AN EMPIRICAL STUDY ON NORMALIZATION IN MAMBA

Anonymous authors
Paper under double-blind review

ABSTRACT

Normalization layers are crucial for enhancing both the training efficiency and stability of deep neural network architectures. The recently proposed Mamba network has shown considerable promise in achieving competitive results alongside Transformers. However, ensuring training stability in Mamba, as in many deep architectures, remains a significant challenge, where normalization techniques are crucial for solving this issue. In this paper, we conduct a systematic investigation into the effects of various normalization techniques and their combinations within the Mamba architecture. Our analysis encompasses both long sequence modeling and image classification tasks. For long sequence modeling, we perform extensive experiments to assess the impact of applying normalization layers both before and after S6 Modules. The results show that normalization across layers leads to enhanced training stability and improved model performance. Furthermore, we validate these findings on large-scale V-Mamba models and offer practical suggestions for selecting appropriate normalization techniques. We hope that our insights will contribute to mitigating training instabilities in deep learning and fostering the development of more robust architectures. All code and models used in this study will be open-sourced on GitHub.

1 INTRODUCTION

The Mamba architecture (Gu & Dao, 2023) is a state-of-the-art model designed to efficiently handle long sequences in sequence modeling (Gu & Dao, 2023), leveraging structured state-space models (SSM) (Gu et al., 2021). However,training Mamba presents significant challenges due to the inherent unpredictability of state space model(SSM). Normalization plays a crucial role in the effective training of deep neural networks (Huang et al., 2023), particularly for architectures like Mamba. There are various usage of normalization techniques in the variants of Mamba architecture (Liu et al., 2024; Liang et al., 2024), but without providing the sufficient evidences for supporting why one should use like that. Therefore, it is essential to conduct a comprehensive investigation into various normalization strategies to optimize Mamba's performance. In this study, we examine the impact of several widely used normalization techniques, including Batch Normalization (BN) (Ioffe, 2015), Layer Normalization (LN) (Ba, 2016), Group Normalization (GN) (Wu & He, 2018), Instance Normalization (IN) (Huang & Belongie, 2017), and Root Mean Square Normalization (RMSN) (Zhang & Sennrich, 2019). We systematically evaluate the effect of normalization type, position, and combinations on Mamba's performance and stability in sequence modeling and image classification tasks.

For sequence modeling tasks, we found that placing normalization before the S6 layer led to performance degradation while placing it after the S6 Module led to performance improvement.Certain combinations of normalization techniques applied before and after the S6 Module enhance the model's performance, while some can be detrimental. Our analysis, rooted in the scale-invariant property of deep learning (Papyan, 2018), revealed that the weight matrices (Huang et al., 2020) in deeper Mamba blocks exhibited significantly larger L2 norms (Luo et al., 2016) than earlier layers. Introducing normalization before the S6 Module exacerbated this issue, whereas placing normalization after the S6 Module helped the model enter a state of *Weight Domination*, whcih is beneficial to keeping scale-invariant property. However, *Weight Domination* also limited the extent to which weights could be updated. By applying normalization before the S6 Module, we found that this *Weight Domination* phenomenon could be alleviated, leading to led to more effective parame-

ter updates under stable training conditions. Finally, we propose an intution for selecting optimal normalization combinations to improve both the efficiency and stability of Mamba.

For image classification tasks, we validated the conclusions derived from sequence modeling. We explored the impact of position on performance by examining BN, IN, LN, RMSN, GN, only RMSN underperformed. However, by positioning the normalization layers before and after the Mamba blocks, we aimed to determine whether the placement of these techniques affected network performance. For IN, GN, and LN, placing normalization before or after the Mamba blocks consistently reduced validation loss and improved accuracy. However, when BN or RMSN were placed before the Mamba blocks, validation loss increased and accuracy declined, while placing these layers after the Mamba blocks resulted in unstable performance. These results suggest that LN normalization techniques are robust to positional changes, whereas BN and RMSN are sensitive to position.

Finally, motivated by those findings in sequence modeling, we investigated combinations of normalization techniques before and after the Mamba blocks to maximize network performance. We explored extensive combinations, such as BN+RMSN, LN+BN, LN+RMSN, etc. Surprisingly, even normalizations that previously detrimented performance,such as RMSN and BN led to combined improvements. Notably, the RMSN+BN combination yielded faster and more significant improvements than the other combinations like LN+LN. Compared to the self-combination of RMSN, IN, GN, and LN, cross-combined with BN resulted in faster and greater reductions in validation loss and higher accuracy. However, LN+RMSN underperformed relative to LN+LN. These results demonstrate: (a) the order of normalization placement affects the Mamba network's performance; (b) placing LN before the Mamba block and another normalization technique after helps mitigate the negative effects of certain normalizations; (c) BN is particularly effective when placed after the Mamba block and is an essential component for enhancing the Mamba network's performance when combined with other normalization techniques.

Contributions In this work, we systematically investigate widely used normalization techniques and aim to address the following questions:

(1) Normalization Position: Where should normalization be placed in the model to ensure high performance and stable training? We found that applying normalization layer after the S6 Module SSM yields better results(if only one can be choosen). However, combining certain normalization types can achieve more outstanding performance.

(2) Normalization Combanition: What are the effects of different normalization combinations? And what is the intuition behind these effects? We found that certain combinations of different normalization techniques produce excellent results.

(3) Combination intution: How can we choose the proper combination?We propose an intution for harmonizing normalization strategies in deep architectures, providing practical guidelines for selecting normalization methods.

2 PRELIMINARY

2.1 MAMBA FOR SEQUENCE MODELING

Mamba (Gu & Dao, 2023) is a state-of-the-art architecture based on the State Space Model (SSM) (Hamilton, 1994). For one-dimensional sequence modeling, we define the sequence as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\} \in \mathbb{R}^{L \times D}$, where L represents the sequence length and D denotes the number of feature channels. Each element $\mathbf{x}_l = \{x_l^1, x_l^2, \ldots, x_l^D\} \in \mathbb{R}^D$ contains D feature values at time step l. For a mini-batch of data with m samples, the input tensor is typically shaped as (m, D, L).

Not only does Mamba excel in handling one-dimensional sequences (Gu & Dao, 2023), but it also extends effectively to two-dimensional image data (Liu et al., 2024).

Because image data usually undergoes a flatten operation to transform the $h * w$ dimension into sequence length L before entering the model. In particular, to allow each pixel to better gather information from all directions and establish a global receptive field in the 2D space, a 2D-Selective-Scan (SS2D) method is proposed in VMamba. In this method, images are divided into patches, and each patch is processed as a sequence along four different traversal paths (Cross-Scan) using parallel S6 blocks (Mamba). The processed sequences are then reshaped and merged to form the

output map (Cross-Merge), thus enabling Mamba to efficiently process both 1D and 2D data. In our experiments, to better study the Mamba block's nature, we removed the FFN block, which is an MLP branch following the VSS Block in VMamba. For VMamba details, please refer to (Liu et al., 2024).

2.2 NORMALIZATION

Normalization is a crucial technique in deep learning (Hinton & Salakhutdinov, 2006; Ioffe, 2015; He et al., 2015; Huang et al., 2023) that aims to stabilize and accelerate the training process of neural networks (Desjardins et al., 2015), as defined in Equation 1. It helps adjusting input data distribution or intermediate activations into normal distribution (Huang et al., 2023) that is beneficial tothe convergence of model training. Furthermore, normalization methods, such as BN,LN and GN address issues like internal covariate shift (Wang et al., 2022), which occurs when the distribution of inputs to each layer changes during training (Wang et al., 2022). Models can maintain consistent representation learning and effectively manage varying input scales by incorporating normalization layers. Thus, selecting the appropriate normalization method according to specific requirements is essential for training with complex date or large parameters.

$$\text{Norm}(x) = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \cdot \gamma + \beta \tag{1}$$

Various normalization variants have been proposed, including Iterative Normalization (ItN) for addressing normalization computational inefficiencies (Huang et al., 2019), Scale Normalization (Scale) for ensuring consistent value scaling of features (Papyan, 2018). However, the five most widely adopted methods by researchers are still BN, LN, GN, IN, and RMSN, as they have been well-integrated into pytorch libraries and cater for most models. Accordingly, our experiments focus on these commonly used normalization methods.

Huang et al. (2023) proposed a unified taxonomy for understanding the similarities and differences between these approaches, specifically, Normalization Representation Area Partitioning (NAP), Normalization Operation (NOP), and Normalization Representation Recovery (NRR). The operational details of these mainstream techniques are clearly presented in Table 1.

Table 1: Details of Mainstream Normalization techniques: We take sequence $\mathbf{X} \in \mathbb{R}^{m \times D \times L}$ 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 set of samples used to compute the statistics. For example, $\mathbf{X} = \Pi_{BN}(\mathbf{X}) \in \mathbb{R}^{D \times (mL)}$, where mL indicates that the statistics (mean and variance) are calculated along the batch, sequence length (time steps)

Method	**NAP**	**NOP**	**NRR**
Batch Normalization (BN)	$\Pi(\mathbf{X}) \in \mathbb{R}^{D \times mL}$	Standardizing	Learnable $\gamma, \beta \in \mathbb{R}^D$
Instance Normalization (IN)	$\Pi(\mathbf{X}) \in \mathbb{R}^{mD \times L}$	Standardizing	Learnable $\gamma, \beta \in \mathbb{R}^D$
Group Normalization (GN)	$\Pi(\mathbf{X}) \in \mathbb{R}^{mg_D \times s_D L}$	Standardizing	Learnable $\gamma, \beta \in \mathbb{R}^D$
Layer Normalization (LN)	$\Pi(\mathbf{X}) \in \mathbb{R}^{m \times DL}$	Standardizing	Learnable $\gamma, \beta \in \mathbb{R}^D$
Root Mean Square Norm (RMSN)	$\Pi(\mathbf{X}) \in \mathbb{R}^{m \times DL}$	Scaling	Learnable $\gamma \in \mathbb{R}^D$

3 THE MAMBA LANDSCAPE

Due to the inherent unpredictability of state space transitions, especially when introducing the multi-head mechanism, Mamba is challenging to train stably during the training process. To mitigate this issue, it becomes essential to implement effective strategies for stabilizing the training dynamics. One such key solution involves incorporating an extra normalization layers in the architecture. In Section 3.1, we describe the process of sequence modeling using Mamba architecture and specify

the positions of the normalization layers we investigated(In Figure 1, the orange layers: Normalization1,before inprojection layer which norms the input activations and Normalization2 ,before outprojection layer which norms the SSM output.). Followed by a detailed discussion of the normalization techniques explored in Section 3.2.

3.1 Mamba Landscape Details

As shown in Figure 1, the input data is first processed in the data factory,the sequence data will be extracted with spatio-temporal features, and the image data will undergo scanning and flatten operations to become sequence data.After being embedded, the data will be projected into high-dimensional feature space through the inprojection layer. A convolution operation is then applied to fuse information from adjacent time steps, followed by processing through the SSM layer and variants including S4 (Gu et al., 2021),S6 (Gu & Dao, 2023),SSD (Dao & Gu, 2024). The output is subsequently combined with the data from the feedforward MLP channel, undergoes an extra normalization layer, and is finally projected to the desired dimensions through the outprojection layer.

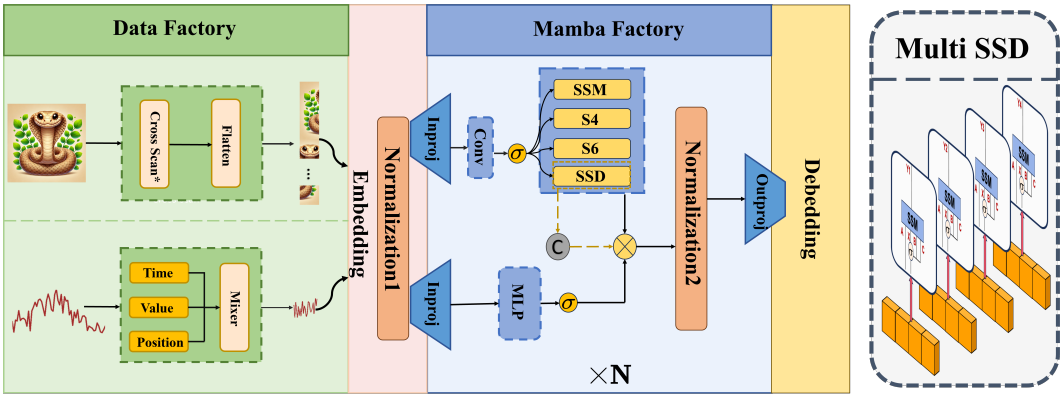

Figure 1: Illustration Data Flow in Mamba Architecture and Our Research Focus:

In previous Mamba varients, many of them has introduced the normalization operation right after the S6 Module. But there is no unified study on where and what normalization to introduce in the architecture. We have extensively explored the the type,position,combanition of different normalization techiniques.

3.2 Extra Normalization

The normalization in Mamba can be broadly categorized into two types:

1. Normalization applied to the input activations before entering the S6 Module.
2. Normalization applied to the output activations after the S6 Module.

We first define the notation of normalization to describe the position of normalization layers within a block. Let Norm1→Norm2 denote the normalization configuration within a block, where Norm1 refers to the normalization applied before inprojection layer, and Norm2 refers to the normalization applied before the output projection layer. For instance, the notation BN→IN represents the application of Batch Normalization before the inproj and Instance Normalization before the outproj, as depicted in Figure 2.

3.3 Datasets

For long sequence modeling, we use the LRA ListOps task (Tay et al., 2021), a challenging benchmark designed to test a model's capacity to handle long-range dependencies. The task requires the model to classify sequences of nested operations applied to integers, making it a strong indicator

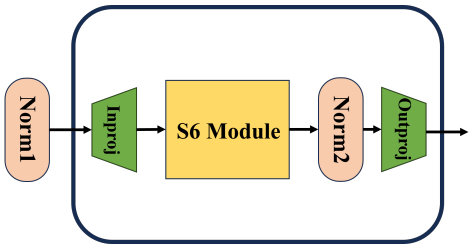

Figure 2: Norm Location: Norm1 is responsible for normalizing the input activations, while Norm2 normalizes the output of the S6 Module (Mamba Block).

of a model's ability to process long sequences. Furthermore, we extend the difficulty by utilizing a longer version of ListOps with an input sequence length of 2000 tokens, making the task more demanding for the model to capture distant dependencies effectively.

For vision tasks, we use the ImageNet-100 (Krizhevsky et al., 2012) classification dataset. ImageNet-100 is a randomly selected subset from the ImageNet-1k dataset of the 2012 Large Scale Visual Recognition Challenge. It contains 100 categories, spanning various objects and scenes, ensuring diversity in vision tasks. The training set has 1,300 images per category, and the validation set contains 50 images per category, totaling 135k images.

3.4 Exploring The Mamba Normalization Landscape

We conduct a comprehensive investigation of commonly used normalization techniques, analyzing their type,position and combinations in both long sequence modeling and image classification tasks. In Section 4.1, we fix everything but combinationevaluate the effects of applying normalization before or after the S6 Module. Our results indicate that applying normalization after the S6 Module usually improves model , while appending it before the S6 Module leads to degraded performance. In Section 4.2, we examine the impact of combining different normalization techniques, with certain combinations resulting in significant performance improvements. A detailed analysis of these findings will follow. Finally, in Section 4.3, we validate these results using the Mamba architecture for image classification, further confirming the observed trends in sequence modeling.

4 Experimental Results and Discussion

4.1 Normalization Position

Table 2: Accuracy Comparison: The left column shows normalization applied before the S6 Module, the middle column shows it applied after, and the right column shows normalization applied both before and after the S6 Module.Underlined None → None is the baseline model.The first arrow(↑) compares with the first column, and the second arrow(↑) compares with the second column.

Normalization	**Accuracy**	**Normalization**	**Accuracy**	**Normalization**	**Accuracy**
None → None	0.390	None → None	0.390	None → None	0.390
BN → None	0.389	None → BN	0.426(↑)	BN → BN	0.439(↑↑)
IN → None	0.357	None → IN	0.380(↑)	IN → IN	0.406(↑↑)
LN → None	0.374	None → LN	0.394(↑)	LN → LN	0.381 (↑↓)
RMSN → None	0.379	None → RMSN	0.394(↑)	RMSN → RMSN	0.401(↑↑)
GN → None	0.390	None → GN	0.394(↑)	GN → GN	0.414(↑↑)

The results of experiments are summarized in Table 2. We can draw three conclusions:

1. We observed that adding normalization before the S6 Module weakens the model's predictive performance, with Instance Normalization (IN) showing the most significant decline.

2. However, when normalization is applied after the S6 Module, the model's predictive accuracy improves compared to normalization applied before the S6 Module, and all results outperform those without normalization.

3. When the same normalization is applied both before and after the S6 Module, there is a further improvement in predictive accuracy. Except for Layer Normalization (LN), all other normalization techniques show additional improvements when applied both before and after the S6 Module.

We plotted the L2 norms of the weights in each Mamba block and conducted an in-depth analysis of the effects of different normalization techniques in a 4-layer model in the following Figure 3. We

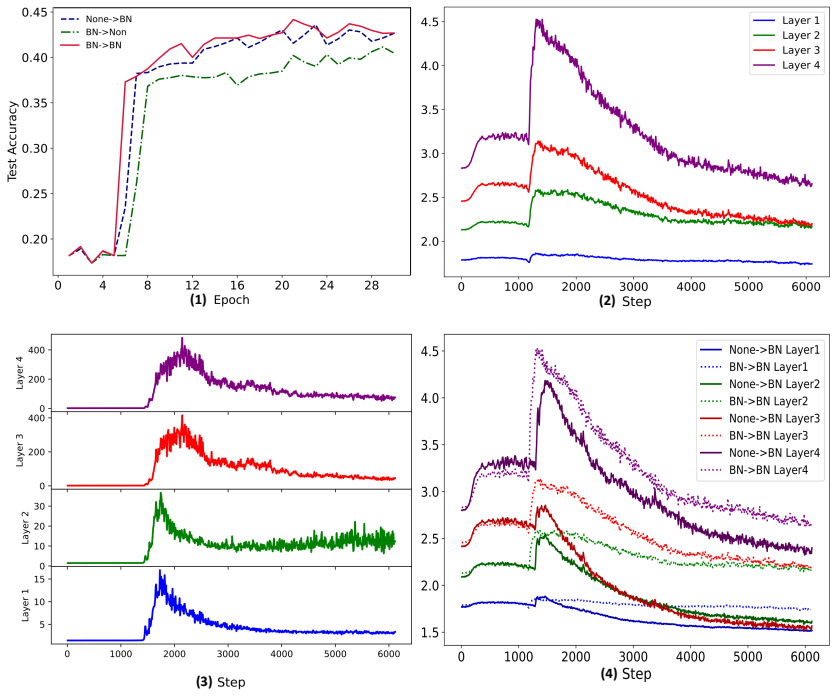

Figure 3: The Performance and L2 norm about BN→BN compared with None→BN and BN→None

present the test accuracy for three configurations: None→BN, BN→None, and BN→BN. Subfigure in subfigure (1), show the L2 norms of the weight parameters for each block in the None→BN configuration in subfigure (2) , depict the L2 norms of the weight parameters for each block in the BN→None configuration in subfigure (3) and illustrate the L2 norms of the weight parameters for each block in the BN→BN configuration in subfigure (4). The solid line represents the results of the None→BN configuration, while the dashed line corresponds to the results of the BN→BN configuration.

We observe that in the BN→None configuration, as the network layer becomes deeper, there is a significant disparity in the L2 norms of the weight parameters across layers, indicating a large-scale variation between layers. This suggests that the later layers of Block dominate the weighting, which is not conducive to network stability. In contrast, in the None→BN configuration, the magnitude of the L2 norms of the weight parameters remain within the same across layers. This indicates smaller scale variations between layers, leading to more stable training. Last but not least, we also observe that the updates in the weight norms are relatively limited across all layers, suggesting that the model enters a state of *Weight Domination*, making it harder to update the parameters effectively. In the BN→BN configuration, the weight norms across layers remain within the same magnitude but with larger magnitude of the update in the L2 norms. This implies that applying normalization before the S6 Module mitigates the issue of *Weight Domination*, facilitating more effective parameter updates. For more cases, please refer to Appendix C

4.2 NORMALIZATION COMBANITION

Additionally, we tried different normalization combinations and listed the excelled ones in the table3.Then,we provide valuable guidance for selecting optimal normalization combinations. For instance, here is the behavior of BN and IN combination. Our analysis reveals that these two normalization techniques exhibit complementary effects on model weights in the final block,which we refer to as "harmonic structure".We observed that the weight matrix Norm updates in different directions and has a large spacing when the two normalizations act alone. The Norm of BN→IN, on the other hand, is exactly the balance of the two. This leads to a 10% improvement in performance compared to using either normalization individually (see Left Figure 4).

Table 3: Combination Comparison: Here are the top 5 combinations and the full results can be found in the table7.

Normalization	**Accuracy**
BN → IN	0.469
IN → GN	0.468
GN → IN	0.448
RMSN →BN	0.448
BN → BN	0.439
None → None	0.390

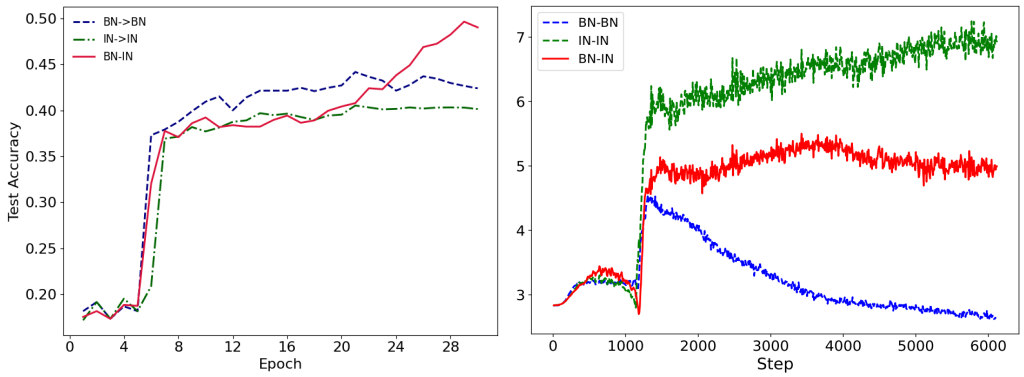

Figure 4: The Performance and L2 in layer 4 of BN→IN

4.3 VISION MAMBA

For vision tasks, we use a tiny version of the VMamba model for image classification . The model we use has a depth of 15 Mamba blocks and contains three downsampling layers. The original Mamba blocks consist of normalization layers, inprojection layer, convolutional layers, activation layers, the S6 module, and feedforward MLP layers, with the same normalization layer applied both before and after the S6 module. Due to the multiple components involved, different combinations can form various styles of Mamba blocks.

Given the important role of normalization layers in network design and training, we first investigate the impact of the normalization layer's position within the Mamba block on model performance. Since the core of the Mamba block is the S6 module, we consider the effect of the normalization layer's position relative to the S6 module, i.e., whether it is placed before or after the S6 module. Currently, the same normalization layer is used both before and after the S6 module, leading to a rather uniform approach. This prompted us to explore whether other diverse combinations of normalization layers could improve model performance, and we further experimented with the effects of combining different normalization layers. The experimental results are shown in Table 3 and Table 4, respectively.

Table 4: The impact of normalization layer positioning relative to S6 module, evaluated with five representative normalization layers. LN → LN is the baseline model adopted in original Vmamba. Accuracy1 represents top 1 accuracy. The first arrow (↑) compares with the first column, and the second arrow (↑) compares with the second column.

Normalization	Accuracy	Normalization	Accuracy	Normalization	Accuracy
None → None	0.196	None → None	0.196	None → None	0.196
BN → None	0.167	None → BN	0.678(↑)	BN → BN	0.746(↑↑)
IN → None	0.702	None → IN	0.838(↑)	IN → IN	0.863(↑↑)
LN → None	0.864	None → LN	0.867(↑)	LN → LN	0.866
RMSN → None	0.294	None → RMSN	0.176(↓)	RMSN → RMSN	0.072(↓↓)
GN → None	0.660	None → GN	0.868(↑)	GN → GN	0.863(↑↓)

From Table 4, we found that placing the normalization layer after the S6 module significantly improves the classification performance of the VMamba model. Among them, GN and LN are superior choices. When they are placed after the S6 module, the model achieved accuracies of 0.868 and 0.867, respectively, while IN also performed well with 0.838. Conversely, placing the normalization layer before the S6 module led to a significant performance drop. For example, BN → None only achieved 0.167, whereas None → BN reached 0.678. RMSN showed the worst performance, regardless of whether it was placed before or after SS, with lower accuracy compared to other methods.These two conclusions corroborate what we drawn in the sequence experiments.

Table 5: The impact of different normalization layer combinations on the VMamba model, sorted by accuracy.

Normalization	Accuracy1	Normalization	Accuracy1	Normalization	Accuracy1
RMSN → BN	0.876	BN → RMSN	0.845	None → BN	0.678
GN → BN	0.872	IN → IN	0.837	RMSN → None	0.294
LN → BN	0.871	GN → RMSN	0.836	None → None	0.196
IN → BN	0.867	LN → RMSN	0.836	None → RMSN	0.176
LN → LN	0.866	IN → RMSN	0.821	BN → None	0.167
GN → GN	0.863	BN → BN	0.746	RMSN → RMSN	0.072

Table 5 results show that combining different normalization layers has a significant impact on the performance of the S6 module in the VMamba model for image classification tasks. Notably, the combination of RMSN → BN achieved the highest accuracy of 0.876, while standalone RMSN → RMSN performed the worst, only 0.072. This indicates that a proper combination of normalization layers can greatly enhance model performance.

We believe that the BN normalization layer is better suited for handling the features processed by the S6 module, thus improving the overall model performance. In contrast, using RMSN alone may disrupt the input feature distribution, leading to a significant performance drop.

We also explored the impact of normalization at different positions on training the VMamba model. The training loss and validation accuracy are shown in Figure 6. In Figure 6, the same color represents the same normalization layer, with dashed and solid lines indicating normalization applied before and after the S6 module, respectively. From 6, we found that shifting the position of GN and IN normalization from before to after the S6 module transforms their negative impact on training acceleration into a positive one. Additionally, LN normalization shows robustness to positional changes, as it consistently accelerates model convergence whether applied before or after the S6 module.

To more intuitively demonstrate the impact of different normalization layers on training VMamba, we also plotted the curves of training loss and validation accuracy during the training of VMamba models with different normalization layers and positions, as shown in Figure 5. BN and RMS normalization layers degraded the performance of the Mamba network architecture, while LN, GN, and IN improved its performance. Additionally, we plotted the training loss and validation accuracy when combining the performance-degrading BN and RMSN layers with other normalization layers, as shown in Figure 7. We observed that their negative effects were eliminated when com-

bined with other normalization layers. Furthermore, the normalization layers combined with BN all outperformed their individual performance.

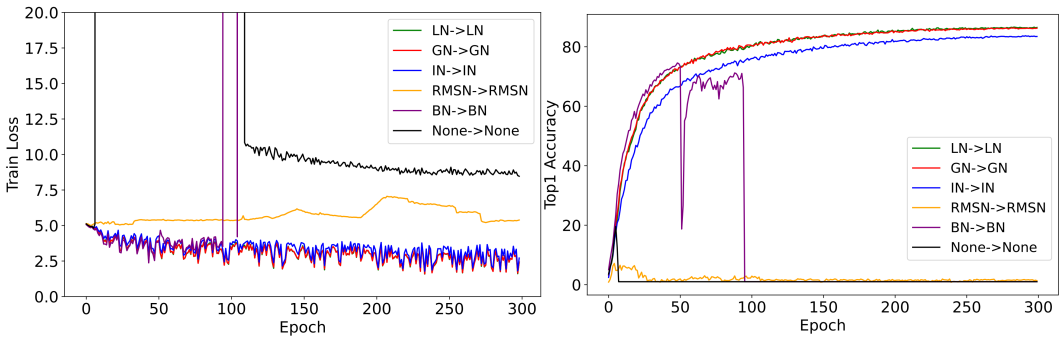

Figure 5: Impact of different Normalization on Vmamba performance. BN and RMS normalization layers degraded the performance, while LN, GN, and IN improved VMamba performance.

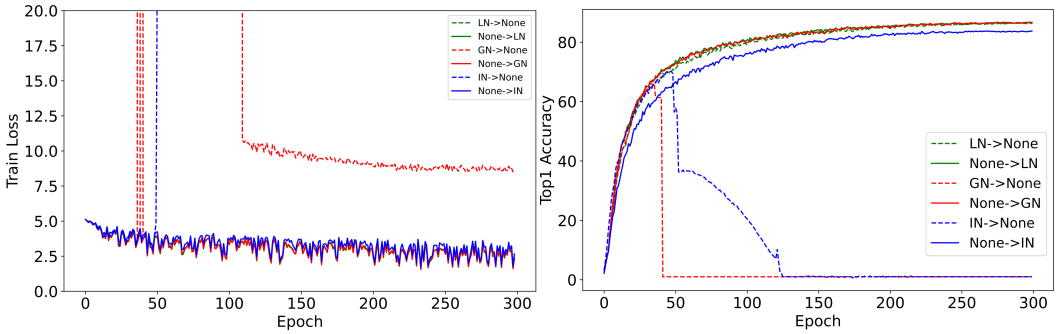

Figure 6: Influence of Location of Normalization:The solid line represents the results where normalization is applied after the S6 Module, while the dashed line represents the results where normalization is applied before the S6 Module

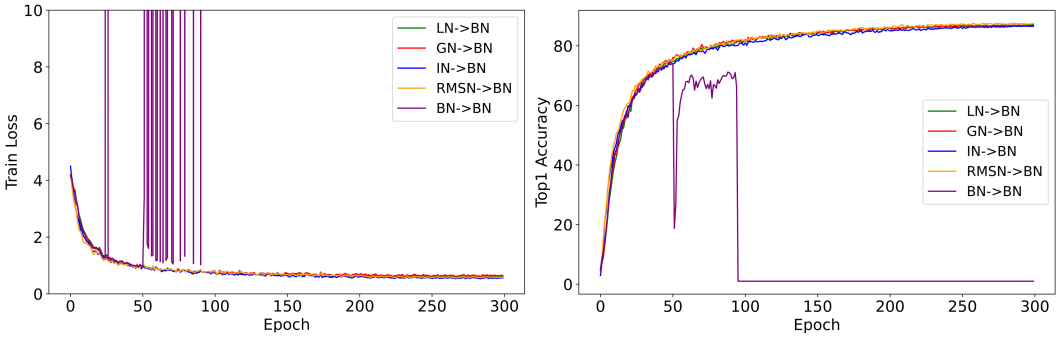

Figure 7: The results of combination other normalization with BN. From the decreasing loss and increasing accuracy, BN improves the performance when combined with other normalization layers, but reduces performance if used alone.

5 Conclusions and Future Work

In this study, we conducted a comprehensive investigation into the types, positions, and combinations of normalization layers within the Mamba architecture. Our findings reveal that applying normalization after the S6 Modules enhances training stability by mitigating large variations in weight

norms. Moreover, specific combinations of normalization techniques not only stabilize the training process but also lead to significant improvements in model performance.

We also proposed an intution for selecting and combining normalization layers to optimize the training of Mamba and other deep architectures. This intution provides valuable insights for managing weight domination and addressing the challenges of training large-scale neural networks. Future research will focus on extending this intution to more complex models and tasks, to refine normalization strategies to enhance both efficiency and robustness further.

The Mamba2 architecture has recently been introduced, and Dao & Gu (2024) found that training Mamba2 is less stable compared to Mamba1. Building upon our research into the normalization of Mamba Blocks, future work could further explore the Mamba2 architecture to address its stability challenges. Additionally, we hope our findings can be extended to other deep learning architectures, offering valuable guidance for designing more complex models.

REFERENCES

Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al. Natural neural networks. *Advances in neural information processing systems*, 28, 2015.

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

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

James D Hamilton. State-space models. *Handbook of econometrics*, 4:3039–3080, 1994.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Lei Huang, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Iterative normalization: Beyond standardization towards efficient whitening. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4874–4883, 2019.

Lei Huang, Jie Qin, Li Liu, Fan Zhu, and Ling Shao. Layer-wise conditioning analysis in exploring the learning dynamics of dnns. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 384–401. Springer, 2020.

Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE transactions on pattern analysis and machine intelligence*, 45(8):10173–10196, 2023.

Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 1501–1510, 2017.

Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12), 2020.

Aobo Liang, Xingguo Jiang, Yan Sun, and Chang Lu. Bi-mamba4ts: Bidirectional mamba for time series forecasting. *arXiv preprint arXiv:2404.15772*, 2024.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model 2024. *CVPR*, 2024.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *Learning*, 10, 2016.

Xiong Luo, Xiaohui Chang, and Xiaojuan Ban. Regression and classification using extreme learning machine based on l1-norm and l2-norm. *Neurocomputing*, 174:179–186, 2016.

Vardan Papyan. The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

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

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

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

A TRAINING STABILITY

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,S6 Module,out_projection layer.

B MAMBA ARCHITECTURE

State space modelling is a method of describing and analysing a system based on matrix theory. Introducing state variables can get more in-depth information about the system. SSM uses first-order differential equations to map the input function x_t to the output function y_t through hidden state h_t, defined as follows:

$$h_t = \boldsymbol{A}h_{t-1} + \boldsymbol{B}x_t, \quad y_t = \boldsymbol{C}h_t + \boldsymbol{D}x_t \tag{2}$$

where state transition matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$,input matrix $\boldsymbol{B} \in \mathbb{R}^{N\times C}$,output matrix $\boldsymbol{C} \in \mathbb{R}^{C\times N}$and forward channel transition matrix $\boldsymbol{D} \in \mathbb{R}^{C\times C}$. The variables N and C refer to the hidden state and dimension factors, respectively. Continuous parameters $\boldsymbol{A}, \boldsymbol{B}$ can be discretized by a first-order difference method or a bilinear transformation method as follows $\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}}$, with the sampling interval Δ. Details of the process can be found in (Gu et al., 2021). We give the discretization result directly here. What should be noted is that the hidden state update mechanism of SSM is similar to Recurrent Neural Network (RNN), which receives current time step input x_t and the previous time step hidden state h_{t-1} and computes the current time step hidden state h_t.

The key design principle of Mamba lies in the introduction of a selective mechanism to parameterize the transition matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \Delta$ in a data-driven manner. In Mamba1, these matrices are defined as functions of the input embedding features, allowing them to adapt dynamically to the data context through a hardware-aware parallel computing algorithm. This enables efficient processing of long sequences with improved computational throughput. In contrast, Mamba2 leverages semi-differentiable matrix factorization to compute the hidden state space efficiently, ensuring that any state space model with state size N and sequence length L can be computed in time $O(TN)$, which means that Mamba2 preserves the controllability of the state space across various sequence lengths.

C IMPLEMENTATION DETAILS

C.1 LISTO 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. One of the tasks included in the LRA benchmark is the *ListOps* task, which is particularly useful for assessing a model's capability to handle hierarchical structures and process long-range sequences.

In the *ListOps* task, the input sequence is represented as a nested list of operations. These symbolic expressions involve various operations, such as `max`, `min`, and `median`, applied to integers. The task requires the model to parse the input sequence, resolve the nested operations, and compute the final result. For example, a typical input could be:

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

In this example, the operations `max`, `min`, and `max` are applied in a hierarchical structure. The model's objective is to correctly parse the operations and compute the final result, which in this case is 5.

The standard *ListOps* task involves input sequences of length 1000. However, to further test Mamba's capacity to handle long-range dependencies, we used an enhanced dataset, *ListOps-New*, which includes input sequences of length 2000. This extended version allows us to assess better the limits of Mamba's ability to manage and process extremely long sequences with complex, hierarchical operations.

Experiment Details All experiments were implemented in Pytorch(Listo (Tay et al., 2021)) and conducted on 12 NVIDIA 4090 24GB GPU using DDP (Li et al., 2020). We set the initial learning rate as 0.0001 and used the ADAMW optimizer (Kingma, 2014) for model optimization. The batch size was set to 32. To maintain stable training, we take the cosine_warmup scheduler (Loshchilov & Hutter, 2016). Detailed model configuration information is presented in Table 6

Table 6: Model Configuration Details

Encoder	**Value**
encoder_name	position
encoder_dropout	0.0
Layer	**Value**
layer_name	mamba
causal	false
Parameter	**Value**
dropout	0.0
n_layers	4
d_model	128
ss_state	64
d_conv	4
expand	2
epoch	30

Complete Result of Combanation We present the results of all possible combinations of commonly used normalization techniques in Table 7. We observed that combinations involving BN and GN often yield better performance. The combinations with GN tend to outperform those with BN, which we hypothesize is due to the inconsistencies between training and inference in BN, leading to performance degradation. The specific reasons behind this require further investigation in future studies.

Table 7: Accuracy Comparison for Various Normalization Combinations: The first arrow indicates a comparison with the combination in the first column, while the second arrow represents a comparison with the combination in the second column. Comparisons with close values are not marked.

Normalization	**Accuracy**	**Normalization**	**Accuracy**	**Normalization**	**Accuracy**	**Normalization**	**Accuracy**
BN → BN	0.439	IN → IN	0.406	BN → IN	0.469(↑↑)	IN → BN	0.376(↓↓)
BN → BN	0.439	RMSN → RMSN	0.401	BN → RMSN	0.401(↑)	RMSN → BN	0.448(↑↑)
BN → BN	0.439	LN → LN	0.381	BN → LN	0.386(↓↑)	LN → BN	0.418(↓↑)
BN → BN	0.439	GN → GN	0.414	BN → GN	0.401(↓↓)	GN → BN	0.429(↓↑)
IN → IN	0.406	LN → LN	0.381	IN → LN	0.398(↓↑)	LN → IN	0.412(↑↑)
IN → IN	0.406	RMSN → RMSN	0.401	IN → RMSN	0.392(↓↓)	RMSN → IN	0.372(↓↓)
IN → IN	0.406	GN → GN	0.414	IN → GN	0.468(↑↑)	GN → IN	0.448(↑↑)
LN → LN	0.381	RMSN → RMSN	0.401	LN → RMSN	0.393(↑↓)	RMSN → LN	0.400(↑)
LN → LN	0.381	GN → GN	0.414	LN → GN	0.404(↑↓)	GN → LN	0.425(↑↑)
RMSN → RMSN	0.401	GN → GN	0.414	RMSN → GN	0.423(↑↑)	GN → RMSN	0.416(↑)

C.2 IMAGENET EXPERIMENT

Experiment Details All experiments were implemented in Pytorch(Listo (Liu et al., 2024)) and conducted on 12 NVIDIA 4090 24GB GPU using DDP (Li et al., 2020). We set the initial learning rate as 0.001 and used the ADAMW optimizer (Kingma, 2014) for model optimization. The batch

size was set to 256. To maintain stable training, we take the cosine_warmup scheduler (Loshchilov & Hutter, 2016).

Result Showcase In Figure 8, we present the training loss and validation accuracy curves during the training of the VMamba model using different normalization layers and positioning them in different locations.

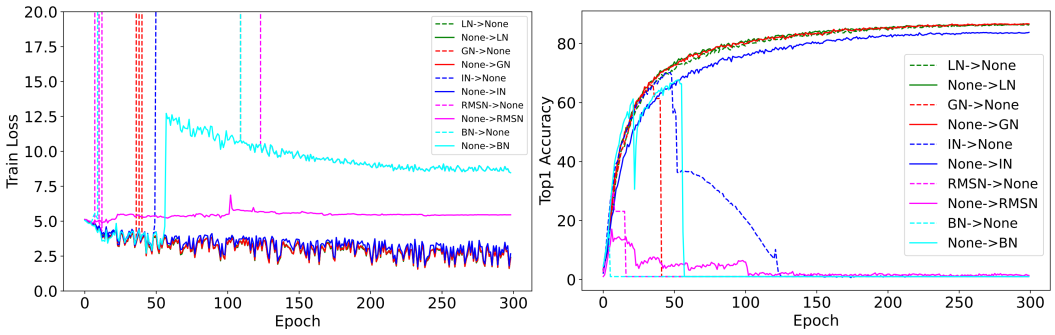

Figure 8: Impact of using five typical Normalization(BN, LN, IN, GN, RMS) in before or after S6 module. The solid line represents the results where normalization is applied after the S6 Module, while the dashed line represents the results where normalization is applied before the S6 Module.