mamba-2 network for music separa-

tion. *ArXiv preprint*, abs/2409.06245, 2024a. URL https://arxiv.org/abs/2409.06245.

Bai, J., Yin, Y., He, Q., Li, Y., and Zhang, X. Retinexmamba: Retinex-based mamba for low-light image enhancement. *ArXiv preprint*, abs/2405.03349, 2024b. URL https://arxiv.org/abs/2405.03349.

Chen, H., Song, J., Han, C., Xia, J., and Yokoya, N. Changemamba: Remote sensing change detection with spatiotemporal state space model. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–20, 2024a. doi: 10.1109/TGRS.2024.3417253.

Chen, K., Chen, B., Liu, C., Li, W., Zou, Z., and Shi, Z. Rsmamba: Remote sensing image classification with state space model. *ArXiv preprint*, abs/2403.19654, 2024b. URL https://arxiv.org/abs/2403.19654.

Chen, M., Zhang, Q., Wang, M., Zhang, X., Liu, H., Ambikairaiah, E., and Chen, D. Selective state space model for monaural speech enhancement. *ArXiv preprint*, abs/2411.06217, 2024c. URL https://arxiv.org/abs/2411.06217.

Chen, T., Ye, Z., Tan, Z., Gong, T., Wu, Y., Chu, Q., Liu, B., Yu, N., and Ye, J. Mim-istd: Mamba-in-mamba for efficient infrared small target detection. *ArXiv preprint*, abs/2403.02148, 2024d. URL https://arxiv.org/abs/2403.02148.

Chiang, H.-Y., Chang, C.-C., Frumkin, N., Wu, K.-C., and Marculescu, D. Quamba: A post-training quantization recipe for selective state space models. *ArXiv preprint*, abs/2410.13229, 2024. URL https://arxiv.org/abs/2410.13229.

Dao, T. and Gu, A. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *ICML*, 2024.

Deng, R. and Gu, T. Cu-mamba: Selective state space models with channel learning for image restoration. *ArXiv preprint*, abs/2404.11778, 2024. URL https://arxiv.org/abs/2404.11778.

Dong, W., Zhu, H., Lin, S., Luo, X., Shen, Y., Liu, X., Zhang, J., Guo, G., and Zhang, B. Fusion-mamba for cross-modality object detection. *ArXiv preprint*, abs/2404.09146, 2024. URL https://arxiv.org/abs/2404.09146.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Fan, H., Zhu, M., Hu, Y., Feng, H., He, Z., Liu, H., and Liu, Q. Tim4rec: An efficient sequential recommendation model based on time-aware structured state space duality model. *ArXiv preprint*, abs/2409.16182, 2024. URL https://arxiv.org/abs/2409.16182.

Gao, H., Shen, W., Qiu, X., Xu, R., Hu, J., and Yang, B. Diffimp: Efficient diffusion model for probabilistic time series imputation with bidirectional mamba backbone. *ArXiv preprint*, abs/2410.13338, 2024. URL https://arxiv.org/abs/2410.13338.

Glorioso, P., Anthony, Q., Tokpanov, Y., Whittington, J., Pilault, J., Ibrahim, A., and Millidge, B. Zamba: A compact 7b ssm hybrid model. *ArXiv preprint*, abs/2405.16712, 2024. URL https://arxiv.org/abs/2405.16712.

Gong, H., Kang, L., Wang, Y., Wan, X., and Li, H. nnmamba: 3d biomedical image segmentation, classification and landmark detection with state space model. *ArXiv preprint*, abs/2402.03526, 2024. URL https://arxiv.org/abs/2402.03526.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *ArXiv preprint*, abs/2312.00752, 2023. URL https://arxiv.org/abs/2312.00752.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *ArXiv preprint*, abs/2111.00396, 2021. URL https://arxiv.org/abs/2111.00396.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., and Shao, L. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):10173–10196, 2023a. doi: 10.1109/TPAMI.2023.3250241.

Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., and Shao, L. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE transactions on pattern analysis and machine intelligence*, 45(8):10173–10196, 2023b.

Huang, Y., Miyazaki, T., Liu, X., and Omachi, S. Irsrmamba: Infrared image super-resolution via mamba-based wavelet transform feature modulation model. *ArXiv preprint*, abs/2405.09873, 2024. URL https://arxiv.org/abs/2405.09873.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 1(4):7, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Kuehne, H., Arslan, A. B., and Serre, T. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR), year =*.

Li, Y., Luo, Y., Zhang, L., Wang, Z., and Du, B. Mambahsi: Spatial-spectral mamba for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

Liang, J., Meyer, B., Lee, I. N., and Do, T.-T. Self-supervised learning for acoustic few-shot classification. *ArXiv preprint*, abs/2409.09647, 2024. URL https://arxiv.org/abs/2409.09647.

Lin, R., Yu, C., Han, B., Su, H., and Liu, T. Layer-aware analysis of catastrophic overfitting: Revealing the pseudo-robust shortcut dependency, 2024. URL https://arxiv.org/abs/2405.16262.

Linsley, D., Ashok, A., Govindarajan, L., Liu, R., and Serre, T. Learning long-range spatial dependencies with horizontal gated recurrent units. *Advances in Neural Information Processing Systems*, 31, 2018.

Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., and Liu, Y. Vmamba: Visual state space model 2024. *CVPR*, 2024.

Ma, S., Kang, Y., Bai, P., and Zhao, Y.-B. Fmamba: Mamba based on fast-attention for multivariate time-series forecasting. *ArXiv preprint*, abs/2407.14814, 2024. URL https://arxiv.org/abs/2407.14814.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150. Association for Computational Linguistics, 2011.

Phung, H., Dao, Q., Dao, T., Phan, H., Metaxas, D., and Tran, A. Dimsum: Diffusion mamba–a scalable and unified spatial-frequency method for image generation. *ArXiv preprint*, abs/2411.04168, 2024. URL https://arxiv.org/abs/2411.04168.

Pierro, A. and Abreu, S. Mamba-ptq: Outlier channels in recurrent large language models. *ArXiv preprint*, abs/2407.12397, 2024. URL https://arxiv.org/abs/2407.12397.

Su, J. and Huang, Z. Mlsa4rec: Mamba combined with low-rank decomposed self-attention for sequential recommendation. *ArXiv preprint*, abs/2407.13135, 2024. URL https://arxiv.org/abs/2407.13135.

Tang, S., Ma, L., Li, H., Sun, M., and Shen, Z. Bi-mamba: Towards accurate 1-bit state space models. *ArXiv preprint*, abs/2411.11843, 2024. URL https://arxiv.org/abs/2411.11843.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Long range arena: A benchmark for efficient transformers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Ting, W.-Y., Ren, W., Chao, R., Lin, H.-Y., Tsao, Y., and Zeng, F.-G. Mc-semamba: A simple multi-channel extension of semamba. *ArXiv preprint*, abs/2409.17898, 2024. URL https://arxiv.org/abs/2409.17898.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization, 2017. URL https://arxiv.org/abs/1607.08022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Wang, J., Wu, J., and Huang, L. Understanding the failure of batch normalization for transformers in nlp. *Advances in Neural Information Processing Systems*, 35:37617–37630, 2022.

Wei, Y., Abrol, A., Hassanzadeh, R., and Calhoun, V. Hierarchical spatio-temporal state-space modeling for fmri analysis. *ArXiv preprint*, abs/2408.13074, 2024. URL https://arxiv.org/abs/2408.13074.

Wu, D., Wang, Y., Wu, X., and Qu, T. Cross-attention inspired selective state space models for target sound extraction. *ArXiv preprint*, abs/2409.04803, 2024. URL https://arxiv.org/abs/2409.04803.

Wu, Y. and He, K. Group normalization, 2018. URL https://arxiv.org/abs/1803.08494.

Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent neural network regularization, 2015. URL https://arxiv.org/abs/1409.2329.

Zeng, C., Liu, Z., Zheng, G., and Kong, L. C-mamba: Channel correlation enhanced state space models for multivariate time series forecasting. *ArXiv preprint*, abs/2406.05316, 2024. URL https://arxiv.org/abs/2406.05316.

Zhang, B. and Sennrich, R. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Zhou, X., Han, Y., Liu, C., Ding, Y., Jia, Z., and Liu, Y. Bitmamsleep: Bidirectional temporal mamba for eeg sleep staging. *ArXiv preprint*, abs/2411.01589, 2024. URL https://arxiv.org/abs/2411.01589.

A. Normalization Taxonomy

An unified taxonomy was proposed by (Huang et al., 2023b) to understand the similarities and differences among these methods, specifically including Normalization Representation Area Partitioning (NAP), Normalization Operation (NOP), and Normalization Representation Recovery (NRR). The operational details of these mainstream techniques are clearly presented in the following Table 3.

Method	NAP	NOP	NRR
BN	$\Pi_{\text{BN}}(\mathbf{X}) \in \mathbb{R}^{D\times mL}$	Standardizing	Learnable $\gamma, \beta \in \mathbb{R}^D$
IN	$\Pi_{\text{IN}}(\mathbf{X}) \in \mathbb{R}^{mD\times L}$	Standardizing	Learnable $\gamma, \beta \in \mathbb{R}^D$
GN	$\Pi_{\text{GN}}(\mathbf{X}) \in \mathbb{R}^{mg_D\times s_D L}$, $g_D \times s_D = D$	Standardizing	Learnable $\gamma, \beta \in \mathbb{R}^D$
LN	$\Pi_{\text{LN}}(\mathbf{X}) \in \mathbb{R}^{m\times DL}$	Standardizing	Learnable $\gamma, \beta \in \mathbb{R}^D$
RMSN	$\Pi_{\text{RMSN}}(\mathbf{X}) \in \mathbb{R}^{m\times DL}$	Scaling	Learnable $\gamma \in \mathbb{R}^D$

Table 3. Details of mainstream normalization techniques.

Taking an batch sequences $\mathbf{X} \in \mathbb{R}^{m\times D\times T}$ as an example. The NAP operation determines how $\mathbf{X}$ is reshaped into $\mathbf{X} \in \mathbb{R}^{S_1\times S_2}$, where S_2 indexes the sample set used to compute the statistics. For example, in $\Pi_{BN}(\mathbf{X}) \in \mathbb{R}^{D\times(mT)}$, the mT indicates that the statistics (mean and variance) are computed along the batch and sequence length (time steps) dimensions.

B. Statistical description of tensor element

Given a normed vector space V, we refer to the L_1 and L_2 norm to be the special cases of the following general L_p norm of a give vector $x \in V$, by setting $p = 1$ and $p = 2$:

$$\|x\|_{L^p} = \left(\sum_{j=1}^{\infty} |\xi_j|^p\right)^{1/p}$$

We evaluate the training stability using L_2 norms. We analyzed L2 norm of the weight matrix of the whole Mamba Block,including in_projection layer,conv1d layer,SSM Module,out_projection layer.

C. Implementation Details

C.1. ImageNet Experiment

Experiment Details We implemented our solution using VMamba's open-source code (Liu et al., 2024). In the original VMamba VSS Block, it not only includes the Mamba Block but also adds FFN and LN modules afterward. To avoid the impact of these modules and ensure a fair comparison, we removed the FFN and LN modules in our experiments, while keeping other parameter settings consistent with VMamba's default configuration, such as learning rate, model depth, etc. In subsection 4.5 Validation Experiment, due to computational cost and time constraints, the number of training epochs on the ImageNet-1k dataset was reduced from 300 to 100. The comparative values presented in Table 5 are the results from experiments trained for only 100 epochs.

C.2. ImageNet Experiment

Experiment Details We implemented our solution using VMamba's open-source code (Liu et al., 2024). In the original VMamba VSS Block, it not only includes the Mamba Block but also adds FFN and LN modules afterward. To avoid the impact of these modules and ensure a fair comparison, we removed the FFN and LN modules in our experiments, while keeping other parameter settings consistent with VMamba's default configuration, such as learning rate, model depth, etc. In subsection 4.5 Validation Experiment, due to computational cost and time constraints, the number of training epochs on the ImageNet-1k dataset was reduced from 300 to 100. The comparative values presented in Table 5 are the results from experiments trained for only 100 epochs.

C.3. ListOps Experiment

Dataset The Long Range Arena (LRA) benchmark (Tay et al., 2021) is designed to evaluate the ability of models to capture long-range dependencies across various tasks. The *ListOps* task requires models to process nested mathematical operations (e.g., `max`, `min`, `median`) on integers. For example:

$$\texttt{max}(2, \texttt{min}(3, 9), \texttt{max}(4, 5))$$

While the standard task uses sequences of length 1000, we employed *ListOps-New* with sequence length 2000 to further test Mamba's capabilities.

Experiment Details Experiments were conducted on 12 NVIDIA 4090 GPUs. The model configuration included 4 layers with a hidden dimension of 128 and state dimension of 64. We used the ADAMW optimizer (learning rate: 0.0001) with cosine_warmup scheduler and batch size 32. The architecture featured a convolution kernel size of 4 and expansion factor of 2, trained for 30 epochs.

C.4. IMDB Experiment

Dataset The IMDB dataset (Maas et al., 2011) contains 50,000 movie reviews (25,000 each for training and testing) labeled as positive or negative. In our experiments, reviews were encoded at character level with maximum sequence length of 4096.

Experiment Details The model comprised 4 layers with hidden dimension 128 and state dimension 64. Training utilized ADAMW optimizer (learning rate: 0.0001, weight decay: 0.1) with constant_warmup scheduler (2000 steps). The architecture included convolution kernel size 4 and expansion factor 2. Training ran for 65 epochs with batch size 32 on a single GPU. The random seed was set to 2222.

C.5. CIFAR Experiment

Dataset The CIFAR dataset (Krizhevsky et al., 2009) includes CIFAR-10 and CIFAR-100 versions. Both contain 32×32 pixel images, with CIFAR-10 having 10 classes (6,000 images per class) and CIFAR-100 having 100 classes (600 images per class). Each version splits into 50,000 training and 10,000 test images.

Experiment Details For CIFAR-100, images were converted to grayscale. The model used 4 layers with hidden dimension 64. Training employed ADAMW optimizer (learning rate: 0.001, weight decay: 0.1) with cosine_warmup scheduler (2000 steps). The model trained for 40 epochs with batch size 50.

C.6. Pathfinder Experiment

Dataset The Pathfinder-64 dataset (Linsley et al., 2018) consists of 64×64 pixel grayscale images, designed to test models' ability to identify connecting paths between two points, emphasizing long-range spatial dependencies.

Experiment Details The Mamba_Norm model used 6 layers with hidden dimension 128. Training utilized ADAMW optimizer (learning rate: 0.0002, weight decay: 0.05, $\beta_1 = 0.9$, $\beta_2 = 0.95$) with constant_warmup scheduler (5000 steps). The architecture included state dimension 64, kernel size 4, and expansion factor 2. Training ran for 50 epochs with batch size 32 across 10 GPUs.

D. NormVary Strategy

D.1. Algorithm Description

The motivation for this design stems from our observation that, after determining that the normalization after the SSM layer in the Mamba architecture is most effectively set to GN or LN, automatically selecting the most suitable normalization based on gradient updates during training can efficiently handle features with different scales and modalities. This greatly simplifies the selection of the appropriate normalization combination for different data distributions. NormVary1d is an adaptive normalization algorithm that dynamically integrates five different normalization strategies through a learnable

weight mechanism. While preserving the unique advantages of various normalization methods (BN, GN, IN, LN, RMSN), it adaptively adjusts their importance using a softmax weighting approach, achieving optimal normalization across different data distributions and network layers. The algorithm introduces independent mean and variance weight parameters, allowing for more precise control of feature statistics, while ensuring training stability through a moving average mechanism.

D.2. Mathematical Representation

D.2.1. Input

Given that Mamba is specifically designed for sequence modeling tasks, we consider sequence data as our primary example. For a typical sequence input tensor $X \in \mathbb{R}^{m \times D \times L}$, where m represents the batch size, D denotes the feature dimension, and L corresponds to the sequence length. This three-dimensional representation is particularly suited for sequential data processing, where each sequence in the batch can be effectively normalized across different dimensions while preserving temporal dependencies.

D.2.2. Statistical Computation

1. Batch Normalization:

$$\mu_{BN} = \frac{1}{m}\sum_{n=1}^{m}\mu_{IN,n}, \quad \sigma^2_{BN} = \frac{1}{m}\sum_{n=1}^{m}\sigma^2_{IN,n} \tag{8}$$

2. Group Normalization ($g \in [1, G]$):

$$\mu_{GN} = \frac{1}{(D/G)L}\sum_{c\in g}\sum_{l=1}^{L}x_{n,c,l}, \quad \sigma^2_{GN} = \frac{1}{(D/G)L}\sum_{c\in g}\sum_{l=1}^{L}(x_{n,c,l} - \mu_{GN})^2 \tag{9}$$

3. Instance Normalization:

$$\mu_{IN} = \frac{1}{L}\sum_{l=1}^{L}x_{n,c,l}, \quad \sigma^2_{IN} = \frac{1}{L}\sum_{l=1}^{L}(x_{n,c,l} - \mu_{IN})^2 \tag{10}$$

4. Layer Normalization:

$$\mu_{LN} = \frac{1}{D}\sum_{c=1}^{D}\mu_{IN,n,c}, \quad \sigma^2_{LN} = \frac{1}{D}\sum_{c=1}^{D}\sigma^2_{IN,n,c} \tag{11}$$

5. RMS Normalization:

$$\mu_{RMS} = 0, \quad \sigma^2_{RMS} = \frac{1}{DL}\sum_{c=1}^{D}\sum_{l=1}^{L}x^2_{n,c,l} \tag{12}$$

D.2.3. Weight Fusion

We use softmax function to compute normalization weights:

$$w_i = \frac{e^{z_i}}{\sum_{j=1}^{5}e^{z_j}}, \quad \mu = \sum_{i=1}^{5}w_{\mu,i}\mu_i, \quad \sigma^2 = \sum_{i=1}^{5}w_{\sigma,i}\sigma^2_i \tag{13}$$

D.2.4. Fusion Normalization

$$y = \gamma\frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{14}$$

D.2.5. TRAINING UPDATES

$$\mu_{running} = \alpha\mu_{running} + (1-\alpha)\mu_{BN} \tag{15}$$

$$\sigma^2_{running} = \alpha\sigma^2_{running} + (1-\alpha)\sigma^2_{BN} \tag{16}$$

NormVary1d implements dynamic integration of multiple normalization methods through an adaptive weight learning mechanism. The algorithm automatically adjusts the importance of different normalization methods during training based on changes in data distribution, allowing each method to contribute its advantages. By introducing independent mean and variance weight parameters, along with a moving average mechanism, the algorithm provides more precise control over feature statistics while maintaining model stability. This design enhances the model's performance and offers an effective normalization solution for handling complex sequence data.