Mamba vs SOTA for Vision Tasks

Clay Crews†
Department of Computer Science
University of South Carolina
Columbia, SC, United States
jccrews@email.sc.edu

Lexington Whalen†
Department of Computer Science
University of South Carolina
Columbia, SC, United States
LAWHALEN@email.sc.edu

***Abstract*—This paper explores the performance of the Mamba architecture, a variant of Structured State Space Sequence Models (SSMs), in comparison to state-of-the-art models such as Transformers, U-Net, and ResNet for vision tasks. We focus on image segmentation and classification, with the goal of maintaining high accuracy while keeping parameter counts low. Developing compact models with fewer parameters is crucial for the future of AI, not only to reduce energy consumption but also to enable deployment on resource-constrained edge devices. By leveraging the selective state spaces in Mamba blocks, we aim to achieve efficient and effective segmentation and classification performance while maintaining the linear scalability of SSMs. We implement Mamba-based architectures and compare their results to popular models like U-Net, ResNet, and Transformer-based approaches on the tasks of segmentation and classification. This work highlights the potential of Mamba and SSMs for efficient vision tasks, contributing to the development of compact yet accurate models in image segmentation and classification. We open-source our code to facilitate further research and exploration of these architectures: https://github.com/lxaw/mamba-vs-else-vision**

***Index Terms*—Mamba, SSMs, Image Segmentation, Image Classification**

I. Introduction

The rapid advancement of deep learning has led to remarkable achievements in various domains, including computer vision, natural language processing, and speech recognition. However, this progress has been accompanied by a significant increase in the size and complexity of neural network models. State-of-the-art (SOTA) models often have several million or even billions of parameters, making them computationally expensive and challenging to deploy on resource-constrained devices [30]. This trend poses concerns regarding energy consumption and the feasibility of deploying these models on edge devices, which have limited memory and processing power.

Many SOTA models for vision tasks, such as image classification and segmentation, rely on architectures like Transformers [2], U-Net [3], and ResNet [4]. While these models have achieved impressive performance, they often come with a high parameter count. Transformers, in particular, have gained significant attention due to their ability to capture long-range dependencies and achieve superior results in various tasks [5]. However, the self-attention mechanism in Transformers scales quadratically with the input sequence length, making them computationally expensive and difficult to apply to long sequences or high-resolution images [12]. This motivates the development of models that can achieve low parameter counts, accurately model long dependencies, and have fast inference and training.

To address the challenges of large parameter models and enable efficient deployment on edge devices, there is a growing interest in developing more compact and computationally efficient architectures. One promising approach is the recently proposed Mamba architecture, which is based on Structured State Space Sequence Models (SSMs) [6]. Mamba combines the modeling power of Transformers with scaling linearly in sequence length, allowing for efficient processing of long sequences while maintaining high performance [17]. By leveraging SSMs, Mamba can achieve competitive results with significantly fewer parameters compared to Transformers and other large models.

In this paper, we explore applications of Mamba for vision tasks, specifically image segmentation and classification. We aim to demonstrate that Mamba-based models can achieve high accuracy while keeping parameter counts low, and can be a competitive alternative to Transformer and other SOTA architectures.

II. Current Vision Task SOTA

Over the past decade, deep learning has revolutionized the field of computer vision, with various architectures achieving SOTA performance on tasks such as image classification and segmentation. Convolutional Neural Networks (CNNs) [28] have been at the forefront of this revolution, with architectures like AlexNet [27], VGGNet [7], and Inception [8] pushing the boundaries of classification accuracy on large-scale datasets.

CNNs achieve their efficiency and effectiveness through their unique architecture and the use of convolutional operations. The convolutional layers in CNNs apply a set of learnable filters (kernels) to the input image or feature map. These filters slide over the input, performing element-wise multiplications and summing the results to produce a new feature map [28]. This process allows CNNs to automatically learn and extract relevant features from the input data.

One of the most significant advancements in CNNs came with the introduction of ResNet [4], which addressed the problem of vanishing gradients in deep networks by introducing residual connections. ResNets allowed for the training of much

† Equal contributions

deeper networks, leading to improved performance on various vision tasks. The success of ResNet sparked a wave of research into more efficient and effective CNN architectures, such as MobileNet [9], EfficientNet [10], and RegNet [11].

While CNNs have been highly successful in vision tasks, they struggle to capture long-range dependencies and global context, which are crucial for understanding complex scenes and objects. To address this limitation, the Vision Transformer (ViT) [2] was introduced, adapting the self-attention mechanism from natural language processing to vision tasks. ViT treats an image as a sequence of patches and applies multi-head self-attention to learn global relationships between these patches. ViT has achieved impressive results on image classification tasks, often outperforming more classical CNN models.

Building upon the success of ViT, several transformer-based architectures have been proposed for other vision tasks, such as image segmentation. SegFormer [24] is a transformer-based model for semantic segmentation that employs a hierarchical structure and a novel attention mechanism called Efficient Self-Attention (ESA). ESA reduces the computational complexity of self-attention by performing attention operations in a local window and aggregating global information through a depth-wise convolution. This allows SegFormer to efficiently process high-resolution images while capturing both local and global context.

Despite the impressive performance of these SOTA models, they often come with a high computational cost and large number of parameters. The self-attention mechanism in transformers scales quadratically with the input sequence length, making them challenging to apply to long sequences or high-resolution images [12]. CNNs, while more efficient than transformers, still require a significant number of parameters to achieve SOTA performance, especially for complex tasks like segmentation.

The high parameter counts and scaling issues of these models pose challenges for deployment on resource-constrained devices and raise concerns about energy consumption. As the demand for efficient and sustainable AI solutions grows, there is a pressing need for more compact and computationally efficient architectures that can maintain high accuracy while reducing the parameter footprint. This has led to increased interest in techniques like model compression, quantization, and the development of novel architectures that can achieve SOTA performance with fewer parameters [13], [14].

III. State Space Models (SSMs)

Structured State Space Models (SSMs or S4 models) are a powerful mathematical framework for sequence modeling and processing temporal signals that are built on State Space Models (SSMs). They map an input signal $x(t)$ to an output signal $y(t)$ via a latent state $h(t)$. The latent state evolves according to a linear differential equation involving the input signal and a set of parameter matrices (A, B, C, D) that define the SSM's behavior [6].

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t) \tag{1}$$

$$y(t) = \mathbf{C}h(t) + \mathbf{D}x(t) \tag{2}$$

In general, the behavior of a system over time can be described by a differential equation. To produce an equation $h(t)$ that describes the state of the system for all time steps, the system must first be discretized. Finding this $h(t)$ function on a continuous system can prove difficult. Discretizing a system by sampling at multiple time steps will produce an approximation of the continuous function, which an $h(t)$ can be found for. Below, we show one simple way the system can be discretized using Euler's method. There are many other ways; [17] uses the zero-order hold (ZOH) rule.

Beginning with the continuous state space model equation (1), the definition of a derivative

$$\lim_{\Delta \to 0} \frac{h(t+\Delta) - h(t)}{\Delta} = h'(t)$$

is used to find the recurrent formula of the system:

$$h(t+\Delta) \cong \Delta h'(t) + h(t) \tag{3}$$

Substituting for $h'(t)$ with (1) (the $\mathbf{D}$ matrix is considered only as a skip connection, and thus as it does not directly play a role in the differential equation, was ignored) and discretizing the discretized parameters of the model, the "continuous parameters" $(\Delta, \mathbf{A}, \mathbf{B})$ are transformed to "discrete parameters" $(\bar{\mathbf{A}}, \bar{\mathbf{B}})$

$$\begin{aligned} h(t+\Delta) &\cong \Delta h'(t) + h(t) \\ &\cong \Delta(\mathbf{A}h(t) + \mathbf{B}x(t)) + h(t) \\ &\cong \Delta\mathbf{A}h(t) + \Delta\mathbf{B}x(t) + h(t) \\ &\cong (I + \Delta\mathbf{A})h(t) + \Delta\mathbf{B}x(t) \\ &\cong \bar{\mathbf{A}}h(t) + \bar{\mathbf{B}}x(t) \end{aligned} \tag{4}$$

where I is an identity matrix and the discrete parameters $(\bar{\mathbf{A}}, \bar{\mathbf{B}})$ are

$$\begin{aligned} \bar{\mathbf{A}} &= I + \Delta\mathbf{A} \\ \bar{\mathbf{B}} &= \Delta\mathbf{B} \end{aligned}$$

we derive the recurrent equations of the system:

$$h_t = \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t \tag{5}$$

$$y_t = \mathbf{C}h_t \tag{6}$$

The above derivation was done with reference to [31]. The recurrent equation can be used to predict any time step given the initial state where a previous time step, h_{t-1} is used to produce the output of the system with equations (6) and (5). This is directly similar to the mechanism of an RNN. Each token is computed one at a time with a constant memory and computation requirement. The constant computation requirement is great for inference of the system, however, a recurrent SSM is not ideal for training. Parallelizing this computation to caluclate the loss was a more practical solution presented in

[6]. With a initial state of 0, the convolutional representation was derived:

$$h_0 = \bar{\mathbf{B}}x_0$$
$$y_0 = \mathbf{C}h_0 = \mathbf{C}\bar{\mathbf{B}}x_0$$

$$h_1 = \bar{\mathbf{A}}h_0 + \bar{\mathbf{B}}x_1 = \bar{\mathbf{A}}\bar{\mathbf{B}}x_0 + \bar{\mathbf{B}}x_1$$
$$y_1 = \mathbf{C}h_1 = \mathbf{C}(\bar{\mathbf{A}}\bar{\mathbf{B}}x_0 + \bar{\mathbf{B}}x_1) = \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}x_0 + \mathbf{C}\bar{\mathbf{B}}x_1$$

$$\begin{aligned} h_2 &= \bar{\mathbf{A}}h_1 + \bar{\mathbf{B}}x_2 = \bar{\mathbf{A}}(\bar{\mathbf{A}}\bar{\mathbf{B}}x_0 + \bar{\mathbf{B}}x_1) + \bar{\mathbf{B}}x_2 \\ &= \bar{\mathbf{A}}^2\bar{\mathbf{B}}x_0 + \bar{\mathbf{A}}\bar{\mathbf{B}}x_1 + \bar{\mathbf{B}}x_2 \\ y_2 &= \mathbf{C}h_2 = \mathbf{C}(\bar{\mathbf{A}}^2\bar{\mathbf{B}}x_0 + \bar{\mathbf{A}}\bar{\mathbf{B}}x_1 + \bar{\mathbf{B}}x_2) \\ &= \mathbf{C}\bar{\mathbf{A}}^2\bar{\mathbf{B}}x_0 + \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}x_1 + \mathbf{C}\bar{\mathbf{B}}x_2 \end{aligned}$$

$$y_k = \mathbf{C}\bar{\mathbf{A}}^k\bar{\mathbf{B}}x_0 + \mathbf{C}\bar{\mathbf{A}}^{k-1}\bar{\mathbf{B}}x_1 + \cdots + \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}x_{k-1} + \mathbf{C}\bar{\mathbf{B}}x_k \tag{7}$$

Now, the kernel can be constructed from the convolutional form (7) and output y can be computed.

$$\bar{\mathbf{K}} = (\mathbf{C}\bar{\mathbf{B}}, \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}, \ldots, \mathbf{C}\bar{\mathbf{A}}^k\bar{\mathbf{B}}, \ldots) \tag{8}$$
$$y = x * \bar{\mathbf{K}} \tag{9}$$

The above derivation was done with reference to [31]. This convolution can be computed in parallel as y_k does not depend on previous outputs.

During training of the SSM, building the kernel can be expensive from a memory and computational perspective due to the nature of the convolutional operations. In inference, hwoever, the recurrent equation (4) is used, resulting in a constant computation cost for any token output in the sequence. In comparison to the Transformer architecture, where the computation cost of each token increases with the sequence lenth, the S4 model provides a much more efficient method. Additionally, for a given input of multiple dimensions, such as what is produced as an input embedding, the multi-head attention mechanism of a Transformer divides the dimensions into groups by the number of heads in the multi-head attention. Each head produces an output for the group of dimensions that were provided. Where as in an SSM architecture an independent state space model is dedicated for each dimension for an input.

For the state space model to perform well, data must be accurately captured about a previous state to produce a new token. The structure of the $\mathbf{A}$ matrix is carefully considered and the HiPPO (Hight-Order Polynomial Projection Operator) framework, presented in [15], is employed to maintain proper behavior of this matrix. The HiPPO framework is a technique for approximating a representation of the entire input history that makes use of orthogonal polynomial functions. Recent context of the input sequence is captured well and tokens of past information decays exponentially. This provides the proper information to the latent state, $h(t)$, making the technique a powerful tool for modeling long-range dependencies in sequence data.

IV. Mamba

Mamba (S6) [17] is a recently proposed foundational architecture that builds on the SSM (S4) architecture. The S6 block combines the modeling power of Transformers with the linear scaling efficiency of SSMs. The key idea is to augment SSMs with a selection mechanism that allows the model parameters to vary based on the input, enabling them to perform content-aware reasoning.

Standard SSMs map an input sequence $x(t)$ to an output $y(t)$ via a latent state $h(t)$ using a linear time-invariant system, shown by equations (1) and (2), where the parameter matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ are fixed. This allows SSMs to be computed efficiently as a convolution. However, the time-invariance means the model cannot change its behavior for specific inputs, making it difficult to solve tasks requiring selective focus, such as selective copying or induction heads [17].

To address these challenges, Mamba introduces a selection mechanism where the SSM parameters $\mathbf{B}$, $\mathbf{C}$, Δ (the state step size) are time-varying for each input token x_t. This allows the model to selectively propagate or forget information based on the current token. The selective SSM is computed using a parallel scan operation to maintain linear complexity, Algorithm 1.

Algorithm 1 SSM + Selection *Mamba Block (S6)*

Input: x: (B, L, D)
Output: y: (B, L, D)
1: $\mathbf{A}$: (D, N) ← Parameter
▷ Represents structured $N \times N$ matrix
2: $\mathbf{B}$: (B, L, N) ← $S_B(x)$
3: $\mathbf{C}$: (B, L, N) ← $S_C(x)$
4: $\mathbf{\Delta}$: (B, L, N) ← τ_Δ (Parameter $+S_\Delta(x)$)
5: $\bar{\mathbf{A}}$, $\bar{\mathbf{B}}$: (B, L, D, N) ← discretize($\mathbf{\Delta}$, $\mathbf{A}$, $\mathbf{B}$)
6: y ← SSM($\bar{\mathbf{A}}$, $\bar{\mathbf{B}}$, $\mathbf{C}$)(x)
▷ Time-varying: recurrence (*scan*) only
7: **return** y

The parallel scan decreases the time complexity from $O(N)$ to $O(N/T)$ by implementing T threads to perform binary operations in parallel, syncronizing at each step.

A major slow down that comes with having a time-varying system is that every input uses a different kernel. Mamba addresses this by considering the relationships of memory in the hardware of modern GPUs. When an operation on a tensor is performed, the tensor is copied from the High-Bandwidth Memory (HBM), which is slow, of the GPU to the SRAM memory, which is fast, where the operation is performed, then is copied back to the HBM memory. Minimizing the number of times a tensor is copied between the fast and slow memory greatly improves the computation speed and efficiency of a tensor operation. Mamba employs kernel fusion to create a custom CUDA kernel as that multiple tensor operations are computed in SRAM without the need for unnecessary copy operations to and from the HBM, preventing an IO bound process.

To compute the gradient of loss function during back-propagation, the gradient of each node in the network is needed for this computation. The chain rule is used to calculate the these gradient as each node in a forward pass. The results would need to be cached and copied for calculation during back-propagation. Reducing memory requirements of the model, these calculations are recomputed for the loss function, preventing the slow copy operations from the HBM.

The Mamba block interleaves the selective SSM with linear projections, convolutions, and activations:

1) Linear projection to expand the embedding dimension
2) Convolution to mix information between dimensions
3) Selective SSM via parallel scan to efficiently propagate information
4) Linear projection scale back to the target embedding size

Algorithm 2 Mamba (S6) Block Pipeline

Input: x: (B, L, D)
Output: y: (B, L, D)
RMS_Norm: r:(B, L, D) $\leftarrow$ x:(B, L, D)
Branch 1 $\rightarrow$
    Linear: $b1$:(B, L, N) $\leftarrow$ r:(B, L, D)
    Conv $b1$:(B, L, N) $\leftarrow$ $b1$:(B, L, N)
    SiLU $b1$:(B, L, N) $\leftarrow$ $b1$:(B, L, N)
    SSM $b1$:(B, L, N) $\leftarrow$ $b1$:(B, L, N)
Branch 2 $\rightarrow$
    Linear: $b2$:(B, L, D) $\leftarrow$ r: (B, L, N)
    SiLU: $b2$:(B, L, N) $\leftarrow$ $b2$:(B, L, N)
Element-Wise Multiply:
z:(B, L, N) $\leftarrow$ $b1$:(B, L, N) $\times$ $b2$:(B, L, N)
Linear: z:(B, L, D) $\leftarrow$ z:(B, L, N)
Add: z:(B, L, D) $\leftarrow$ z:(B, L, D) + x:(B, L, D)
$y \leftarrow z$
return y

These S6 blocks are stacked to form the Mamba architecture. Compared to Transformers, Mamba can achieve comparable modeling performance while scaling linearly in sequence length rather than quadratically [17]. Overall, the selective SSM leverages the strengths of RNNs, CNNs and Transformers - it maintains an unbounded context window like RNNs, can be parallelized like CNNs, and achieves content-aware reasoning like Transformers, while being more efficient than all three. This makes Mamba a promising foundation model for processing long sequences across various domains.

V. Related Works

Prior to an analysis of how the Pyramidal U-Mamba compares, we shall explain what models we chose for comparison and why.

For this our analysis, we compare four models against our own developed ones. The models are U-Net [3], ResNet18 [4], ResNet50, U-Mamba [18], and SegFormer [24]. Below we explain our choices.

A. U-Net

The core idea behind U-Net is to complement a traditional contracting network with successive layers that replace pooling operations with upsampling operators. These upsampling layers aim to increase the resolution of the output. Subsequent convolutional layers can then learn to assemble a precise output based on this high-resolution information [3]. A key modification in the U-Net architecture is the inclusion of a large number of feature channels in the upsampling part. This allows the network to effectively propagate contextual information to higher resolution layers. As a result, the expansive path of the network becomes more or less symmetric to the contracting part, leading to a distinctive U-shaped architecture. To predict pixels at the border regions of the image, the missing context is extrapolated by mirroring the input image. This tiling strategy is crucial for applying the network to large images, as it circumvents resolution limitations imposed by GPU memory constraints [3]. Due to its encoder-decoder architecture, skip connections, multi-scale feature extraction, and its efficiency, U-Net has been used in many denoising and diffusion models. For instance, DDPMs (Denoising Diffusion Probabilistic Models) use an architecture similar to U-Net for denoising and sample generation [20] while the popular Stable Diffusion architecture uses a U-Net based architecture for the diffusion process [19]. We have chosen to compare our novel segmentation architecture against the U-Net architecture due to its well-established reputation and widespread adoption in the field of medical image segmentation.

B. ResNet

We have also selected ResNet18 and ResNet50 as additional benchmarks. These architectures, introduced by He et al. in their seminal work "Deep Residual Learning for Image Recognition" [4], have revolutionized the field of deep learning by addressing the problem of vanishing gradients in deep neural networks. The key innovation in ResNets is the introduction of residual connections, which allow the network to learn residual functions with reference to the input layer, thereby facilitating the training of much deeper networks. ResNet18 and ResNet50, with 18 and 50 layers respectively, have been widely adopted in various computer vision tasks, including image classification, object detection, and segmentation. These models have demonstrated exceptional performance and generalization ability across diverse datasets. By comparing our proposed architecture against ResNet18 and ResNet50, we aim to assess its effectiveness in relation to these well-established and highly influential architectures. This comparison will provide valuable insights into the capabilities of our model and its potential to advance the state-of-the-art in image segmentation tasks.

C. U-Mamba

As our model takes much inspiration from the recently developed U-Mamba design [18], we also choose to incorporate it in our comparison. U-Mamba addresses these limitations by introducing a novel hybrid CNN-SSM block that leverages

the strengths of both architectures. The convolutional layers in the block are responsible for local feature extraction, while the State Space Sequence Models (SSMs) [6], a new family of deep sequence models, are known for their strong capability in handling long sequences and capturing long-range dependencies. By integrating these two components, U-Mamba achieves a balance between local and global information processing, enabling it to effectively handle long-range dependencies in biomedical image segmentation tasks. Moreover, U-Mamba incorporates a self-configuring mechanism that allows it to automatically adapt to various datasets without manual intervention, enhancing its versatility and usability.

D. SegFormer

SegFormer [24] is a transformer-based semantic segmentation model that employs a hierarchical structure and a novel attention mechanism called Efficient Self-Attention (ESA). The SegFormer architecture consists of a transformer encoder for capturing long-range dependencies and a lightweight All-MLP decoder for generating high-resolution segmentation masks. The ESA module reduces the computational complexity of self-attention by performing attention operations in a local window and aggregating global information through a depth-wise convolution. This allows SegFormer to efficiently process high-resolution images while capturing both local and global context. Moreover, SegFormer introduces a position-sensitive embedding scheme that encodes both spatial and channel-wise information, enhancing the model's ability to capture fine-grained details. By comparing U-Mamba against SegFormer, we aim to evaluate the effectiveness of our hybrid CNN-SSM approach in relation to the transformer-based segmentation mechanism employed by SegFormer. This comparison will provide insights into the strengths and weaknesses of both architectures and their ability to handle complex biomedical image segmentation tasks. Furthermore, it will help us understand the potential of transformer-based approaches and hybrid approaches in advancing the state-of-the-art in biomedical image segmentation.

E. UltraLight VM-UNet

The UltraLight VM-UNet [21] is a lightweight neural network architecture designed for skin lesion segmentation tasks. It is built upon the Vision Mamba module, which is a state-space model (SSM) that can efficiently handle long-range dependencies in sequences, making it well-suited for image segmentation tasks.

The key innovation of the UltraLight VM-UNet is the proposed Parallel Vision Mamba Layer (PVM Layer), which processes deep features in parallel using multiple Vision Mamba blocks. Specifically, the input feature map is split into multiple sub-feature maps, each processed by a separate Vision Mamba block with a reduced channel count. This parallel processing approach allows the UltraLight VM-UNet to maintain high segmentation performance while significantly reducing the number of parameters and computational complexity.

The UltraLight VM-UNet is reported to have only 0.049 million parameters and a computational cost of 0.060 GFLOPs, which is significantly lower than traditional convolutional neural networks and transformers used for image segmentation tasks. Despite its lightweight nature, the authors demonstrate that the UltraLight VM-UNet achieves competitive performance on three publicly available skin lesion segmentation datasets, outperforming several state-of-the-art lightweight models.

The success of the UltraLight VM-UNet can be attributed to the authors' in-depth analysis of the key factors influencing the parameters of the Vision Mamba module. By identifying the number of input channels as a critical factor affecting the parameter count, they were able to design the PVM Layer to process features in parallel while keeping the overall channel count constant, leading to a significant reduction in parameters without compromising performance.

VI. Application: Image Segmentation

Here we investigate Mamba-based architectures with regards to the task of image segmentation.

A. Input Data

We shall be using a Brain MRI segmentation dataset found in [23]. This dataset has been used in several papers regarding classification of the shape and severity of tumors, and provides an adequate dataset for research purposes in the field of medical image analysis, particularly in the area of brain MRI segmentation. Researchers have utilized this dataset in various studies focusing on the classification of tumor shapes and the assessment of tumor severity. With its comprehensive collection of brain MRI scans, along with corresponding segmentation masks, the dataset offers valuable resources for developing and evaluating algorithms aimed at automating tumor detection and analysis.

The dataset contains brain MRI images along with their segmentation masks for 110 patients. There were a total of 3143 train images, 393 validation images, and 393 test images. The distribution of the tumor and non-tumor images is shown in Figure 1.

B. Output Data

The output of this model shall be segmentation masks for new, never before seen images.

VII. Our Models

Inspired by [18] and [21], we sought to implement both models on a different problem: tumor segmentation.

Our goal was to 1) maintain dice scores comparable to state-of-the-art (SOTA) methods such as those listed above, while 2) being smaller than those above.

We now go into what we did for each of our models.

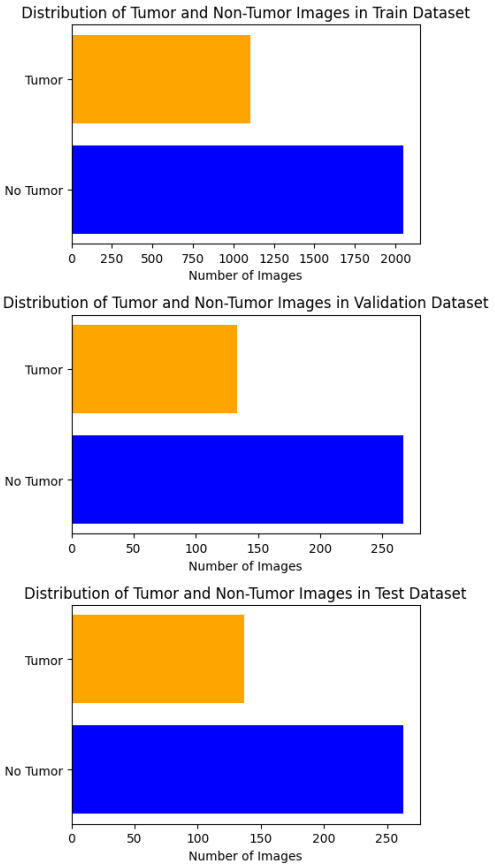

Fig. 1: Distributions of tumor and non-tumor images in the train, validation, and test datasets.

A. UMambaBot-PP

This model incorporates pyramidal pooling, which is a strategy used in convolutional neural networks (CNNs) to capture context and incorporate multi-scale information [22]. Pyramidal pooling involves parallel pooling operations at different scales, followed by concatenation of the resulting feature maps. This approach has been shown to improve the performance of CNNs in various computer vision tasks, including segmentation, by enabling the model to capture both local and global context.

In our UMambaBot-PP model, we integrated pyramidal pooling into the U-Mamba architecture, aiming to leverage the strengths of both the Mamba module and multi-scale feature extraction.

B. UL-VM-UNet-v1

For UL-VM-UNet-v1, the initial channel list was modified from [8, 16, 24, 32, 48, 64] to [16, 32, 64, 128, 256], increasing the number of channels at each step of the encoder and decoder. Additionally, the depth of the U-Net was reduced by one layer to keep the parameter count low while increasing the number of parameters in each step. The intention was to improve performance by increasing the capacity of the model while maintaining a reasonable parameter count.

C. UL-VM-UNet-v2

In UL-VM-UNet-v2, the number of parallel branches in the Parallel Vision Mamba (PVM) block was increased from 4 to 8. This modification directly decreases the parameter count, further proving the idea proposed in the UltraLight VM-UNet paper [21] that processing features in parallel with reduced channel counts can significantly reduce parameters while maintaining performance. The channel list is the same as Ultra LightM-UNet's, [8,16,24,32,48,64].

D. UL-VM-UNet-v3

UL-VM-UNet-v3 combines the modifications from UL-VM-UNet-v1 and UL-VM-UNet-v2. It increases the number of parallel branches in the PVM block to 8 and modifies the channel list to [16, 32, 64, 128, 256]. Additionally, the depth of the U-Net was reduced to 5 layers. This approach aims to balance the trade-off between model capacity and parameter efficiency.

E. UL-VM-UNet-v4

This model incorporates pyramidal pooling, similar to UMambaBot-PP, but within every PVM block in both the encoder and decoder in the UltraLight VM-UNet architecture. The goal was to leverage the benefits of multi-scale feature extraction while maintaining the parameter efficiency of the UltraLight VM-UNet. Incorporating pyramidal pooling on the UL network scored slightly worse than without pyramidal pooling but required approximately 400,000 more parameters. The channel list was [16,32,64,128,256,512].

F. UL-VM-UNet-v5

UL-VM-UNet-v5 is meant to be a comparison to the -v4 and -v6 UltraLight variations. This model trained and performed well with 0.7M parameters. v5 has 8 parallels and modifies the channel list to [16,32,64,128,256,512].

G. UL-VM-UNet-v6

Another variation to further test the effects of incorporating pyramidal pooling in the UltraLight VM-UNet architecture was made by only using pyramidal pooling in the bottleneck layer, similarly done in UMambaBot-PP. The main goal here, again, was to integrate pyramidal pooling for the benefit of multi-scale feature extraction while keeping a low parameter count. The channel list was also [16,32,128,256,512].

VIII. Results for Segmentation Task

We now compare our models against ResNet18, ResNet50, standard U-Mamba bottleneck, UNet, and UltraLight VM-UNet. We train for 100 epochs on all models, and use dice-loss with Adam optimizer. We train on roughly 3000 images, and validate on roughly 400. we then test on roughly 400. We show the distributions of our data in Figure 1. Model results are shown in Table I and Figures 2,3.

IX. Application: Image Classification

Here we investigate Mamba-based architectures with regards to the task of image segmentation.

Model Name	#Params (M) ↓	Dice Score ↑
UMambaBot-PP(Ours)	10.0	0.8867
UMambaBot	9.8	0.8799
ResNet18	13.9	0.8648
ResNet50	42.9	0.8672
SegFormer	17.8	0.8454
UNet	**7.8**	**0.8929**
UL-VM-UNet	0.049	0.79955
UL-VM-UNet-v1(Ours)	0.20	0.8409
UL-VM-UNet-v2(Ours)	**0.042**	0.8243
UL-VM-UNet-v3(Ours)	0.176	0.8109
UL-VM-UNet-v4(Ours)	1.1	0.8548
UL-VM-UNet-v5(Ours)	0.7	**0.8633**
UL-VM-UNet-v6(Ours)	0.77	0.8628

TABLE I: Comparison of different segmentation models.

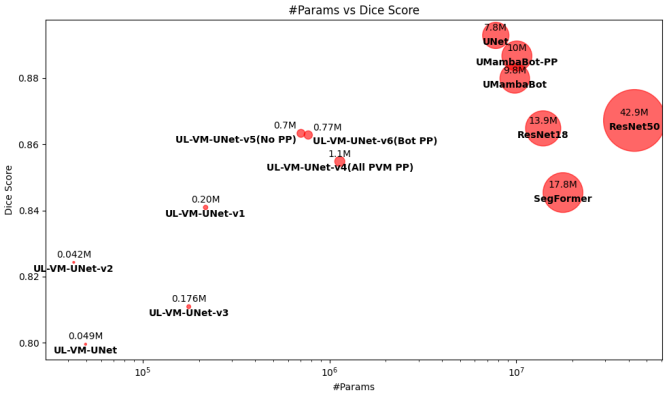

Fig. 2: A scatter plot of Dice Score vs model parameter count, where "M" means "millions. The points are scaled to help represent size.

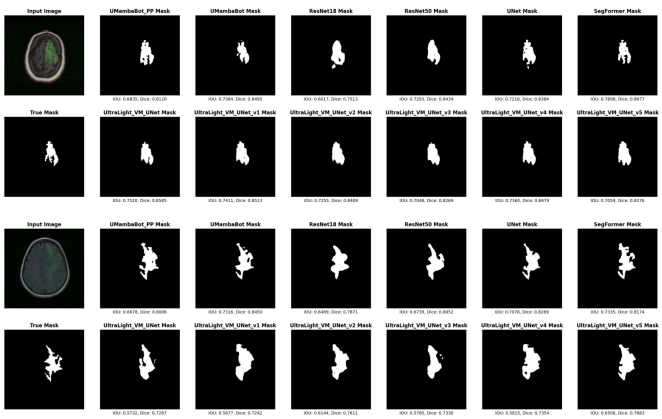

Fig. 3: Example segmentations.

A. Input Data

To further demonstrate the viability and versatility of the Mamba architecture, we conducted experiments on the CIFAR-10 dataset [25], a widely-used benchmark for image classification tasks. CIFAR-10 consists of 60,000 32x32 color images across 10 classes, with 6,000 images per class. The dataset is split into 50,000 training images and 10,000 test images.

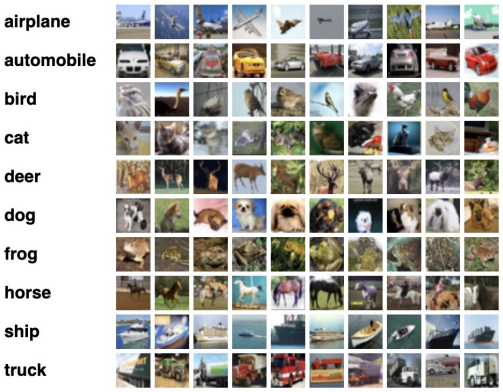

Fig. 4: Example images from CIFAR10.

B. Output Data

The output of this model will be a classification for one of the 10 classes in CIFAR10.

X. Image Classification Models

For this experiment, we employed the Vision Mamba (ViM) architecture proposed by Zhu et al. [29]. ViM is an efficient visual representation learning framework that leverages a bidirectional state space model to capture long-range dependencies in visual data. By combining the strengths of CNNs and SSMs, ViM aims to achieve competitive performance while maintaining computational efficiency.

We compared the performance of ViM against the Vision Transformer (ViT) architecture [2] on the CIFAR-10 dataset. The ViM model had 1,974,664 parameters, while the ViT had 1,767,602. Both models were trained for 100 epochs using the same experimental setup, including hyperparameters, optimization algorithm, and data augmentation techniques. Training and validation was done with batch size of 64, and upon every training iteration we do a validation iteration to obtain the validation scores. To see more on the training, please check our GitHub. The experiments were conducted on an NVIDIA GeForce RTX 3090 Ti GPU to ensure a fair comparison of training and validation times. Figure 5 presents the validation F1 score and loss curves for both ViM and ViT over the course of 100 epochs, Figure 6 shows the validation loss values over the 100 epochs, while Figure 7 compares the training and validation times for both models. Note that ViM achieves similar performance with less training and validation time.

XI. Results for Classification Task

These experiments demonstrate that the Mamba architecture, as implemented in the ViM model, can achieve competitive performance on image classification tasks while maintaining computational efficiency. The results highlight the potential of Mamba-based models as an alternative to transformer-based

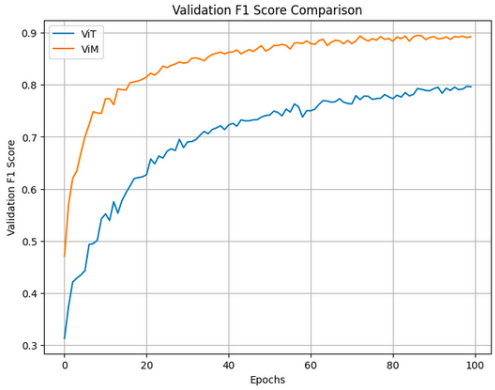

Fig. 5: F1 scores for ViM and ViT over 100 epochs.

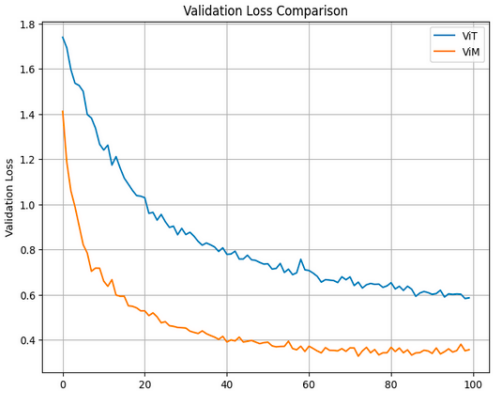

Fig. 6: Loss scores for ViM and ViT over 100 epochs.

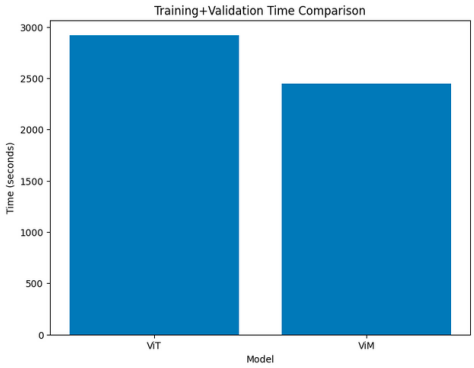

Fig. 7: Training and validation time for ViT and ViM.

approaches, particularly in scenarios where both accuracy and efficiency are crucial considerations.

XII. Conclusion

We investigate the Mamba architecture with regards to the vision tasks of segmentation and classification. We develop and fine-tune models for both tasks, showing that they can outperform the SOTA in parameter count or accuracy. We hope to show that there are viable alternatives to transformers, and that transformer-level performance can be achieved with Mamba in the realm of computer vision.

References

[1] T. B. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165 [cs.CL], 2020.

[2] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," arXiv:2010.11929 [cs.CV], 2021.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," arXiv:1505.04597 [cs.CV], 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv:1512.03385 [cs.CV], 2016.

[5] A. Vaswani et al., "Attention Is All You Need," arXiv:1706.03762 [cs.CL], 2017.

[6] A. Gu, K. Goel, and C. Ré, "Efficiently Modeling Long Sequences with Structured State Spaces," arXiv:2111.00396 [cs.LG], 2022.

[7] K. Simonyan and A. Zisserman, "Published as a conference paper at ICLR 2015 VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION." Available: https://arxiv.org/pdf/1409.1556

[8] C. Szegedy et al., "Going deeper with convolutions." Available: https://arxiv.org/pdf/1409.4842

[9] A. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." Available: https://arxiv.org/pdf/1704.04861

[10] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." Available: https://arxiv.org/pdf/1905.11946

[11] I. Radosavovic, R. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing Network Design Spaces." Accessed: Apr. 30, 2024. [Online]. Available: https://arxiv.org/pdf/2003.13678

[12] K. Choromanski et al., "Published as a conference paper at ICLR 2021 RETHINKING ATTENTION WITH PERFORMERS." Accessed: Apr. 30, 2024. [Online]. Available: https://arxiv.org/pdf/2009.14794

[13] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "IEEE SIGNAL PROCESSING MAGAZINE, SPECIAL ISSUE ON DEEP LEARNING FOR IMAGE UNDERSTANDING 1 A Survey of Model Compression and Acceleration for Deep Neural Networks." Accessed: Apr. 30, 2024. [Online]. Available: https://arxiv.org/pdf/1710.09282

[14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," 2015. Accessed: Apr. 30, 2024. [Online]. Available: https://arxiv.org/pdf/1503.02531

[15] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "HiPPO: Recurrent Memory with Optimal Polynomial Projections," Advances in Neural Information Processing Systems, vol. 33, pp. 1474-1487, 2020.

[16] A. Vaswani et al., "Attention Is All You Need," arXiv:1706.03762 [cs.CL], 2023.

[17] A. Gu and T. Dao, "Mamba: Linear-Time Sequence Modeling with Selective State Spaces," arXiv:2312.00752 [cs.LG], 2023.

[18] J. Ma, F. Li, and B. Wang, "U-Mamba: Enhancing Long-range Dependency for Biomedical Image Segmentation," arXiv:2401.04722 [cs.CV], 2024.

[19] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," arXiv:2112.10752 [cs.CV], 2021.

[20] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," CoRR, vol. abs/2006.11239, 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[21] R. Wu, Y. Liu, P. Liang, and Q. Chang, "UltraLight VM-UNet: Parallel Vision Mamba Significantly Reduces Parameters for Skin Lesion Segmentation." Accessed: Apr. 20, 2024. [Online]. Available: https://arxiv.org/pdf/2403.20035.pdf

[22] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid Scene Parsing Network," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6230-6239, doi: 10.1109/CVPR.2017.660.

[23] "Brain MRI segmentation," www.kaggle.com. https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation

[24] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers." Available: https://arxiv.org/pdf/2105.15203.pdf

[25] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Tech Report, 2009.

[26] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," arXiv:1611.05431 [cs.CV], 2017.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, vol. 25, 2012.
[28] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.
[29] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, "Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model," arXiv.org, Feb. 10, 2024. https://arxiv.org/abs/2401.09417
[30] H. Naveed et al., "A Comprehensive Overview of Large Language Models." Available: https://arxiv.org/pdf/2307.06435
[31] U. Jamil, "hkproj/mamba-notes," GitHub, Apr. 27, 2024. https://github.com/hkproj/mamba-notes (accessed Apr. 30, 2024).

APPENDIX

A. Supplementary Figures

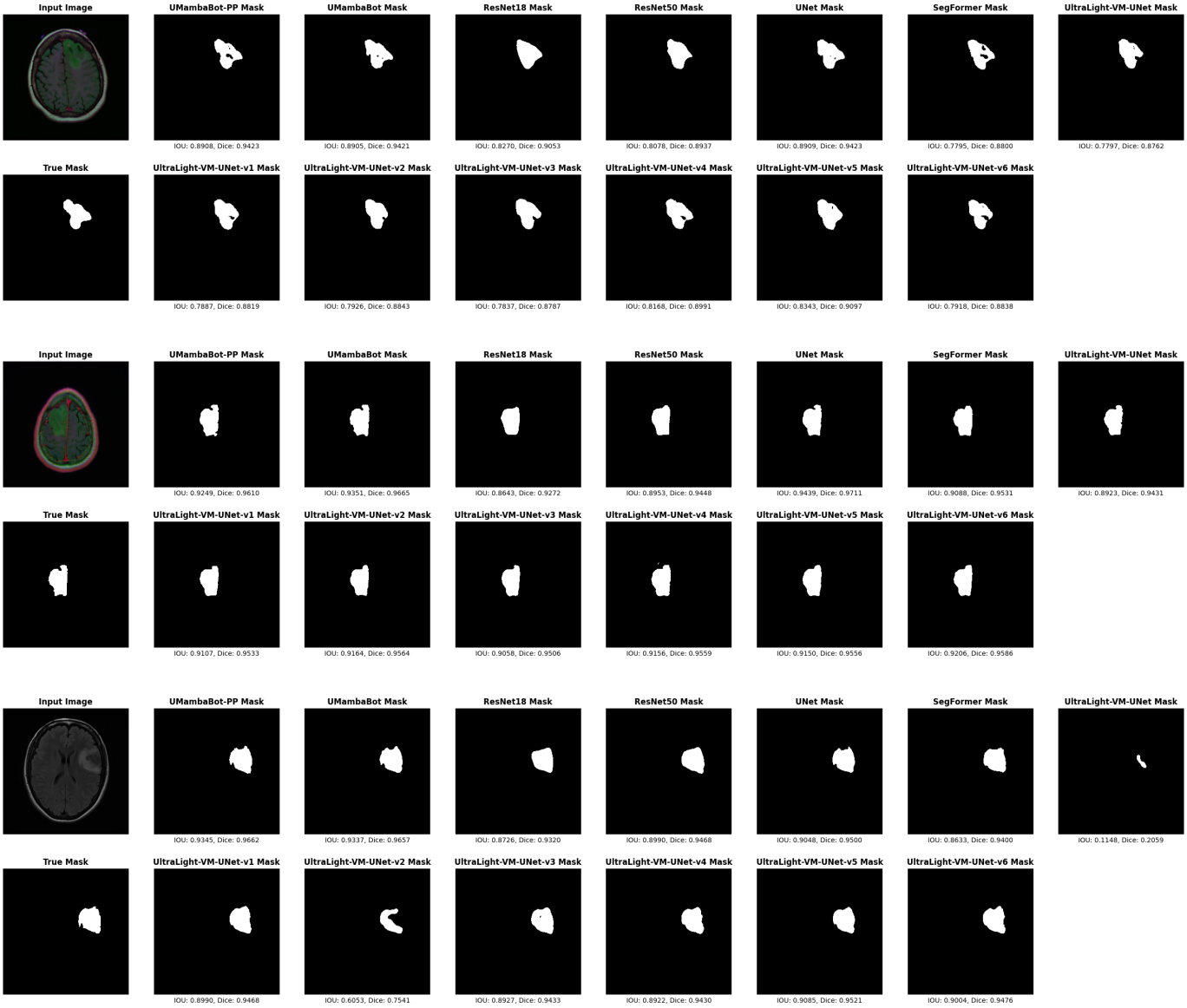

Fig. 8: In some cases, such as this one, it was observed that models with too few parameters (UltraLight-VM-UNet, UltraLight-VM-UNet-v2) don't detect sections of the tumor during segmentation.

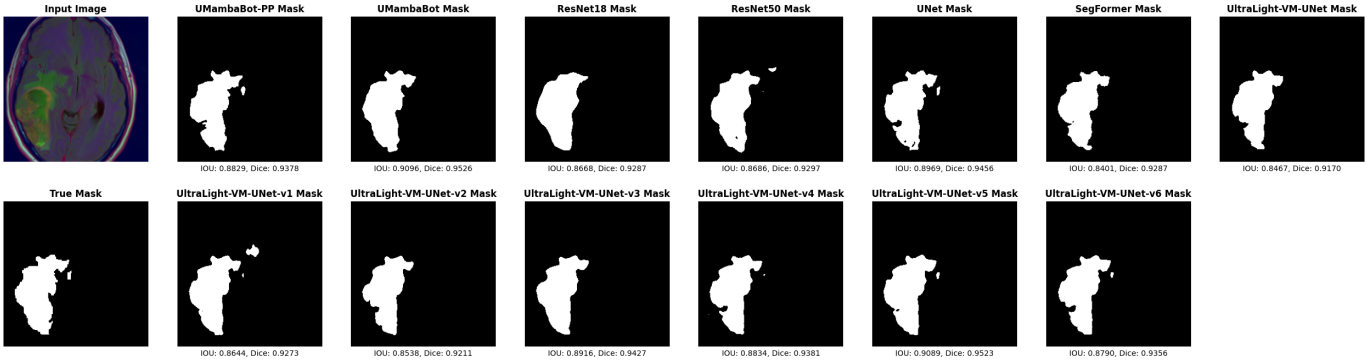

Fig. 9: UltraLight-VM-UNet-v5 achieved segmentation results equivalent to UMambaBot with approximately 9.1 M less parameters.

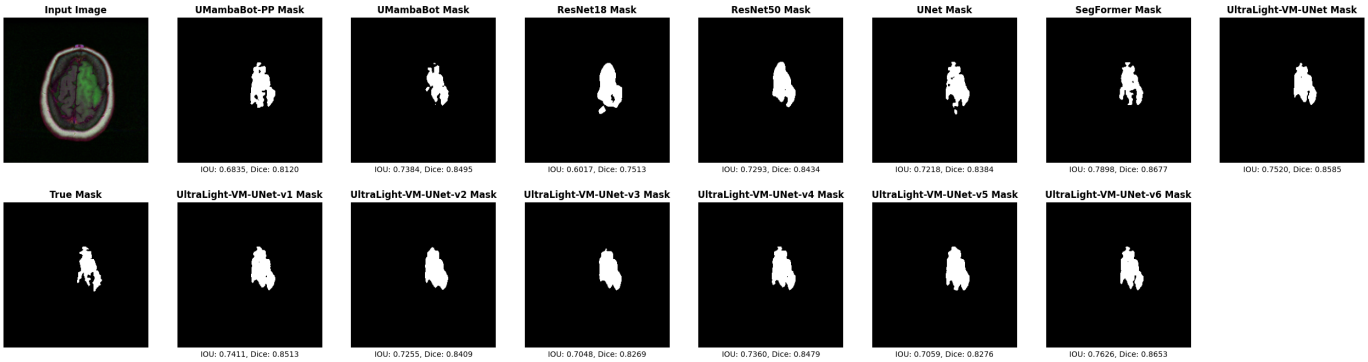

Fig. 10: An example of models with pyramidal pooling in the PVM blocks (UltraLight-VM-UNet-v4, UltraLight-VM-UNet-v6) achieved an improvement over a model without (UltraLight-VM-UNet-v5).