Team 23: Exploring SSMs for Vision Tasks

Shreyansh Tripathi
7056894

Shahriyar Haseeb Mansoor
7063015

Abstract

In the recent few months, Mamba, which is a novel architecture based on the State Space Machines was introduced and has shown promise to efficiently model long range dependencies with faster training and inference times. However, since these models are new, there is limited work in checking the robustness of these models to transformations applied to images for tasks like classification. In this project, we analyze the performance of the VMamba model across various transformations and compare it with the performances of Swin-Transformer and ConvNext on these transformations. We observed that VMamba consistently performs better than SwinTransformer and ConvNext across all the transformations that were taken into account. We further fine-tuned UMamba models on a Road Segmentation dataset to check how well the model generalizes to a different task. We tried several fine-tuning methods like full fine-tuning and gradient freezing. Results show that fine-tuning the UMamba models produces good performance on Dice Metrics.

1. Introduction

Self-attention is a core mechanism that enables transformers [16] to excel at capturing contextual relationships between different elements in a sequence. However, the same thing is a disadvantage when it comes to tasks that have long-range dependencies due to quadratic computation complexity. This is because the self-attention mechanism scales quadratically with the input size making them resource-intensive. What makes State Space Models(SSMs) [6] different and advantageous is their linear-computational complexity during inference. The capability of efficiently handling long-range dependencies was introduced to the SSMs by the Structured State Space for Sequences (S4) [7] paper. The S4 was still naive in comparison to transformers in their ability to focus on particular inputs and ignore unnecessary information and therefore Mamba [5] was introduced to solve this problem. Mamba uses a selective scan algorithm which allows the model to filter irrelevant information and it's implementation was designed in a hardware-aware fashion.

Mamba was originally designed to process one-dimensional data but this poses a challenge while transferring the same methodology to image data which originally lacks a sequential arrangement of visual components. VMamba [10] was introduced wherein it proposes 2D Selective Scan (SS2D), a four-way scanning mechanism that allows spatial domain traversal and makes the image data suitable to be fed into the Mamba model. SS2D helps in bridging the gap between the ordered nature of 1D array scanning and the non-sequential structure of 2D data. SS2D ensures that each image patch gains contextual knowledge exclusively through a compressed hidden state computed along the corresponding scanning path, thereby enabling linear complexity. On the other hand, UMamba [13] was introduced with the motivation to apply SSMs for the task of biomedical image segmentation. CNNs [8] are inherently good at extracting local features and UMamba was essentially designed to make them capable of long-range dependency modeling.

2. Related Work

- **Mamba-Transformer Comparison for various tasks:** In previous work, Mamba has been compared against several baselines, including the standard Transformer architecture (GPT-3) [2] and Transformer++, which is based on the PaLM [3] and LLaMA [15] architectures. Notably, Mamba achieves 4-5× higher inference throughput compared to a Transformer of similar size, demonstrating its efficiency in handling inference tasks.

- **VMamba and UMamba:** The authors in VMamba paper have assessed the model on various downstream tasks including object detection and instance segmentation. For the task of image classification they evaluated the performance on ImageNet-1K [4], achieving top-1 accuracy of 83.9% with their Base model. U-Mamba has been compared with two CNN-based and two Transformer-based segmentation networks, both widely used in medical image segmentation competitions. U-Mamba outperforms both types of networks, achieving higher average Dice Similarity Coefficient (DSC) scores on abdomen CT and MR datasets, demonstrating its superior segmentation accuracy.

- **Finetuning Mamba:** Authors in [1] have used parameter-efficient finetuning technique, LoRa to finetune Mamba and Transformer based models on a questionanswering dataset SlimOrca [9] to check the finetuning stability of Mamba models. They analyzed that the Mamba follows the same pattern of instability as the other Transformer models.

3. Method

3.1. Performance Validation of VMamba on different Transformations

3.1.1 Models Used

Following are the models that were taken into consideration to compare their performances on the transformed dataset :

1. VMamba: VMamba-Base
2. Swin-Transformer [12]: Swin-Base
3. ConvNeXt [11]: ConvNeXt-Base

Each of these models are approximately of 89M parameters. These models were chosen based on factors such as model size, image classification performance, and novelty. We ran the model inference on the transformed dataset using each model.

3.1.2 Dataset Preparation and Transformations

A dataset was prepared by applying various transformations to images sampled from the ImageNet-1K Validation Dataset. This was used as the test set to evaluate all the above models. It has been confirmed in the research works that this validation dataset was never used to train the models. ImageNet-1K validation dataset has 1000 classes and 50000 images, i.e., 50 images per class. 15 images per class were sampled randomly in order to avoid long computational time. The transformations that were applied are as follows: Additive Gaussian Noise, Affine, Crop, Hue and Saturation, Crop and Pad, Cutout, Dropout, Edge Detect, Emboss Transform, Piecewise Affine, Salt, Fast Snowy Landscape, Fliplr, Gaussian Blur, Grayscale, Inversion, Linear Contrast, Multiply, Multiply 0, Perspective in total of 21 transformations. Thereby a dataset of size 15000 images per transformation was created.

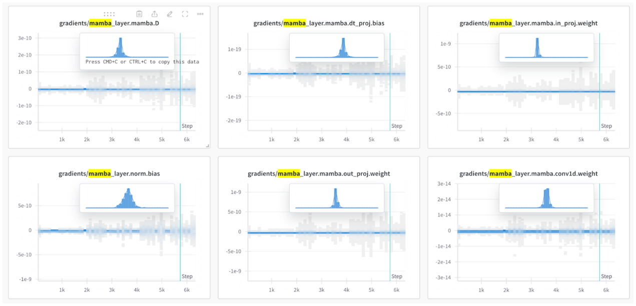


Figure 1. Gradient for different parameters in mamba layers

3.2. Finetuning UMamba

We fine-tuned U-Mamba to check the feasibility of Mamba models being fine-tuned over a dataset that is completely different from the one it was trained on. The original U-Mamba is trained on the medical datasets for medical image segmentation applications. The fine-tuning was done on the Massachusetts road image segmentation dataset [14].

We followed two different fine-tuning techniques. We started with the fully trained segmentation U-Mamba model and:

1. **Full Finetuning:** Finetuned the model completely i.e. all the weights of the model were fine-tuned.
2. **Gradient Freezing:** During full-finetuning, we observed that the gradients for some of the weights were really low and close to zero, indicating, that during the course of training, little to zero weight updates were being done for these weights. Figure 1 shows near zero gradients of some of the weights. These weights can then be frozen and can be restricted from taking part in training and thereby increasing the speed of training.

Baseline: For baseline, we trained a fresh model from scratch, not using the pretrained weights. The training was done for 6k iterations for the full training, full fine-tuning and gradient freezing approaches.

4. Experimental Results and Analyses

4.1. Performance Metrics

1. **Top-1 Accuracy:** Top-1 accuracy measures how often a model's top prediction matches the true class in multi-class classification tasks. It's used to evaluate the performance of models like VMamba, Swin-Transformer, and ConvNeXt across various transformations, with higher accuracy indicating better classification performance.

$$\text{Top-1 Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{\hat{y}_i = y_i\}$$

where n is the total number of samples, $\hat{y}_i$ is the predicted class for the i-th sample, y_i is the true class label, and $\mathbf{1}\{\cdot\}$ is the indicator function that returns 1 if the predicted class equals the true class and 0 otherwise.

2. **Dice Loss:** Dice score is a common metric used to measure the accuracy of segmentation maps from a segmentation model. Since to check the accuracy of these maps, we need to calculate the proportion of overlap, the dice score is defined as:

$$\text{Dice Score} = 2 * \frac{|A \cap B|}{|A| + |B|}$$

where A and B are the predicted and ground truth maps respectively.

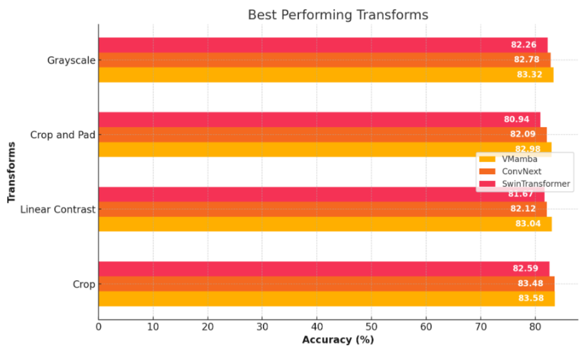
Dice loss is defined as the error made in dice score and since the maximum value of the dice score is 1, the dice loss becomes:

$$\text{Dice loss} = 1 - \text{Dice Score}$$

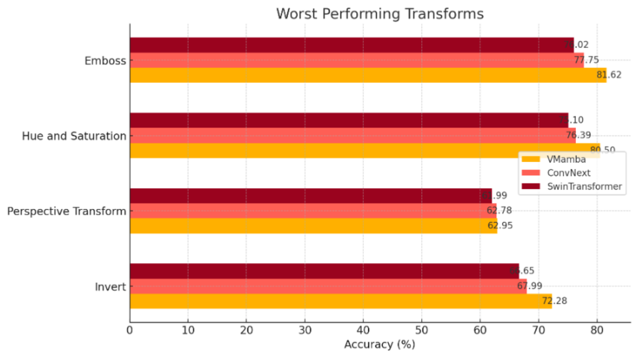
4.2. Results and Analysis on different transformations

4.2.1 Results

Figure 2 shows the performances of the VMamba, Swin-Transformer and ConvNeXt models across 8 transformations. Figure 2 (a) depicts the transformations on which the three models have performed the best and figure 2 (b) depicts the transformations on which the models have performed the worst. It can be seen that VMamba consistently outperforms the Swin-Transformer and ConvNeXt models across all the transformations. In comparison to any other transformation and the performance on the original ImageNet-1K validation images which is 83.90%, the best performance of the VMamba model is on the *Crop* transformation with a top-1 accuracy of 83.58% and the worst performance is on the *Perspective Transform* transformation with a top-1 accuracy of 62.95%.

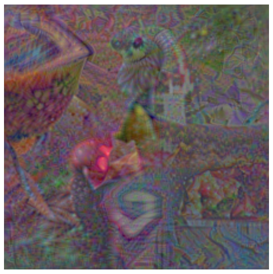


(a)

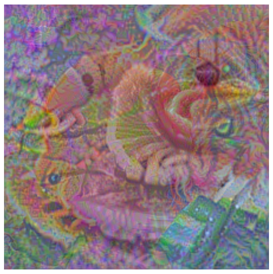


(b)

Figure 2. Shown in the top panel are the performances of the VMamba, Swin-Transformer and ConvNeXt models across various transformations. (a) depicts the transformations on which the three models have performed the best and (b) depicts the transformations on which the models have performed the worst.



(a) Guacamole



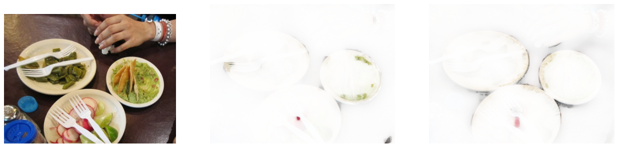
(b) Plate

Figure 3. Learned Features VMamba(Perspective transform)

4.2.2 Analysis

1. **Learned Features:** We have analyzed a class *Guacamole* which was misclassified as *Plate* by VMamba for the Perspective Transform. Figure 3 is a visualization of the learned features that refer to the internal representations that the VMamba model used to make its classification decision. These images show the patterns the VMamba model 'thought' were representative of each class after applying a perspective transformation to the original image. Despite the image originally belonging to the 'Guacamole' class, the transformed image activated features that the model associates with the 'Plate' class. This misalignment in features likely caused the model to misclassify the image. The visualization shows that the model was perhaps more sensitive to certain patterns or textures that are common between the two classes (like circular shapes or specific color patterns), leading to the wrong prediction.

2. **Localized Features:** The figure 4 is related to the



(a) Original (b) Guacamole (c) Plate

Figure 4. Localized Features for classification VMamba(Perspective transform), (b) The features the VMamba model highlighted (or localized) as contributing to the classification of 'Guacamole' and (c) The features that led to the incorrect classification as 'Plate'

classification and feature localization process for analyzing why the VMamba model misclassified an image. The images labeled (b) Guacamole and (c) Plate show what the model perceives or focuses on when making its classification decisions. The figure 4(b) represents features the VMamba model highlighted (or localized) as contributing to the classification of Guacamole. These are the features the model should have ideally focused on (likely regions around the guacamole itself, such as color, texture). The model has incorrectly prioritized the features in figure 4(c), which may include irrelevant regions like the white space around the plate or the structure of the plate itself that led to the incorrect classification as Plate.

4.3. Results and Analysis from Finetuning UMamba

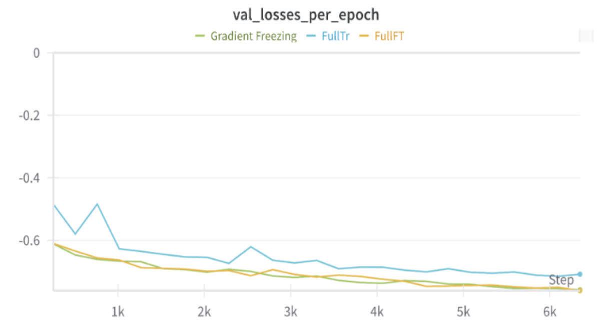
4.3.1 Results

Figure 5 shows the results for all the training and fine-tuning methods. It can be observed from both 5 (a) and (b) that the best-performing method in terms of both accuracy and speed is the gradient freezing method. We trained the model using all the approaches for 6k iterations and found the gradient freezing to be about 20% faster. We can see for 5 (a) that the loss is negative this is because the val loss is the summation of the cross entropy loss and the unnormalised dice loss which can be negative.

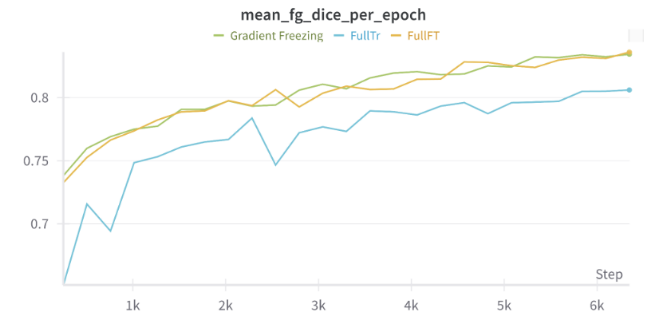
4.3.2 Analysis

The U-Mamba models trained on completely unrelated Road Segmentation data give dice scores of more than 80% which is a desirable performance. This shows that the Mamba models are capable of being fine-tuned to specific tasks even if the new task's data is completely different.

We observed that in U-Mamba models, the Mamba layers exhibit low and nearly negligible gradients during fine-tuning. This indicates that these layers may not require extensive fine-tuning, as the representations they learn for one task can potentially generalize well to other tasks. Future research could explore the broader applicability and generalizability of this finding.



(a)



(b)

Figure 5. The graph shows the model training stats for the U-Mamba model training using three different methods: full-training, full-finetuning, and gradient-freezing. (a) The val loss curve per epoch. As can be seen, the val loss continues to decrease for all methods, but full-finetuning and gradient freezing perform better, gradient freezing being the fastest. (b) The mean dice score per epoch for all three approaches, the figure shows full fine-tuning and gradient freezing performing better than the full training with both full fine-tuning and gradient freezing showing almost equal performance, gradient freezing being the fastest.

We also observed that it is quite feasible to implement parameter-efficient fine-tuning methods for Mamba models and the training/finetuning time can be reduced further.

5. Conclusion

The analysis in section 4.2 shows that after the perspective transform, the model's internal representations were more aligned with features it associated with the 'Plate' class rather than 'Guacamole'. Understanding these learned and localized features can guide further refinement of the model thereby making it more robust to transformations. Also, SSMs like ConvNeXt and Swin-Transfomers can be finetuned, and parameter-efficient finetuning methods can be implemented similarly.

References

[1] Lucas Brennan-Almaraz, Daniel Guo, and Sarvesh Babu. *Wrestling Mamba: Exploring Early Fine-Tuning Dynamics on Mamba and Transformer Architectures*. Stanford CS224N Custom Project, 2024. 2

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[3] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022. 1

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1

[5] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. 1

[6] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in Neural Information Processing Systems*, 35:4401–4414, 2022.

[7] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022. 1

[8] Yann LeCun and Yoshua Bengio. *Convolutional networks for images, speech, and time series*, page 255–258. MIT Press, Cambridge, MA, USA, 1998. 1

[9] Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and ”Teknium”. Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification, 2023. 2

[10] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model, 2024. 1

[11] Zhuang Liu, Hanzi Mao Hu, Yutong Lin, Zhuliang Cai, Trevor Cao, Zheng Zhang, Chao-Yuan Wei, Jiawei Xie, Xiyang Dai, Ze Liu, Stephen Lin, and Baining Guo. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022. 2

[12] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. 2

[13] Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation, 2024. 1

[14] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013. 2

[15] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1