Project B: State Space Models

ECE1512 Digital Image Processing and Applications
Siyu Zhang
1010403653
Date due: December 10, 2024
Date handed in: December 10, 2024

Part 1 Summarization

Key points [1]

Modern machine learning tasks like language, audio, and genomics modeling rely heavily on Foundation Models (FMs), which are typically based on Transformers. While Transformers are effective at processing information within finite context windows, they face key limitations: they are computationally inefficient for long sequences due to quadratic scaling, and they struggle to capture dependencies beyond their fixed context. To address these issues, Structured State Space Models (SSMs) were introduced. These models scale efficiently with sequence length and perform well on continuous data like audio. However, SSMs are less effective for tasks involving discrete and context-sensitive data, such as text, because they lack the ability to adapt to the input dynamically. To overcome this, Selective SSMs were developed, enabling the model to focus on relevant information and ignore irrelevant context using input-dependent dynamics. Building on these improvements, the Mamba model simplifies architecture by integrating Selective SSMs, removing attention and MLP layers. This design achieves exceptional efficiency, scaling to handle sequences of up to a million tokens, while delivering competitive or superior performance across various modalities like language, audio, and genomics.

Technical contributions

Mamba cleverly combines the best features of different models while solving their key limitations. The main challenge with Transformers is that they need too much memory when processing long sequences, as their attention mechanism becomes increasingly demanding with longer inputs. RNNs and LSTMs work differently, they use a fixed-size memory to store past information and it limits their capacity to capture long-term dependencies. While LSTMs have improved this with special gates to control memory, they still cannot match the flexible memory management of Transformers. State Space Models offer an interesting alternative by using fixed rules to process information across time, making them computationally efficient but less adaptable to different contexts. Mamba brings together these approaches by creating a selective mechanism that can adjust to the input like Transformers, while maintaining the efficiency of State Space Models. It manages to train like a Transformer by adapting to inputs, but runs efficiently like an RNN during actual use, creating an effective balance between adaptability and speed.

The most important part in the SSMs is implicit latent state, $h(t)$. The equations show the relationships between the input, $x(t)$, and output, $y(t)$ and the latent space, $h(t)$.

$$h'(t) = Ah(t) + Bx(t) \quad \text{(Eq.1a)}$$

$$y(t) = Ch(t) \quad \text{(Eq.1b)}$$

$$h(t) = \bar{A}h(t-1) + \bar{B}x(t) \quad \text{(Eq.2a)}$$

$$y = x * \overline{K} \quad \text{(Eq.2b)}$$

$$\overline{K} = (C\bar{B}, \overline{CA\bar{B}}, \dots, C\bar{A}^k\bar{B}) \quad \text{(Eq.2c)}$$

In the structure state space model, the equations are Eq.1a and Eq.1b, it represents continuous relationships. The parameters, A, B, and C are fixed for every sequence, making SSMs time-invariant. These fixed parameters are transformed into discrete parameters, $\bar{A}$ and $\bar{B}$ using a

process called discretization, such as Zero-Order Hold (ZOH), allowing SSMs to operate efficiently in discrete time while maintaining properties like resolution invariance. Another key component in SSMs is $\overline{K}$, the convolutional kernel, which encodes the influence of the parameters, $\bar{A}$, $\bar{B}$ and C over time. $\overline{K}$ is used to efficiently transform the input sequence into the output by capturing dependencies between past inputs and the current output. In Transformers, however, the parameters A, B and C are dynamically built during each forward pass, as tokens in Transformers rely heavily on both the input and past context. In LSTM, A and B adapt based on the last hidden state, using gates to control information flow dynamically. Mamba builds upon these models by combining the efficiency of SSMs with the adaptability of Transformers and LSTMs, introducing a new architecture called Selective State Space Models. In Selective SSMs, $\bar{A}$ and $\bar{B}$ become input-dependent, allowing the model to filter out irrelevant inputs and focus on relevant ones along the sequence. This dynamic computation makes Selective SSMs as flexible as Transformers while keeping the efficiency of SSMs, allowing them to adapt to context effectively.

Besides the core innovation of the selection mechanism in Mamba, the Hardware-Aware Algorithm and redesigned model block also represent significant contributions to its efficiency and performance. The Hardware-Aware Algorithm is particularly notable for its innovative use of GPU memory hierarchies. By processing data in the faster Static Random Access Memory (SRAM) and only transferring outputs and hidden states back to the slower high-bandwidth memory (HBM), Mamba overcomes the high memory usage problem and enhances both efficiency and processing speed. Additionally, the updated model block simplifies deep sequence models by integrating SSM architectures with elements inspired by the multi-layer perceptrons (MLP blocks) used in Transformers. These design choices contribute to the core properties of Mamba: high quality, fast training and inference, and the ability to handle long contexts.

Building on these strengths, Mamba demonstrates exceptional performance across a range of tasks, including synthetic benchmarks like selective copying and induction heads, as well as real-world applications in audio and genomics, such as speech generation and DNA sequence modeling. It excels in language modeling as the first linear-time sequence model to achieve Transformer-level quality while delivering 5x generation throughput compared to previous state-of-the-art models.

Structure of Mamba Model

The block-level details with Mamba model is shown in Fig.1.The first layer of Mamba model is token embedding. This part basically splits sentences by word, sub-word or characters based on the requirements. Each word, subword, or character is known as a token, then each token will assign a number based on a predefined vocabulary. These numbers will convert into vectors to help the model to understand the meaning of each token. After that the results will pass through multiple Mamba Blocks and get final prediction. In each Mamba Block, there contains many operations, Root Mean Square Normalization (RMS Norm), Projections, Conv1D, SiLU Activation, Selective SSM and Skip Connections.

$$RMS(x) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2} \qquad \text{(Eq.3)}$$

RMS Norm scales the features based on their root mean square (RMS) value instead of using mean and variance. The reason to use RMS Norm is because the computation is fast compare to the traditional methods. The equation is shown in Eq.3, where d refers to the number of features and x represents the vector of feature values. The next operation is linear projections, it used to adjust dimensionality, usually it will expand to higher dimension to separate and process different features easily. Conv1D is used to capture patterns in sequences for local context. After capturing local patterns, the data passes though the Sigmoid Linear Unit (SiLU) activation function, shown in Eq.4. It is particularly effective for handling this relationship because it can avoid cutoffs.

$$SiLU(x) = x \cdot sigmoid(x) \quad \text{(Eq.4)}$$

The projection sequence (x_proj, dt_proj, out_proj), it reduces dimensions to focus on the most essential features and discard less relevant information, then increase then back to the original dimension. Selective SSM is core of Mamba blocks, it selects relevant features, containing important information and use it as an efficient processing of sequential data. Skip connections mean to pass some operations, it helps original inputs to flow through the network and retain the raw input features.

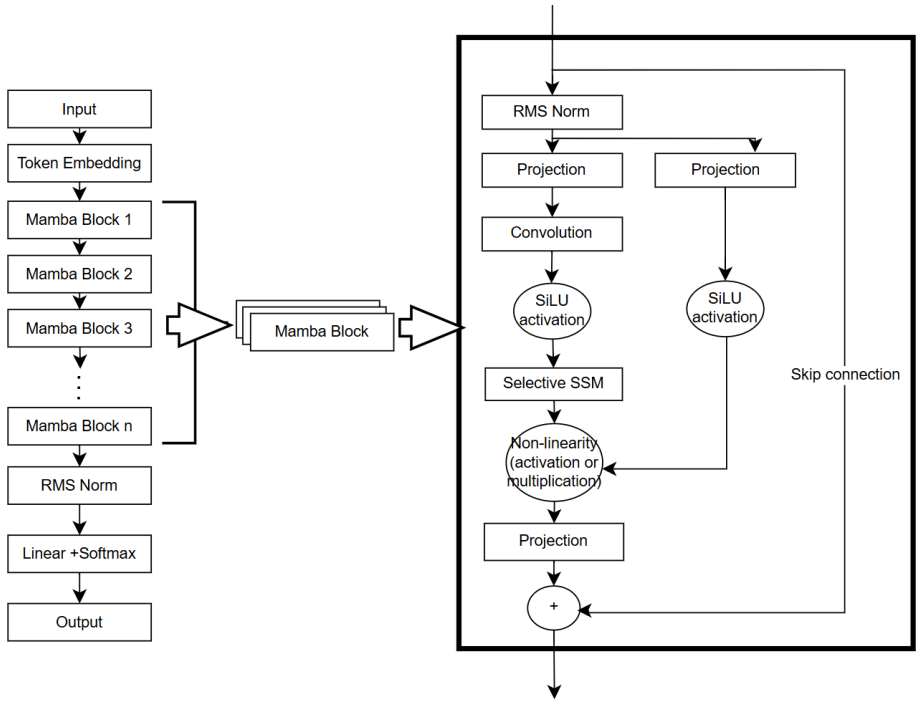

Fig.1 Mamba model with detailed layer[2]

Fig.2 shows a real example of the structure of the Mamba Model. It uses Mixer Model as its backbone, which means it use MLPs, instead of traditional attention mechanisms, which is one of main differences between Mamba and Transformer architecture. The first layer is the embedding layer, where the input has 50280 dimensions, representing the total number of tokens in a predefined vocabulary, and the output dimension is 2560. After that, the values pass through 64 Mamba blocks. Each Mamba block acts as a mixer and includes several operations: first, an in_proj layer, which takes 2560 input features and projects them to 10240 features without using a bias term. Next is a

conv1d layer, operating on 5120 features, with a kernel size of 4, a stride of 1, padding of 3, and grouped into 5120 groups for efficiency. This is followed by the SiLU activation function, introducing non-linearity. Within the block, x_proj, dt_proj, and out_proj are all linear projections that refine the features, and finally, RMS Norm is applied for normalization. After passing through all 64 Mamba blocks, the results are processed by another RMS Norm layer, and the language model head (lm_head) takes the 2560 features, mapping them back to the predefined vocabulary of 50280 dimensions to generate predictions.

```
MambaLMHeadModel(
  (backbone): MixerModel(
    (embedding): Embedding(50280, 2560)
    (layers): ModuleList(
      (0-63): 64 x Block(
        (mixer): Mamba(
          (in_proj): Linear(in_features=2560, out_features=10240, bias=False)
          (conv1d): Conv1d(5120, 5120, kernel_size=(4,), stride=(1,), padding=(3,), groups=5120)
          (act): SiLU()
          (x_proj): Linear(in_features=5120, out_features=192, bias=False)
          (dt_proj): Linear(in_features=160, out_features=5120, bias=True)
          (out_proj): Linear(in_features=5120, out_features=2560, bias=False)
        )
        (norm): RMSNorm()
      )
    )
    (norm_f): RMSNorm()
  )
  (lm_head): Linear(in_features=2560, out_features=50280, bias=False)
)
```

Fig.2 A real example structure of Mamba model, MambaLMHeadModel

Areas for improvement

Despite these achievements, Mamba still has room for expansion and improvement. For instance, while it performs well with discrete data like text and genomics, its capabilities with continuous data, such as audio, could be further enhanced. Scaling Mamba to handle larger datasets with millions or billions of tokens remains an open area for exploration. Furthermore, parameter tuning, including dynamic parameterization or alternative initialization techniques, could unlock additional performance gains. Leveraging Mamba's strengths for real-time discrete datasets, such as live or streaming data, could further establish its utility as a flexible and powerful tool in sequence modeling.

Part 2 State Space Models (SSMs)

Efficiency improvement

From understanding the basic structure, advantages, and improvements of the Mamba model, efficiency and accuracy are the two main points where the Mamba model aims to make significant progress. The parameters in the state matrices, A, B, C in every layer of the Mamba block is always quite large and consume a significant amount of memory and this problem is also experienced in the Mamba extension for image classification in Mamba Extension. From the paper [1], the author mentions the low-rank projection in the parameterization of Δ. Instead of using low-rank projection, low-rank factorization can be utilized to decompose a large matrix into two smaller matrices to reduce memory and computation. The algorithm that can analyze and capture the main components

of the matrix is Singular Value Decomposition (SVD). SVD can capture the important parts of the signal and help identify and analyze the key components while also separating residual, less significant parts. This is a good approach to improving the audio signals that the authors mentioned for enhancement.

SVD can be applied to the input projection layer, which is a fully connected linear layer for transforming the input features. W_{in_proj} projects the original dimension, d_{model} into an expanded space, which is scaled by a factor of two with d_{model}, resulting in d_{inner}. The weight matrix, W_{in_proj} has the dimension is $2 \cdot d_{inner} \cdot d_{model}$ and this is the part used for SVD method.

$$W = U\sum V^T \quad \text{Eq.5a}$$

$$W \approx P \cdot Q \quad \text{Eq.5b}$$

$$Q = \sum_r \cdot V_r^T \quad \text{Eq.5c}$$

$$P = U_r \quad \text{Eq.5d}$$

$W_{in\_proj} \in R^{(2 \cdot d_{inner}) \times d_{model}}$ is decomposed into two smaller matrices, $P \in R^{(2 \cdot d_{inner}) \times r}$ and $Q \in R^{r \times d_{model}}$, where r is the rank used to control the dimension of the reduced space. The Eq.5a shows the general equation for SVD, and the Eq.5c and Eq.5d represents Q and P, respectively. Q maps input features to the reduced r-dimensional space and it carries the most important information. P is basic maps the reduced dimension back to the output space. If it possible, it is a good chance to give few percent on the less important parts, where can be found as original weights matrix minus the low-rank weights matrix, to add as a backup classification. For the size of W_{in_proj} reduces from $2 \cdot d_{inner} \cdot d_{model}$ to the size of P and Q in total, which is $(2 \cdot d_{inner}) \times r + r \times d_{model}$, which is much smaller than the original one and the requirements of memory space is reduced a lot. Besides that, the calculation involves the weight matrix and input is simplified. Output used to be $Output = W \cdot input$, now becomes to $Output = P \cdot (Q \cdot input)$. Since SVD calculation is quite completely and easy to use, this application can be easily added into Mamba Model.

For the training part, P and Q are directly used as trainable parameters, replacing the matrix W. This means that during backpropagation, P and Q need to be updated. Although there are two parameters, their dimensions are much smaller than W. Furthermore, since P and Q are initialized during the setup, no additional effort is required to reconstruct W.

For the testing part, it is important to add the total training and testing times and FLOPs of the two models, one using the original matrix and one using the proposed method. Additionally, the accuracy of both models should be evaluated and compared to ensure the proposed method maintains performance while improving efficiency.

Mamba Model Extension

Introduction

The extension of the Mamba model for this project is medical image classification. The inspiration comes from its ability to capture long-range dependencies and sequential patterns in complex datasets. Medical images, while discrete, often exhibit hidden relationships between spatial regions that reflect transitions signaling disease progression. Testing the capabilities of Mamba in medical image classification could help it identify critical features and pinpoint the primary issues present in the images. Beyond classification, the sequential modeling abilities of Mamba could enable it to

predict the sequence of problems that might arise from the identified issue. If successful, this capability could assist doctors in detecting disease progression, anticipating subsequent complications, and enhancing patient care through early intervention.

Image classification is a well-established area with numerous methods available to solve the problem, such as Random Forests and Convolutional Neural Networks (CNNs) with various architectures like AlexNet, VGG, and ResNet. To expand into predicting the sequence of problems that might follow, it is a good idea to combine CNN models with the Mamba model. With the Mamba layer and its state-space modeling capabilities, it can better capture long-term dependencies and potentially improve the model's accuracy and efficiency. This approach also highlights the extension capabilities of Mamba, showing the types of models or problems that it can address.

Dataset

The dataset used in this project is MHIST dataset [5], formally known as the Minimalist Histopathology dataset. It contains fixed-size images, 214 x 214 pixels, of colorectal polyps. It comprises 3152 images in total: 2175 for training and 977 for testing. The MHIST dataset contains two classes of colon polyp images: 2162 images of hyperplastic polyps (HP), which are benign, and 990 images of sessile serrated adenomas (SSA), which are considered precancerous. In the test dataset, 360 images of SSA and 617 images of HP, the images are shown in Fig.3 and Fig.4 in results.

Method

For the project, the baseline classification model is CNN with a simple ResNet18 architecture, since ResNet18 is a common methodology to classify the medical histology images. The extension is combing ResNet18 with the Mamba to perform a hybrid Model, where the hybrid model use ResNet18 as a backbone to extract the features, and pass to Mamba Sequential Processing to enhance feature representation and sequential processing to compute the prediction.

The baseline model uses a pretrained ResNet18. The first four layers in the model consist of convolution operations, batch normalization, ReLU activation functions, and pooling layers. The final fully connected layer in the model is customized with a linear transformation layer. This layer reduces the feature dimensionality to 256 and adds layer normalization for stable training. The activation function in this part is the Gaussian Error Linear Unit (GELU) activation function because it provides smoother gradients compared to ReLU. Dropout is added with a value of 0.1. It is used in the baseline model to prevent overfitting and to overcome the memory overwhelming problem in the hybrid model. Since this is a baseline, the setup should remain the same. The last transformation layer maps the features from 256 dimensions back to the number of classes to perform the classification task.

The hybrid model begins with a pretrained ResNet18 backbone using the same four layers (convolution, batch normalization, ReLU, and pooling) as the baseline model. Instead of connecting directly to the classification layer, the features are first normalized using layer normalization. The model then introduces a Mamba layer for sequential state-space modeling. This layer starts with an input projection that doubles the model dimension, creating separate paths for the main features and gating mechanism. A 1D convolution with kernel size 4 processes the temporal patterns in the

features. The core of the Mamba layer is its state-space dynamics, which uses four key parameters: A (learns how to optimize past information), B (controls input influence on hidden states), C (maps hidden states to outputs and determines relevant features), and D (provides skip connections to preserve direct input patterns). The layer uses SiLU activation and projects the processed features back to the original dimension. Finally, the classification head follows the same structure as the baseline model: a linear layer reducing to 256 dimensions, followed by layer normalization, GELU activation, dropout (0.1), and a final classification layer for distinguishing between SSA and HP classes. The basic structure of the two models is shown in Fig.5.

The rest of the training and testing procedures are identical for both models, using the Adam optimizer and cross-entropy as the loss function. For comparison purposes, the total training and testing times are recorded. The code also provides confusion matrices for comparing binary classification performance, and use FLOPs (Floating Point Operations) to measure the computational complexity of the models.

Discussion

The results of the testing phase for the two models are shown in Fig.6. The differences between the two models are not substantially large. The test loss for the baseline and hybrid models is 0.5246 and 0.5929, respectively. The complexity of these two models is comparable, with both requiring over 300 seconds for testing. However, in terms of overall performance, the hybrid model outperforms the baseline model. The overall accuracy increased from 81.99% to 83.93%, with the testing time decreasing by less than 5 seconds and the training time reducing by approximately 27 seconds for the same datasets, despite the hybrid model having slightly higher complexity than the baseline. Examining the confusion matrix for specific class performance, both models show similar recall for the SSA class at 0.6861, but the hybrid model demonstrated improved precision for both SSA and HP classes, increasing from 0.7968 to 0.8488 and from 0.8306 to 0.8353, respectively. The hybrid model shows particularly strong performance in identifying HP class samples, with the recall value increasing significantly from 0.8979 to 0.9287, resulting in an impressive F1-score of 0.8795 for the HP class. These results suggest that although adding the Mamba layer for sequential state-space modeling increased the model's complexity, it effectively improved both the efficiency and accuracy of the model in capturing relevant features.

For the model improvements, the hyperparameters tuning should be accounted. For example, in the Mamba layer, the model and state dimension, kernel size in the convolution layer and the expansion factors should be considered based on the datasets and the operation system. Additionally in the training part, learning rate, batch size and number of epochs are all should be carefully selected. The dataset used in this model is not so large, but for the large datasets, the model should add more Mamba layers to process the features. For each Mamba layers, splitting the features in different size simultaneously, and gather the information about the features. This way could help the model preform even better.

Result

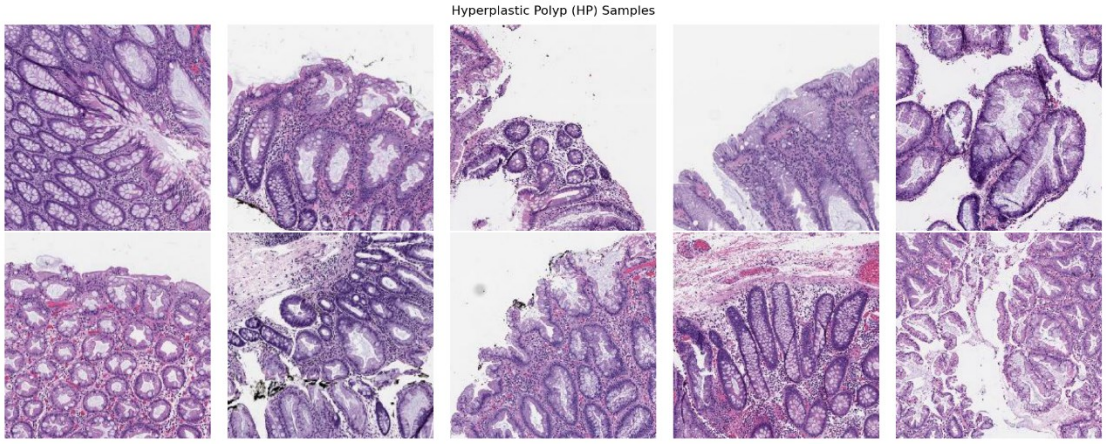

Fig.3 Randomly selected 10 images in HP class from the MHIST dataset

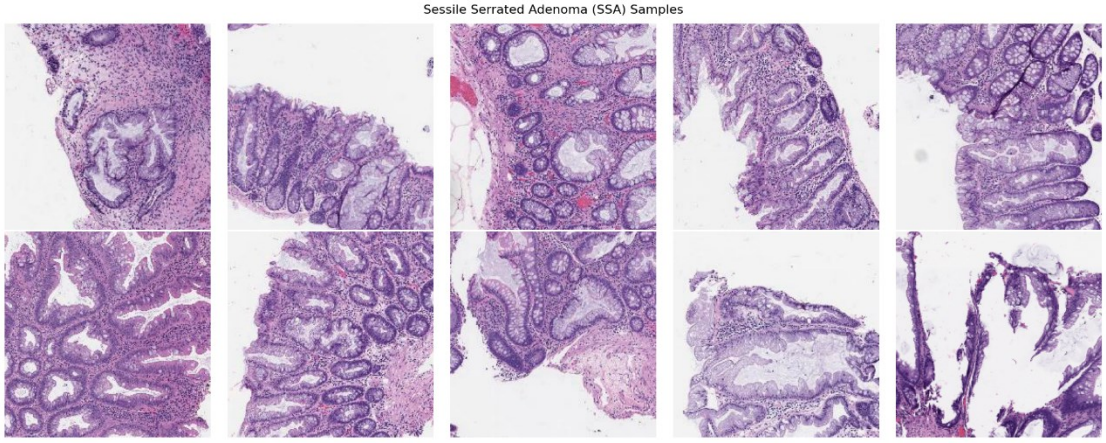

Fig.4 Randomly selected 10 images in HP class from the MHIST dataset

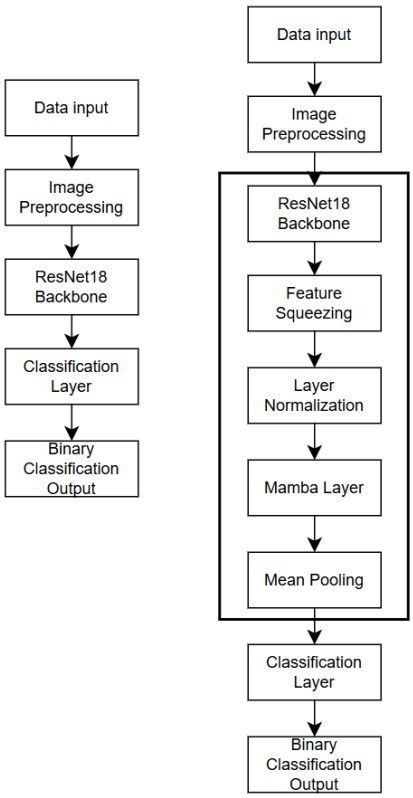

Fig.5 Architecture of the baseline model and optimized model for Mamba extension

```
Test Results:
Test Loss: 0.5246
Overall Accuracy: 81.99%

Detailed Classification Report:
              precision    recall  f1-score   support

         SSA     0.7968    0.6861    0.7373       360
          HP     0.8306    0.8979    0.8629       617

    accuracy                         0.8199       977
   macro avg     0.8137    0.7920    0.8001       977
weighted avg     0.8181    0.8199    0.8166       977


Training Time: 778.52s
Testing Time: 310.97s
Model FLOPs: 5.959859e+08
```

```
Test Results:
Test Loss: 0.5929
Overall Accuracy: 83.93%

Detailed Classification Report:
              precision    recall  f1-score   support

         SSA     0.8488    0.6861    0.7588       360
          HP     0.8353    0.9287    0.8795       617

    accuracy                         0.8393       977
   macro avg     0.8420    0.8074    0.8192       977
weighted avg     0.8403    0.8393    0.8350       977


Training Time: 751.11s
Testing Time: 305.63s
Model FLOPs: 5.968051e+08
```

Fig.6 The detailed testing results of the baseline and hybrid models

Codes

https://github.com/Alicesyz/ECE1512_2024_ProjectB_SSMs_VLMs_SiyuZhang

Appendix

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

[2] Grootendorst, M. (2024, February 19). *A Visual Guide to Mamba and State Space Models.* Exploring Language Models. Retrieved from https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

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

[4] Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling, 2023.

[5] Wei, J., Suriawinata, A., Ren, B., Liu, X., Lisovsky, M., Vaickus, L., Brown, C., Baker, M., Tomita, N., Torresani, L., Wei, J., & Hassanpour, S. (2021). A Petri Dish for Histopathology Image Analysis. In *Artificial Intelligence in Medicine: 19th International Conference on Artificial Intelligence in Medicine, AIME 2021, Proceedings* (pp. 11–24). Springer.

[6] ECE1512_Project_B_Details

[7] ECE1512_AssignmentB