Comparative Analysis of Attention-Based, Convolutional, and SSM-Based Models for Multi-Domain Image Classification

by

Mondrita Biswas
21101056
Sayeedur Rahman
21101281
Syeda Farhat Tarannum
21101016
Dipro Nishanto
21101032
Md Aqeed Safwaan
21101066

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
BRAC University
January 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Mondrita Biswas
21101056

Sayeedur Rahman
21101281

Syeda Farhat Tarannum
21101016

Dipro Nishanto
21101032

Md Aqeed Safwaan
21101066

Approval

The thesis/project titled "Comparative Analysis of Attention-Based, Convolutional, and SSM-Based Models for Multi-Domain Image Classification" submitted by

1. Mondrita Biswas(21101056)
2. Sayeedur Rahman(21101281)
3. Syeda Farhat Tarannum(21101016)
4. Dipro Nishanto(21101032)
5. Md Aqeed Safwaan(21101066)

Of Fall, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 31, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Amitabha Chakrabarty
Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Md Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

The increasing frequency and severity of environmental and societal challenges, such as natural disasters, medical diagnostics, and agricultural threats require the development of efficient and scalable detection and classification systems. Lightweight and fast models deployed on edge devices, such as surveillance drones, portable diagnostic tools, or agricultural sensors, can address constraints of network delays, adverse conditions, and bandwidth limitations often faced by autonomous technologies. Transformer-based models using attention mechanisms, trade off computational costs to achieve high accuracies in these classification tasks. Recently, State-space models (SSMs) have emerged as a promising alternative in areas where long-range dependence on data is crucial but computational efficiency is particularly important. This research explores the application of Attention-Based, Convolutional, and SSMs, particularly Vision Mamba (ViM), in diverse domains: wildfire detection, plant disease identification, and skin cancer diagnostics. Finally, the feasibility of knowledge distillation in ViM is examined using the information gathered from a thorough evaluation and model comparisons. Evaluations highlight that while CNN models consistently achieved the highest accuracy, ViM Tiny is the most memory-efficient, requiring only 0.03GB of GPU memory. ViM Tiny (7.60M params) achieved 70.60% accuracy in wildfire detection, matching DeiT Base's (85.80M Params) 70.62% accuracy. The SSM-based models also had the fastest convergence rate. These models achieved promising accuracies in plant disease classification (98.71%–99.65%) and skin cancer detection (87.03%–90.16%), highlighting their potential for efficient and scalable vision tasks. In the context of wildfire detection, knowledge distillation with EfficientNet B7 as a teacher model further improved ViM Tiny's accuracy from 70.6% to 85.32%, highlighting its potential for lightweight, high-performance applications in critical scenarios.

Keywords: State Space Models, Vision Mamba, Mamba, Knowledge Distillation, Multi-Domain Applications, Attention-Based Models, Convolutional Models

Acknowledgement

First and foremost, we would like to express our deepest gratitude to our supervisor, Dr. Amitabha Chakrabarty, for their unwavering support and guidance throughout the research process. His experience, intelligent feedback, and patience helped shape the direction of this thesis. His dedication to perfection pushed us to push ourselves and aim for the greatest academic standards.

We are also very grateful for our parents' love and encouragement. Their conviction in our ability and constant support have been our greatest motivators. We owe our work, in part, to their steadfast faith in our abilities.

This thesis would not have been possible without the collective encouragement and assistance of these individuals. We are forever grateful for their contributions and the confidence they instilled in us.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN Convolutional Neural Network

FM Foundation Model

$HiPPO$ High-order Polynomial Projection Operators

LLM Large Language Models

$LSTM$ Long Short Term Memory

MLP Multi-Layer Perceptron

MoE Mixture of Experts Architecture

RNN Recurrent Neural Network

$S4$ Structure State Space Sequence Model

$SOTA$ State of the Art

SSM State Space Model

ViM Vision Mamba

ViT Vision Transformer

ZOH Zero Order Hold

Chapter 1

Introduction

The emergence of large-scale neural network-based models, also called Foundation Models (FMs), has been instrumental in the surge in application-specific advancements in Machine Learning and Artificial Intelligence. These models, built to handle enormous volumes of data, have greatly improved our capacity to resolve challenging issues in various fields. These models can be divided into three categories based on their area of focus; FMs can be specifically trained for language modeling (GPT, BERT, LaMDA) or vision-related tasks (ViT, DALLE) or both, with some models like CLIP [26] combining both modalities. This flexibility has allowed FMs to excel in tasks ranging from text generation to image processing.

It is important to note that the models mentioned above draw heavily from the Transformer architecture and its core attention module [43], which has transformed how sequence modeling is approached. Sequence models, as opposed to feed-forward models, are designed to process and generate outputs from data sequences, even in cases where the data may be highly dependent on long-distance connections between inputs. Because of this characteristic, the Transformer architecture has outperformed the previous models, such as Long Short-Term Memory (LSTM) and RNNs, in various NLP tasks which have struggled to capture long-term dependencies in processing long text data sequences. Building on the pioneering work done on the Transformer architecture, the Vision Transformer (ViT) model was proposed in [25] which brings the major advancements in sequence modeling to Vision backbones. ViT treats each image as a sequence of size 16 by 16 and adds positional embeddings to better learn spatial information.

Purely neural network-based architectures have found it difficult to match the sequence modeling capabilities of transformers. This is because sequence models require operating on arbitrarily long data sequences where the data may or may not have dependencies, that are essential for generating the correct outputs. Pure neural network architectures fail to store information over lengthy sequences due to vanishing or exploding gradients or the lack of feedback loops in their architectures. Transformers, on the other hand, successfully overcome this fundamental limitation by enabling the model to give each sequence segment equal attention through its self-attention mechanism within some arbitrary context window.

However, more recently, researchers have explored avenues to improve the efficiency of transformer architectures from the existing quadratic time to something closer to linear time. The benefit here is two-fold: it will bring massive efficiency improvements to the models enabling them to use less computational resources and it will allow efficient upscaling of context windows which would further improve sequence modeling capabilities enabling models to learn over much longer sequences. A huge body of work has gone into making attention more efficient[42][18][10], all of which come with their limitations. Despite improvements in memory efficiency and context window sizes, there is still a gap in addressing very long sequences efficiently and effectively.

Recently, state space models have emerged as a promising alternative with long sequence dependence modeling capabilities in linear time[30][52]. These include a wide range of proposed models such as the Structured State Space Sequence (S4) model, the HIPPO framework[19], and very recently, the Mamba[50] architecture.

1.1 Research Problem

The current boom in AI, primarily driven by groundbreaking advances in generative models such as DALLE and GPT-3, has compelled people to look for ways to make transformers and attention less computationally expensive to minimize costs and emissions. Moreover, the ubiquity of AI in everyday life means that people are increasingly looking for ways to incorporate AI into previously untapped domains.

The computation cost of transformers on downstream domain-specific tasks depends on several factors during pre-training, training, and inference. With respect to input sequence length, N, both computational and memory costs of transformers are quadratic. This limits the usefulness of self-attention in settings where long-sequence dependencies are instrumental for correct outputs. In addition, it limits the context window sizes which is a fundamental limitation that is preventing models from learning over very long sequences of data.

1.2 Research Objective

Over the years, a lot of research has gone into making transformers and attention more efficient. However, these efforts have mostly yielded models and architectures with limitations and haven't achieved linear-time sequence modeling capabilities to match transformers. Lately, a new class of State Space Models has been proposed, which claim to perform sequential modeling in linear time with performance on par or even better than transformer-attention-based architectures [30][50][78].

Originally inspired by the Kalman Filter which dates back to the 1960s, State Space Models have emerged as a viable alternative to transformer-based architectures. While Mamba has proved very effective in sequence modeling in natural language processing, its ability to act as vision backbones in downstream image classification

and segmentation remains unexplored. Many different Mamba-based vision backbones have emerged, some of which have proposed incorporating lightweight, linear scaling attention mechanisms [53]. However, Vision Mamba (ViM) [78] is by far the leading mamba-based vision backbone with zero percent attention.

Vision Mamba proposes a bidirectional architecture, inspired by BERT, a positional encoding scheme, and a patch sequence representation scheme inspired by ViT and achieves encouraging results in the ImageNet-1k classification task. Nevertheless, its effectiveness on domain-specific classification tasks remains unexplored. In addition, it remains to be seen whether vision backbones based on Mamba exhibit the same generalization capabilities of Foundational Models based on ViT. Our research aims to bridge this gap by applying a pre-trained version of Vision Mamba to three important domain-specific classification tasks and exploring the feasibility of knowledge transfer between Vision Mamba models and between other models and Vision Mamba. Knowledge distillation is still relatively unexplored in SSMs, particularly in SSM-based vision backbones, we aim to bridge this gap too as this is an interesting prospect to explore whether SSMs transfer and/or learn knowledge from other models like pure Neural Network models do.

1.3 Research Outline

Chapter 2 sets a background for our research, discussing the key theoretical aspects. Chapter 3 provides a thorough review of all past works related to our research objectives. Chapter 4 discusses the datasets in each of our chosen domains and provides an analysis of these datasets. Chapter 5 explores the inner workings and architectures of all seventeen models and their variants which were used for this research. Chapter 6 discusses our core methodology for implementation, such as dataset preprocessing techniques, training setups, and chosen evaluation metrics. Chapter 7 demonstrates our results. Chapter 8 provides a thorough analysis of our key findings. Furthermore, Chapter 8 also explores the future directions of our research. Chapter 9 is our final chapter providing a conclusion.

Chapter 2

Background

State-space models were originally inspired by the Kalman Filter and have extensive applications in control engineering where it is primarily used for representing dynamic states of a system. To understand their relevance in machine learning, we must take a detour to discuss some fundamental limitations of existing machine learning paradigms, namely, the sequence modeling limitations of feed-forward neural networks, the vanishing and exploding gradient problems in RNNs and the quadratic time complexity with respect to the sequence length and context windows of attention blocks.

2.1 Feed-forward models and sequential data

In the mid-2010s, feed-forward neural network architectures were the state-of-the-art models available for vision-related tasks.

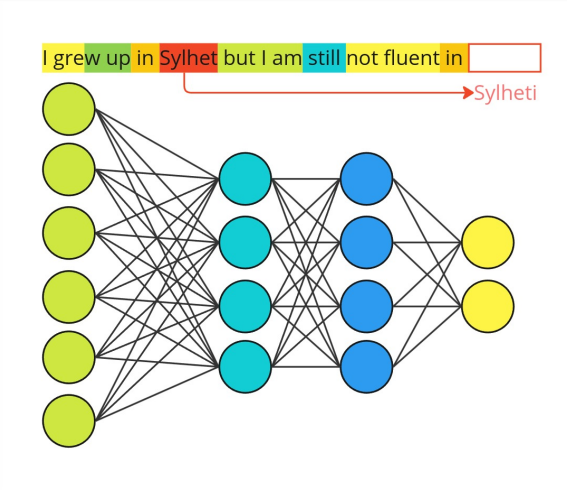


Figure 2.1: Feed-forward neural networks for predicting the next word in a sequence

The advent of Convolutional Neural Networks (CNNs) revolutionized computer vision, however, they struggled to model data with long-range dependencies due to

the absence of sequential information available to the model. This was particularly problematic for textual data, as illustrated in 2.1, and vision-related tasks, particularly for high-resolution images. Figure 2.1 illustrates a classic NLP task of predicting the next word in a sequence of words where the sequence is broken down into tokens. To correctly predict the last word in the sentence, the models need to be aware of the sequence of the text generated in the past. Let's assume that each word is a token in some language model. Therefore, the model needs to hold some state of previous tokens to be able to generate the correct next word.

Sequence modeling in computer vision is interesting because images have a spatial aspect to them. Modern sequence models such as the Vision Transformer and similar architectures circumvent this problem by adding position embeddings to "patches" of an image, where each image patch is a small portion of the whole image typically of the dimensions 16×16. Therefore, we can observe the same kind of long-range dependencies in images as previously observed in the NLP task of predicting the next word in a sequence of words.

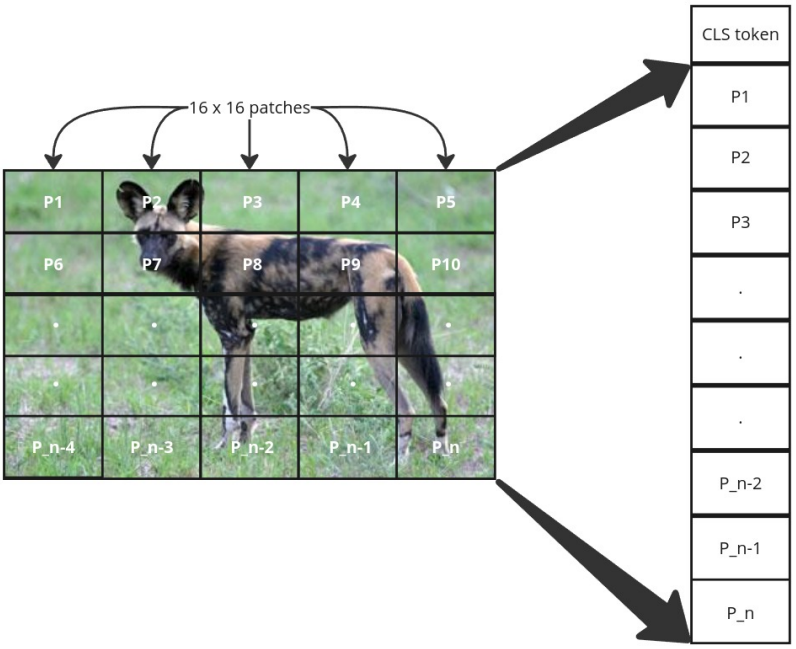


Figure 2.2: Images are divided into patches and projected onto a vector space which introduces sequence dependencies.

Figure 2.2 illustrates how an image might be represented as a sequence of patches to be fed into a transformer. The image is projected onto an n-dimensional vector space where each patch is associated with a positional embedding to learn the spatial features of the image. It is easy to imagine how sequential dependencies might be relevant here, particularly for high-resolution images. The distance between the patch containing the head of a dog and the patch containing the tail of a dog might be large but they are essential for the model to be able to classify the images as that

of a dog.

2.2 Recurrent Neural Networks and the hidden state

To address sequence modeling limitations of traditional feed-forward neural networks Recurrent Neural Network (RNN) was proposed as a possible alternative to simple feed-forward networks. This architecture ensured that models had access to some prior state, which would be continuously learned with data progression along the model in a closed loop. This prior state, more commonly referred to as the Hidden State H_t, was then combined with input x_t, to generate loss or to predict outputs. Since H_t is learned or updated at every time step, it behaves like any other learned parameter. However, over time, it learns sequential patterns better than a simple feed-forward network. A slightly different version of the backpropagation algorithm, known as the Backpropagation Through Time (BPTT) algorithm, is used to learn these parameters, which leads to quadratic time complexity with respect to the hidden state parameters and linear time scaling for sequence length [17].

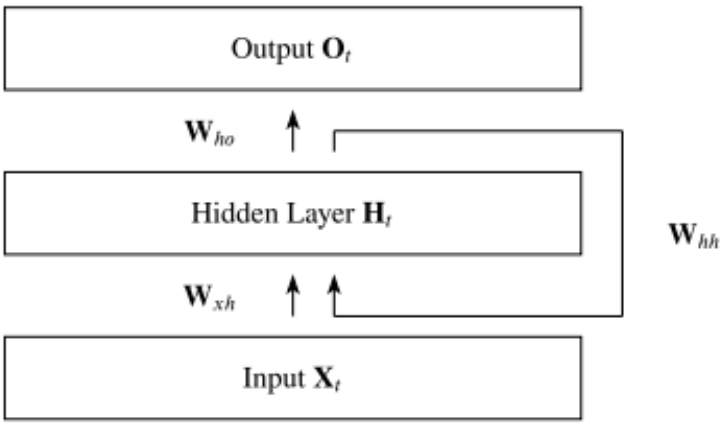


Figure 2.3: Recurrent Neural Network incorporating hidden layer

2.3 Vanishing and Exploding gradients

Due to the chain rule of derivatives [17], the BPTT algorithm requires several matrix multiplications between partial derivatives. This means that if the partial derivative is very close to zero or has a large magnitude, it could result in very small or substantial gradients. This either overstates the contribution of the hidden state or reduces its effect on the model.

To address this problem, Long Short-Term Memory (LSTM) units were introduced. LSTMs store information outside the path of data flow in structures known as gated cells. However, LSTMs proved to be computationally expensive, and their ability to

model long-sequence dependencies, although much better than normal feed-forward networks, proved inadequate in real-world scenarios.

2.4 Transformers and Attention

The transformer architecture and attention [43], initially inspired by the inherent human ability to selectively focus on certain relevant information in the environment while ignoring less relevant things in their surroundings, revolutionized deep learning. At the core of the transformer architecture lies the attention block which along with the MLP layers enables the transformer to learn fine-grained, generalizable information over many layers to learn nuanced representation of data and later use that nuanced representation to make accurate predictions.

While the transformer is comprised of blocks of attention followed by MLP and normalization layers, our core focus here is the attention block. Initially, an embedding matrix consisting of all patches/tokens with respect to context window size is passed into the attention block. Each block consists of three learnable parameter matrices: the query matrix (Q), the key matrix (K), and the value matrix (V). The query and key matrices are multiplied with each patch or token in the embedding matrix to produce a tensor called the attention map as illustrated in table 2.1. It should be noted that we are restricting our discussion to only self-attention here with a token context window of size 6 for illustrative purposes. However, the same principle applies to cross attention with the only difference being that the keys are multiplied with a different embedding matrix.

To summarize the purpose of the attention map, at a very high level, it helps the model learn relevant information from all the other tokens in its context windows. By computing the dot product, the model emphasizes tokens that are closer to each other in the n-dimensional projection space, where n is the dimensions of the query and key vectors. Finally, a softmax function is applied along each column and then each column is multiplied with the learnable value parameters.

	$\mathbf{P_1} \times \mathbf{Q} = \mathbf{X_1}$	$\mathbf{P_2} \times \mathbf{Q} = \mathbf{X_2}$	$\mathbf{P_3} \times \mathbf{Q} = \mathbf{X_3}$	$\mathbf{P_4} \times \mathbf{Q} = \mathbf{X_4}$	$\mathbf{P_5} \times \mathbf{Q} = \mathbf{X_5}$	$\mathbf{P_6} \times \mathbf{Q} = \mathbf{X_6}$
$\mathbf{P_1} \times \mathbf{K} = \mathbf{Y_1}$	$X_1 \cdot Y_1$	$X_2 \cdot Y_1$	$X_3 \cdot Y_1$	$X_4 \cdot Y_1$	$X_5 \cdot Y_1$	$X_6 \cdot Y_1$
$\mathbf{P_2} \times \mathbf{K} = \mathbf{Y_2}$	$X_1 \cdot Y_2$	$X_2 \cdot Y_2$	$X_3 \cdot Y_2$	$X_4 \cdot Y_2$	$X_5 \cdot Y_2$	$X_6 \cdot Y_2$
$\mathbf{P_3} \times \mathbf{K} = \mathbf{Y_3}$	$X_1 \cdot Y_3$	$X_2 \cdot Y_3$	$X_3 \cdot Y_3$	$X_4 \cdot Y_3$	$X_5 \cdot Y_3$	$X_6 \cdot Y_3$
$\mathbf{P_4} \times \mathbf{K} = \mathbf{Y_4}$	$X_1 \cdot Y_4$	$X_2 \cdot Y_4$	$X_3 \cdot Y_4$	$X_4 \cdot Y_4$	$X_5 \cdot Y_4$	$X_6 \cdot Y_4$
$\mathbf{P_5} \times \mathbf{K} = \mathbf{Y_5}$	$X_1 \cdot Y_5$	$X_2 \cdot Y_5$	$X_3 \cdot Y_5$	$X_4 \cdot Y_5$	$X_5 \cdot Y_5$	$X_6 \cdot Y_5$
$\mathbf{P_6} \times \mathbf{K} = \mathbf{Y_6}$	$X_1 \cdot Y_6$	$X_2 \cdot Y_6$	$X_3 \cdot Y_6$	$X_4 \cdot Y_6$	$X_5 \cdot Y_6$	$X_6 \cdot Y_6$

Table 2.1: Tabular representation of a single head self-attention map with context window size set to 6 for illustrative purposes.

A useful way of thinking about keys and queries is to think of the keys as matching the queries when they align with each other in the smaller dimensional project space

of Q and K matrices. The directions in this space encode some high-level semantic meaning. To measure how well they align, a dot product is computed between the two vectors where a high positive value indicates greater relevance. A full discussion of attention and the transformer architecture is beyond the scope of this paper but the overall process that be written concisely in the form of equation 2.1.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.1}$$

Note: Each value in the attention map is divided by the square root of the number dimensions in the embedding space for mere numerical stability. It has no relevance in the discussion related to the overall architecture and is therefore omitted from our visualization of the attention map.

From the above discussion, we can assert that the overall cost of computing each attention map is quadratic with respect to context size. Compounded by the fact that this needs to be multiplied by the number of heads in a multi-head attention block and then multiplied again by the number of attention blocks in the overall model, it's clear that there is room for exploration of more efficient alternative mechanisms.

2.5 MAMBA: Selective State-Space Models

Mamba is a new class of state-space models that are inspired by the Kalman Filter and state-space representation of dynamic systems from control theory. Similar to the transformer architecture, it was first proposed for language modeling and has now been adapted for various other tasks in AI [50].

Mamba builds on the success of the Structure State Space Sequence (S4) [30] model by packaging the continuous-time, recurrent, and convolutional views of the S4 model into a single efficient model. It then adds time-variant learnable parameters, an efficient hardware-aware parallel scan algorithm, and incorporates the HiPPO (High-order Polynomial Projection Operators) [19] matrix into the evolution parameter. The final model achieves excellent performance on sequence modeling tasks, scales linearly with sequence length, and performs on par with Transformer models for short sequences.

State Space Models

In control theory, a State Space refers to the minimum number of variables required to fully describe a dynamic system which contains information about the current condition of the system, the future possible conditions of the system, and how those conditions may be realized.

The data structures that hold this information about the current condition, or state, of the system are referred to as the state vector. State vectors are analogous to

embedding vectors in transformer architectures except that they are not learned using attention blocks.

The full system can be fully described using only two equations.

$$h'(t) = Ah(t) + Bx(t) \tag{2.2}$$
$$y(t) = Ch(t) + Dx(t) \tag{2.3}$$

Equation 2.2 is known as the state equation while equation 2.3 is called the output equation. *A, B, C, and D* are all matrices of learnable parameters. This is one of the core innovations in Mamba that these parameters are learning and time-variant, the latter implies that the parameters depend on the input at any given time t. Figure 2.4 describes an SSM block where matrix A is called an evolution parameter which dictates how the hidden/latent h(t) state evolves, matrix B and C are called the projection parameters where B dictates how the input at any given time t affects the hidden state and C dictates how the hidden state affects the output. Matrix D is a skip connection that ensures that the input has some direct influence on the output and is often left out of the state space equations. The dimensions of the *A, B, C, and h* matrices are (m, n), (1, m), (m, p), and (n, m) respectively where n is the input vector's dimensions and p is the output vector's dimensions.

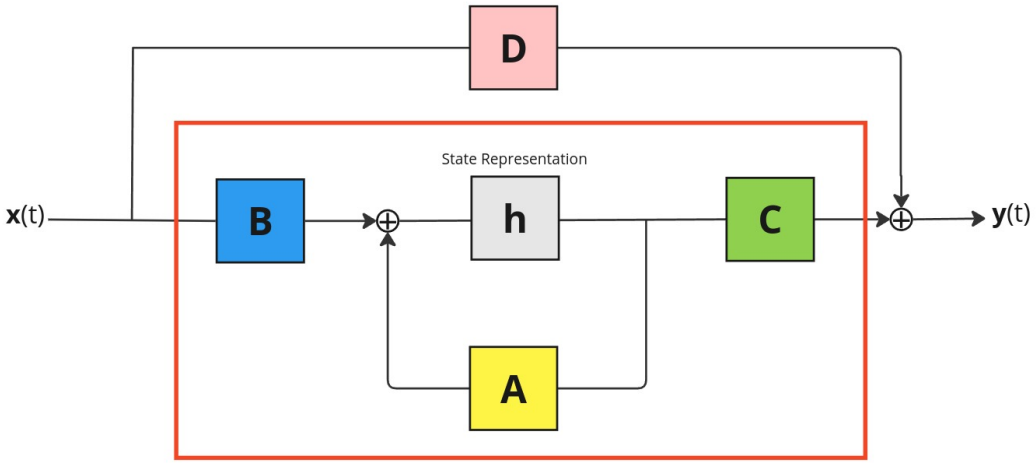


Figure 2.4: Visual representation of a SSM block

There's one last crucial aspect of State Space Models that needs to be addressed. State Space Models take continuous data as input and produce continuous outputs which is problematic as most machine learning tasks deal with discrete data. Therefore, there needs to be a way to feed the model's discrete input data in continuous form and sample discrete data from continuous outputs. There are various ways to achieve this behavior, among which the Zero-Order Hold (ZOH) method is used with Mamba and S4. In addition, the evolution and projection parameters also need to be discretized according to equations 2.4 and 2.5 [50]. With Mamba, even the time step, Δ required for ZOH, is learnable.

$$\overline{A} = \exp(\Delta A) \tag{2.4}$$
$$\overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B \tag{2.5}$$

State Space Representations

A state space model has three distinct representations: the continuous-time representation, the recurrent representation, and the convolutional representation, each of these representations is useful at different stages of the model's lifecycle.

Once the input, output, and parameters are discretized, it is quite useful to represent the State-Space model using the recurrent representation where a new hidden state is recomputed at the end of each time step.

$$\text{Timestep 0:} \quad h_0 = Bx_0, \quad y_0 = Ch_0 \tag{2.6}$$
$$\text{Timestep 1:} \quad h_1 = Ah_0 + Bx_1, \quad y_1 = Ch_1 \tag{2.7}$$
$$\text{Timestep 2:} \quad h_2 = Ah_1 + Bx_2, \quad y_2 = Ch_2 \tag{2.8}$$

There's one other representation called the convolutional representation which is similar to traditional convolution operations applied to other popular neural network architectures except the kernel is derived from the SSM parameters as specified below in equation 2.9 and the output is as specified in equation 2.10. [30]

$$\text{kernel} \longrightarrow \overline{K} = \left(C\overline{B}, C\overline{AB}, \ldots, C\overline{A^k B}, \ldots\right) \tag{2.9}$$

$$y = x * \overline{K} \tag{2.10}$$

Together, these two representations are incredibly powerful. The convolutional representation makes training fast, efficient, and parallelizable while the recurrent representation is fast and efficient at inference time.

Initializing the evolution parameter

Theoretically, the evolution parameter, A, can be initialized to any random value and learned as we would typically learn any other learnable parameter in deep learning. However, the evolution parameter is not just any other parameter here as it needs to remember all relevant information from all previous examples or tokens. Keeping this in mind, it has been experimentally shown that SSMs perform significantly better when initialized using High-order Polynomial Projection Operators

(HiPPO)[19]. The HiPPO matrix is defined as follows:

$$A_{nk} = \begin{cases} \sqrt{(2n+1)(2k+1)}, & \text{if } n > k \text{ (below the diagonal)}, \\ n+1, & \text{if } n = k \text{ (on the diagonal)}, \\ 0, & \text{if } n < k \text{ (above the diagonal)}. \end{cases} \tag{2.11}$$

It attempts to preserve context from tokens seen recently and gradually decays information from tokens seen a long time ago. This bypasses the need for attending to every single token with equal importance and is more powerful than the simple latent state representation in RNNs. It is also interesting how this closely resembles a masked attention map but without the quadratic time complexity.

Time-Variant parameters

So far, the projections parameters B and C have remained the same for all input tokens. However, this means that Mamba can only hold a limited amount of state and forget other important details from the past similar to RNNs. To circumvent this limitation, Mamba proposed creating a new set of C and C matrices for each sample in the batch of tokens.

Therefore, the new dimensions of projection parameters are *(B, L, N)* where B is the batch size, L is the sequence length, and N is the dimensions of the hidden state. Note, that the evolution parameter is still time-invariant as it needs to hold information about the whole dataset. In addition, Mamba also makes the Δ parameter learnable and dependent on the input with dimensions *(*(B, L, D)) where D is the size of the input vectors.

Hardware aware Parallel Scan Algorithm

Since the projection parameters are now dependent on the input, they cannot be used to compute the Kernels according to equation 2.9. A detailed discussion on this algorithm is beyond the scope of this paper but from a high level, the algorithm works by assuming associativity between each time step and computing previous hidden states where necessary. [51]

2.6 Knowledge Disitillation

A simple recipe for improving the performance of machine learning models on any given task is to train multiple different models on the same data and then take the average of their outputs as the output of the overall system. This method is more popularly known as ensemble learning [1]. However, a huge drawback of this method is the overall inefficiency of training all the different models and deploying them for real-world tasks. Both training and inference require increased computing that increases costs and time which may not be available due to constraints. The overall

increase in the size of the overall models also means that it is virtually impossible to deploy these models to devices with limited computational capabilities.

Nevertheless, the idea of polling different models and averaging their outputs to come to an output with a reasonable degree of confidence is an intriguing concept. Moreover, it's analogous to many real-world systems. The opinion of multiple different 'experts' is almost certainly going to be more reliable than that of a single 'expert.' In addition, there are countless examples of this phenomenon in nature, where species adapt different organs and processes to excel in different situations. One particularly intriguing aspect of nature is the knowledge transfer that takes place between parents and their offspring. Inspired by this phenomenon, researchers started exploring ways to transfer knowledge from larger 'teacher' models to smaller, more portable 'student' models. In addition, researchers also explored ways to create generic models that transfer the expertise of many different 'expert' models to a single large general model neural network-based model. As a whole, this field is known as knowledge distillation and was proposed by Hinton, et al in 2015 [6] and is more popularly known in recent times as the mixture of experts (MoE) architecture for Large Langauge Models (LLMs).

Knowledge distillation works by defining a more abstract view of knowledge. Traditionally, the knowledge of a model is thought of as the internal weights, activities, and biases it uses to produce output. Interestingly, a more abstract way of defining knowledge in neural networks is to only consider the raw logits of its output layer. This is experimentally proven to be very effective and is rather convenient for every practitioner as it requires minimal changes to the internal architecture of 'teacher' models [14]. Optimizing loss over the raw logits, the input data, and the output of student models helps the smaller models generalize well over training data. Figure 2.5 is an illustration of the process of knowledge distillation for simple feed-forward neural networks.

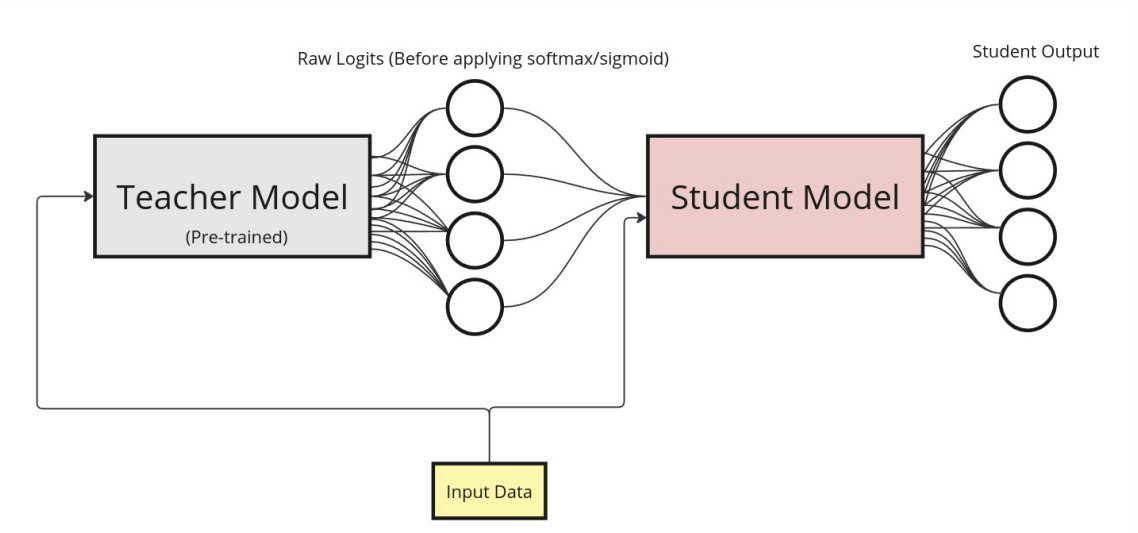


Figure 2.5: Knowledge Distillation from a teacher model to a student model

The idea is to capture the raw logits from the output of a larger pre-trained teacher model before activations are applied to it. These raw logits will act as 'soft' targets

for the student model. In addition, the student model will also take the data as input and treat them as another part of its target. With distillation, there are two approaches we can take: we can either treat the raw logits and input data with equal importance or we can overstate or understate the importance of one of the other. Distillation is done using an added hyperparameter usually denoted by α and another hyperparameter denoted by τ which is the temperate of the final softmax outputs of the two models as shown in equation 2.12.

$$q_i = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)} \tag{2.12}$$

There are two types of distillation that we can do: 'soft' and 'hard' distillation. In soft distillation, the model loss is defined as the KL divergence between the softmax of the outputs of the two models while with hard distillation the loss is defined as the Cross Entropy Loss between the student and teacher models. In both bases, it has been experimentally proven [6] that using a higher value of α produces improved outcomes. This implies that giving more importance to the soft targets of the teacher model improves the transfer of knowledge.

The final loss is defined by equation 2.13.

$$\text{loss} = \text{base\_loss} \cdot (1 - \alpha) + \text{distillation\_loss} \cdot \alpha \tag{2.13}$$

The Mixture of Experts (MoE) approach has yielded more performant and less intensive Large Language Models [79]. It is particularly useful in self-supervised learning which is a perfect fit for training LLMs on enormous bodies of data. However, their application in Computer Vision remains under the shadows of more resource-hungry models and leaves room for careful experimentation.

Chapter 3

Literature Review

In this section, we have coined our in-depth review of CNN models, Transformer models, SMMs, RNNs, and various MAMBA models. The papers discussed below provide theoretical overviews of the models mentioned earlier and have applied the models to tasks (such as NLP, computer vision, object detection, etc.) in various domains. Many of the papers also provide a comparative analysis of transformer- and SSM-based models by evaluating their performance and efficiency in various tasks and across different benchmarks. Furthermore, we have explored several studies that have proposed different variants of the Mamba model that are suited for particular applications such as remote sensing, multi-dimensional data modeling, processing spatiotemporal data, and many more. This chapter shapes our understanding of the important subjects relevant to our work.

3.1 CNN And Transformer-Based Models

In [38] and [34], in-depth examinations of transformers and the evolution of transformer topologies are provided, which were created to meet particular issues in specialized sectors. Paper [34] focuses on transformer optimization for handling extended sequences while minimizing memory and overall computing costs. This novelty dealt with the usage of traditional transformers, which suffered from self-attention mechanisms, making them impractical for long sequences of images, videos, or text documents. This gave rise to further efficient transformer models which had fixed or learnable attention patterns (e.g., Longformer, Reformer), low-approximations (e.g., Informer, Performer), and sparse attention mechanisms (e.g., Big Bird, Sparse Transformer). Innovations like these have greatly lowered computational and memory needs while maintaining competitive performance. This paper also discusses the applications of efficient transformers in a variety of domains, including bioinformatics, computer vision, and natural language processing. Paper [38] examines the transformer models initially proposed for NLP in computer vision problems. Computer Vision received a paradigm shift in every manner with the introduction of transformers, previously dominated by CNNs. The transformers leveraged self-attenuation to capture long-range dependencies in image data. ViT was the first major visual transformer that treated images as sequences of patches like how

normal transformers handled texts. Since then, visual transformers have been applied in image classification, object detection, semantic segmentation, and video processing. CNNs were quite efficiently outmatched by models like ViT, DeiT, and PVT. DETR and Max-DeepLab demonstrated extreme efficiency in object detection and segmentation by capturing both visual details in global and local frontiers. Both [38] and [34] address significant problems with transformer models. The paper [34] focuses on efficient transformers to improve memory and compute efficiency, whereas [38] examines problems such as data and resource needs for training and the lack of inductive biases for additional improvements and advancements. Both studies aim to improve efficiency in their respective domains while optimizing pretraining procedures and broadening the use of transformers to more complicated and resource-intensive jobs. Overall, the papers demonstrate transformers' expanding importance in various fields and sectors, as well as ongoing efforts to make them more scalable and practical for real-world applications.

In [56], a thorough overview of optimizing lightweight CNNs for edge computing is provided. It emphasizes the need to develop efficient models that can function on devices with limited computational power, memory, and energy. The research uses datasets such as ImageNet, MS COCO, and Pascal VOC to assess CNN models for classification, object detection, and segmentation. Key optimization approaches, including quantization, pruning, depthwise separable convolutions, and inverted residual blocks, are presented alongside models such as MobileNet, ShuffleNet, and EfficientNet, which provide a balance of performance and efficiency. Real-time object detection is investigated using YOLO models, specifically smaller forms such as Tiny-YOLO for edge deployment. Hybrid methods, such as MobileViT, are addressed, and they combine the benefits of CNNs and transformers. The study discovered considerable performance increases, particularly with MobileNetv2 and MobileViT, which attain inference speeds of 0.9ms to 7.28ms on devices like the iPhone Neural Engine. ConvNeXt models delivered cutting-edge results that outperformed traditional CNNs and transformers in accuracy. While these models provide efficiency and low latency for edge AI, they have disadvantages such as difficulty with complex tasks, high memory utilization, and poorer transformer performance. Memory optimization, additional refining of hybrid CNN-transformer models, and hardware-specific improvements are potential areas for future research. Overall, the study emphasizes the benefits and disadvantages of using sophisticated AI models in resource-constrained situations and the importance of ongoing development toward lightweight, efficient edge AI systems.

The papers [44] and [41] focus on the application domains of ViT architectures in specific domains, such as human activity recognition (HAR) and biometric authentication through finger vein recognition. Both papers lay out the advantage of using ViTs as they extract global variables and features from images and long sequences, which makes them highly suitable for complex visual tasks. In the case of [44], the ViT-ReT model leverages the ViT to capture features that are spatial from every single video frame and combines with the Recurrent transformer (ReT) to form temporal dependencies between every frame, which is a monumental improvement from the limited and traditional CNN-RNN architectures in HAR tasks. [41] again uses the ViT but as FV-ViT, which is the first application of this in finger vein recognition. The global feature extraction improves biometric authentication, especially

in domains with small datasets. [44] expands how ViT-ReT utilizes spatial feature extraction and the ReT for learning temporal dependencies across frames, which CNNs struggle with, and RNN-based models, although efficient, are computationally intensive. This hybrid model provides a double speedup, maintaining efficiency and accuracy, making it highly suitable for real-time applications in contrast to traditional CNN-RNN models. Both studies stress the potential of transformers in visual tasks, which are becoming more prevalent in fields beyond NLP, which should always work on refining spatial and temporal understandings to improve efficiency and adaptability.

Paper [25], like [44] and [41], provides further information on ViT, transforming image recognition with transformer architecture. Similar to NLP, it processes image patches as tokens sequentially. The model archived high results on the ImageNet dataset, but it needs a large amount of data to train well. This paper has established the foundation for transformers in computer vision applications, and it is frequently mentioned in other studies to demonstrate the efficiency of doing large-scale visual tasks. More work has to be done on hybrid architectures that combine ViT with CNNs to improve feature extraction.

In [28], a low-cost and reliable technique is described for detecting river floods using raw images acquired by off-the-shelf cameras. The system addresses the problem of unaffordable flood warning systems in developing nations such as Brazil, where flooding has serious economic and social consequences. The method employs Deep Neural Networks (DNNs) for semantic segmentation to properly detect water surfaces in rivers, which are subsequently processed by a basic computer vision algorithm to estimate the river's water level. When the water level surpasses a predetermined unsafe threshold, the system automatically alerts authorities and the general public, providing critical warnings to help limit potential flood damage. The method was tested in two rivers in São Carlos, Brazil, utilizing 167,897 photos. The DeepLabv3 DNN architecture was chosen and fine-tuned using transfer learning with the Microsoft COCO dataset. The system operated admirably, with a Mean Absolute Error (MAE) of 3.32 cm, sufficient for anticipating impending river overflow. Despite its shortcomings, the system has the potential to drastically minimize flood damage in urban areas, especially in locations with limited resources for traditional flood detection systems. The authors propose several enhancements, including developing a mobile application for faster alert alerts, inventing portable hardware for easier deployment, and expanding the system to monitor more rivers.

The paper [40], like paper [28], tackles the current issues of traditional techniques of responding to natural disasters. The study focuses on the challenges of measuring damage after natural disasters and recommends the efficient use of CNNs to interpret satellite photos. Traditional methodologies, such as ground surveys, are usually time-consuming and expensive, making CNN-based methods more feasible for rapid and accurate disaster response, especially in flood-, storm-, and heatwave-prone areas. The study employs databases such as xView and Functional Map of the World (FMoW), which contain millions of high-resolution satellite shots, and FloodNet, a library of UAV-captured images, for post-disaster assessment. The methodology takes a two-step approach: semantic segmentation, followed by pixel-by-pixel classification, with CNNs outperforming older techniques. Although these models are

extremely effective, the paper highlights the issues of data labeling costs and class imbalance. The authors recommend developing semi-supervised learning strategies to lessen labeling dependency and reduce class imbalance. Future studies could look at multi-temporal image processing and deeper integration with AI systems to enable faster (less time complexity) and more precise damage assessment. The findings show CNNs' ability to dramatically improve disaster response, particularly in resource-constrained environments, increasing post-disaster damage detection and recovery operations.

The paper [39], working in a similar line of disaster response like [28] and [40], investigates several neural network models for wildfire detection using visible-range cameras that have been placed strategically. The researchers utilize transfer learning techniques, in which models pre-trained on the ImageNet-1K dataset are fine-tuned on a fresh dataset containing 35,000 images. ResNetV2, MobileNetV3, BiT, EfficientNetV2, DeiT, Swin Transformer, and ConvNeXt are assessed on parameters like accuracy, true detection rate, false alarm rate, and latency. Swin Transformer-tiny had the largest area under the curve (AUC) and accuracy (97.95%), indicating more effectiveness in detecting wildfire occurrences. DeiT-tiny had the best overall accuracy (98.13%). On the other hand, ConvNeXt-tiny had the lowest false alarm rate despite being somewhat less accurate. In comparison, ResNetV2-50 had a high false alarm rate of 6.99%. MobileNetV3-small has the least implementation time, making it the fastest model for image processing. However, transformer-based models like Swin and DeiT had slower detection latencies, raising worries about real-time applications. The paper highlights the potential of transfer learning for quick model adaptation even with limited data. Still, it identifies key limitations, such as high false alarm rates in some CNN-based models and the absence of thermal or multispectral data, which could enhance detection reliability. Future research could concentrate on hybrid models that integrate deep learning with classic machine learning approaches and utilize various data types to reduce false alarms and enhance detection speed, offering significant insights for companies planning to install wildfire detection systems. Papers [28], [40], and [39] all emphasize improving the response time of detection by reducing the time complexity, paving the path for future research on SSM-based architectures.

Similar to paper [39], paper [69] and [29] offer a more detailed review of deep-learning models that utilize computer vision for wildfire detection as well, especially neural networks like CNNs and transformers. The paper [69] talks about the drawbacks of real-time monitoring, similarly noting that the readings would not be accurate on bad weather days. Related work overlaps with [25], where transformers are investigated in visual tasks. The multi-scale transformer model for classification and segmentation is evaluated on datasets of UAV-captured wildfire imagery in paper [29]. The model outperforms current SOTA models but struggles when environmental factors like smoke or occlusions reduce visibility. In the future, we can explore combining different data modalities, e.g., thermal and visual imagery, to increase accuracy in diverse environmental conditions.

3.2 Recurrent Neural Networks (RNNs)

The paper [13] introduces a unifying theoretical tool, the Dynamical System Framework (DSF), to compare foundation model architectures, Softmax Attention, SSMs, and RNNs—by examining their similarities and contrasts. While attention models are dominant, the authors argue that SSMs and RNNs provide alternatives with potential benefits, particularly in terms of computational complexity reduction. Attention mechanisms have made models like Transformers so successful due to their ability to capture dependencies over long sequences- but that success comes at the expense of their quadratic complexity with sequence length, making them inefficient in such cases. The standard attention block uses three matrices (WQ, Wk, WV) to generate queries (q), keys(k), and values(v). They are combined within the formula 3.1.

$$y = \zeta\left(\frac{qk^T}{\sqrt{d_k}}\right)v \tag{3.1}$$

where $\zeta(\cdot)$ is the softmax function. SSMs are more computationally efficient (recurrent nature and linear complexity) as they work by preserving and updating a hidden state throughout time, making them ideal for jobs that need sequential input. SSMs are based on the recurrence equations $h_i = Ah_{i-1} + Bu_i$ and $y_i = Ch_i + Du_i$, (h_i: hidden layer, A, B, C, D: learned dynamic matrices). One mentionable selective SSM architecture, S6, parameterizes the recurrence with learned matrices and the discretization step. RNNs process sequential data by maintaining a hidden state updated at each time step. Long Short-Term Memory (LSTM), a very well-known RNN), follows the equations: $x_i = f_i \odot Ox_{i-1} + i_i \odot Ou_i$ and $y_i = o_i \odot \tanh(x_i)$, ($f_i, i_i, o_i$ are forget, input and output gates). RNNs efficiently capture sequential dependencies, particularly with state expansion, allowing them to handle longer sequences and preserve information over more extended periods. The DSF demonstrates that state expansion enhances RNN expressivity and performance, mainly using SSM principles, as shown in qLSTM's S6 transition. It also indicates that SSMs can approach attention mechanisms and efficiently manage extended sequences by linking state changes to input matrices.

For recursive predictions combined with mean loss functions, paper [64] suggests the DB-RNN model to address the problem of blurring in precipitation nowcasting. The authors propose a deblurring network with an adversarial and gradient loss forecasting subnetwork. Previous datasets are employed to validate the model simulations, which suggest that, in general, precipitation forecasts have been made clearer and less biased. In summary, DB-RNN is superior to other RNN-based predictors (i.e., ConvLSTM and TrajGRU) for both short-term and, more particularly, long-term forecasting. Despite these gains, challenges remain, particularly with autoregressive error accumulation. Future work could be spent to enlarge the range of scenarios that DB-RNN applies in other domains that require long-term forecasting as presented in [48].

For example, in [36], the authors suggest a hybrid model for landslide risk prediction using TCNs augmented by an attention mechanism with RNNs. In this study, the authors specifically target overcoming issues resulting from abrupt environmental changes that standard approaches miss. Thanks to a nice combination of TCNs

as temporal feature extractors and RNNs for modeling sequential dependencies, the model reports improved accuracy on landslide prediction datasets relative to stand-alone TCNs or RNNs. This model is tested on multiple geographical datasets (rainfall and terrain features of different geographies). A limitation of this model is a lowered performance in high noise conditions, and studies may consider investigating novel attention mechanisms or multi-modal data to enhance predictive performances. Conceptually, this paper resembles [64], as both use hybrids for long-sequence modeling tasks.

In [15], a recurrent convolutional neural network (RCNN) is applied to perform change detection in spectral-spatial-temporal feature learning from multi-channel data. In this paper, we combine these two approaches as the traditional CNN and RNNs cannot deal with such data. Remote sensing datasets (e.g., Landsat images) are employed for model performance validation. Compared with other classic methods, the RCNN demonstrates better results in time series change detection. This is an analogous study to [36], which uses RNNs for learning time-dependencies in environmental data. In future work, one can consider integrating an attention module into RCNN to perform feature selection; furthermore, many unsupervised learning techniques that leverage the strength of labeled data might be studied for these cases.

Paper [24] also offers an in-depth review of RNN language models and their advancements over time. These models have greatly furthered natural language processing or NLP tasks like machine translation and speech recognition. Tomas Mikolov's introduction of the RNN models in 2010 is key for capturing long-range dependencies in text. This new capability addressed the significant gap in older models, including expert rule-based models (which were rigid and unscalable) and the statistical models of the 1990s struggling with data sparsity. The processing of sequences was made effective by the more flexible approach taken by the RNNs. This paper highlights the key improvements made to RNNs, especially with the introduction of variants like Long Short-Term Memory (LSTM) and Gate Recurrent Units (GRU). Problems such as vanishing gradients, which hindered the training of deep networks in long sequences, were dealt with by the novelty of these models. Early RNN models faced computational challenges despite advancements, especially while processing large vocabularies and long text sequences. To deal with uprising problems, techniques like Noise Contrastive estimation (NCE) and Pointer Networks were introduced, all in the hope of efficiency. Future directions discussed in this paper include a key area of research in hybrid models. Combining RNNs with other neural architectures such as transformers would create a potent and efficient model, which can handle much larger datasets. With newer data sets, this paper emphasizes the need for increased complexity in the scaling models which would not explode the computational costs. Thus, RNN's future lies in improving efficiency, scalability, and ability to handle larger language tasks with greater accuracy.

3.3 State Space Models (SSMs)

Paper [66] then comprehensively surveys SSMs and discusses the possibility of SSMs being an alternative to transformers in long-sequence modeling tasks in multiple domains. The paper [66], [47], [60] addresses the transformer's computational inefficiencies (their quadratic complexity $O(N^2)$. to input sequence length) in long-range tasks. Contrarily, SSMs are highlighted due to their ability to handle long sequences with linear complexity $O(N \log N)$. Paper [66] categorizes SSMs into three main paradigms: gating architectures (uses gating mechanisms to enhance control over sequence modeling, for example, GSS, Mega), structural architectures(models the structural dynamics of sequences like S4, HiPPO, etc.), and Recurrent architectures (captures dependencies between sequences without relying on self-attention like RWKV, LRU, etc.). The research concentrates on Mamba, a well-known SSM that is an upgraded version of the architectures listed above and is designed for long-sequence modeling workloads. It employs selective state propagation to retain useful information and a channel mixing module to improve stability during training, particularly for vision tasks. Mamba outperforms transformers in long-sequence modeling and time-series forecasting jobs because it scales linearly with sequence length rather than quadratically. The paper contrasts SSMs' performance on several benchmark datasets with transformers. Mamba outperformed transformers in Long-sequence NLP testing (the LRA benchmark), obtaining 91% accuracy on sequential CIFAR-10, but transformers dominated visual tasks. Mamba-360 enhances the performance of the base model, hence addressing stability difficulties. The paper [66] investigates the use of SSMs in a variety of fields, including NLP, vision and video, medical applications, speech recognition, and so on. It emphasizes the necessity for additional study on optimizing SSMs for Long-Sequence Vision tasks by improving the SSM architectures of newer models like the Mamba-360.

Building upon the theoretical groundwork laid out in paper [13], the paper [72] shifts focus to a practical application of SSMs in time series forecasting. This paper provides a unique probabilistic time series forecasting method that blends SSMs and deep learning, notably RNNs. The goal is to harness the characteristics of SSMs, such as data efficiency and interpretability while using the pattern recognition skills of deep learning to model complicated temporal patterns in huge datasets. The paper uses a linear SSM to capture temporal patterns (e.g., trend and seasonality) and an RNN to map covariates to time-varying parameters. Kalman filtering is used for fast likelihood computation and training to maximize the likelihood of observed time series data. The paper qualitatively and quantitatively evaluates the performance of the model using several datasets: Electricity dataset (hourly electricity consumption), Traffic dataset (Hourly occupancy rate), Tourism Dataset (monthly tourism demand), etc. The model accurately captures underlying time series patterns from synthetic data and outperforms standard approaches such as ARIMA, ETS, and DeepAR on real-world datasets, performing well in both small and large data regimes by properly predicting seasonality and shared patterns. The model is very data efficient, scalable, and interpretable and consistently outperforms standard forecasting approaches; yet, it has disadvantages such as assuming Gaussian likelihoods, relying on fixed covariates, and increasing complexity. The authors recommend extending the model to handle non-Gaussian likelihoods, better

capturing nonlinear dynamics, and optimizing parameter initialization to increase performance. This research broadens the scope for developing a more generalized version of deep SSM to handle more complex data.

Extending the exploration of state-space models, papers [70] and [51] provide an in-depth overview of employing SSM as a basis for various deep-learning neural network models to solve the constraints of existing architectures such as Transformers, which were mentioned in paper [13], as well as [72]. The authors of publications [70] and [51] describe how SSMs can potentially replace attention processes utilized in models that rely significantly on them for sequential data processing, such as GPT. SSMs are helpful because they use a recurrent architecture to handle large sequences efficiently. The papers [70] and [51] investigate the use of linear and structured forms of SSMs and their impact on overall performance. The paper [70] does a comparative analysis of SSMs versus Transformers through the LRA benchmark, where the Mamba architecture (a selective SSM) demonstrates more computational efficiency in learning longer dependencies. In addition, SSMs provide greater mathematical transparency than Transformers, which are black boxes. However, the author observes that current SSM research is driven by empirical performance rather than systematic control-theoretic design, which has constraints when dealing with nonlinear dynamics. The work [51] advocates the usage of SSMs by providing an in-depth examination of their growth and application in several fields, including NLP, Computer Vision (CV), graph learning, multi-modal data, and time series analysis, demonstrating their versatility across domains (text, photos, videos, graphs, etc.). Both papers [70] and [51] advocate for more principled approaches, hybrid models that combine the capabilities of SSMs and Transformers, and better initialization procedures to increase SSM-based systems' long-term memory and efficiency.

In paper [3], a novel SSNN architecture for effectively modeling dynamical nonlinear systems is proposed [3]. This compensates for the inability of conventional methods to capture dynamic system behaviors and time dependencies. This is at the bottom of a standard state-space method and uses hierarchical neural networks with multiple layers for nonlinear modeling; recurrent connections help memorize temporal patterns. This research uses benchmark nonlinear system datasets and simulated chaotic datasets to test the model, producing a similar performance graph plot on trajectory prediction compared to typical RNNs or feed-forward Neural networks. However, the current work is interfered with by data noise, making extending the SSNN technique to real-world systems difficult. The authors suggest that adaptive learning approaches might be used to improve the model and that extending it to multidimensional modeling would improve interaction robustness.

Following that, papers [37] and [30] provide further experimental evidence, showing the advantages of combining SSMs with Transformers in specific benchmarks. In [37], SPADE is introduced, which integrates a global information-capturing SSM in its bottom layer and local attention mechanisms in the following layers while combining SSMs with Transformers to increase the efficiency of lengthy sequence modeling. Results from experiments indicate that SPADE performs better on the autoregressive tasks and the Long Range Arena benchmark than the S4 and vanilla Transformer models. Future studies should investigate additional local attention processes to improve SSMs further, even though its focus on language modeling

issues may further limit its applicability. The Structured State Space (S4) model outperforms more conventional models like RNNs and Transformers in managing long-range relationships in sequence data, according to [30]. The S4 model performs highly across multiple benchmarks, including 98.3% on voice command classification and 91% accuracy on sequential CIFAR-10, with an average accuracy of 80.48% on Long Range Arena tasks. This is obtained by replacing state-space models with more computationally efficient ones. S4 also reduces memory and processing overhead remarkably compared to previous methods. Studies in the future may focus on enhancing computational kernels and utilizing S4 for a wider range of data modalities despite potential numerical instabilities in some scenarios.

After exploring all the possible comparisons, papers [49] and [83] offer novel methods to improve signal estimate accuracy and processing efficiency in various applications. With a reported 3% reduction on the ETTh1 dataset, the first paper [49] presents token merging strategies that attempt to improve computational efficiency in time series processing. These methods maintain a minor loss of accuracy while offering astounding speedups of up to 5400%. The evaluation, which highlights the importance of token merging across several datasets and shows how Autoformer and FEDformer also benefit from these techniques, includes models such as Chronos and HyenaDNA. To be applied in future research, token merging approaches need to be expanded to various architectures and real-world scenarios. Two primary kernel-based methods for adaptive signal estimation are presented in the work [83]: High-Dimensional Kernel Learning (HDKL) and Kernel-Based State-Space Modeling (KSSM). KSSM uses kernels to give reliable signal and parameter estimation, and HDKL improves localized estimation by building a feature space from several real-valued spaces. Although exact accuracy rates are not generated, simulations demonstrate significant improvements in signal estimation accuracy, especially for tracking body movements and estimating wind speed.

Additionally, Advanced techniques to improve inference and parameter estimation in complex models are reviewed in [8] and [80]. A review of particle approaches for parameter estimation in nonlinear, non-Gaussian state-space models in [8] emphasizes the value of Sequential Monte Carlo (SMC) methods. It includes several particle techniques, including particle filtering and Markov Chain Monte Carlo (MCMC), that provide precise numerical approximations in complex state inference problems. The findings indicate that these methods significantly enhance state sequence estimation and improve both Bayesian and maximum likelihood approaches, though specific accuracy rates are not provided. High computing cost is a disadvantage that limits scalability to bigger datasets and highlights the necessity of more extensive applications in dynamic models. A method called Multi-Fidelity Bayesian Optimization for State-Space Models (MFBO-SSM) is presented in paper [80] and is intended to enhance parameter selection and inference in intricate dynamical applications. This paper mentions how SMC approaches are implemented to handle big systems and investigate Reinforcement Learning (RL) applications for decision-making scenarios. Testing with synthetic gene expression data and real-world stock price data (VIX) increases the accuracy rates when employing VIX data, MFBO-SSM performs prominently better than conventional techniques. Future studies might concentrate on applying MFBO-SSM methods to larger datasets.

After that, the work in [48] aims to compress the S4 model — a widely used state-space-based model architecture for long-sequence processing, through balance truncation and diagonal SS layers that productively simplify the complex system. We also consider reducing memory consumption and computational complexity while keeping model accuracies. Experiments in time-series forecasting tasks, including temperature and stock price prediction, showed that the model maintains more than 90% of original performance with a fraction amount parameter-wise compared to vanilla ResNet. Although the approach is powerful, it has trouble handling higher dimensionality compression quality. This should have partially covered (similar to [3]) what one can do with state-space models in time series prediction, and a focus should have been on making these models faster, as in the case of the methodology from this paper.

Finally, papers [2], [81], and [59] explore the progress in state space modeling and the need for better interpretability of models. To improve data modeling capabilities, the Nonlinear Switching State-Space Models (NSSMs) in Paper [2] combine NSSMs with Hidden Markov Models (HMMs), capturing short-term dynamics while managing longer-term changes. A Bayesian ensemble learning technique is presented for parameter optimization, showing better phoneme segmentation performance than conventional models such as plain HMMs. The NSSM and HMM combination exhibits better segmentation results when the phoneme sequences are known, but relevance to other data types has not been evaluated. In the paper [81], different control and signal processing tasks are addressed in broad SSMs. Online optimization problems are addressed by combining gradient-based searches with Sequential Monte Carlo (SMC) techniques. Although no particular accuracy rates are presented, novel algorithms for sensor scheduling, risk-sensitive control, and decentralized inference are proposed, highlighting the methodologies' relevance in engineering domains. The Mamba architecture, a Selective State Space Sequence Model (SSM) intended for effective long-sequence processing, is examined in Paper [59]. MambaLRP, based on Layer-wise Relevance Propagation (LRP), is developed to improve the explainability of Mamba models. To improve transparency and address dishonest explanations, our method exhibits SOTA performance across a range of models and datasets, with exceptional accuracy rates of 91.97% on SST-2 and 94.15% on the Mamba-1.4B model. Future research may focus on extending MambaLRP to other architectures and high-risk areas, even though the focus is still on Mamba models.

In [46] a new sequencing modeling approach known as Mamba is introduced. It employs chosen state spaces to achieve linear temporal complexity while maintaining Transformer-level performance on various applications, including language modeling, audio, and genomics. The Mamba model incorporates Selective SSMs, in which parameters vary in response to incoming data, allowing the model to focus on important information while discarding irrelevant content. It includes a hardware-aware parallel technique for efficient recurrent computation, which addresses standard SSM restrictions. Mamba streamlines neural network design by removing attention and MLP blocks, which improves speed and performance. The authors of the paper [46] used the Pile dataset (for language modeling tasks), human genome datasets (for genomics), and the YouTubeMix dataset (for audio tasks). On the Pile dataset, Mamba beat Transformers of comparable size by $\times 5$ times inference throughput,

and in language modeling, it performed equivalent to Transformers twice its size. It excelled at tasks like Selective Copying with 99.8% accuracy and performed brilliantly on Induction Heads, extrapolating to sequences of 1 million tokens. In genomics, it outperformed HyenaDNA and Transformer++ in dealing with long-range dependencies, while in audio modeling, it achieved cutting-edge performance, lowering FID by more than 50% on speech datasets. Mamba outperforms Transformers in content-based reasoning and areas such as language, genetics, and music, with $\times 5$ quicker inference and linear scalability. However, it strongly relies on input-dependent parameterization and hardware-specific optimization, which may restrict its usefulness. Improvements could center on better handling of nonlinear dynamics and creating hybrid models that combine Mamba's efficiency with attention mechanisms. Mamba can be used for multi-modal learning and long-sequence tasks in developing areas like video, opening up new research opportunities.

3.4 Related Works in MAMBA

3.4.1 Introduction to Mamba

The Mamba architecture is examined in paper [67] as a new Transformer substitute, especially for managing long-range relationships in deep learning tasks, as suggested by papers [70] and [51] as well. Compared to Transformers, which have computing difficulties because of the quadratic complexity of attention mechanisms, Mamba models, based on classical SSMs, have a near-linear scalability benefit. According to the study, Mamba is a potential paradigm for a wide range of applications since it can handle several data types while retaining high memory management and scalable sequence processing capabilities. The study thoroughly assesses deep learning models such as CNNs, RNNs, and Transformers before selecting Mamba as an appropriate approximation. It explains how the design of Mamba, including its memory-efficient techniques and scanning modes, has evolved to work surprisingly well on tasks related to long-range dependency modeling. The survey highlights Mamba's efficacy and efficiency in NLP, CV, and healthcare. Future research areas include integrating Mamba with other advanced models and increasing its memory management skills. The report also indicates that Mamba models may struggle to capture complicated patterns compared to Transformer-based designs, highlighting the need for additional research to broaden their application across many tasks and domains.

The papers [74] and [68] provide further details on the previously discussed MAMBA model, which has shown quite some potential and promise in advancements of computer vision tasks. Mamba turned out to be a versatile architecture by introducing linear scalability and efficient long-sequence modeling in applications of various vision tasks like image classification, object detection, video analysis, and 3D point cloud processing. Both papers emphasize Mamba's selective attention mechanism, which enhances performance across various domains, keeping computational complexity at a low level. [74] explores how the Mamba model was formed, highlighting the hardware-aware form and design with a simplified state space architecture. This

allows the efficient long handling of sequences. It also shows how Mamba has been applied in pure and hybrid backbones to make many architectures more robust like with CNNs or transformers for performance optimization. The success of Mamba in image processing and more complex tasks like video analysis and segmentation are also mentioned in this paper. Paper[68] adds to the previous discussion by providing a detailed taxonomy of Mamba models, which categorizes them based on various tasks or vision tasks. It is a comparison of Mamba with CNNs and ViTs. It showcases how Mamba's selective SSM provides the balance between global and local feature extraction, with efficient scanning models like zigzag, spiral, and radial patterns that enhance flexibility. Both papers consider Mamba to hold the potential to be a core foundational architecture in vision tasks. Both papers suggest that for future works, multi-modal and in-context learning application research is needed.

3.4.2 Mamba in Remote Sensing

The paper [47] introduces ChangeMamba, an innovative architecture for remote sensing change detection (CD) tasks, building on the Mamba architecture. It proposes different frameworks for the three main CD subtasks: binary change detection (BCD), which uses MambaBCD; semantic change detection (SCD), which uses MambaSCD; and building damage assessment (BDA), which uses MambaBDA. ChangeMamba comprises spatio-temporal relationship modeling to exploit multi-temporal image features for accurate CD. Images are scanned in different spatial directions, enhancing its ability to detect changes. The architecture employs the Visual Mamba (VMamba) encoder, efficiently learning global spatial contextual information from high-resolution remote sensing images. The model is assessed on five benchmark datasets (including xBD and additional CD datasets). It achieves SOTA performance in all three CD tasks without requiring complex training procedures and even with damaged data, confirming its effectiveness in real-world circumstances. It outperformed CNN and transformer models for BCD (high precision and recall), SCD (more thorough classification), and BDA. Similar to study [66], paper [47] emphasizes transformers' quadratic computing complexity, which makes them unsuitable for large-scale remote sensing. Mamba overcomes this by providing linear scalability. CNNs suffer in this domain due to their restricted receptive area; however, ChangeMamba can efficiently capture local and global contexts. Hence, promising scopes open for applying this model in remote sensing using geospatial datasets as a means of disaster control.

The publications [76] and [55] aim to improve the analysis and fusion of hyperspectral and multispectral data in remote sensing employing Mamba models, similar to the paper [47]. [76] presents SSRFN (Spatial-Spectral Interaction Super-Resolution CNN-Mamba Fusion Network), which combines the strengths of CNNs and Mamba to fuse hyperspectral and multispectral images. A novel mutual guidance mechanism is conceived to provide spatial resolution of HSI and spectral resolution of MSI. The extraction will be such that the data types complement each other. CNN is used for local feature extraction while Mamba is used for long-range global feature modeling, and their merging allows the framework to fuse the images efficiently. Similar to [76], [55] describes the 3DSS-Mamba framework, which is intended exclusively

for HSI classification. This model uses a 3D-Spectral-Spatial selective scanning approach to capture global and local information, addressing issues with typical CNNs and transformers. This also includes a Spectral-Spatial Token Generation (SSTG) module, which turns HSI cubes into tokens representing both dimensions. These are later scanned by the 3DSS model employing five different scanning routes to prioritize spectral or spatial dimensions. This approach can classify hyperspectral images with high accuracy, low computational costs, and complexity, in contrast to other methods and frameworks.

3.4.3 Mamba in Image Processing

The paper [60] introduces another extension of the Mamba architecture, Mamba-ND, which enables efficient multi-dimensional data modeling (such as images, videos, weather data, etc.). Selective state propagation extends Mamba's state-space model to handle multi-dimensional data, like 2D images and 3D videos. Mamba-ND processes data by alternating 1D SSM layers across several dimensions, which improves efficiency while retaining performance. The model was evaluated against various benchmarks, including HMDB-51 and UCF-101 for video classification, ERA5 for weather forecasting, and BTCV for 3D medical segmentation. It achieved SOTA results even after significantly reducing the number of parameters compared to Transformer-based models. On ImageNet-1K, its accuracy increased by +3.8% compared to ViT and reduced parameters by 20.7%. In HMDB-51, the model boosts accuracy by +1.3% compared to the Video Swin Transformer with only 44% of the parameters. On ERA5, it increases the anomaly correction coefficient (ACC) by 0.7% compared to Cli-ViT while lowering parameters by 44.5%. In the BTCV dataset, Mamba-ND surpassed ViT with fewer parameters, receiving a DICE score of 84.7. The results clearly show the model's efficiency in using fewer computational resources than earlier transformer-based models. Using Mamba-ND suggests improving climatic and atmospheric modeling accuracy while incurring lower computational costs by using satellite picture sequences, radar data, and other spatiotemporal geospatial data.

A new infrared image super-resolution model called IRSRMamba is presented in the paper [58] by fusing wavelet transform feature modulation with Mamba-based backbone networks. The model successfully captures long-range dependencies and improves local-global information to tackle the problem of handling sparse infrared (IR) image features. The wavelet transform feature modulation block makes multiscale feature extraction efficient and significantly improves the reconstruction of infrared images. For handling long-range dependencies, IRSRMamba's architecture uses convolution layers for shallow feature extraction and the Vision State-Space Module (VSSM) and Residual State-Space Groups (RSSG) for deep feature refinement. In tests on the M3FD and other benchmark datasets, IRSRMamba achieved SOTA performance with a PSNR of 39.33 and SSIM of 0.9439 on the result-A dataset and 44.50 PSNR and 0.9719 SSIM on the CVC10 dataset, surpassing existing models such as RGT and PSRGAN. Using IRSRMamba for other infrared tasks, such as object detection and thermal image processing, may be investigated in future studies.

Papers [63] and [62] present advanced hybrid network architectures for medical picture segmentation, addressing distinct issues. As proposed in paper [63], U-Mamba combines CNNs and SSMs for local feature extraction and long-range dependency handling. In this area, U-Mamba outperforms traditional CNNs and Transformers. It can work with different datasets using a self-configuring technique that eliminates the need for manual calibration. Tests on four biomedical segmentation tasks show that U-Mamba performs better than SOTA approaches in various functions with superior Dice Similarity Coefficient (DSC) and F1 scores. The approach uses a hybrid design to improve long-range interdependence by incorporating a Mamba block into an encoder-decoder structure that records local and global contexts. U-Mamba achieved notable results in 3D abdominal organ segmentation with an average DSC of 0.8683 and in 2D organ segmentation, 0.7625. Future research into applying U-Mamba for other biological imaging applications is also suggested, as are further optimizations for a larger variety of datasets.

Paper [62] introduces a Semi-Mamba-UNet that combines the benefits of Mamba-based networks with the popular U-Net model; this design highlights the drawbacks of labeled data in medical picture segmentation. Cross-supervised learning and pixel-level contrastive learning processes are the primary innovations in Semi-Mamba-UNet. These methods enable the model to generate pseudo-labels from unlabeled input and self-supervise during training. Tests conducted on publicly accessible datasets show a notable increase in segmentation accuracy compared to semi-supervised methods. The Semi-Mamba-UNet architecture overcomes the shortcomings of CNNs and ViTs like Swin-UNet in segmentation tasks by combining Mamba-based encoding-decoding with U-Net to handle long-range associations efficiently. The approach improves feature learning from labeled and unlabeled data using pixel-level contrastive learning. The network can produce pseudo-labels to enhance performance through cross-supervised learning, particularly when there is a shortage of labeled data. Experiments demonstrate that Semi-Mamba-UNet achieved a Dice coefficient of 0.9114 on the MRI cardiac dataset and 0.7627 on the MRI prostate dataset, outperforming seven other semi-supervised frameworks. The research shows that Semi-Mamba-UNet may be used for increasingly challenging tasks, like volumetric data segmentation, and it improves the model's performance in more challenging medical imaging scenarios. According to the study [39], additional validation on a wider variety of medical imaging datasets is necessary to generalize its conclusions completely. A variety of medical specializations with various picture qualities also require customization of the framework.

Paper [57] introduces MambaMIR and its GAN-based variation, MambaMIR-GAN, for medical picture reconstruction (work in the medical domain like paper [63] and [62]. The authors suggest an arbitrary-mask approach that adds unpredictability to the inference process and facilitates Monte Carlo-based uncertainty assessment. In addition to providing uncertainty maps to evaluate the dependability of the outcomes, this enhances the model's reconstruction quality. MambaMIR employs Arbitrary-Masked S6 blocks for image patch processing, while both models are tested on sparse-view CT and fast MRI datasets. For MRI and CT, MambaMIR obtains a Peak Signal-to-Noise Ratio (PSNR) of 27.13 and 37.02, respectively. In high-acceleration situations, MambaMIR-GAN significantly enhances perceptual quality. The paper highlights the model's superior performance compared to SOTA meth-

ods in these tasks. Future work includes expanding MambaMIR to other medical imaging modalities like ultrasound and optimizing it for real-time applications on resource-constrained devices. However, more research is needed to improve its uncertainty estimation mechanism and demonstrate its usefulness in non-medical imaging activities.

A U-shaped Vision Mamba network called UVM-Net is introduced in paper [77] as an effective single-image dehazing method. It addresses the computational difficulties presented by Transformer models, which are resource-intensive even though they are good at capturing long-range dependencies. By combining convolutional layers and State Space Sequence Models (SSMs) through a Bi-SSM block, UVM-Net presents a hybrid structure that is effective and can manage long-range relationships in image restoration tasks. The design efficiently captures local and global information by combining Mamba blocks with U-Net. The Bi-SSM block improves long-range modeling even further by implementing feature map scrolling over the channel domain. With a PSNR of 40.17 and an SSIM of 0.996, UVM-Net outperforms competing models like DehazeFormer in dehazing tasks, according to experiments conducted on public datasets, such as the RESIDE dataset. The work demonstrates UVM-Net's potential for image restoration tasks other than dehazing. The authors recommend further enhancements for low-resource devices to maintain performance and efficiency. However, the study also notes that additional testing is required to apply UVM-Net to more general image-processing tasks. Additional fine-tuning may be required to adapt the model for different kinds of images and specific application scenarios.

3.4.4 Mamba in Object Detection, Tracking and Anomaly Detection

The paper [54] also addresses the previously discussed shortcomings of CNNs and transformer-based models in handling long-range dependency and computational efficiency. A new architecture, MambaAD, is proposed that targets multi-class unsupervised anomaly detection (AD). The model comprises Locality-Enhanced State Space (LSS) modules and Hybrid State Space (HSS) blocks to capture global and local information. The LSS modules use HSS blocks to capture global and local details by combining different information scales. The HSS blocks capture long-range interactions using five scanning methods in eight directions. This hybrid technique improves MambaAD's ability to recognize abnormalities and analyze complex data patterns. MambaAD is tested on six benchmark datasets, including MVTec-AD (a real-world industrial anomaly detection dataset with 5354 photos), VisA (images of 12 different objects), Reak-IAD (150,000 images in 30 categories), and others. MambaAD achieves 98.6% AU-ROC at the image level, 97.7% AU-ROC at the pixel level, and an F1-max of 59.2%, outperforming other baseline models on MVTec-AD. It also achieved 94.3% AU-ROC at the image level and 98.5% at the pixel level on VisA. MambaAD does not sacrifice performance for accuracy because it employs the best features of state-space models and convolutional networks. Furthermore, the model is scalable, performing well on huge datasets while remaining computationally economical. The research emphasizes further optimizing the model for

real-time manufacturing, surveillance, and medical image processing applications.

Like paper [54], paper [65] also proposes a novel object-detection model called Mamba-YOLO that has a hybrid architecture combining strengths of CNN-based local convolutions and SSM-based global attention, making the model computationally efficient while keeping performance top tier. Mamba-YOLO uses SSM for global dependency modeling, which differs from exclusively CNN or Transformer designs by working on linear time complexity, as previously discussed in other papers. The ODSSBlock is a new module in Mamba-YOLO that adapts the SSM structure to object detection by solving the problems of bigger pixel sizes and channel dimensions. It introduces two specialized components: the LSBlock, which improves the capture of local image dependencies to mitigate SSM's weaknesses in local spatial modeling, and the RGBlock, which combines gated aggregation and residual connectivity to improve channel correlation handling, increasing the model's overall robustness. Two benchmark datasets for object detection were used: MS COCO and PASCAL VOC. On the COCO dataset, Mamba-YOLO outperforms YOLOv8 by 8.1% in terms of mean average precision (mAP). The model surpasses other YOLO models in accuracy and speed, making it ideal for real-time object recognition. Even while the model has lower computing complexity than transformers, it adds architectural complexity. Both articles [54] and [65] recommend looking into various hybrid architectures, such as Mamba-YOLO, to balance performance and computational overhead.

TrackingMamba, which combines the efficiency of Mamba with object tracking methods, aligning with the work done in papers [54] and [65], is described in [71] as a visual state-space model. It is appropriate for real-time tracking in unconstrained environments and has been validated against benchmarks such as OTB100 and VOT2018. TrackingMamba exceeds other traditional trackers in terms of accuracy, although it struggles in cluttered surroundings. This work takes advantage of Mamba's potential for visual tasks, however, it focuses on dynamic object tracking rather than static segmentation. In the future, reinforcement learning could improve adaptability, and the model could be extended to generalize to multi-object tracking tasks.

3.4.5 Mamba for Video Processing: Spatio-Temporal Data Modeling

Similar to paper [60], paper [73] introduces VideoMamba, which is used for video recognition and builds on the Mamba architecture to process spatio-temporal data. It uses spatiotemporal SSM to accomplish video tasks with linear complexity, allowing the model to capture long-range temporal dependencies that transformer-based models cannot. The model employs a Spatio-Temporal Forward and Backward SSM, which captures spatio-temporal interdependence through bidirectional scanning and three backward scanning methods: spatial, temporal, and spatio-temporal reversal. A 3D convolutional video tokenizer turns video clips into token embeddings by dividing them into tubelets and utilizing pre-trained image models. Furthermore, positional embeddings improve the model's capacity to capture spatiotemporal relationships. The paper [73] trains the model using the following datasets,

all of which contain vast amounts of validation and training videos: Kinetics-400 (K400), Something-Something V2(SSV2), and HMDB51. VideoMamba outperforms or is comparable to models such as VideoSwin and VideoMAE on numerous video datasets, notably in computing efficiency. On the K400, it obtains a top-1 accuracy of 75.7% in 32 frames. When pre-trained on K400, it achieves 68.6% for 16 frames and 75.7% for 32 frames on HMDB51, outperforming conventional transformer models. On the SSV2 dataset, the model also shows significant performance increases, showing its usefulness in action recognition tasks. However, VideoMamba complicates training, particularly in developing adequate scanning algorithms. VideoMambaPro (VMP) builds upon the shortcomings of its predecessor in paper[73] and is presented in paper [61]. VideoMambaPro tackles element contradiction and historical deterioration to lessen computing complexity, enabling it to manage long-range relationships. VideoMambaPro achieves a top-1 accuracy of 91.9% on K400, with significantly fewer parameters, and an accuracy of 90.3% on SSV2, outperforming VideoMAE, InternVideo, and its predecessor by a notable margin. Future research should explore VideoMambaPro's application in other video tasks, such as spatiotemporal localization and video segmentation, focusing on optimizing computational efficiency. However, further fine-tuning of its masked backward mechanism and residual connections is needed to ensure robust performance across various video datasets.

The Mamba architecture in [75], Mamba-PCGC, is designed for geometry compression of point clouds and is geared towards 3D rendering and virtual reality. This model generally uses selective state-space modeling so that the computational cost is reduced while retaining the necessary geometric information. It achieves competitive compression rates on benchmark point cloud datasets while maintaining decent detail. However, it may struggle in complex scenarios with a lot of information. The study builds on the approaches described in paper [66], in which Mamba was used to compress spatial data on long text sequences. Future studies could combine Mamba-PCGC and 3D scene segmentation models to enable end-to-end inference in real-world settings.

Table 3.1: Summary of Papers

Ref	**Study**	**ML Model**	**Dataset/Benchmark**	**Result**	**Research Gap**
[66]	Compares accuracy and efficiency of Mamba with Transformers	Mamba, Transformers	LRA Benchmark	Mamba: 91% accuracy on CIFAR-10; Transformer outperformed in visual tasks	Optimizing SSMs for long-sequence vision tasks
[47]	Proposes an SSM-based architecture for remote sensing change detection	Change-Mamba	xBD dataset, CD datasets	Outperformed CNN and Transformer, achieving SOTA performance	Applying the model in remote sensing and real-world applications
[60]	Proposes an SSM-based architecture for efficient multi-dimensional data modeling	Mamba-ND	HMDB-51, UCF-101, ERA5, BTCV, ImageNet-1K	+3.8% accuracy on ImageNet-1K, +1.3% on HMDB-51, +0.7% on ERA5	Improving climate modeling accuracy with lower costs using SSMs on satellite, radar, and spatio-temporal data

Ref	Study	ML Model	Dataset/Benchmark	Result	Research Gap
[54]	Proposes a Mamba variant for multi-class unsupervised anomaly detection	MambaAD	MVTec-AD, VisA, Reak-IAD	MVTec-AD: 98.6% AU-ROC (image-level), 97.7% AU-ROC (pixel-level)	Optimizing the model for real-time surveillance and medical image processing
[65]	Proposes a hybrid Mamba architecture (combining CNN and SMM) for object detection	Mamba-YOLO	MS COCO, PASCAL VOC	Achieves 8.1% higher mAP than YOLOv8 on the COCO dataset	Developing more hybrid architectures with less architectural complexity
[73]	Proposes another Mamba model for video recognition to process spatio-temporal data	VideoMamba	K400, SSV2, HDMB51	Top-1 accuracy of 75.7% in 32 frames (K400 and HMDB51), outperforming transformer-based models	Applying SSM-based models in real-world video tasks
[13]	Introduces DSF as a theoretical tool to compare Attention, SSM, and RNN architectures	Attention, SMM, RNN	N/A	Attention gives high accuracy but has quadratic time complexity; SSMs have linear complexity	Applying SSM-based models in tasks commonly done by transformer-based models
[70]	Overview of using SSM as a foundation for NN models to address shortcomings of Transformers	Mamba	LRA Benchmark	Achieves better computational efficiency and mathematical transparency than transformers	Use of SSM in NLP and CV, developing hybrid architectures
[51]	Studies SSMs' potential to replace Transformers, reducing computational complexity	SSM, Mamba, Transformer	Multiple tasks: NLP, computer vision, etc.	Shows efficiency over Transformers in various domains	Improving SSM models and applications across different domains
[72]	Combines SSMs with deep learning for time series forecasting	SSM, RNN	Public datasets (electricity consumption, traffic, tourism)	Outperforms ARIMA, ETS, and DeepAR with SOTA performance	Extension to non-Gaussian likelihoods and other state-space models
[46]	Proposes the Mamba architecture combining SSMs and deep learning for sequence modeling	SSM, Mamba	Language, audio, genomics datasets	Achieves SOTA performance, outperforming Transformers	Improving nonlinear dynamics and integrating Mamba's efficiency with attention mechanisms
[56]	Review of lightweight CNN designs optimized for edge computing in image processing	MobileNet, EfficientNet, ConvNeXt	ImageNet, MSCOCO	MobileNetv2/MobileViT: 0.9ms-7.28ms (iPhone); ConvNeXt superior accuracy vs. CNNs/Transformers	Development of lightweight, efficient edge AI systems
[28]	River flooding detection using deep learning and computer vision	DNN, DeepLabv3	Rivers in São Carlos, SP, Brazil (167,897 images)	High accuracy, Mean Absolute Error (MAE) of 3.32 cm, robust to lighting changes	Addressing night vision limitations, alert speed, and camera viewpoint variations
[40]	Comparative study of models for flood image classification using the FloodNet dataset	ResNet-18, VGG-16, MobileNet V2, EfficientNet, ViT, ConvNeXt, RegNet	FloodNet dataset	ResNet-18: accuracy 98.6%, F1 score 94.9%	Improving speed and accuracy of damage assessment, need for domain-specific methods

Ref	Study	ML Model	Dataset/Benchmark	Result	Research Gap
[39]	Survey on wildfire detection using transfer learning and neural networks	ResNet, MobileNet, BiT, EfficientNet, Swin Transformer, ConvNeXt	Custom datasets: HPWREN, FIRESENSE (35,000 images)	Swin Transformer: highest AUC; ConvNeXt: lowest false alarm rate	Incorporating thermal and multispectral imagery, developing hybrid models
[2]	SPADE model addresses long sequence modeling challenges by combining SSMs and Transformers	SPADE, S4, Transformer	Long Range Arena, Language datasets	Outperformed S4 and Transformer on long sequence tasks	Explore local attention, new architectures, broader dataset testing
[8]	Proposes token merging for time series processing to improve computational efficiency	Chronos, Autoformer, HyenaDNA, Transformers	ETTh1, Traffic	Up to 5400% speed-up, 3% accuracy loss on ETTh1; 2.27x speed-up with no accuracy loss in traffic	Explore broader architectures, real-world sequence tasks, decoder integration
[80]	Proposes a nonlinear switching SSM combining HMMs and NSSMs	NSSM, HMM, MLP	Speech datasets	Improved phoneme segmentation vs. HMMs and other models	Generalize to other time series, explore complex hierarchical models, handle noisy data
[30]	Reviews SMC and MCMC particle methods for parameter estimation in nonlinear non-Gaussian SSMs	SMC, Particle Filtering, MCMC, Kalman Filter	Non-Gaussian SSMs	Enhanced state sequence and parameter estimation	Explore dynamic/hierarchical models, adaptive methods for real-time data
[81]	Proposes MFBO-SSM to enhance parameter estimation	SMC, MFBO-SSM, MDPs, RL, BKF, POBDS	VIX stock prices, gene expression	MFBO-SSM improved speed and accuracy in VIX, high accuracy in gene prediction	Extend to larger datasets, explore new RL methods, apply POBDS in other domains
[59]	S4 models enhance efficiency in handling long-range dependencies	S4, Transformer Variants, LSSL	CIFAR-10, LRA, Speech Commands	91% accuracy on CIFAR-10, 88% on LRA, 98.3% on Speech Commands	Explore S4 for broader data modalities, address numerical instabilities
[49]	Explores SMC with gradient-based methods for control and signal processing on embedded devices	SMC, HMM, Gradient-Based, RML, EM, Dynamic Graphs	N/A	Focus on method development, no specific accuracy rates	Generalize SMC for broader problems, improve online parameter estimation
[37]	Proposes MambaLRP, an explainability method for Mamba models	Mamba, Mamba-130M, Mamba-1.4B, Vim-S	SST-2, Medical BIOS, Emotion, SNLI	91.97% (SST-2), 94.15% (SST-2), 90.30% (Medical BIOS), 93.65% (Emotion), 91.05% (SNLI)	Extend MambaLRP to other architectures, explore high-risk domains
[83]	Examines HDKL and KSSM for adaptive signal estimation	SVR, HDKL, KSSM, MCMC, PF	Motion tracking, wind speed estimation	Improved trajectory tracking and wind speed predictions, no specific accuracy rates provided	Explore advanced kernels, apply KSSM to more domains
[82]	Introduces probabilistic sequence models with RNN-RBM fusion, trained via Hessian-free optimization for long-term dependencies	RNN, TRBM, RBMs, HF Optimizer, Gradient Descent with Momentum	Character-level language modeling, optimal control	Improved RNN training for long-term dependencies, no specific accuracy rates provided	Broaden sequence modeling tasks, refine initialization techniques

Ref	Study	ML Model	Dataset/Benchmark	Result	Research Gap
[4]	Investigates MRNN for text generation using multiplicative connections and Hessian-Free optimization	RNN, MRNN, RBMs, HF Optimizer, Sequence Memoizer, PAQ	Wikipedia, text corpora	MRNN achieved 1.60 bpc (Wikipedia), better than Sequence Memoizer (1.66 bpc), slightly behind PAQ (1.51 bpc)	Explore MRNN in other sequence tasks, scale model for larger hidden states
[20]	Reviews RNN optimizations for embedded devices	RNN, LSTM, GRU, Delta RNN, ConvLSTM	N/A	Focus on optimizations for embedded systems, no specific accuracy rates provided	Explore more algorithmic optimizations, enhance flexibility in hardware design
[11]	Proposes a GRU-based RNN with a novel activation for hyperspectral image classification	RNN, GRU, LSTM, PRetanh, CNN	Pavia University, Houston, Indian Pines	RNN-GRU-PRetanh outperformed SVM and CNN on multiple datasets, highest OA on Indian Pines	Explore deep RNNs for hyperspectral data, investigate convolutional-RNN architectures
[63]	U-Mamba integrates CNNs with SSMs for biomedical segmentation	U-Mamba, CNNs, Transformers, UNETR, Swin-UNETR, S6	CT, MRI, 2D organ segmentation	DSC of 0.8683 (CT), 0.8501 (MRI), 0.7625 (2D organ segmentation), F1 of 0.5607 (cell segmentation)	Explore other biomedical tasks, optimize for diverse applications and datasets
[62]	Semi-Mamba-UNet integrates semi-supervised learning with contrastive learning and cross-supervision for medical segmentation	Semi-Mamba-UNet, U-Net, ViTs, Swin-UNet	MRI cardiac, MRI prostate	Dice coefficient: 0.9114 (cardiac), 0.7627 (prostate); outperformed seven other frameworks	Explore volumetric segmentation, enhance effectiveness in challenging scenarios
[77]	UVM-Net combines CNNs and SSMs for efficient image dehazing	UVM-Net, SSMs, CNNs, DehazeFormer	RESIDE	PSNR of 40.17, SSIM of 0.996; outperformed DehazeFormer	Explore other image restoration tasks, optimize for low-resource devices
[67]	Reviews Mamba as a Transformer alternative for foundation models, achieving near-linear scalability	Mamba, Mamba-2, Transformers, RNNs, CNNs, GNNs	N/A	Mamba shows efficiency in long-range modeling tasks, outperforming traditional models	Explore integration with other models, improve memory management and generalization
[57]	MambaMIR and its GAN variant enhance medical image reconstruction, improving uncertainty estimation and perceptual quality	MambaMIR, MambaMIR-GAN, SSMs, CNNs, ViTs	Fast MRI, Sparse-View CT	PSNR of 27.13 (MRI), 37.02 (CT); GAN variant improved perceptual quality	Expand to other modalities (e.g., ultrasound), optimize for real-time clinical use
[61]	VideoMambaPro enhances Mamba for video understanding	VideoMambaPro, Video-Mamba, Transformers, SSMs	Kinetics-400, Something-Something V2	91.9% accuracy on Kinetics-400, 90.3% on Something-Something V2; fewer parameters than transformers	Apply to other video tasks (e.g., segmentation), reduce computational overhead
[58]	IRSRMamba integrates a Mamba backbone with wavelet transform for infrared image super-resolution, boosting PSNR and SSIM	IRSR-Mamba, Wavelet Transform Feature Modulation, RSSG, VSSM	M3FD, CVC10	PSNR of 39.33, SSIM of 0.9439 (M3FD); PSNR of 44.50, SSIM of 0.9719 (CVC10)	Apply to other infrared tasks, optimize for real-time hardware-constrained environments

Ref	Study	ML Model	Dataset/Benchmark	Result	Research Gap
[3]	Proposes a state-space neural network (SSNN) to model nonlinear dynamic systems	SSNN	Two simulation examples of multivariate nonlinear dynamic systems	Successfully modeled multivariate nonlinear systems using SSNN, demonstrating strong identification performance	Need for exploring real-world system modeling, further refinement of model's robustness
[48]	Develops an S4 model compression method using diagonal state-space layers and balanced truncation	S4 model with diagonal state-space layers	Performance benchmarks on synthetic and real-world datasets	Compression reduced the complexity of S4 models while maintaining accuracy in sequence prediction tasks	Potential improvements in memory and computation efficiency in larger-scale tasks
[64]	Introduced DB-RNN for nowcasting and deblurring precipitation data	DB-RNN	Precipitation datasets with real-time observations	Achieved better deblurring performance compared to baseline models for precipitation forecasting	Further enhancements needed for robustness in diverse weather conditions
[36]	Proposed a landslide risk prediction model using attention-based TCN connected to RNN	Attention-based TCN + RNN	Landslide datasets	Improved prediction accuracy of landslide risks in real-time settings using the combined model	Handling imbalanced datasets and integrating more spatial information could improve results
[15]	Develops an RCNN for spectral-spatial-temporal feature learning in multispectral imagery	RCNN	Multispectral imagery datasets	RCNN outperformed baseline models in change detection tasks	Potential improvements in processing efficiency for large-scale imagery
[75]	Introduced Mamba-PCGC, a compression method for point cloud geometry using Mamba-based techniques	Mamba-PCGC	Point cloud datasets	Achieved higher compression efficiency without significant loss of reconstruction quality	Need for further exploration into real-time compression scenarios for streaming data
[71]	Proposed TrackingMamba, a visual state-space model for object tracking tasks in dynamic environments	Tracking-Mamba	Object tracking benchmarks	Demonstrated improved tracking accuracy in real-time applications compared to traditional tracking models	Improvements needed for occlusion handling and adaptability in more complex scenes
[25]	Explored transformers for image recognition, comparing image patching methods to traditional CNNs	ViT	ImageNet dataset	ViTs outperformed CNNs on large-scale image recognition tasks	High computational cost compared to CNNs, especially for smaller datasets
[69]	Critical review of computer vision methods for wildfire detection using various deep learning techniques	CNNs, Transformer variants	Wildfire image datasets	Identified key challenges in real-time wildfire detection and segmentation using vision models	Need for better integration of multimodal data for improved accuracy and early detection
[29]	Surveys deep learning and Transformer methods for UAV-based wildfire detection and segmentation	CNNs, Transformers, Hybrid Models	UAV wildfire datasets	Achieved SOTA wildfire detection and segmentation accuracy using hybrid transformer models	Need for faster real-time processing and integration of additional environmental data

Ref	Study	ML Model	Dataset/Benchmark	Result	Research Gap
[24]	In-depth review of RNN language models and their advancements over the years	RNNs, LSTM, GRU	N/A	New techniques like NCE and Pointer Networks for increasing efficiency	Hybrid models needed for further enhancement with new datasets
[38]	Extensive review of transformers and evolution of various architectures	ViT, DeiT, PVT, DETR, MaxDeepLab	N/A	Shift to transformers from CNNs in image classification, object detection, and video processing	Lack of data/resources for training, lack of inductive bias
[34]	Extensive review of transformers and evolution of various architectures	Longformer, Reformer, Informer, Performer, Big Bird, Sparse Transformer	N/A	Reduced computational and memory demands, efficient handling of long sequences	Improve memory and computation efficiency with newer standards
[44]	Application of ViT	ViT-ReT	Unspecified HAR dataset	Double speed in runtime than CNN, maintaining comparable accuracy	Lower latency environments and improve model generalization
[41]	Application of ViT	FV-ViT	FV-USM dataset, SDUMLA-HMT dataset	SOTA results with EER of 0.042% on FV-USM dataset, 1.033% on SDUMLA-HMT dataset	Multimodal biometric systems and exploring its application in low-resource environments
[74]	Mamba model formation and usages in various ways	Mamba and SSMs	Not specified	Promising results in image classification, object detection	Improve scalability and explore multimodal approaches
[68]	Mamba in contrast to CNNs and ViTs	Mamba variant models	Not specified	High promise in computer vision, almost a foundational resource	Focus on adaptive scanning methods and applying Mamba to new fields
[76]	A novel CNN-Mamba framework that uses mutual guidance to improve resolution	CNN-Mamba, MSI, HSI, SSTG	Hyperspectral Image datasets	Superior performance compared to SOTA models in spatial and spectral quality	Further optimization needed to handle larger datasets
[55]	3D-Spectral-Spatial Mamba framework designed for hyperspectral image classification	3DSS-Mamba	Hyperspectral Image datasets	Outperformed SOTA HSI classification, achieving higher accuracy	Optimizing model for larger datasets

3.5 Research Gap

As can be seen from 3.1, transformer-based models have demonstrated exceptional performance in various machine-learning tasks, including computer vision and NLP. Despite their advantages, a number of study papers have consistently pointed out that transformers' quadratic temporal complexity presents serious difficulties for real-time detection systems and long-sequence modeling jobs. These well-established algorithms might not be appropriate for environmental monitoring jobs like diagnosing floods or wildfires since they cannot process vast amounts of satellite data in real-time. Therefore, the drawbacks of transformers have led to a research void in creating a more efficient basic design that can manage remote dependencies without

sacrificing efficacy.

Many papers have suggested SSMs, particularly the Mamba architecture, as a promising alternative to the traditional transformer-based models. These models provide linear time complexity, reducing the computational cost and improving performance in tasks requiring long sequences and high temporal resolution. However, some studies have revealed that many of these fully SSM-based models have not been optimized regarding accuracy, architectural complexity, non-linear dynamics, etc. Some hybrid architectures that combine the strengths of both transformers and SSMs have mitigated some of the issues. More research is needed to develop optimized versions of these hybrid models, as integrating SSMs with attention mechanisms can help improve accuracy in environmental applications.

Some of the proposed Mamba-based architectures, such as ChangeMamba, MambaAD, VideoMambaPro, etc., have outperformed traditional CNNs and transformers with their SOTA performance in a range of tasks. However, the use of these models in actual environmental planning has not yet been investigated. Our study attempts to close this gap using an SSM-based model, such as Vision Mamba, for classification tasks in multiple domains, namely environmental disaster, medical, and agriculture. Its performance will also be thoroughly examined to compare it to transformer-based and CNN models.

Chapter 4

Datasets

The datasets explored span diverse domains of applications, addressing multi-critical issues such as detecting wildfires, monitoring the health status of agricultural plants, and diagnosing medical images. Hence, by choosing the datasets from these distinct but correlated domains, the research intends to assess the versatility and stability of developed vision transformer models across different tasks and difficulty levels.

In fire and no-fire classification for wildfire detection, the FLAME dataset offers an opportunity to analyze drone-captured aerial images to help develop and support timely disaster response. PlantVillage dataset aims at plant disease classification which helps support sustainable agricultural practices by diagnosing infections on the crops. Finally, the Skin Cancer dataset focuses on the early and timely identification of malignant or benign skin growth, to which, deep learning has proven beneficial for healthcare diagnosis.

Each dataset used in this research includes its special features and difficulties, the environmental changes inherent in wildfires, complex patterns of leaves in the context of the agricultural industry, and the fine boundaries of the lesion typical for medical imagery.

4.1 The Flame Dataset

Timely detection of wildfires can significantly help firefighting efforts and boost prompt actions. Unmanned Aerial Vehicles (UAVs) and drones, both excellent technologies for reliable aerial images or videos, can be very beneficial in collecting images of wildfires from various perspectives. This study selected an aerial imagery FLAME (Fire Luminosity Airborne-Based Machine Learning Evaluation) dataset from IEEEDataPort, collected using drones during prescribed pile burn in Northern Arizona, USA. The dataset, which includes raw video, resized image frames, and raw heat map footage recorded by an infrared thermal camera, is focused on research involving a binary image dataset for Fire vs. No Fire classification and Fire Segmentation.

Zenmuse X4S and DJI Phantom 3 were used for data collection and original res-

olution and frame rate were preserved while resizing the frames. The frames captured variations in intensity, background conditions, and environmental elements like smoke and lighting.

The training dataset consists of 39,375 frames (1.3 GB) in JPEG format and some samples are shown in Fig 4.1. The test dataset consists of 8,617 frames (5.2 GB), also in JPEG format. Both the training and testing dataset images have been labeled in fire and no-fire folders. The images were already resized to a uniform resolution of 254×254, which simplifies the downstream processing and matches the input size for most of our chosen models. The compact file sizes also ensure computational efficiency during training and testing.

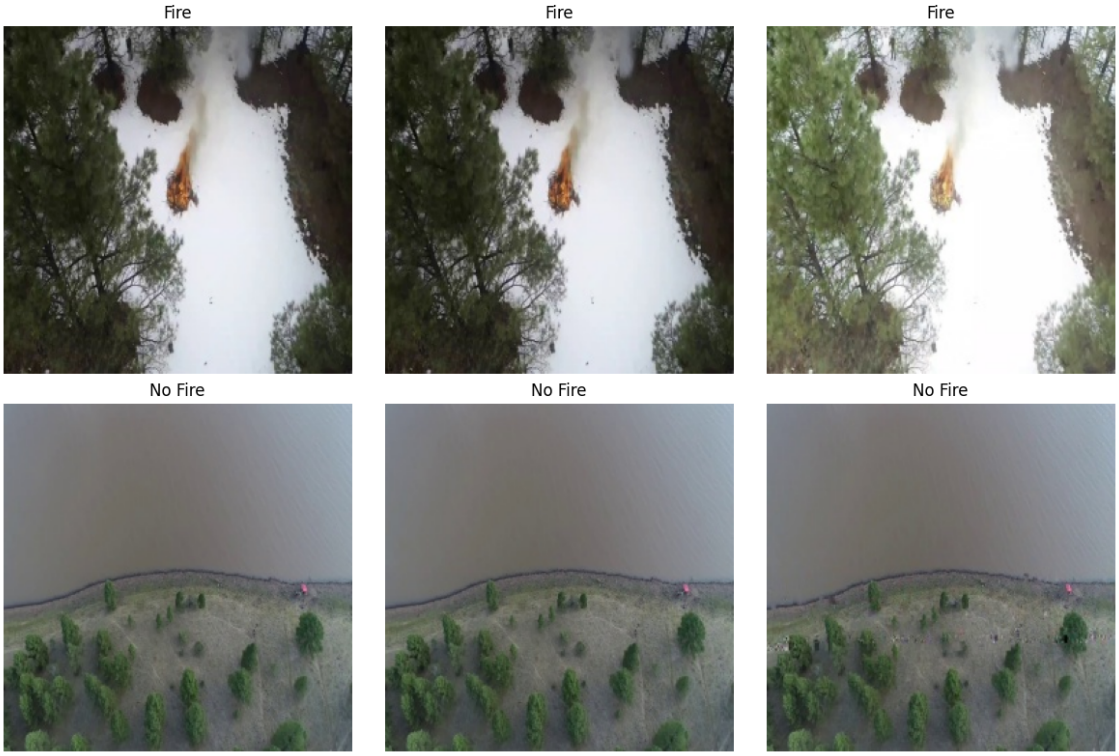


Figure 4.1: Sample images with labels from the FLAME Dataset

The dataset contains a significant number of images for training and testing, providing us with a strong foundation for our chosen deep-learning models. The size of the dataset ensures statistical reliability in the models' performance evaluation. Since the images are collected from video footage, they capture real-life wildfire scenarios, making them relevant for deployment in practical wildfire detection systems.

However, the dataset also comes with a few limitations, creating challenges in the training process. There is a notable imbalance in the training set with $25,018$ fire images (63.52%) and $14,357$ NoFire images (36.46%) as shown in Fig 4.2. This can lead to biased predictions favoring the majority class (Fire) and therefore had to be properly handled to avoid sub-optimal model performance.

Figure 4.2: Class distribution of Flame's training dataset before pre-processing

The dataset did not explicitly provide any validation set, requiring us to create our own stratified split in order to analyze the training process more fruitfully. This introduced additional complexity by risking unintentional bias during the validation process. Furthermore, since the dataset is derived from video footage, consecutive frames have very high inter-frame similarity, making splitting risky by significantly increasing the chance of data leakage between training and validation subsets. Although the data set captures real-world scenarios, its diversity in different environments such as weather conditions, vegetation types, and lighting variations is not well documented. The videos were used to extract a large number of frames, inflating the dataset size but not proportionally increasing the diversity of unique scenarios. This had the potential to limit its generalization to other regions and scenarios by overfitting the dataset.

Figure 4.3: Mean image of each class from Flame Dataset

The mean images in Fig 4.3 for both classes (Fire and NoFire) reveal subtle differences, with the Fire class having noticeable lighter regions which may be caused by

the brightness of the flames, and the NoFire class mean image appearing darker, representing the absence of fire. The smoothness of the mean images suggests some level of homogeneity within each class, which could be beneficial in reducing noise in data. However, the similarity in the mean images of the two images poses challenges to the models to perform well under conditions with smoke or low light.

The RGB color distribution in 4.4 shows peaks in the lower-intensity ranges for all the channels, suggesting that most images have darker tones. This perfectly aligns with the natural environment that may be captured in wildfire scenarios with backgrounds often having forests and smoke. The red channel shows slightly higher intensities compared to blue and green, corresponding to the fire's brightness and warmth tones. The overlap in RGB distributions across the classes indicates that the dataset relies on subtle differences in color intensity rather than stark contrast. This highlights the need for advanced deep learning models for advanced feature extraction and capturing fined-grained differences, as also observed from the mean image analysis.

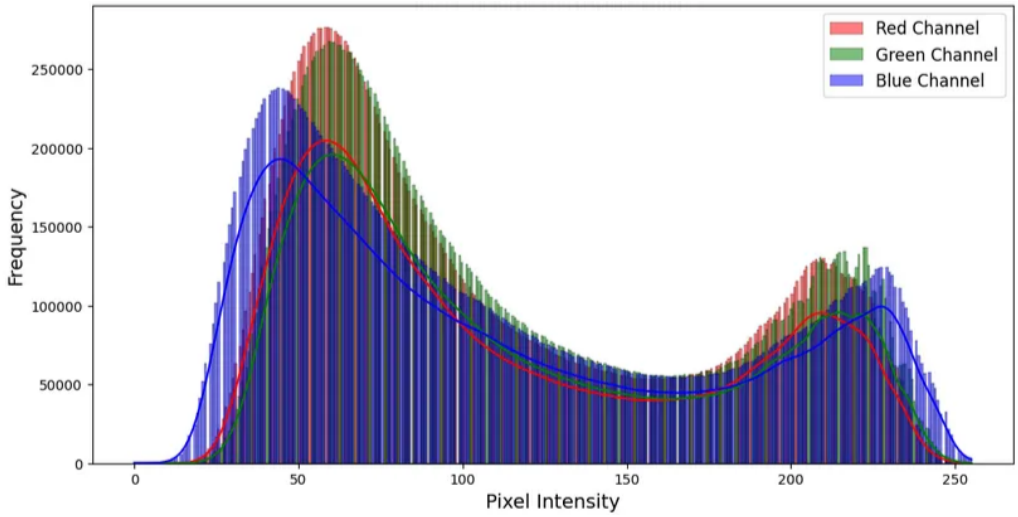


Figure 4.4: RGB Color Distribution of Flame dataset

4.2 PlantVillage Dataset

The PlantVillage dataset [7], which is sourced from the Kaggle platform, is a widely recognized standard benchmark in agricultural research and plant pathology. It contains a substantial dataset of annotated images, 20,638 images, of three different plant species (tomato, potato, and bell pepper) that are infected with a multitude of diseases and healthy samples. It includes 15 different labels in total, with each one of them representing a diseased or healthy condition. The primary purpose of the dataset is to facilitate the development of machine learning models to classify plant disease from image data, which is essential for agriculture productivity and food security.

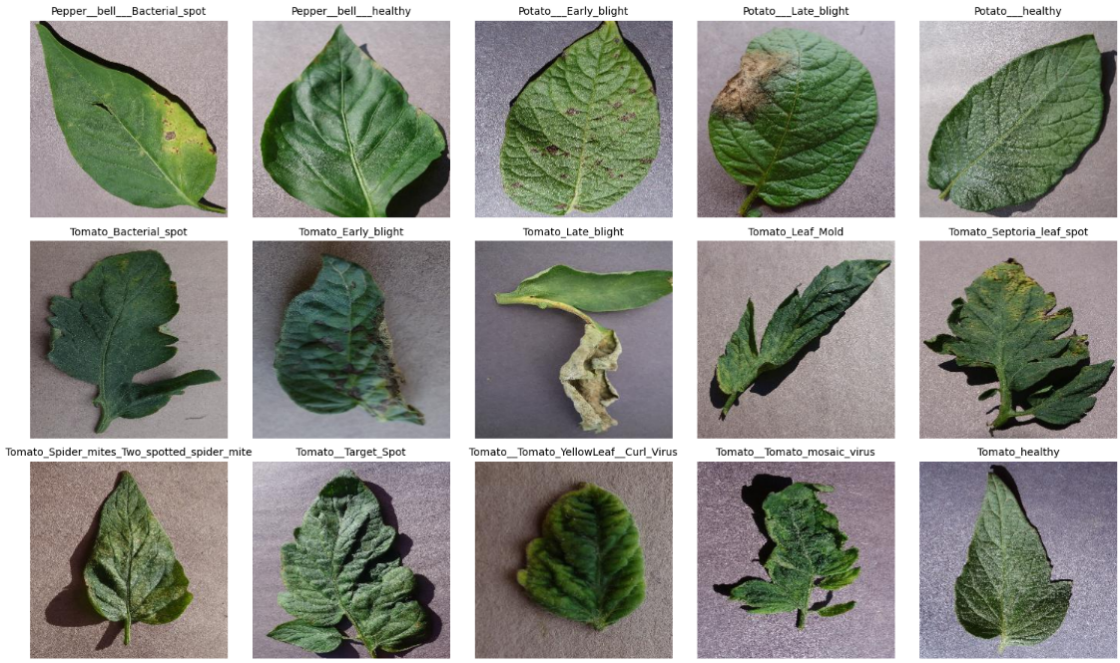


Figure 4.5: Sample Images from PlantVillage Dataset

As seen in Fig 4.5, each label is represented by a unique visual pattern. This is an important step towards understanding the differences between classes and visualizing potential obstacles in classification tasks.

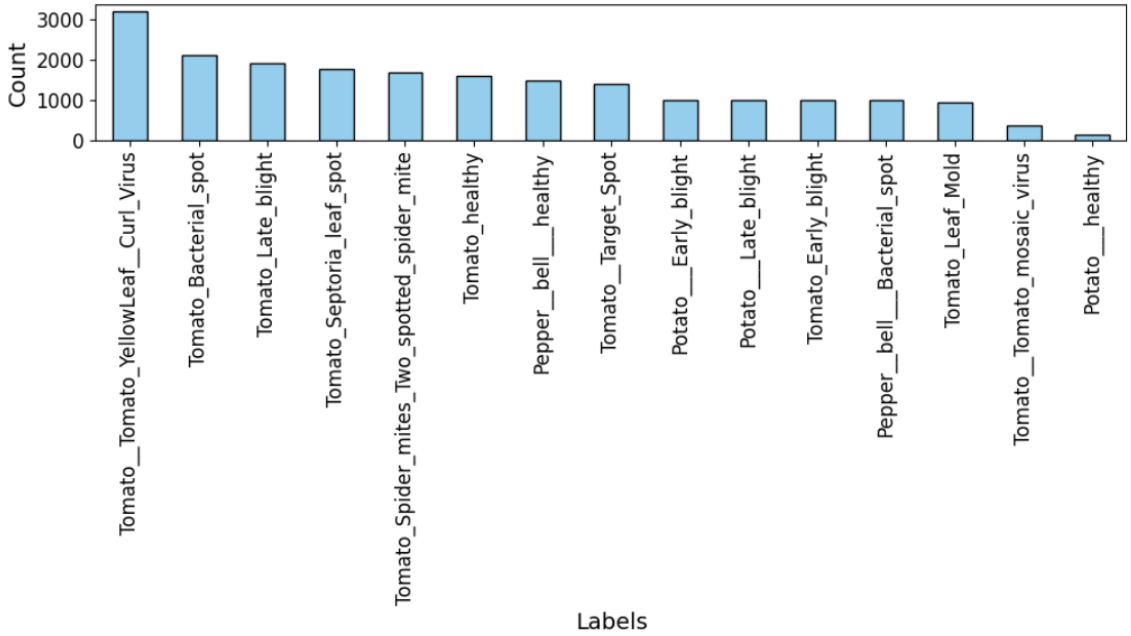


Figure 4.6: Label Distribution of PlantVillage Dataset

The 15 distinct labels of the PlantVillage dataset and their class distribution are represented in the bar chart 4.6. The dataset shows a slight imbalance, with certain labels like "Tomato_Tomato_YellowLeaf_Curl_Virus" being heavily represented, while others like "Potato_healthy" have fewer samples. The imbalance in

this dataset is not significant enough to lead to biased models. Oversampling was not done on this dataset to maintain the integrity of the dataset and avoid potential overfitting to the oversampled labels. Other methods were employed to address the imbalance which will be discussed thoroughly in the methodology chapter.

The hexbin plot of image size distribution 4.7, reveals that the images in the dataset have uniform sizes of 256×256 pixels, which is a compatible input value with many of our chosen deep-learning models.

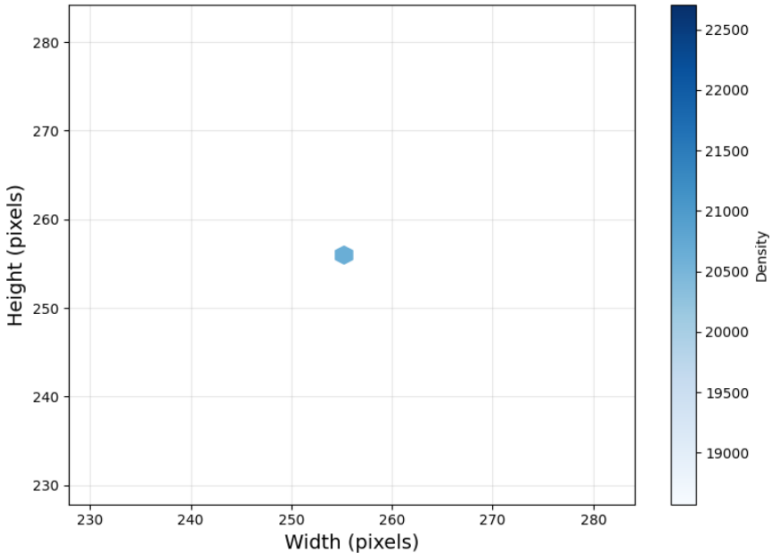


Figure 4.7: Uniform Image size of PlantVillage Dataset

To facilitate model training and evaluation, all image labels were mapped to unique integer IDs as shown in table 4.1.

ID	**Label**
0	Pepper___bell___Bacterial_spot
1	Pepper___bell___healthy
2	Potato___Early_blight
3	Potato___Late_blight
4	Potato___healthy
5	Tomato_Bacterial_spot
6	Tomato_Early_blight
7	Tomato_Late_blight
8	Tomato_Leaf_Mold
9	Tomato_Septoria_leaf_spot
10	Tomato_Spider_mites_Two_spotted_spider_mite
11	Tomato___Target_Spot
12	Tomato___Tomato_YellowLeaf___Curl_Virus
13	Tomato___Tomato_mosaic_virus
14	Tomato_healthy

Table 4.1: ID and Label Mapping for PlantVillage

4.3 Skin Cancer: Malignant vs. Benign Dataset

Skin cancer is one of the most common cancers globally and early detection plays a critical role in reducing mortality rates. Medical imaging has become an essential tool for detecting skin abnormalities. Machine learning techniques, especially deep learning can significantly aid the process of early detection. The chosen dataset, "Skin Cancer: Malignant vs. Benign" from Kaggle, contains high-resolution images of benign and malignant skin moles as shown in the samples of 4.8. It provides an ideal foundation for exploring the potential of the deep-learning models applied to the previous two datasets in skin cancer diagnosis.

Our research focuses on the binary classification of the image dataset (171.61 MB). The training data comprises 2637 images while the test data contains 660 images. The training and test datasets have been labeled in malignant and benign folders and resized to 224×224 pixels.

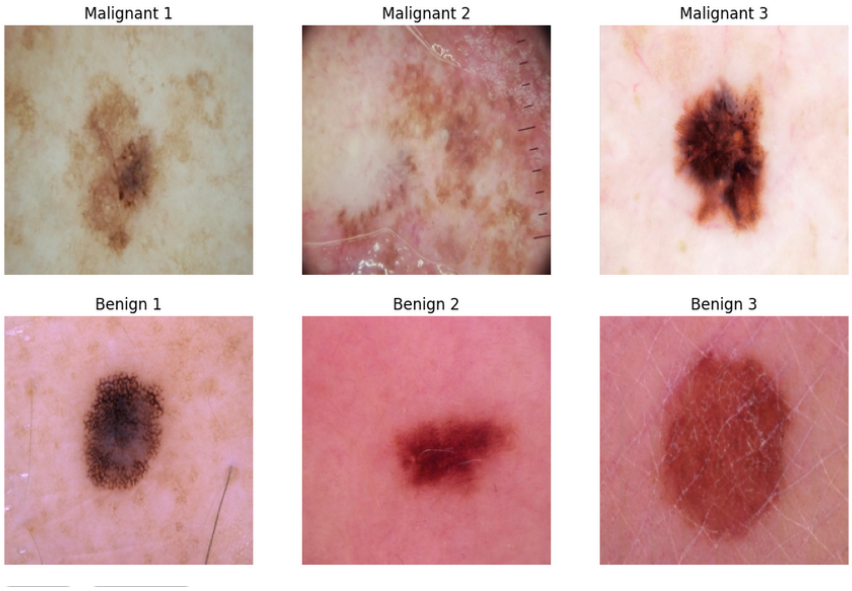


Figure 4.8: Sample images with labels from the Skin Cancer: Malignant vs. Benign dataset

One significant drawback of the medical dataset was its comparatively limited size. This is a problem because large datasets are usually necessary for deep learning models to perform well and generalize. Even though this dataset is smaller than others, it is impossible to tell if it is adequate for the classification objective without first training and assessing the models.

The dataset did not come with a pre-defined validation set, and the training set was deliberately not split into a training and validation set as the dataset did not provide ample images. Both the training and testing datasets were balanced as shown in 4.9, giving an equal representation of both classes. The training dataset had 1440 benign images and 1197 malignant, therefore no oversampling and under-sampling were needed during pre-processing as the chance of overfitting was low during training.

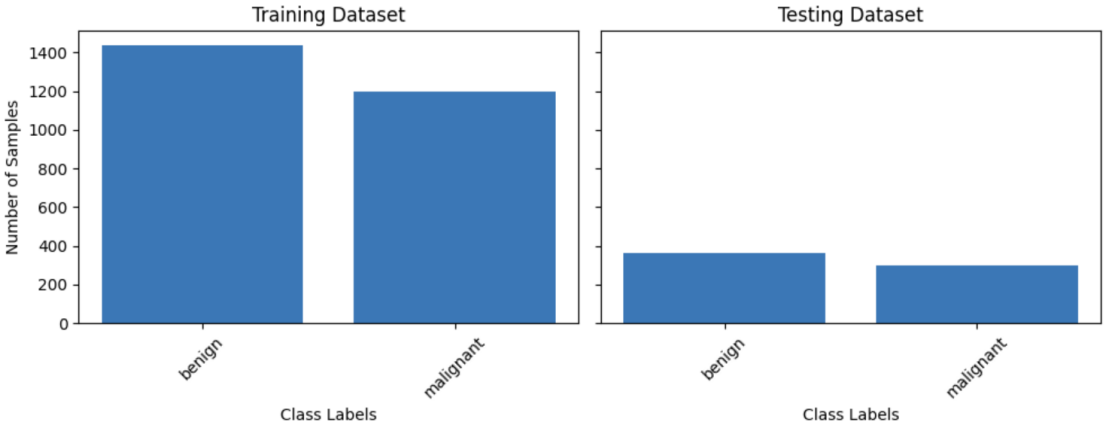


Figure 4.9: Class distribution of training and testing datasets

4.4 Dataset comparison

The datasets used in this study include a wide range of visual imagery that is specifically designed to assess model performance in several tasks and domains (binary classification and multi-class classification). As was covered in detail above, each dataset has its own set of benefits and drawbacks.

Specifically designed for wildfire identification applications, the FLAME collection offers high-resolution aerial imagery that captures the nuances of fire breakouts under a variety of environmental situations. Its use ensures that the research is grounded in real-world scenarios where prompt and accurate fire detection is essential to minimizing damage. The dataset's specificity and relevance make it an ideal choice for evaluating how effectively deep learning models generalize to complex geospatial situations.

Although the Plant Disease dataset is designed for a different use case, it provides a valuable benchmark for testing models on high-resolution photos with intricate classifications. Likewise, the Skin Cancer dataset introduces an additional layer of complexity by requiring the computers to detect subtle patterns in medical images, which is analogous to detecting indicators of small-scale wildfires.

Together, these datasets show a range of challenges in binary classification, including differences in topic matter, image quality, and the visual complexity of feature distinction. Furthermore, the notable variations in dataset sizes further highlight the models' ability to function with varying amounts of data. Utilizing this range of datasets, the study thoroughly assesses the generalizability and adaptability of CNNs, EfficientNet, Transformer-based architectures, and novel State Space Models such as Mamba.

The unique characteristics of each dataset and their impact on model performance are highlighted via comparative analysis. As an example:

- The FLAME dataset focuses on long-range interdependence and geographic

patterns to evaluate models' capacity to capture contextual linkages in large-scale aerial photography.

- The datasets for Skin Cancer and Plant Disease require a high level of sensitivity to small details, making it difficult for the models to identify minute changes in color and texture.

The paper examines the trade-offs between computing efficiency and predictive accuracy across many domains using these varied datasets, in addition to benchmarking model performance. This multi-dataset method highlights the stability and scalability of state space and deep learning models, offering insights into their real-world uses in healthcare, agriculture, and disaster response.

Chapter 5

Model Architectures

5.1 DeiT-Base Distilled Patch16-224

The optimized Data-efficient Image Transformer (DeiT) provides greater data efficiency along with superior learning performance at a comparable level of computational efficiency when compared to the Vision Transformer (ViT). DeiT transforms the basic ViT architecture by adding a distillation token as proposed by Touvron et al. (2021) [23] to enable learning from both source data and a teacher network. The enhanced performance of DeiT occurs through an approach that costs fewer training datasets than a typical ViT-based model.

Architectural Design of DeiT-Base Distilled Patch16-224

Patch Embedding and Tokenization

Similar to typical transformer-based architectures, DeiT initializes by dividing the input image into separate patches without overlap. The system creates an embedding space of 768 dimensions by flattening and projecting 16×16 patches into sequential tokens. The embedding process can be formally defined as:

$$Z_p = W_e \cdot x_p + b_e \tag{5.1}$$

where Z_p represents the patch embedding, W_e is the learned projection matrix, and x_p is the raw image patch.

Transformer Encoder

Each of the model's 12 transformer blocks contains Multi-Head Self-Attention (MHSA) as well as Feedforward Network (FFN) components. The self-attention mechanism operates on input token embeddings to model long-range dependencies, as defined

in Equation (5.2)

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5.2}$$

where Q, K, and V are query, key, and value matrices, and d_k is the dimensionality of the key vector.

Distillation Token

The main innovation in DeIT involves adding distillation tokens that allow the model to acquire knowledge simultaneously from teacher models as well as ground truth labels. By incorporating a distillation token DeiT extracts distilled knowledge which improves generalization capabilities when working with smaller data sets. The classification steps add together CLS token classification alongside the Distillation token to enable supervisory refinement of predictions. The process can be visualized in detail from Figure 5.1

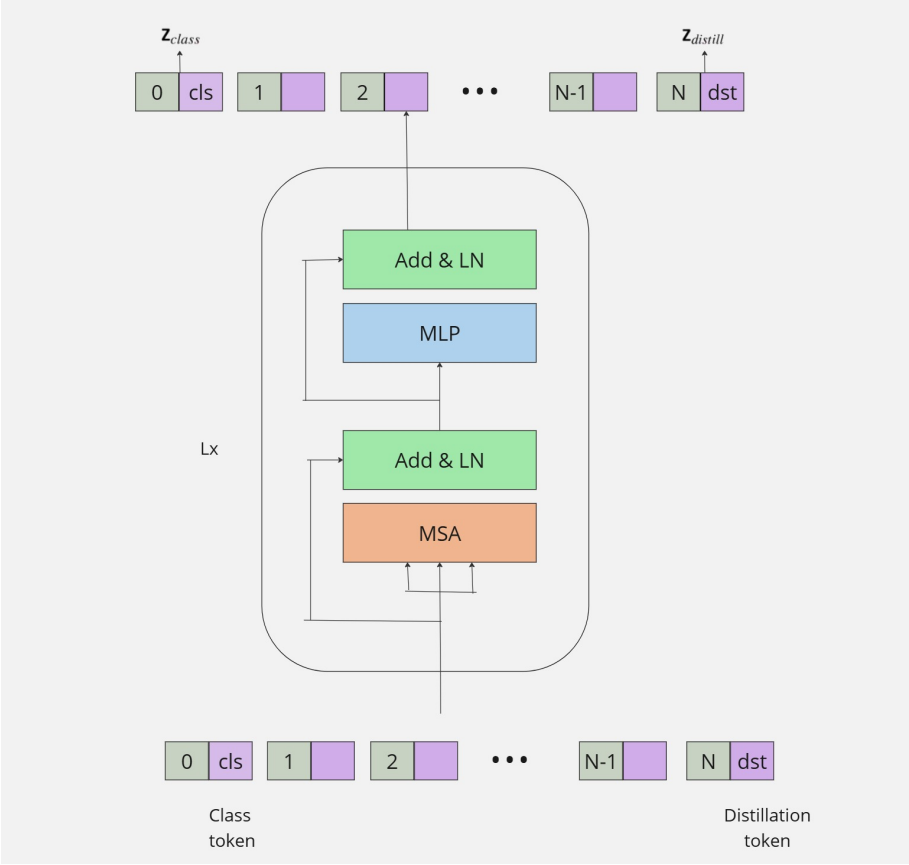


Figure 5.1: Overall architecture of DeiT

Model Variants and Their Applications

The pre-trained weights from the ImageNet database initiated the model before applying binary classification training by replacing the output layer with two new

logits. The Hugging Face Transformers library enabled the use of **DeiTForImageClassification** pre-trained model alongside **AutoImageProcessor** normalization for input pictures before starting training on the Wildfire and PlantVillage datasets. As part of the PlantVillage dataset development, they implemented a checkpoint system that enabled efficient weight initialization during domain adaptation tasks. The **timm** library delivered optimized loading capabilities for smaller datasets to process the Skin Cancer dataset. The model received GPU acceleration throughout training procedures across all target datasets to achieve efficient training execution. Within DeiT the distillation token improved feature extraction capabilities and delivered strong classification results throughout wildfire detection and both skin cancer classification and plant disease identification.

5.2 Swin Transformer

The Swin Transformer extends the versatility and power of transformers which was originally developed for natural language processing, into the vision domain data successfully. Proposed by Liu et al. in their 2021 paper "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows" [31], this model is aimed to overcome the major drawbacks of the previously proposed Vision Transformers (ViTs) which were computational inefficiency and hierarchical representations.

Architectural Design of the Swin Transformer

Patch Partitioning and Tokenization

The Swin Transformer starts with splitting the input image into non-overlapping patches of size 4×4 (for example, 224×224 into 56×56 patches). Each is then projected into a higher-dimensional embedding space [31] [32]. This tokenization step is mathematically represented as:

$$z_p = \text{Linear}(\text{Flatten}(x_{4\times4})) \tag{5.3}$$

Where $x_{4\times4}$ represents a patch, and z_p is its corresponding token embedding.

Shifted Window Multi-Head Self-Attention (SW-MSA)

The built-in Swin Transformer is different from the global self-attention, which attends across the entire image. Rather, it only attends locally using non-overlapping windows. For instance, if the window size is 7×7, there are marks made within the image in smaller windows where attention is calculated locally in each of them. It means, the computational cost comes down from $O(n^2)$ where n is the number of tokens, to $O\left(\frac{m^2 \cdot n}{w^2}\right)$ where m is the number of windows and w is the size of the windows.

Hierarchical Representation

The model passes images through several stages, gradually decreasing the resolution of feature maps while increasing the number of channels at the same time. For example:

- Stage 1: Input resolution 56×56, with 96 feature dimensions.
- Stage 2: Resolution 28×28, with 192 feature dimensions.
- Stage 3: Resolution 14×14, with 384 feature dimensions.
- Stage 4: Resolution 7×7, with 768 feature dimensions.

At each level, an integration layer (patch merging layer) merges adjacent patches with less spatial dimensions but with deeper channels.

MLP and Layer Normalization

The Swin Transformer comprises blocks that contain a Multi-Layer Perceptron (MLP) and a Layer Normalization (LN) operation. The MLP is composed of two dense layers with an added GELU activation function and it aims to change the representation of features, obtained by the self-attention mechanism. Layer normalization helps to train the model more stable due to the normalization of the input features to zero mean and unit variance. The mathematical operation in the MLP block can be expressed as:

$$\mathrm{MLP}(h) = \mathrm{GELU}\left(\mathbf{W}_2\,\mathrm{GELU}\left(\mathbf{W}_1 h + \mathbf{b}_1\right) + \mathbf{b}_2\right) \tag{5.4}$$

Where h represents the input features, and $W1$, $W2$, $b1$, $b2$ are the weights and biases of the fully connected layers.

Classification Head

The last layer of the Swin Transformer combines the features learned by the network through global average pooling and performs a classification through a fully connected layer. For the classification head, the output layer was replaced by two logits, representing the classes in the respective datasets. The layers are represented in Figure 5.2.

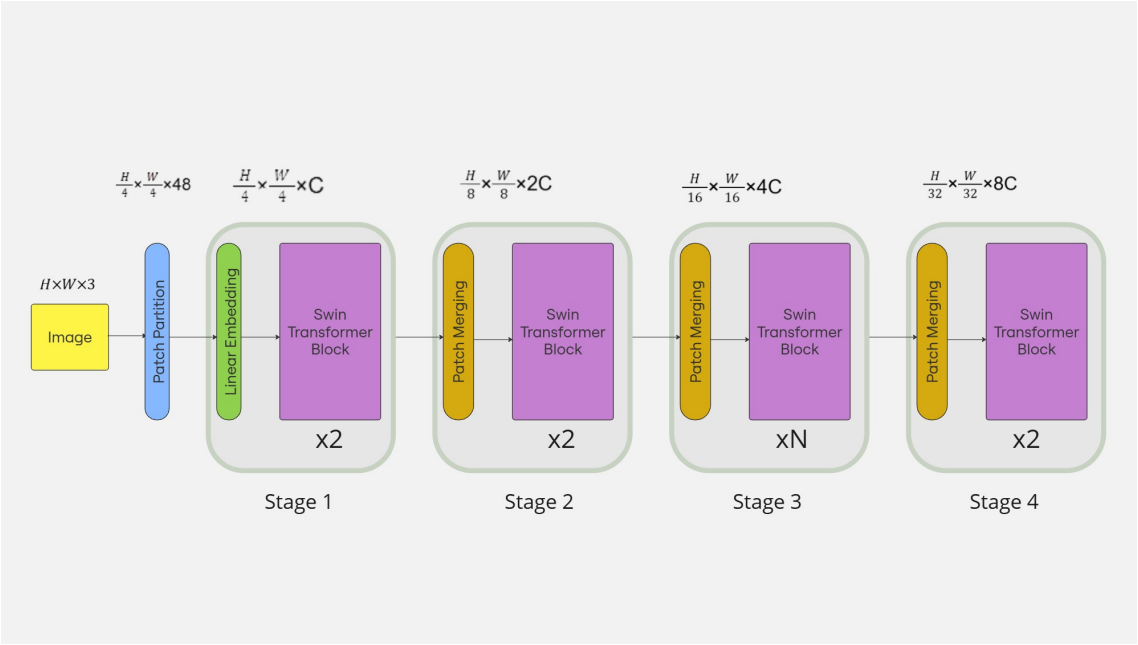


Figure 5.2: Overall architecture of Swin Transformer

Model Variants and Their Applications

Swin-Tiny Patch4 Window7-224 architecture was used in all three datasets but small changes were to suit each project's needs. The model started with pre-trained ImageNet weights to perform binary classifications after rearranging the output layer to generate two logit values (two class outputs). **Swin-Tiny Patch4-Window7-224** was implemented in both Wildfire and PlantVillage datasets through the Hugging Face Transformers library. TrainingArguments, **Trainer**, **SwinForImageClassification**, and **AutoImageProcessor** helped to speed up training while the **AutoImageProcessor** normalized image data before processing. Skin Cancer model setup used the **timm** library because it works better with small datasets. The model automatically leveraged the GPU power when available during all dataset training processes. With its structured attention architecture and sliding attention methods, Swin Transformer showed exceptional results across multiple tasks including wildfire recognition medical image analysis for cancer detection and plant disease identification.

5.3 PVTv2

The Pyramid Vision Transformer v2 (PVTv2) [35] is a Transformer-based Neural Network architecture. Introduced in 2021 by Wang et. al. [27] as PVT, it was later improved in 2022 by the same group who wished to improve their model, which was already capable of taking on dense prediction tasks, such as object detection, semantic segmentation, and image classification. This architecture combines the global modeling capabilities of transformers with the hierarchical design of Convolutional Neural Networks (CNNs). Among its other variants, PVTv2-B0 stands out

Stages	Attention Layer Count	Channel Count
Stage 1	2	32
Stage 2	2	64
Stage 3	6	160
Stage 4	2	256

Table 5.1: Number of Attention Layers and Channels at each Stage of PVTv2

as a lightweight model, making it suitable for mobile applications.

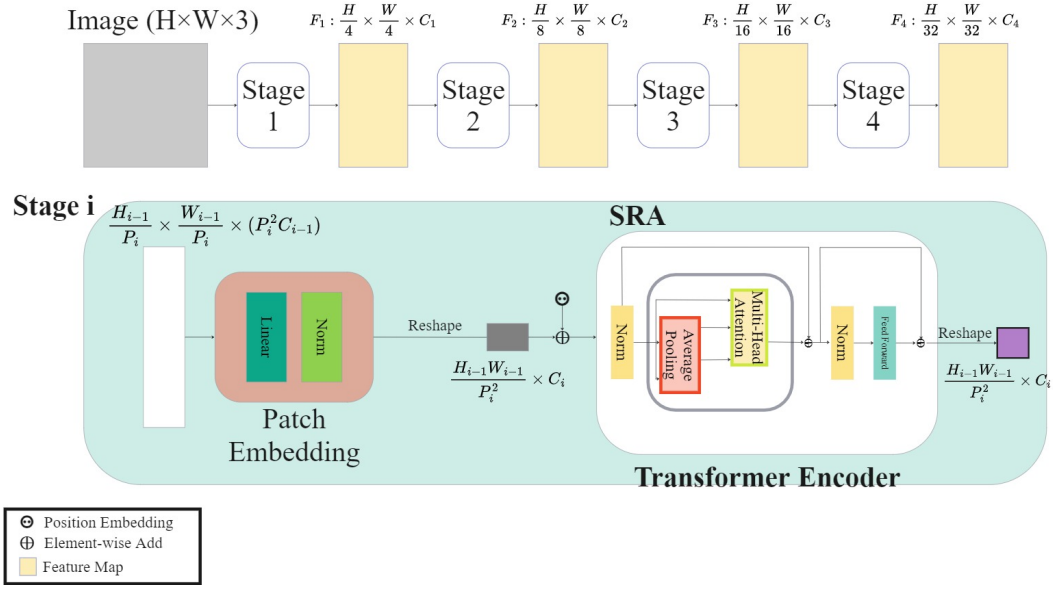


Figure 5.3: Overall architecture of Pyramid Vision Transformer v2 (PVTv2)

Overall Architecture

The PVTv2-B0 architecture has an input size of 224×224 pixels, and a parameter count of 3.7 million, which is the lowest among all PVTv2 variants. The input is passed through four hierarchical layers, progressively reducing spatial resolution and increasing feature channels.

Within each stage, a Patch Embedding mechanism forms overlapping patches of 4×4 pixels to improve feature continuity, a feature that replaced the original patch embedding from the first iteration of the PVT. Afterward, the reshaped input is passed to the Transformer Encoder, where the Spatial Reduction Attention (SRA) mechanism reduces computational complexity, followed by a Feed Forward network to enhance local feature extraction; yet another improvement introduced in PVTv2 from its original iteration.

Internal Workings

The model relies on the following procedures to prepare the input data for classification:

- **Input Feature Extraction:** The input is resized to 224×224 pixels, and overlapping 4×4 patches which are embedded into feature vectors. Positional encodings are added to retain spatial information.
- **Hierarchical Processing:** The input is passed through the aforementioned stages of the hierarchical layers, each of which increases feature channel count while reducing spatial resolution.
- **Spatial Reduction Attention (SRA):** To achieve linear complexity, this mechanism applies self-attention after reducing the spatial dimensions of feature maps.
- **Feed-Forward Networks (FFNs):** A feed-forward network is placed after each attention block that incorporates convolutional layers, allowing the model to effectively capture both global and local patterns.
- **Residual Connections:** Shortcut connections are placed where input and output dimensions are the same, which preserves essential information and improves gradient flow during training.

Model Variants and Applications

PVTv2-B0 architecture was used in the Wildfire dataset, and **PVTv2-B2** architecture was used in the Skin Cancer and PlantVillage datasets, and small adjustments were made to suit each project's needs. For the Skin Cancer and Wildfire datasets, the model was imported from the PyTorch library, along with pre-trained weights to perform binary classification.

For the PlantVillage dataset, a pre-trained model was imported from the HuggingFace Transformers library (**OpenGVLab/pvt_v2_b2**). The image data was prepared and processed, and the model was prepared to perform multiple classifications. PVTv2 provided satisfactory results across multiple tasks such as object recognition and image analysis.

5.4 ViT B-16

The Vision Transformer (ViT) was introduced in the 2020 paper *"An Image is Worth* 16×16 *Words: Transformers for Image Recognition at Scale"* by Dosovitskiy et. al. [25], which is an architecture that applies transformers directly to image recognition tasks.

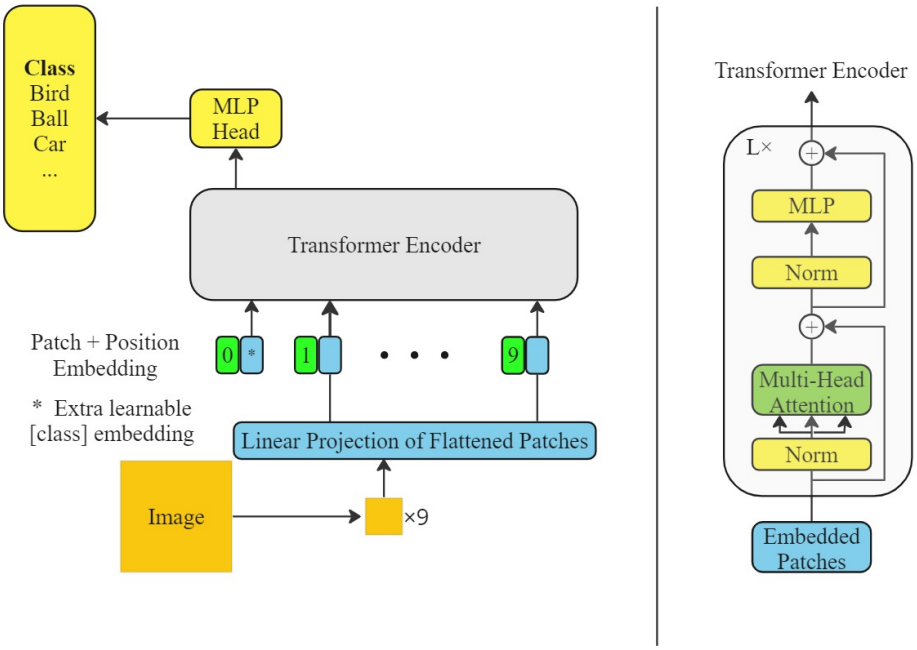


Figure 5.4: Overall architecture of Vision Transformer (ViT B-16)

Although transformers were initially introduced for natural language processing tasks, this architecture demonstrated that state-of-the-art results in image classification can be obtained by dividing images into patches and processing them as a sequence of tokens, provided that they were pre-trained on sufficiently large datasets. The ViT-B16 variant is a base version with 16×16 pixel patches and a highly scalable and adaptable model for computer vision tasks.

Overall Architecture

ViT-B16 is a transformer-based neural network designed for image classification, that employs self-attention mechanisms to capture global dependencies across the entire image, as opposed to local feature extraction in convolutional neural networks (CNNs). The input size for this architecture is 224×224 pixels, with overlapping patches of 16×16 pixels, resulting in a total of 196 patches per image. The architecture consists of 768 Embedding Dimensions, 12 Transformer Encoder Layers, 12 Attention Heads per layer, 3072 Feed-Forward Network (FFN) dimensions, and a total Parameter Count of 86 million.

Internal Workings

The ViT-B16 architecture relies on the following mechanisms to perform image classification:

- **Patch Embedding:** The 224×224 pixel image is divided into 196 patches of size 16×16 pixels, which are flattened into a 1D vector. A trainable linear projection maps each of these patches into 768-dimensional embedding space.
- **Positional Embeddings:** Learnable position embeddings are added to the patch embeddings to retain spatial information, allowing the model to identify each patch's location within the image.
- **Transformer Encoder:** The patch embedding sequence, as well as a classifier token is passed through 12 transformer encoder layers. Within these layers, a Multi-Head Self-Attention (MHSA) mechanism computes the attention of patches across the sequence, which captures relationships between the patches. A Later Normalization (LN) network stabilizes training and improves gradient flow. Afterward, a Feed-Forward Network (FFN) consisting of 2 layers (MLP, GELU activation) enhances the model's ability to learn complex patterns. Lastly, Residual Connections preserves information which stabilizes the training process.
- **Classification Head:** The classifier token (CLS) is passed through a classification head, which enhances the image classification process.

Model Variants and Application

The ViT-B16 architecture is a high-performance, scalable, and flexible model with high accuracy scores on benchmarks such as ImageNet, and supports classification, object recognition,n, and segmentation tasks. For the Wildfire and Skin Cancer datasets, a pre-trained **ViT B-16** model with weights from **IMAGENET1K** was imported from **torchvision.models** in PyTorch, and the model was trained for binary classification.

For the PlantVillage dataset, the model was imported from the HuggingFace Transformers library (**google/vit-base-patch16-224-in21k**), and the model was prepared for performing multiple classification tasks.

5.5 ResNet-18 and ResNet-50

ResNet is a novel framework that introduces the idea of Residual Networks for training deep neural networks that enable scaling to unprecedented depths by simplifying optimization. Where traditional networks suffer from degradation due to optimization challenges once they get deeper, ResNets provide a solution in the form of residual functions and mappings as shown in 5.5. This form of reformulation mitigates vanishing gradients and overall improves convergence. It allows the training of networks as deep as 152 layers.

Overall Architecture

ResNets utilize identity shortcut connections to bypass layers and combine the input with the residual output through element-wise addition. No additional computational overhead is required by these shortcuts and ensure efficient information flow through the network. With the solution of the degradation problem, ResNets achieve significant accuracy improvements alongside increased depth. This makes them ideal for large-scale datasets like ImageNet and tasks that fall under object detection and segmentation. [5] explains the model on an architectural level.

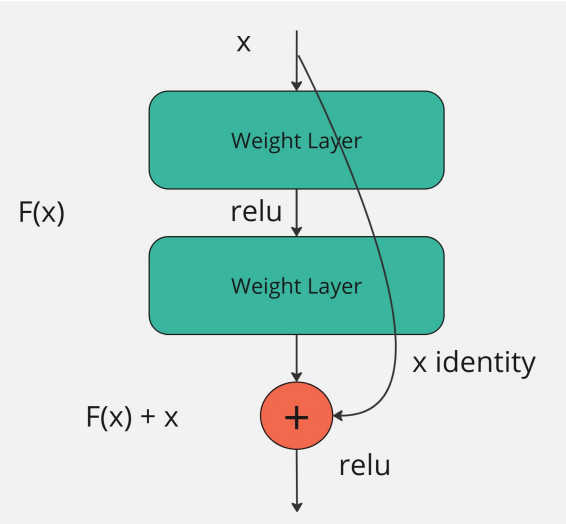


Figure 5.5: Residual Learning Building Block

The architecture of this model has two core building blocks. There are basic blocks with two 3×3 convolutions for shallower models and bottleneck blocks with three-layer designs of 1×1, 3×3, and 1×1. Batch normalization is applied after every convolution. Shortcut connections use 1×1 convolutions to adjust the dimensions when necessary. These verily allow the utilization of depth without excessive computational cost.

Model Variants and Their Applications

For the Wildfire and Skin Cancer datasets, a pre-trained `resnet18` and `resnet50` model with weights from `IMAGENET1K_V1` were imported from `torchvision.models` in PyTorch, and the model was trained for binary classification.

For the PlantVillage dataset, the model was imported from the HuggingFace Transformers library (`microsoft/resnet-18` and `microsoft/resnet-50`), and the model was prepared for performing multiple classification tasks.

5.6 VGG16

Simonyan and Zisserman introduced VGG16 as a deep CNN model in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" (2015) [12] which is one of the most widely recognized CNNs. The architecture prioritizes depth through many layers instead of using wide filters to improve its ability to represent images. VGG16 proves its strength in transfer learning applications because of its direct design that works effectively in various fields and domains [12].

Architectural Design of VGG16

Input and Preprocessing

The network requires images that measure $224 \times 224 \times 3$ as input. These size requirements ensure that the model works with training weights taken from ImageNet.

Convolutional Layers

VGG16 organizes a total of 13 convolutional layers across five blocks instead of presenting them all in one sequence. The filter count progressively increases throughout the blocks with 64 filters in the initial block and 512 filters in the last block. The 3×3 filters let the network extract detailed patterns across images while adding more filters enables it to spot advanced features.

Fully Connected Layers (Dense Layers)

After the convolutional blocks, the feature maps are flattened and passed through three fully connected layers:

- 4096 neurons use ReLU activation in the initial two fully connected layers.
- The final processor includes 1000 neurons with a softmax activation function to classify multiple input classes. To perform binary classification tasks, the final layer contains two output neurons with sigmoid activation ability.

All layers of the network implement the Rectified Linear Unit (ReLU) activation function as their standard method. The model learns sophisticated patterns by using this function that adds non-linear capabilities with minimum computational costs.

Pooling Layers

After convolutional blocks, we add max-pooling layers with 2×2 kernel size and step 2 to reduce feature map size by half. The resulting output dimensions can be determined using the standard convolutional formula, as given in Equations (5.5) and (5.6).

Weights and Parameter Count

With 138 million learnable parameters, VGG16 offers a complex model configuration. Although training VGG16 requires many computational resources, its strong performance proves valuable for large datasets such as the ones from ImageNet. With transfer learning, pre-trained ImageNet weights help the model learn faster and improve its performance. For a convolutional operation with a kernel size $k \times k$, stride s, and padding p, the output dimension $W_{\text{out}} \times H_{\text{out}}$ for an input of size $W_{\text{in}} \times H_{\text{in}}$ can be calculated as:

$$W_{\text{out}} = \frac{W_{\text{in}} - k + 2p}{s} + 1 \tag{5.5}$$

$$H_{\text{out}} = \frac{H_{\text{in}} - k + 2p}{s} + 1 \tag{5.6}$$

All 16 layers of the architecture are shown extensively in Figure 5.6

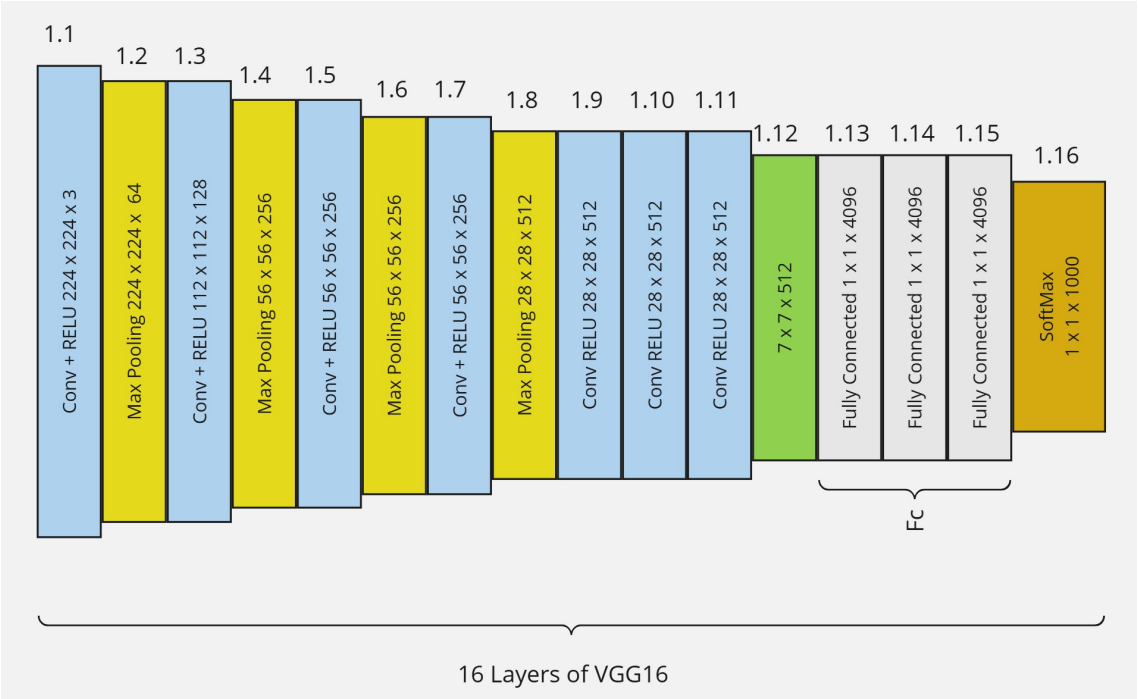


Figure 5.6: Overall architecture of VGG16

Model Variants and Their Applications

The VGG16 neural network architecture with image classification was imported from the `torchvision.models` library and operated on all three datasets (Wildfire, Skin Cancer, and PlantVillage) through unique task adaptations. The model needed input images of 224×224 pixels because it uses its basic operating setup. The final full connection layer of the classifier was swapped out to make two output logits for binary classification tasks. The model separated wildfire images into classes by Fire or No Fire, and plants by examining health (Diseased or Healthy) and cancer type (Benign or Malignant). The model took advantage of PyTorch's CUDA system to boost GPU performance while automatically using the CPU when it was requested. Similarly, for PlantVillage data our model used `id2label` and `label2id` features to match labels and adjusted its trainable parameters to strike a balance between performance and runtime speed. The VGG16 system showed good results across multiple tasks because its design stayed steady while adapting to different types of images.

5.7 InceptionV3

The authors Szegedy et al. (2016) introduced InceptionV3 to computer vision in the paper "Rethinking the Inception Architecture for Computer Vision" [9] as a deep neural network that combines efficient computation with high representation power. After the V1 and V2 releases, InceptionV3 brought essential architectural gains through factorizing convolutions and auxiliary classifiers while enhancing grid size and reduction techniques. The new architecture made inception popular for image classification work and helped it succeed with many source materials.

Architectural Design of the InceptionV3

Factorized Convolutions

Inception V3 increases efficiency through factorized convolutions that maintain performance results. Traditional 5×5 convolutions take too much computing power, so two smaller 3×3 convolutions run consecutively instead. The parameter reduction and performance benefits of this factorization let networks handle more input data at lower computational cost [9]. By breaking spatial convolutions into asymmetric settings (3×1 and 1×3), InceptionV3 reduces floating-point operations (FLOPs) without losing spatial dependency benefits. InceptionV3 needed this innovation to handle big datasets and many-layered networks.

Auxiliary Classifiers

Because deep neural networks encounter difficult gradient decay challenges, InceptionV3 uses auxiliary classifiers within network layers to support effective training. During the training, these classifiers boost early-stage feature learning and supply extra gradient directions. A helper classifier working on the 17×17 feature map helps create regularized models that function well in many environments. Each of the support classifiers includes a condensed convolutional network performing global pooling, fully connected calculation, and a softmax classifier. The training system uses the auxiliary classifiers temporarily to make optimization gradients flow better.

Grid Size Reduction

With InceptionV3, 1×1 convolutions boost max-pooling effectiveness for spatial dimension reduction. This solution keeps more information from each feature map while overcoming traditional pooling problems. Through its design, Inception V3 maintains excellent accuracy results on multiple tasks while preventing the model from becoming too specialized for new data and reducing the risks of over-fitting.

Inception Modules

At its core, InceptionV3 uses a series of Inception modules that identify features in both small and large details. Multiple convolutional blocks run side by side with pooling filters at 1×1, 3×3, or 5×5 to extract the smallest to largest patterns from input images. By processing multiple level inputs at once, the network acquires detailed insights alongside global overview data. Each Inception module can be mathematically represented by Equation (5.7), which defines how multiple convolutional operations are combined.

$$F(x) = \sum_{i=1}^{N} W_i * x_i + b_i \tag{5.7}$$

Here, x represents the input feature map, W_i denotes the weights of each parallel operation, and b_i signifies the biases. The outputs of these operations are concatenated to form the final output of the module.

Final Classification

InceptionV3 concludes with global average pooling that structures feature maps but saves critical information. It takes the generated features and puts them through a full connection layer to produce multi-class outputs when using the softmax activation or binary classes using the sigmoid activation. During testing, the network included two output values to match the binary categories in the dataset.

Computational Efficiency

InceptionV3 reaches top results on ImageNet with fewer parameters than its equivalent networks. It uses three strategies including factorized convolutions to help auxiliary classifiers and grid size reduction techniques which decrease the accuracy vs. compute trade-off as illustrated in Figure 5.7.

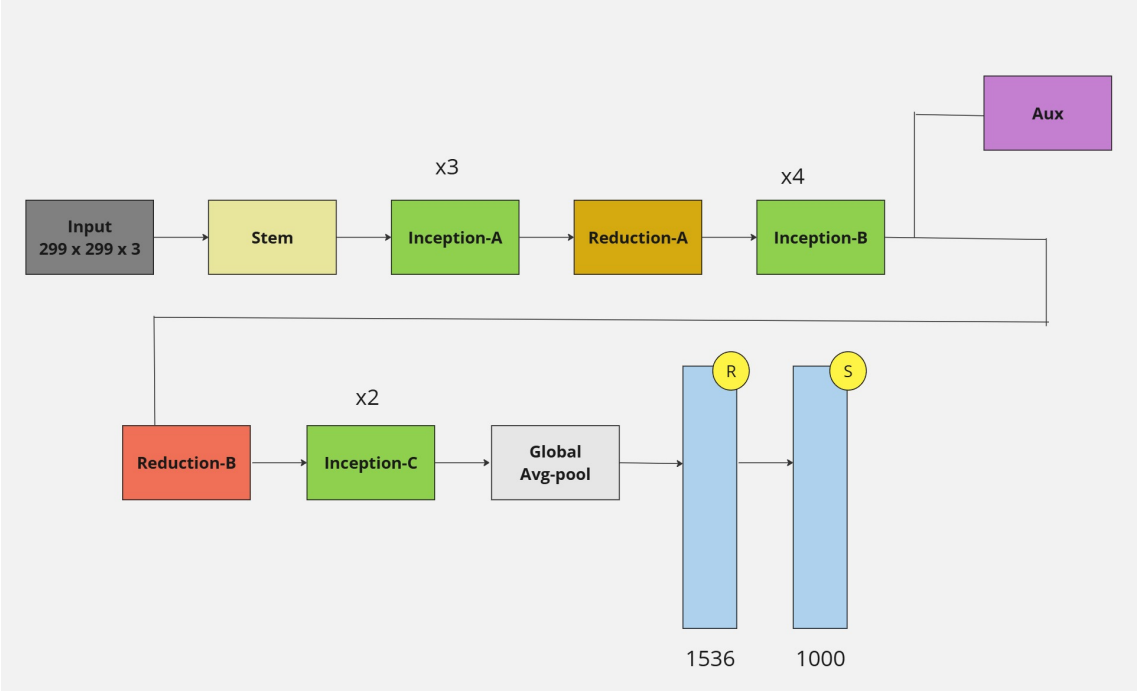


Figure 5.7: Overall architecture of InceptionV3

Model Variants and Their Applications

To study different classification topics the InceptionV3 architecture started with `IMAGENET1K_V1` weights from the `torchvision.models` library which it applied to all three datasets. Our model needed 299×299 pixel input images to work properly with its design. In the final model layer, we replaced the traditional full connection network with our custom binary classifier to identify Fire vs Non-fire, Healthy or Diseased Plants, and Malignant or Benign occurrences. For PlantVillage data a simple tool transformed label numbers into names while setting `ignore_mismatched_sizes=True` maintained layer compatibility. With this setup, the InceptionV3 model demonstrated superior results in both wildfire monitoring and medical image analysis before transitioning to identifying plant diseases effectively.

5.8 Xception

In 2017 François Chollet introduced Xception (short for Extreme Inception) as a new development of the Inception architecture. This model applies depth-wise separable

convolutional to split spatial spatial filtering tasks from channel processing which leads to beneficial performance and reduced power consumption. The Xception model performs better than standard CNN models on all image processing tasks including both object detection and image classification.

Architectural Design of the Xception Xception expands beyond Inception principles through depth-wise separable convolutions which replace traditional convolutional layers across the network design. The new method replaces regular convolutions to decrease processing costs without lowering performance [21].

Depth-wise Separable Convolutions

Unlike standard convolutions, which combine spatial filtering and channel projection into a single operation, Xception separates these into two distinct steps:

- Depth-wise Convolution: It performs independent spatial convolutions on each input channel.
- Point-wise Convolution: Depth-wise convolution results are connected using 1×1 convolution operation.

Mathematically, the operation can be expressed as:

$$\text{Output} = \sigma\left(W_{\text{pointwise}} * \left(W_{\text{depthwise}} * X\right)\right) \tag{5.8}$$

Where X is the input tensor, $W_{\text{depthwise}}$ refers to the weights for the depthwise convolution, $W_{\text{pointwise}}$ represents the weights for the pointwise convolution, and σ is the activation function, such as ReLU.

This decoupling reduces the parameter count and computational complexity, making the model highly efficient. For example, a standard convolution with C_{in} input channels, C_{out} output channels, and a kernel size of $k \times k$ requires:

$$C_{\text{in}} \times C_{\text{out}} \times k \times k \tag{5.9}$$

Parameters, while a depthwise separable convolution requires:

$$C_{\text{in}} \times k \times k + C_{\text{in}} \times C_{\text{out}} \tag{5.10}$$

This reduction in parameters is particularly beneficial for large-scale datasets.

Architectural Flows

Xception's architecture is organized into three main flows:

1. **Entry Flow:**
    - The system downsamples image pixels to create features of greater depth.
    - A sequence of convolutional layers with max-pooling operators uses depthwise separable convolutions instead of standard convolutions.
2. **Middle Flow:**
    - Each of its eight feature extraction sections handles input data identically.
    - Each module works with depthwise separable convolutions, which extract features without distorting spatial properties.
3. **Exit Flow:**
    - The last layers work to shrink image size while taking out useful deep information.
    - Ends with a global average pooling and a decision-making network for recognizing elements.

Feature Representation and Residual Connections

The Xception structure connects all modules through residual links which helps train data flow correctly while reducing the risk of gradients disappearing. Through residual connections, the model learns strong features efficiently during training and avoids instabilities.

Computational Efficiency and Scalability

Xception holds 23 million parameters but needs less CPU resources than InceptionV3 because depthwise convolutions work faster. The model design fits well with large image datasets because it uses fewer computational resources than other models.

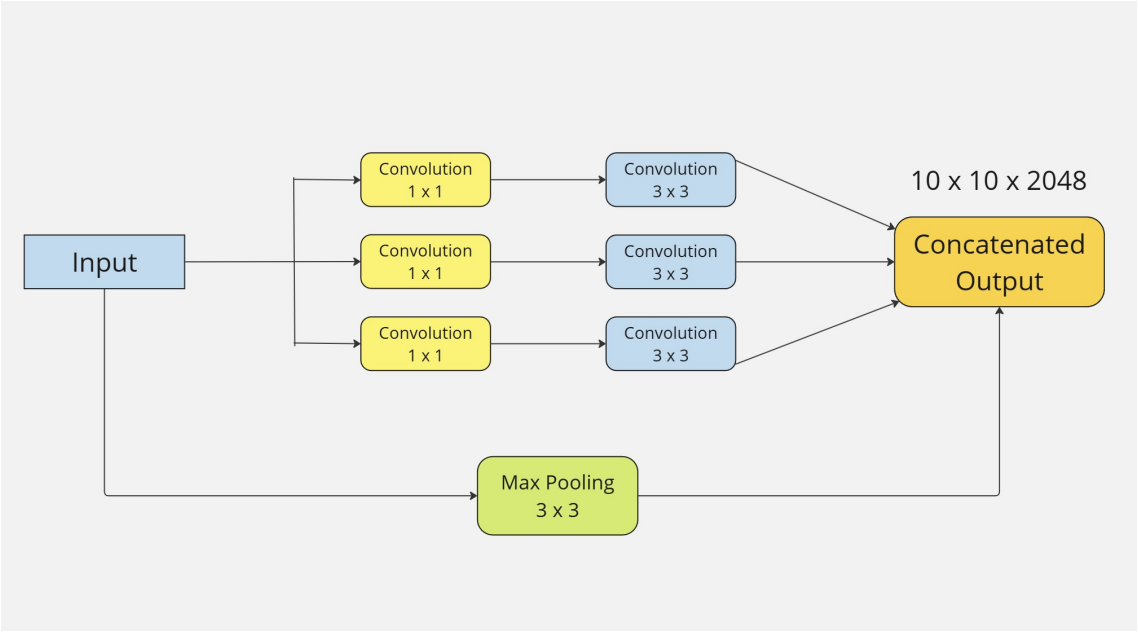


Figure 5.8: Overall architecture of Xception

Model Variants and Their Applications

Xception showed strong performance when working with all three datasets with its ImageNet pre-trained weights adapting directly to binary classification tasks. The basic fully connected layer was replaced with its classification layer while converting all data to 299 × 299 pixel images following the Xception model specifications. The established weights performed well for feature extraction and continued to work smoothly when `ignore_mismatched_sizes=True` was set to transition to different classification purposes. The model used CUDA for fast GPU training and processing but automatically fell back to CPU processing when no GPU resources were available. The PlantVillage dataset demands customized label indexing through modules `id2label` and `label2id` with 24.96 million trainable parameters to achieve maximum performance. With these modifications, Xception demonstrated good results for detecting whether images showed fire or not as well as detecting malignant or benign lesions and healthy versus diseased plants.

5.9 MobileNet V2

MobileNetV2 is a neural network architecture design suitable for resource-constrained and mobile applications and environments. It is an improvement on the original Mobilenet by the introduction of an innovative building block called the inverted residual structure with linear bottlenecks. [16] explains how by balancing performance and efficiency, MobileNetV2 is ideal for image classification, object detection, and semantic segmentation across various fields.

Overall Architecture

The Inverted Residuals and Linear Bottlenecks are very different from the traditional residual blocks. Where the latter usually connects layers with a high number of channels, the former has inverted residuals that connect thin bottleneck layers. A block begins with an expansion phase which goes towards high-dimensional space and applies depthwise separable convolution or filtering. All of these compress back into a low-dimensional space through linear convolution. Essential information is preserved with the help of linear bottlenecks which are narrow layers that remove non-linearities.

This model offers depthwise separable convolutions with an efficient convolutional method splitting traditional convolution into depthwise and pointwise convolutions (1×1). Computational cost is also reduced to 1/8 from 1/9 of standard convolutional layers without any significant effect on accuracy. It also has a good trade-off between expressiveness and capacity. The architecture separates the roles of expensiveness and capacity. Expressiveness is handled by expansion layers and capacity is defined by the bottlenecks. This separation incorporates memory-efficient inference which is extremely crucial for mobile applications. Details are shown in Fig 5.9.

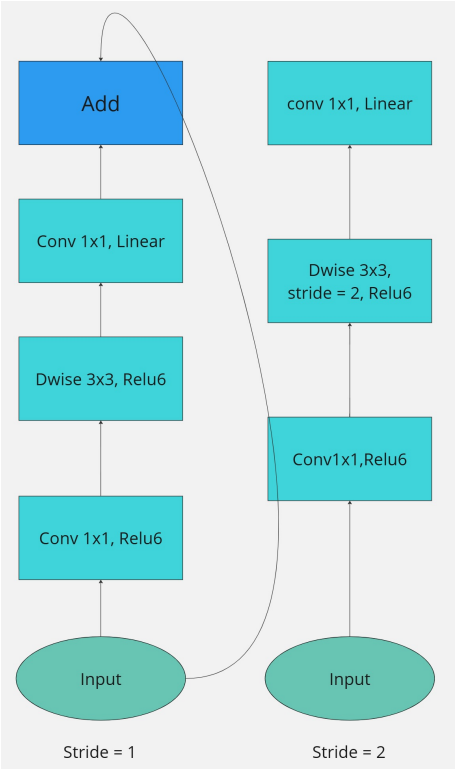


Figure 5.9: Overall architecture of MobileNetV2

Now let us look into further details within the architecture. Primarily in the basic block, the input features are expanded using 1x1 convolution followed by a 3×3 depthwise separable convolution with ReLU6. Finally, the output is projected back using a lower dimension of 1x1 linear convolution. Shortcut connections are also utilized where the input and output dimensions match. Structure wise the initial

layer has a full convolution with 32 filters and stride 2. This is followed by 19 inverted residual bottleneck layers with various expansion factors and strides. The final layer is the 1×1 convolution as mentioned before with average pooling and a final fully connected layer. The aforementioned ReLU6 is used as the non-linearity for robustness in low-precision computing. During training the dropout and batch normalization are applied. The expansion factor is typically set to 6.

Among the advantages of using this model are the reduced computational complexity, fewer operations, and fewer memory accesses. It is tunable and its hyperparameters allow room for adaptation to different resource constraints. It is also scalable and works well across classification, detection, and segmentation. This model represents quite a significant step forward for mobile-centric neural network designs. All its features make it a preferred choice for real-time, on-device AI applications too.

Model Variants and Their Applications

For the Wildfire and Skin Cancer datasets, a pre-trained `mobilenet_v2` model with weights from `IMAGENET1K_V1` was imported from `torchvision.models` in PyTorch, and the model was trained for binary classification.

For the PlantVillage dataset, the model was imported from the HuggingFace Transformers library `google/mobilenet_v2_1.0_224`, and the model was prepared for performing multiple classification tasks.

5.10 ConvNext Base & V2

The ConvNext models are updates to existing convolutional neural networks (ConvNets) where the incorporation of design principles of ViTs and Swin Transformers are applied to modernize the ConvNet family of models to deliver competitive and superior performance on various computer vision tasks. [33] and [45] provide the following architectural details and mechanisms of the models of the base and second versions.

Overall Architecture

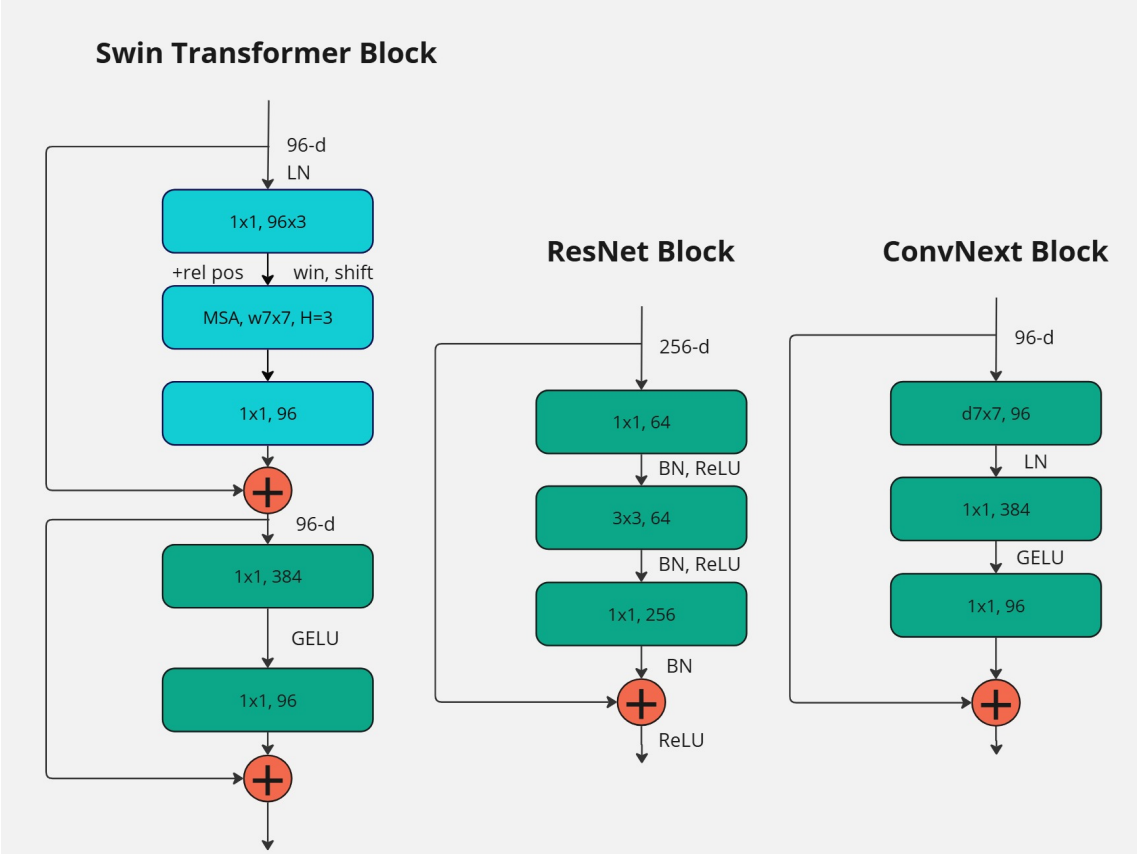


Figure 5.10: Block diagrams of ResNet, Swin Transformer, and ConvNext

In the block diagrams in 5.10, Swin is more sophisticated for the presence of multiple modules and two residual connections. For simplicity, linear layers in Transformer MLP blocks are denoted as '1×1'.

The ConvNets are introduced to features like larger kernel sizes, fewer activation functions, and advanced normalization techniques. Depthwise convolutions, inverted bottlenecks, and a patchy stem for downsampling which is inspired by ViTs are also introduced. As a result, ConvNext achieves SOTA results and becomes suitable for a range of computational budgets and applications. Batch normalization is replaced with Layer Normalisation and ReLU activations with GELU which further optimizes performance and efficiency(as shown in 5.10). ConvNext can also work on multiple sizes which offers flexibility from Tiny to Extra Large.

ConvNext V2 builds upon the ConvNext V1 and addresses the gaps in it. While the first version and ConvNets were optimized for supervised learning, they struggled to adapt self-supervised techniques like autoencoders. This second version proposes and introduces Global Response Normalization (GRN) to enhance feature diversity and sparse convolutions to efficiently process masked inputs. This is a highly scalable model, achieving SOTA performance across diverse benchmarks, ranging from lightweight to high-capacity models.

It retains the original core structure of its predecessor while making critical modifica-

tions to support self-supervised learning. The main feature GRN, as shown in 5.11, normalizes channel activations using global L2-norms, fostering feature competition and preventing the feature from collapsing. This does not increase computational cost. In V2 we add the GRN layer after dimension expansion MLP layer and drop Layerscaler as it becomes redundant. Dense convolutions are replaced by sparse convolutions in specific stages to process masked inputs more efficiently, ensuring no information is leaked during pretraining.

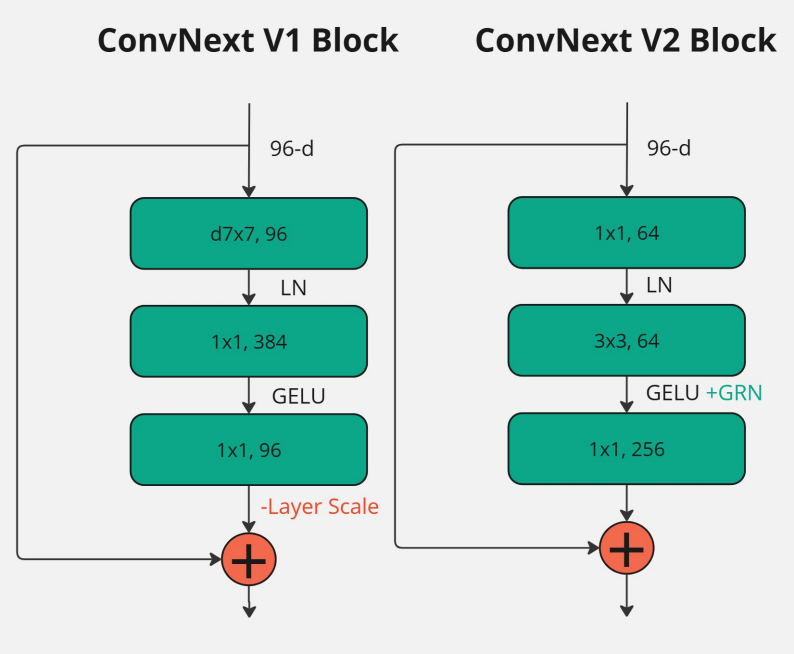


Figure 5.11: ConvNext Block Diagrams

This model proposes the FCMAE framework, a fully convolutional self-supervised learning method that uses sparse convolutions for processing masked input. With lightweight decoders that employ ConvNeXt blocks for image reconstruction, FCMAE prevents information leakage and lowers pretraining complexity in contrast to transformer-based masked autoencoders. A patch-wise normalized mean squared error is used to calculate reconstruction goals, and the framework's performance is further improved by using GRN. All of these improvements make ConvNeXt V2 a ConvNet that can effectively simulate masked images, a field that transformers used to dominate.

Model Variants and Their Applications

For the Wildfire and Skin Cancer datasets, a pre-trained `convnext_base` model with weights from `IMAGENET1K_V1` were imported from `torchvision.models` in PyTorch, and the model was trained for binary classification.

For the PlantVillage dataset, the model was imported from the HuggingFace Transformers library (`facebook/convnext-base-224-22k`), and the model was prepared to perform multiple classification tasks.

For the Wildfire and Skin Cancer datasets, a pre-trained `convnextv2_tiny.fb_in1k` model was imported from `timm` in PyTorch, and the model was trained for binary classification.

For the PlantVillage dataset, the model was imported from the HuggingFace Transformers library (`facebook/convnextv2-base-22k-224`), and the model was prepared for performing multiple classification tasks.

5.11 EfficientNet B0-B7

EfficientNet B0 and B7 fall under the EfficientNet family which is a group of convolutional neural networks, designed in a manner to balance accuracy and efficiency. The compound scaling method (as shown in 5.12) is its core innovation which uniformly scales up a network's depth, width, and input resolution to achieve optimal accuracy under given resource constraints.

Overall Architecture

[22] provides the details with the baseline of this model which is the EfficientNet B0, constructed using neural architecture search to optimize both accuracy and computational cost. To attain improved efficiency, this model utilizes Mobile Inverted Bottleneck Convolutions and Squeeze-and-Excitation optimization. It scales seamlessly from B0 to B7 configurations while maintaining a consistent architectural design. The EfficientNet B0 has 1 initial convolutional layer. Then it has 7 stages of MBConv layers with increasing channels and decreasing resolution. The final layer is a 1×1 convolution, average pooling, and a fully connected layer.

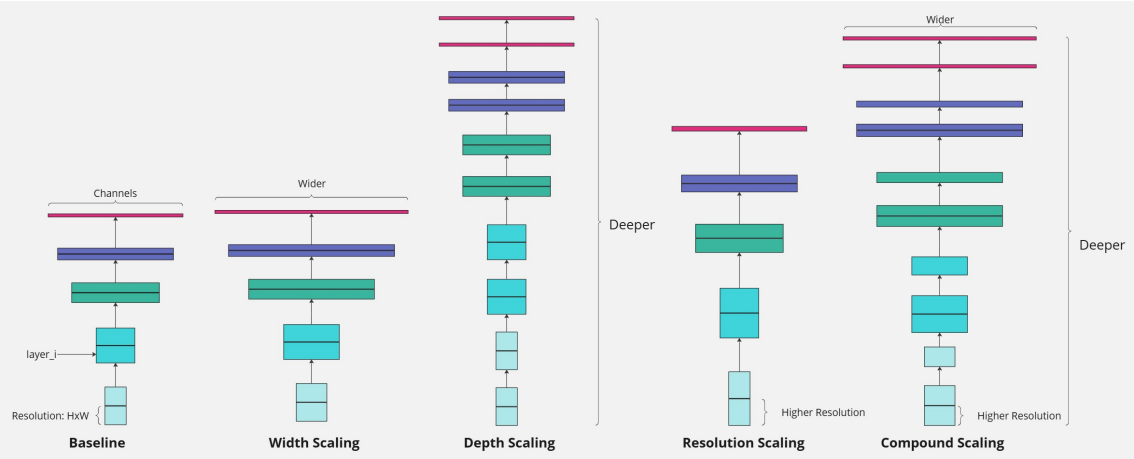


Figure 5.12: Model scaling of different types. The last one is the proposed B7 architecture.

It has highly improved efficiency with significantly fewer parameters and FLOPS compared to traditional ConvNets, without sacrificing accuracy. It is highly scalable

as the compound scaling method provides a simple and effective way. It is also versatile as it can handle tasks like image classification, object detection, and transfer learning. It is a clear demonstration of how models can be families of models can be formed that are both accurate and resource-efficient by careful examination of scaling of depth, width, and resolution.

Model Variants and Their Applications

For the Wildfire and Skin Cancer datasets, a pre-trained `efficientnet_b0` and `efficientnet_b7` model with weights from `IMAGENET1K_V1` was imported from `Torchvision.models` in PyTorch, and the model was trained for binary classification.

For the PlantVillage dataset, the model was imported from the HuggingFace Transformers library (`google/efficientnet-b0` and `google/efficientnet-b7`), and the model was prepared to perform multiple classification tasks.

5.12 Vision Mamba

The Vision Mamba (ViM) model incorporates the Mamba block [50] in an efficient, scalable, bidirectional vision backbone. The model claims to be $\times 2.8$ times faster and uses 86% less GPU memory on batch interference for high-resolution image samples compared to Data-efficient Image Transformers (DeiT) [78].

Like Vision Transformers (ViT), input images in Vim are split into patches and linearly projected as vectors. These patches are used as sequence data when fed into the Mamba model and the model can be trained on unsupervised data. The novelty in ViM stems from the fact that ViM is by far the leading purely SSM-based vision model available. These SSM-based models trace their roots to continuous systems. However, the continuous parameters first need to be discretized using a timescale parameter and zero-order hold (ZOH) transformation.

ViM block

Images are usually 3-dimensional tensors (height, width, color channels) but Mamba can only process 1-dimensional sequences. Hence, the Vim block was introduced to process images into a suitable form that could be fed into a Mamba block.

Initially, the image is divided into patches of size 16. Positional embedding is added to the patches before being flattened (T_i as in equation 5.11) and fed into a Vim encoder block. Inside the Vim encoder blocks, the image patches are normalized using a normalization layer before being linearly projected in $\boldsymbol{x}$ and $\boldsymbol{z}$ vectors of dimensions E and 1-d convolution is applied to reduce the number of tokens. Next, $\boldsymbol{x}$ vector is processed in both forward and backward directions through two Mamba blocks to learn sequential dependencies from either direction as Mamba utilizes

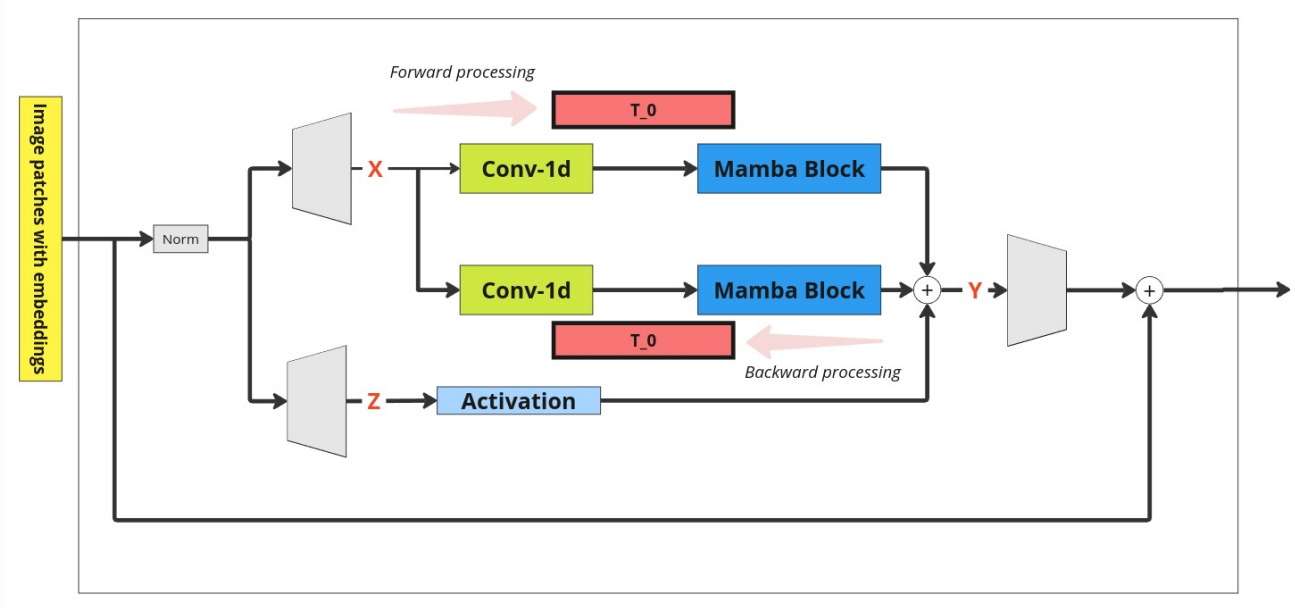


Figure 5.13: Vim Encoder Block

something similar to masked attention. The output of the two Mamba blocks is then combined with a skip connection ($\boldsymbol{z}$ passed to activation) and the output is obtained as shown in figure 5.13 [78].

$$\mathbf{T}_i = \left[t_{\text{cls}}; t_p^1\mathbf{W}; t_p^2\mathbf{W}; \cdots; t_p^J\mathbf{W}\right] + \mathbf{E}_{\text{pos}}, \tag{5.11}$$

Equation 5.11 is the input vector to the conv-1d and mamba blocks. Note the position of the cls token is in the middle when it's passed into the block.

Architecture

The overall ViM model contains L stacked ViM blocks where L is set to a value of 24 by default. The dimensions of the hidden state are set to 192 and 384 and the expanded dimension, E, is set to 384 and 768 for the tiny and small variants respectively,y and is described in figure 5.14. The ViM architecture is heavily inspired by ViT and BERT. Each ViM encoder block processes the sequences from both directions similar to BERT and the the blocks are stacked on top of each other like ViT. Once the sequence passes through all the blocks, the final output cls token is extracted and passed through a normalization layer. The output of the normalization layer is then passed into an MLP head and then a final output layer.

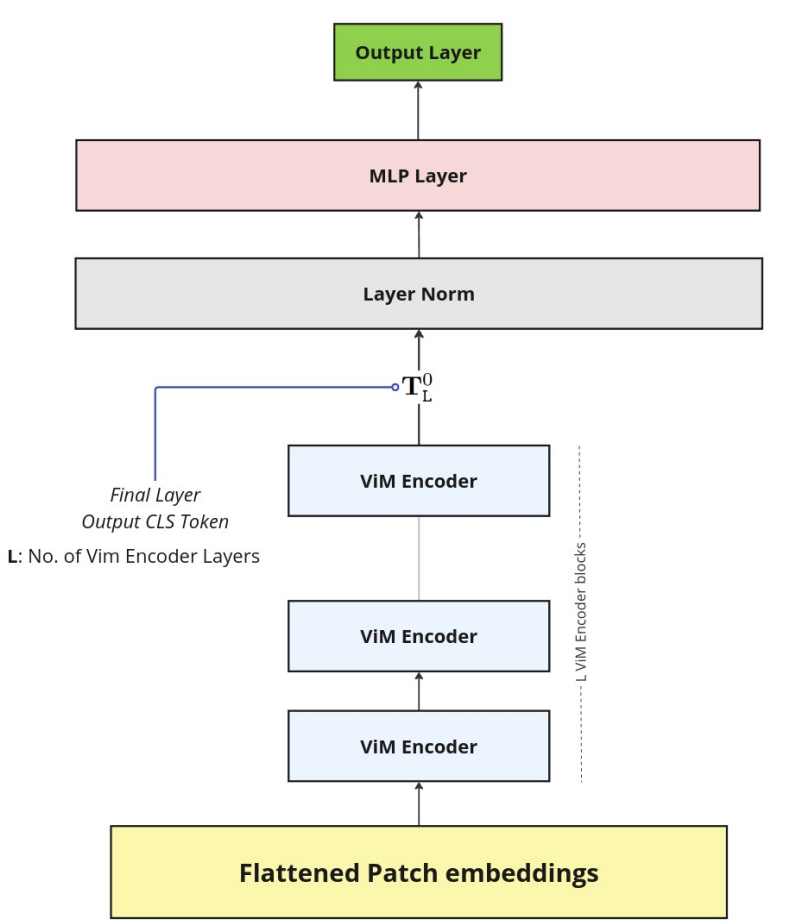


Figure 5.14: Overall architecture of ViM consisting of stacked ViM encoder blocks and MLP layers

Efficiency

The architecture ensures IO efficiency by reducing the number of IO operations required between HBM and SRAM of GPU accelerators from the order of BMEN to the order of EN+BME where B is the batch size, N is the SSM dimensions, E is the dimensions of the projection and M is the sequence length. In addition, to optimize memory usage for long sequences ViM does not store the intermediate states in memory. Instead, it does a recomputation step during the backward pass for both activations and parameters akin to Mamba itself. The computational complexity of ViM comes down to $\Omega(\text{SSM}) = 3M(2D)N + M(2D)N$ which is more efficient than the quadratic complexity of attention.

Chapter 6

Methodology

This chapter provides the fundamental research framework by explaining techniques for reaching research goals through specific setups and processes. It systematically explains the strategies employed to preprocess datasets, train models, and evaluate their performance across three distinct domains. Every dataset required its own specialized approach because it included unbalanced classes and different image quality levels, but needed customized augmentation solutions.

Figures 6.1, 6.2 and 6.3 present concise methodologies that match each dataset by illustrating the complete data preprocessing and model evaluation process in high resolution. The workflows demonstrate how researchers tackled the built-in constraints found within their datasets to create strong training platforms and evaluation frameworks. Throughout all experiments, consistent parameters prevailed including standardized evaluation metrics together with defined hyperparameter values and resource consumption.

The methodology creates a detailed approach for state-of-the-art model assessment through its combination of data augmentation and stratified splitting along with well-defined evaluation results while providing a versatile fundamental structure that can scale to different predictive challenges.

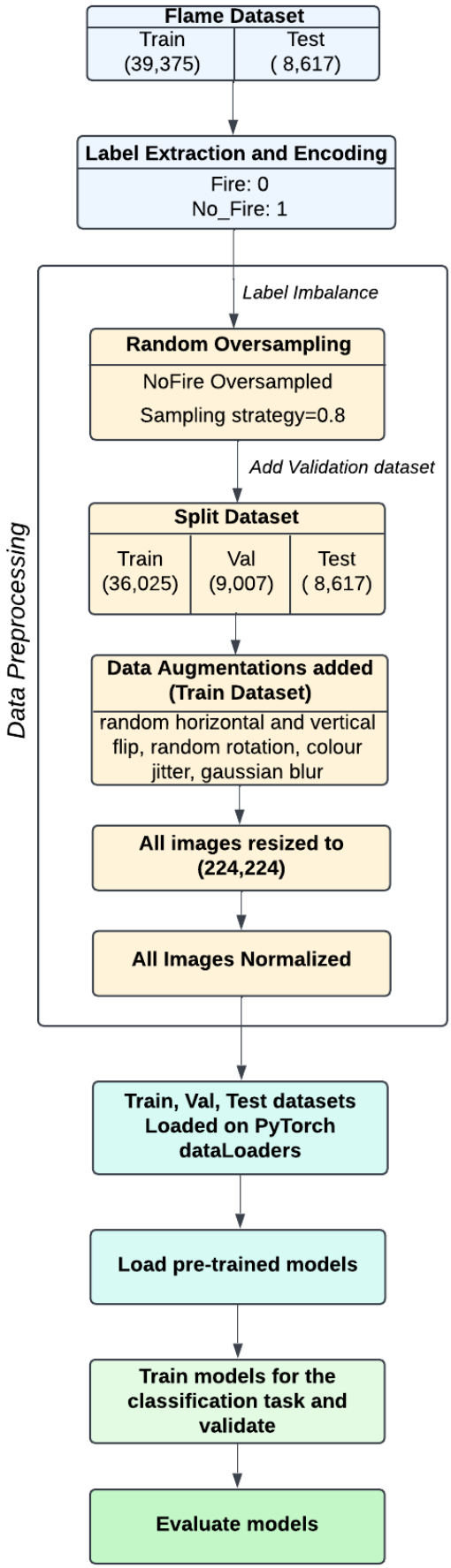


Figure 6.1: High-level Overview of Methodology for Flame Dataset

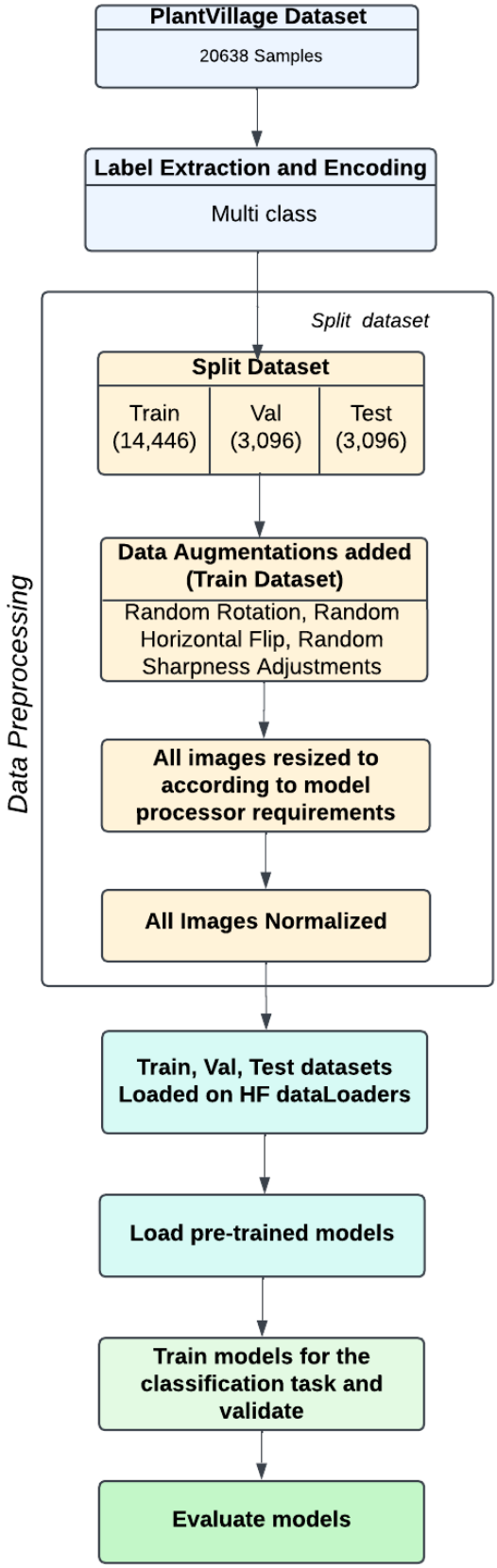


Figure 6.2: High-level Overview of Methodology for PlantVillage Dataset

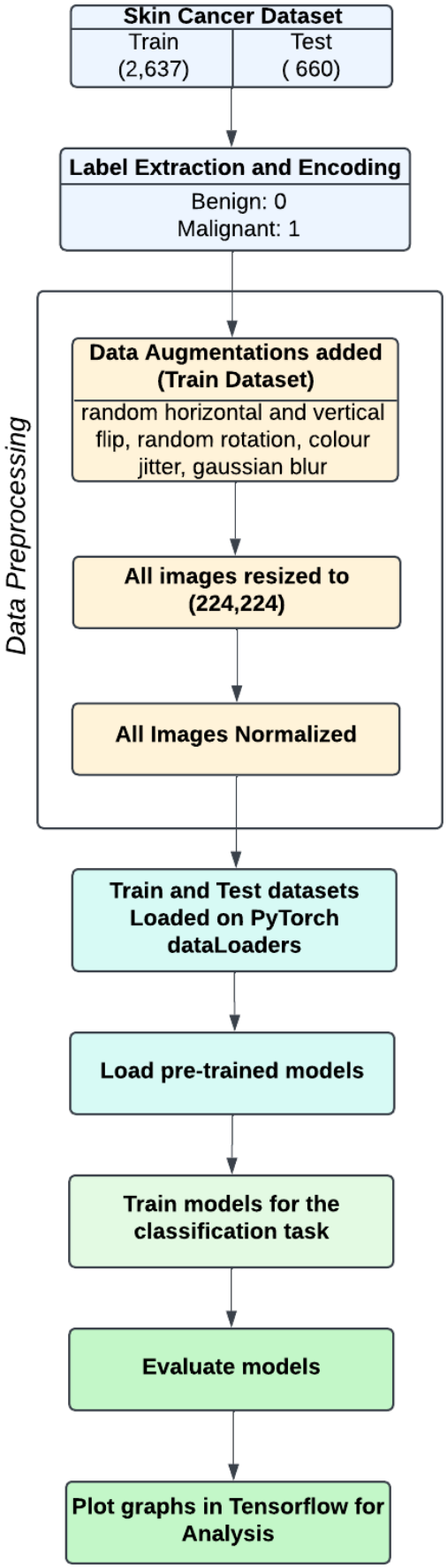


Figure 6.3: High-level Overview of Methodology for Skin Cancer Dataset

6.1 Data Preprocessing

6.1.1 Preprocessing the Flame Dataset

A well-thought-out preprocessing pipeline was implemented to address the challenges presented by the dataset, such as class imbalance, potential data leakage, and limited diversity in environmental conditions. The pipeline sought to improve the model's generalization capability, mitigating over-fitting risks and ensuring a balanced representation of classes during training. A stratified train-validation split was employed to preserve class balance across subsets, and data augmentation techniques were used to replicate various real-world scenarios. The moderate class imbalance in the training set was also addressed using oversampling techniques, which ensured that the minority class (NoFire) was fairly represented.

The series of augmentations that were applied to the training dataset for improving variability is as follows:

- **Random Horizontal and Vertical Flips:** Introduced diversity by randomly flipping pictures either vertically or horizontally with a 50% chance. This reproduces the various orientations in real-world imaging.
- **Random Rotation:** Images were randomly rotated by up to ± 10 degrees to account for variations in camera angles.
- **Color Jittering:** Adjustments to brightness, contrast, and saturation ($\pm 20\%$) were applied to simulate different lighting conditions. Since the RGB distribution reflects real-world conditions, augmentations like color jittering were applied to create variability and prevent over-reliance on specific color patterns.
- **Gaussian Blur:** Introduced mild noise with a kernel size 3×3 and a sigma range of 0.1 to 0.5 to make the model robust to blurry images.
- **Normalization:** Images were normalized using the mean [0.485,0.456,0.406] and standard deviation [0.229,0.224,0.225] of the ImageNet dataset, ensuring consistency with pre-trained model requirements.

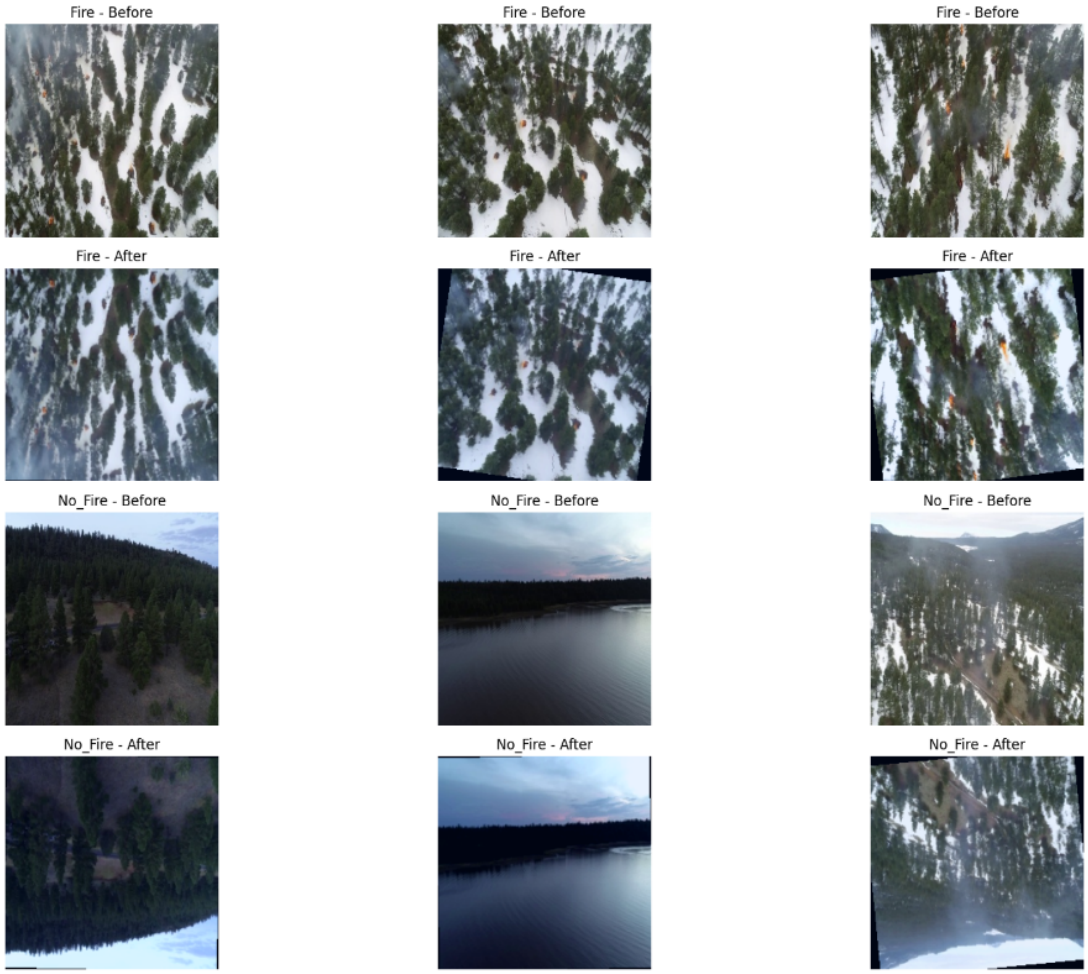


Figure 6.4: Before and After samples of Image Transformations

The training set was over-sampled to handle class imbalance using RandomOverSampler to get a fair representation of the minority class (NoFire). A sampling strategy of 0.8 ensured that NoFire samples constituted at least 80% of the number of Fire samples in the training data. It was ensured that oversampling the NoFire class does not cause the Recall score to drop as giving a false negative for when fire is present can cause more harm, with catastrophic consequences, than giving a few false positives for the Fire class. Recall was given more priority for this particular scenario.

A stratified split was performed on the over-sampled training set to create separate training and validation subsets, both having balanced classes. 80% of the data was used for training (36025 images), while 20% was reserved for validation (9007 images). Stratification ensured that the class distribution was preserved in both subsets. Our final class distribution for the Training and Validation datasets were: Training (Fire: 20014 samples, NoFire: 16011 samples) and Validation (Fire: 5004 samples, NoFire: 4003 samples).

The augmentations done to the validation and test datasets, where images were only resized and normalized, were kept minimal as such heavy augmentations are not realistic for actual wildfire scenarios. Moreover, consistency in evaluation had

to be ensured. The images were only normalized.

The processed datasets were loaded into PyTorch DataLoader objects to enable batch-wise loading during training and evaluation. A batch size of 32 is used for the training loader, and each epoch's sample order is randomly assigned using shuffling. To maintain order for evaluation, validation, and test loaders, a batch size of 32 was used without shuffling.

The preprocessing pipeline was created to ensure the data was ready for efficient training and evaluation while addressing the dataset's inherent constraints and problems. The pipeline reduced the possibility of data leaking, reduced the risks of class imbalance, and increased the dataset's variability by implementing strategies including data augmentation, oversampling, and cautious train-validation-test separation. These procedures were crucial in producing a solid and well-rounded dataset as shown in Fig 6.4, which guaranteed that the deep learning models would perform well when applied to new data. Additionally, by aligning the dataset with the specifications of cutting-edge architectures, changes like normalization and resizing made training effective and reliable. The preprocessing approach established a solid foundation for accomplishing the wildfire detection objective, offering a reliable framework for the creation and evaluation of models.

6.1.2 Preprocessing the PlantVillage Dataset

The PlantVillage dataset [7], sourced from Kaggle, does not come with any predefined train, test, and validation sets. We randomly shuffle the data and divide the images into separate train, test, and validation sets where 70% of the images are used to train the models, while the remaining 30% is divided equally among the validation and test sets. During splitting the data, we stratify the dataset according to the labels to ensure that no label is over-represented or under-represented in any particular set. The distribution of labels in each set is shown in figure 6.5.

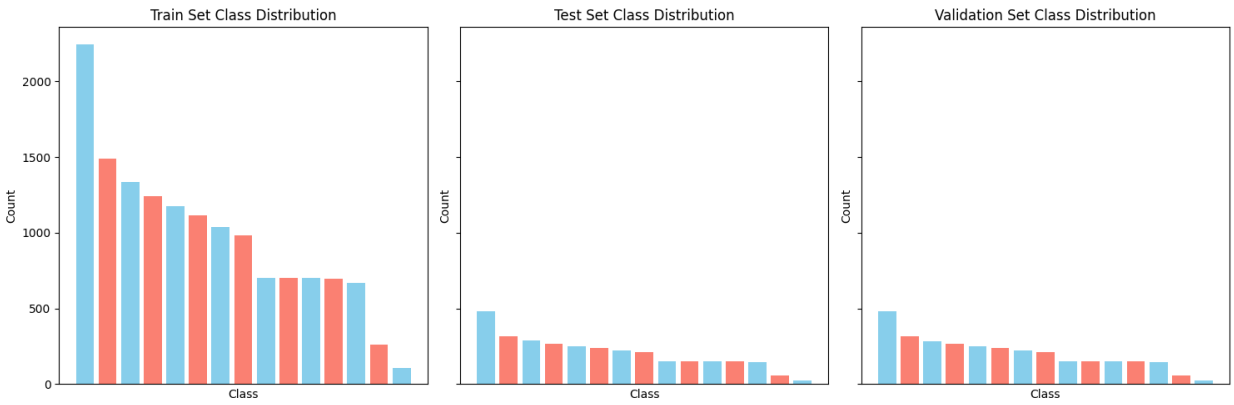


Figure 6.5: Class Distribution after splitting, stratified with respect to labels

All images in the dataset have uniform dimensions of 256×256. No oversampling was required as the dataset is fairly well-balanced. The images were resized according to the input dimensions of each model: center crop was applied to downsize the image

and padding was applied to upscale the image when required. However, most models handle the default dimensions of the images very well and so require no resizing. In addition, to improve the generalizability of the models, the images were normalized with standard image mean and standard deviation as specified by the model's image processor which is usually the same as the ImageNet (as our models were pre-trained on the ImageNet-1k dataset) mean and standard deviation of [0.485,0.456,0.406] and [0.229, 0.224, 0.225] respectively for the RGB channels. Random rotation, random horizontal flipping, and random sharpness adjustments were also made to ensure the models do not overfit the training data. Both train and validation sets were resized and normalized, no other transformations were applied to them. Figure 6.6 shows some random images from the train set after preprocessing.

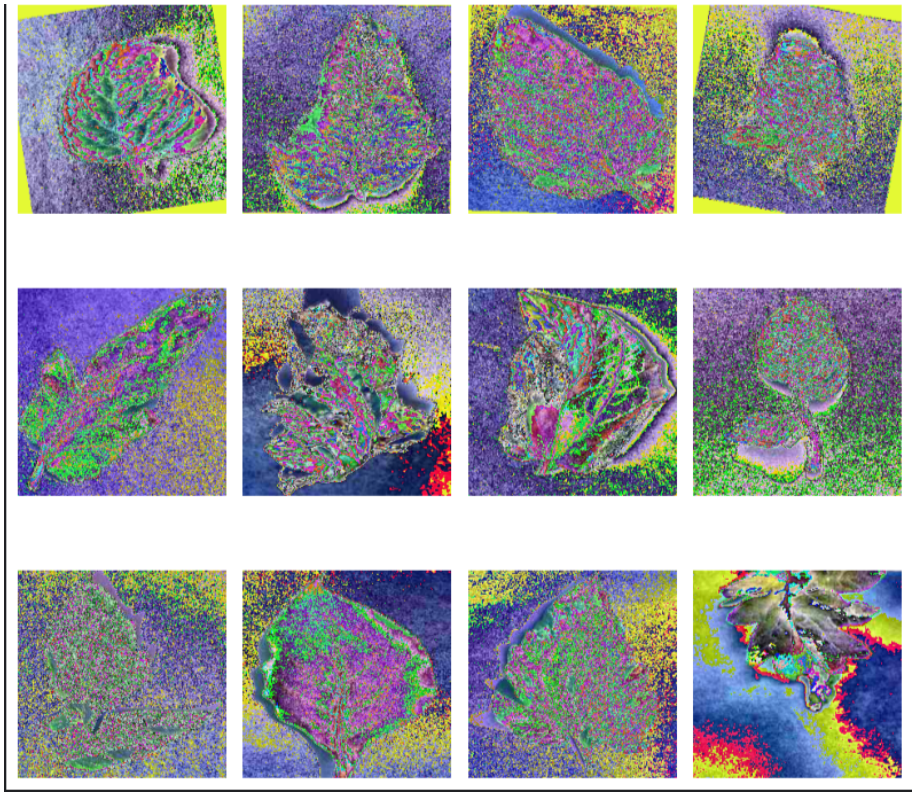


Figure 6.6: Images after preprocessing in the training set

6.1.3 Preprocessing the Skin Cancer Dataset

As discussed in 4.3, the skin cancer dataset does not require excessive preprocessing or data splitting. However, to improve model robustness, handle overfitting, and increase the diversity of the skin cancer training data, the same data augmentation techniques of the wildfire datasets were applied to the training dataset. Similarly, the test dataset was kept simple like the FLAME dataset.

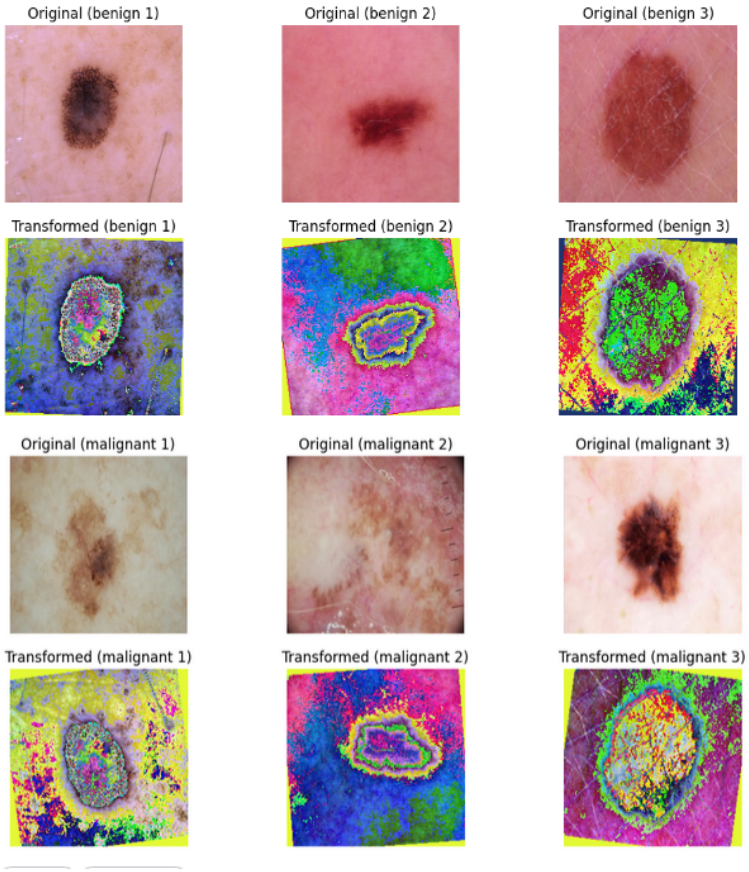


Figure 6.7: Before and After samples of Image Transformations

Data loaders are created for training and testing datasets with a batch size of 32 for efficient training and evaluation. Shuffling was enabled only for the training dataset to introduce randomness but was disabled for testing.

Combining these augmentation methods increases the diversity of the training data and replicates the variety found in skin lesion photos in the real world as shown in 6.7. In addition to lowering the possibility of overfitting, these additions improve the model's capacity to generalize to previously unseen images—a crucial prerequisite for medical applications where variations in imaging settings are unavoidable. The following preparation procedures are essential for preparing the dataset for deep learning models. By ensuring that the input data is balanced, standardized, and enhanced, these procedures enable strong model training and better generalization to new data.

6.2 Experimental Setup

6.2.1 Training Setup

Wildfire Detection and Skin Cancer Detection (Binary Classification)

The training setup for the binary classification tasks of detecting wildfire and skin cancer ensures that all the models were trained under consistent conditions for a fair comparison. The same codebase, hyperparameters, environment, and evaluation criteria were used across all models during training. The experiments were implemented using PyTorch, and TensorBoard was integrated into the training pipeline to log metrics such as accuracy, learning rates, GPU utilization, evaluation scores, etc. To ensure the reproducibility of results across runs, a fixed random seed (42) was applied across all components (torch.manual_seed(42)). Deterministic operations were enforced for CUDA to further enhance reproducibility. Additionally, PyTorch's backend settings (torch.backends.cudnn.deterministic=True and torch.backends.cudnn.benchmark=False) were configured to eliminate randomness in operations on the GPU.

For the training phase, models were trained for a maximum of 50 epochs. However, an early stopping condition was set to terminate training to avoid overfitting and reduce unnecessary computations. The early stopping condition for the skin cancer classification ensured that training was halted if the training loss did not improve for 5 consecutive epochs, with an additional margin ($\Delta = 0.0$) to account for noise. This mechanism ensured that the training process stopped as soon as the model reached its optimal performance. On the other hand, in the wildfire classification task, early stopping was implemented to monitor the validation loss. If the validation loss did not improve for 5 consecutive epochs by at least a delta (Δ) of 0.0, training was terminated early to avoid unnecessary computations and overfitting. Most models achieved convergence well before the 50-epoch limit, reflecting the effectiveness of early stopping.

Initially, a learning rate of 1×10^{-4} was used across all experiments, and the ReduceLROnPlateau scheduler was employed that reduced the learning rate by a factor of 0.1 when the validation loss (for wildfire dataset) or training loss (for skin cancer dataset) plateaued for 5 consecutive epochs for a finer convergence. This adaptive learning rate strategy ensured that training continued effectively when loss stagnated, preventing overfitting or under-optimization. The AdamW optimizer was used as this particular optimizer is well-suited for tasks involving large datasets and all types of deep-learning architectures. The optimizers' default parameters of β_1=0.9 and β_2=0.999 were used, and the learning rate was set to 1×10^{-4} and weight decay was set as 1×10^{-4} to regularize the model and prevent overfitting. Class weights were dynamically computed using the compute_class_weight to address the very slight imbalance in data. The weighted Cross-Entropy Loss function penalized misclassifications of the minority class more heavily, thereby improving the model's ability to detect under-represented cases. A Batch size of 32 images was chosen for efficient utilization of GPU memory. AMP was not used during the 50-epoch runs due to its observed performance degradation for CNN-based models

in the earlier 5-epoch tests. To ensure stability and consistent results across all architectures, standard precision training was employed. Additionally, the available GPU resources were sufficient to handle the extended training without requiring the memory optimizations offered by AMP.

Data loading was optimized using PyTorch's DataLoader with multi-threaded workers (numworkers=2) to minimize I/O bottlenecks. Training progress including training loss, training accuracy, validation loss, and validation accuracy was logged for each epoch to ensure proper training progress. These data were used to plot learning curves. Additionally, during training, at the end of each epoch, GPU memory utilization, computational speed (Frames Per Second - FPS), and Learning rate adjustments by the scheduler were logged for assessment and analysis. Their graphs were also plotted at the end of the training.

After training, each model's performance was assessed using the test set that was not visible to the model before. Upon completion, the trained models were saved in .pth format for further evaluation and deployment. The path for saving the model weights was standardized as /kaggle/working/.

Plant Disease Classification (Multi-Class Classification)

The training setup for this classifier is relatively simple. Each model was initialized using pre-trained weights from the HuggingFace repository. The models were all pre-trained on the ImageNet-1k classification dataset. The final layers were then overwritten to match the number of classes, 15, and a softmax was applied later to compute metrics during evaluation.

For this multi-class classifier, we computed the top 5 accuracy alongside the top 1 accuracy, recall, precision, and f1 scores in line with the rest of our models. The macro average, one-versus-rest (OVR) auc-roc score was taken where the auc-roc score for each class is computed as if it were a binary classifier and then the average over all classes is computed. To compute the auc-roc score, the labels in each evaluation batch were one-hot-encoded as per the requirements of the scikit-learn library method that calculates the auc-roc score.

In addition, the images were pre-processed as specified by the model processor provided by the HuggingFace API. Broadly speaking, we applied our own transformations as detailed in the pre-processing section and only used the input dimensions, image mean, and deviation constants from the image processor. For models that did not provide a default image processor, the HuggingFace *AutoImageProcessor* was used which picks the most appropriate image processor with respect to the model string.

The HuggingFace trainer API was used in uniformity for training all models. We used the computed loss as our primary metric for model fit which is the default in the trainer. In addition, we passed the following hyper-parameters as training arguments to the trainer: we used a learning rate 3×10^{-6}, a weight decay of 0.02, 5 warm-up steps, a train batch size of 32 and evaluation batch of 16 per GPU. The models were evaluated at the end of each epoch and the best model with respect to

training loss was saved. We saved the model weights, as well as the optimizer state and scheduler for seamless checkpointing.

It should be noted that we did not change the default trainer optimizer, which is Pytorch's *AdamW* optimizer, and the default learning rate scheduler, which is set to *Linear*. Our trial runs yielded satisfactory results for all models. Subsequent grad norm plots were observed to be relatively stable for all models except one (resnet-50). For the sake of maintaining the control environment, we did not proceed with further experimentation.

Overall, we fine-tuned our pre-trained models over a maximum of 50 epochs with early stopping patience set to a constant value of 3. Once again, the HuggingFace early stopping module was used which is exposed to the trainer as a callback. We passed one other callback to the trainer which was used for logging data to Tensorboard. Both callbacks were called at the end of each epoch inline with our training and evaluation strategy.

6.2.2 Evaluation Metrics

This paper evaluates the effectiveness of different deep learning architectures in performing image classification tasks by accessing metrics on unseen test data. This ensures reliable performance in real-world applications. To comprehensively evaluate the performance of each model across all three datasets, we utilized the following set of evaluation metrics:

- **Accuracy:** The percentage of correctly categorized samples across all classes was measured using accuracy as a generic metric. However, to adequately evaluate each model's performance further metrics were needed to account for class imbalance.
- **Precision:** To assess the percentage of true positive predictions compared to all of the model's positive predictions, precision was used.
- **Recall:** The percentage of true positives that were accurately detected out of all actual positive samples was measured using recall, also known as sensitivity. This metric is critical in scenarios where missing a positive instance can have severe consequences, so we prioritize maximizing the recall score for our Flame and skin cancer dataset.
- **F1 Score:** To provide a balanced view of the model's performance, the harmonic mean of precision and recall was utilized.
- **AUC-ROC:** The models' ability to distinguish between classes across all thresholds was assessed using the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). Because it evaluates the trade-off between true positive and false positive rates, this statistic is especially pertinent for datasets with unbalanced distributions.

- **Top-5 Accuracy(only for PlantVillage Dataset):** This metric offers a more complex view of model performance. The models' ability to rank the correct class within the top 5 predictions was evaluated during plant disease classification as it is useful for multi-class classification tasks.

To guarantee a fair comparison, all models were assessed in the same way on all three test datasets. However, the top five accuracy scores were additionally computed on the PlantVillage Dataset.

Furthermore, in addition to evaluating the predictive performance of the models using the standard metrics described above, computation efficiency metrics were also calculated to assess the practical viability of each model. The following set of computational metrics were analyzed:

- **FLOPs (GFLOPs):** The number of floating-point operations required to process an image was computed to determine the computational complexity of the model.
- **Total Parameters and Trainable Parameters:** This reflects the size and learnable capacity of the model.
- **Inference Time:** Measured as the time taken for a batch and per image, providing insights into the latency of the model.
- **Throughput (FPS):** Indicates the number of images processed per second, important for high-performance applications.
- **GPU Memory Utilization:** Includes memory allocated and reserved during training and inference, useful for understanding hardware resource usage.

This holistic evaluation framework adds a practical perspective by also integrating real-world resource utilization. However, the thorough evaluation of each model highlights the trade-offs between model performance and resource efficiency. This study focuses on utilizing these metrics scores to identify models best suited for specific hardware or domains.

6.2.3 Computational Resources

To ensure fairness, the training of all models, except Vision Mamba variants, was carried out in a GPU-enabled environment with the NVIDIA T4 GPU. We were able to utilize this GPU through the Kaggle platform and it proved to be sufficient for training and evaluating our chosen deep-learning models. Kaggle's computational environment provided us with 30GB of RAM. The setup provided ample computational power and memory to handle the data sets and models efficiently.

However, Vision Mamba cannot yet run on platforms like Kaggle, due to their specialized architectures. Training these models required an Ubuntu-based operating system. We utilized the NVIDIA GeForce RTX 4090 for the Vision Mamba model

variants, known for its superior computational power and memory capacity. This experiment was conducted on the Z790 AORUS PRO X, having 64 GB of RAM. The specific details of the desktop configuration and the installation process for Vision Mamba are provided in the appendix for reproducibility. This additional setup highlights the need for robust hardware to accommodate certain advanced model architectures.

6.3 Knowledge Distillation

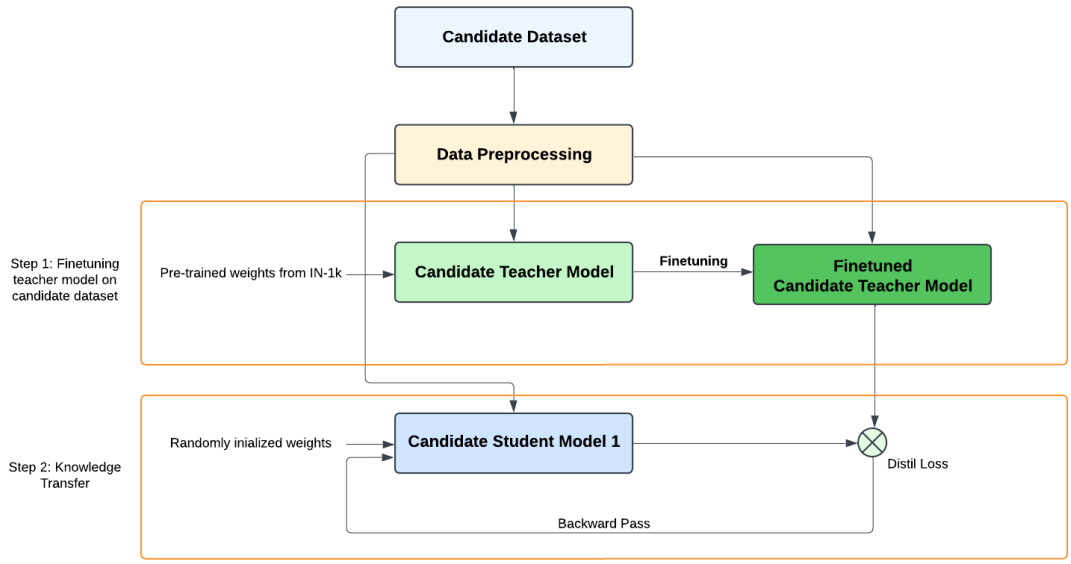


Figure 6.8: High-level Overview of our methodology for knowledge Transfer in SSMs

Figure 6.8 depicts a high-level overview of the overall process of knowledge distillation. The first step of distillation involves training the suitable teacher model. The teacher model can be pre-trained or it could be trained from randomly initialized weights, however, we have initialized the model using pre-trained weights from the ImageNet-1k dataset. Additionally, the number of logits in the output of the teacher model should match the number of output logits of the student model. The next step involves initializing a student model and passing the data into both the untrained student model as well as the pre-trained and finetuned teacher model. A distillation loss is computed according to a suitable scheme under the 'soft' or 'hard' distillation methods before performing a backward pass for each batch through the student model.

It is noteworthy that the teacher model weights, once trained and/or finetuned do not change. We only perform the backward pass on the student model. The only added computation here, besides training the teacher model separately, are the added steps of obtaining an output from the teacher model and computing the distillation loss. Unfortunately, this adds a considerable overhead in computation, especially for larger compute-intensive teacher models, which adds significant time

to the training process.

Furthermore, we perform a manual grid search to obtain optimal hyperparameter values for τ and α. We do so by initially overstating the importance of teacher model output and varying the softmax temperate starting from a value of 3 followed by 5 and then scaling up to 20 using a δ of 5. We varied the value of α with values of 0.5, 0.7, and 0.75 where a larger value of alpha overstates the importance of the teacher model soft targets. Finally, we performed equivalent 'hard' distillations using the same values for α which computes a Cross Entropy Loss with the raw logits from both the models.

Candidate Selection

Selecting suitable datasets and model candidates for knowledge distillation is a time-consuming process. This involves training models of varying sizes and architectures on different datasets and picking candidates from a pool of low-performing and a pool of high-performing candidates. Ideally, the low-performing candidate models will be smaller than those performing well on a particular dataset which would make it worthwhile to transfer knowledge between them.

Our main goal with this step in our research is to test the ability of SSM-based vision backbones to learn and transfer knowledge. So naturally, our immediate choice for a candidate student model is the ViM-Tiny model. This leaves the dataset and teacher model candidates. A suitable candidate dataset is one where models achieve varying degrees of success. Fortunately, this was the case with the FLAME wildfire detection and classification dataset. We did not proceed with distillation for other datasets as the models performed similarly on them and there wasn't much room for improvement. Table 6.1 briefly depicts the varying accuracies of a subset of models we trained on the FLAME wildfire dataset. A more comprehensive view of our results can in table 7.1.

Model	**No. of params (M)**	**Accuracy (%)**
Vim Base	96.80	77.42
Vim Tiny	**7.06**	70.60
ViT	85.80	79.18
EfficientNet B7	**66**	**88.84**

Table 6.1: A brief view of candidate model accuracies for the FLAME dataset

From the above pool of information and those in 7.1, we selected Vim Base and EfficientNet B7 as candidate teacher models while Vim Tiny was selected as the sole student model. This aligns with our research needs which is to test out the ability of SSM-based Vision Backbones to learn from other larger SSM-based backbones and also the ability to learn from models with very different architectures. All other training conditions for training the teacher and student models remain unchanged from the wildfire binary classifier as detailed in section 6.2.1. The results of our experiments are discussed in section 7.4.

Chapter 7

Results

The purpose of this chapter is to present and evaluate the performance of our chosen architectures from chapter 5 on the datasets of chapter 4, following the corresponding methodologies of chapter 6. The results of evaluating the deep-learning models in wildfire detection, plant disease classification, and skin cancer detection pave the way for the discussion on the strengths, limitations, and real-life applicability of these models.

The chapter is structured into four subsections, with the first three corresponding to a specific application. Each subsection provides key trends, insights, and notable observations from the experiments. Moreover, training performance is summed up to put each model's computing requirements in perspective, including convergence rates and GPU resource usage. This chapter offers a thorough assessment of these models in the context of environmental hazard detection, agricultural threat classification, and medical picture classification by contrasting conventional designs with more recent breakthroughs like SSMs. The final section discusses the results obtained from our experiments on *knowledge transfer in SSMs*.

7.1 Performance Comparison of Models for Wildfire Classification

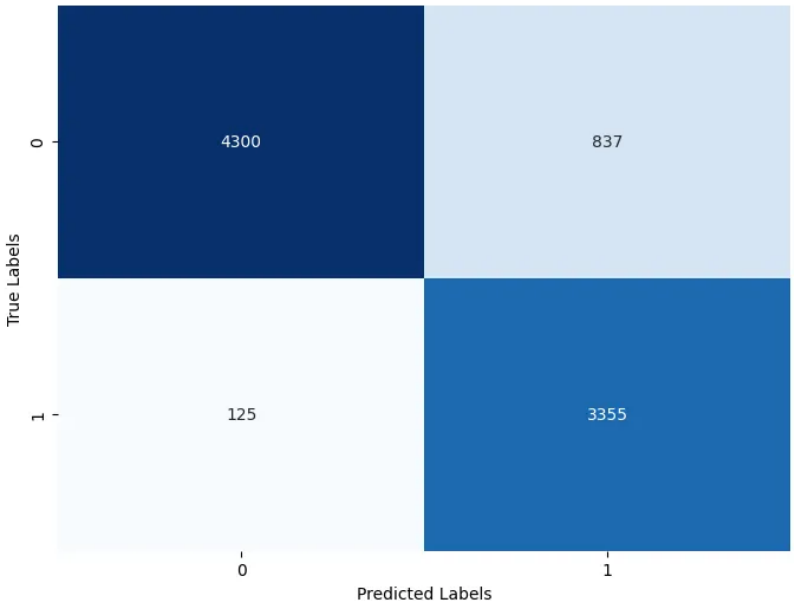
Model	Category	ACC (%)	PRE	REC	F1-Score	AUC-ROC	Conv. Epoch
DeiT-Base Distilled Patch16-224	Transformer-based	70.62	0.6126	0.7408	0.6707	0.7652	13
Swin Transformer	Transformer-based	81.8	0.8149	0.7109	0.7594	0.9103	14
PVT v2	Transformer-based	85.19	0.748	0.9552	0.839	0.9428	21
ViT_b_16	Transformer-based	79.18	0.8059	0.6382	0.7123	0.8512	9
Resnet18	CNN	85.34	0.7836	0.8802	0.8291	0.8901	11
ResNet-50	CNN	77.45	0.8485	0.5376	0.6582	0.9138	11
VGG16	CNN	81.87	0.8305	0.6925	0.7552	0.927	9
InceptionV3	CNN	81.93	0.7523	0.8239	0.7864	0.8955	10
Xception	CNN	86.78	0.7667	0.9670	0.8553	0.914	14
MobileNetV2	CNN	86.81	0.8393	0.8328	0.8360	0.9372	13
ConvNext Base	Modern CNN	83.99	0.7796	0.8414	0.8093	0.9108	9
ConvNextv2	Modern CNN	86.35	0.8069	0.8704	0.8374	0.9306	9
EfficientNet B0	CNN	80.72	0.7080	0.8897	0.7885	0.9184	12
EfficientNet B7	CNN	88.84	0.8003	0.9641	0.8746	0.9506	15
VisionMamba-tiny-patch16	SSMs	70.6	0.6699	0.5339	0.5942	0.7766	6
VisionMamba-small-patch16	SSMs	68.06	0.6732	0.4042	0.5051	0.7405	6
VisionMamba-base-patch16	SSMs	77.42	0.7253	0.7082	0.7166	0.8395	6

Table 7.1: Performance comparison of various models for FLAME dataset

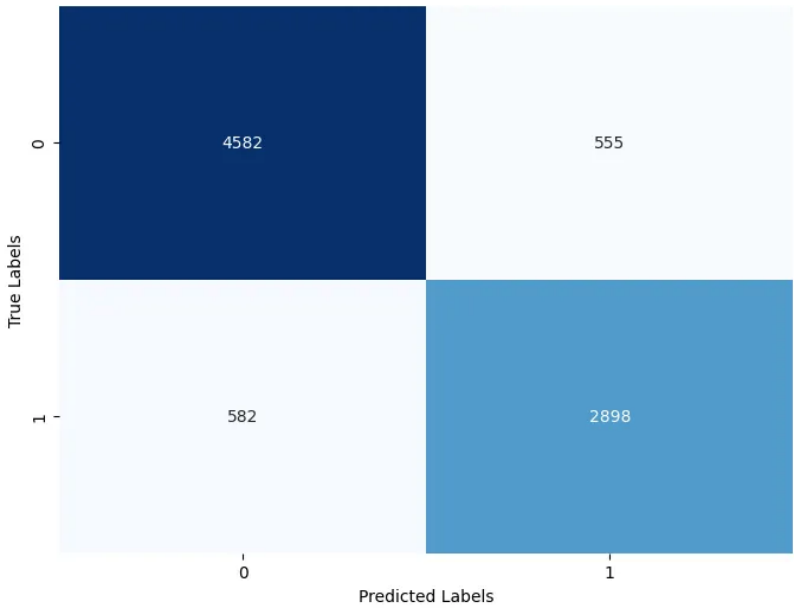
Table 7.1 presents the performance comparison of the deep-learning models for binary classification of wildfire using evaluation metrics such as accuracy (ACC), precision (PRE), recall (REC), F1-score, and AUC-ROC. Additionally, convergence epochs (Conv. Epoch) highlight the number of epochs each model required to stabilize during training, when early stopping was triggered.

Overall Best Performers

The EfficientNet B7, which is a CNN model achieved the highest overall performance with an accuracy of 88.84%, the highest Recall and F1-score of 0.9641 and 0.8746 respectively, and AUC-ROC of 0.9506. We prioritized the Recall score for this task as classifying Fire as No_Fire could have severe impacts, and EfficienNetB7 excelled in this. The MobileNetV2, another CNN model also stoop out in terms of performance, achieving a high accuracy of 86.81% and AUC-ROC of 0.9372 with a significantly smaller parameter size, showcasing its efficiency. All its other scores were above 80% as well, making it noteworthy.



(a) Confusion Matrix of EfficientNet B7



(b) Confusion Matrix of MobileNetV2

Figure 7.1: Comparing Confusion matrices for two CNN models with significantly different parameter counts

Fig 7.1 compares the confusion matrices of the two best-performing models on the wildfire dataset. Here "0" represents fire and "1" represents non-wildfire images. The more computationally efficient and smaller model, MobileNetV2 detects more wildfires (higher TP) than EfficientNet B7 but at the expense of more false positives (FP). EfficientNet B7 sacrifices some wildfire detection to maintain a better balance between false positives and negatives.

Transformer-Based Models

Among all transformer models, PVT v2 achieved the best performance with an accuracy of 85.19% and an AUC-ROC of 0.9428. Its high recall score of 0.9552 indicates its effectiveness in recognizing fire cases with high accuracy while maintaining a high precision of 0.748. This model can minimize false negatives and false positives quite well as shown in the confusion matrix 7.2, thereby being effective at detecting wildfires correctly and also avoiding false alarms.

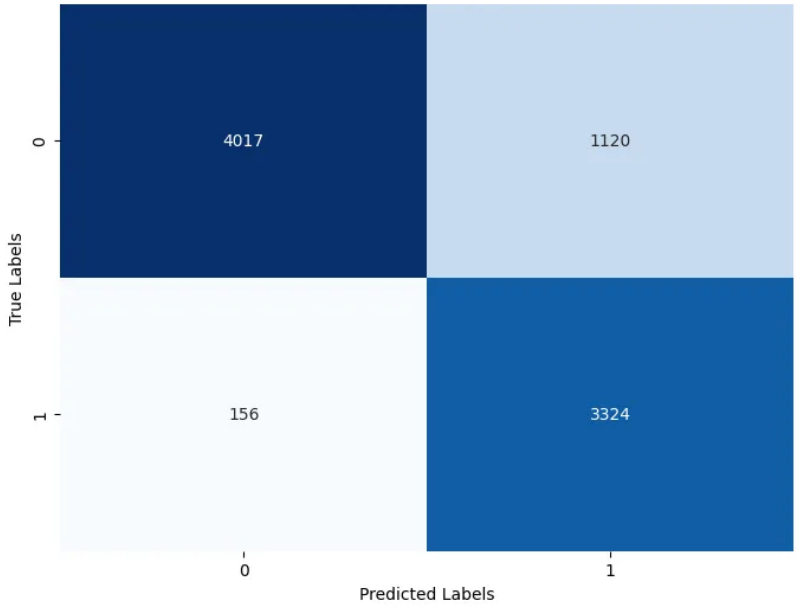


Figure 7.2: Confusion Matrix of PVT v2

CNN Models

As seen in the table 7.1, the CNN models performed extremely well in this binary classification task. Almost all CNN models achieved an accuracy greater than 80.%; the only exception being Resnet-50 with its accuracy of 77.45%, which is quite decent as well. The CNN models were also among the best performers for this task as mentioned above. Furthermore, the Xception model performed remarkably well with a top1 accuracy of 86.78%, a high recall of 0.9670, and an F1-score of 0.8533. Resnet-18 also demonstrated a balanced performance, achieving an accuracy of 85.34%, and a high recall of 0.8802.

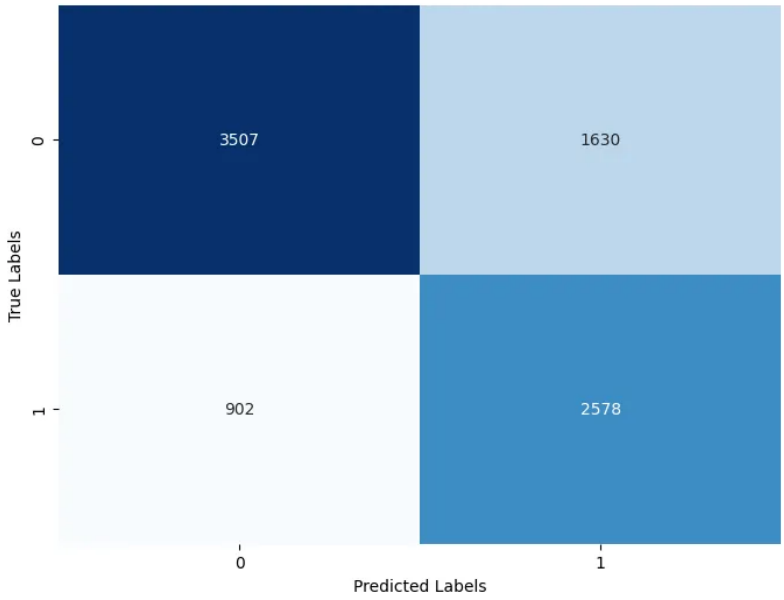
SSMs

VisionMamba-base-patch16 performed the best among SSMs, with an Accuracy of 77.42%, F1-score of 0.7166, and AUC-ROC of 0.8335. However, VisionMamba vari-

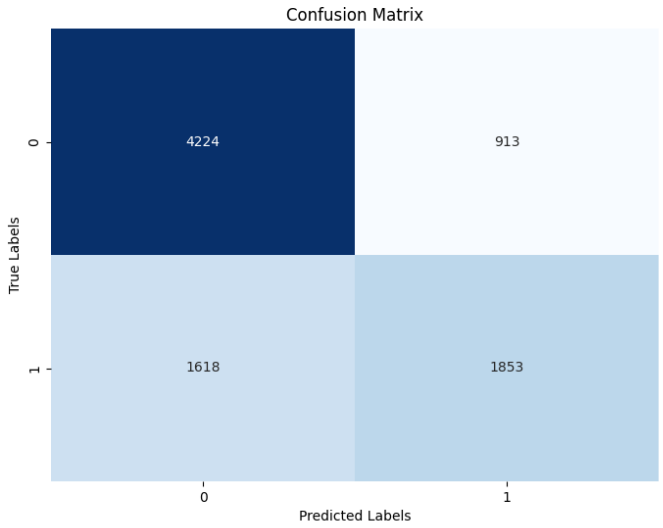
ants showed lower overall performance compared to transformer-based and CNN architectures, especially in precision and recall.

Efficiency and Convergence

SSM models like VisionMamba converged rapidly, requiring only 6 epochs to stabilize, making them computationally efficient compared to all other architectures. PVT v2 required the most number of epochs (21 epochs) to train fully. EfficientNet B7, while achieving the best results, required the second-longest training time (15 epochs). It should be highlighted that VisionMamba-tiny-patch16 achieved parallel accuracy (70.6%) to DeiT-Base Distilled Patch16-224 (70.62%) while having a very low parameter count and training time.



(a) Confusion Matrix of Deit-Base



(b) Confusion Matrix of VisionMamba-tiny-patch16

Figure 7.3: Comparing Confusion matrices for Transformer-based and SSM-based Models

The confusion matrices in 7.3 show that the SSM-based Vision Mamba has better precision but lower recall than the attention-based DeiT model. Therefore, VisionMamba-tiny is better at correctly identifying wildfires but might miss more wildfires. However, DeiT would raise more false alarms.

Result Overview

The results demonstrated that EfficientNet B7 is the top-performing model for wildfire classification, delivering the best balance of accuracy, F1-score, and AUC-ROC. CNN models showed strong consistent performance in this task, even the models with smaller parameter counts (such as MobileNetV2 and Resnet18), thereby showing computational efficiency. Transformer-based models also performed competitively, but they generally took longer to train and also had high parameter counts. Meanwhile, SSM-based VisionMamba models offered rapid convergence but fell short in terms of overall performance metrics compared to other architectures.

7.2 Performance Comparison of Models for Plant Disease Classification

Model	Category	ACC (%)	Top5 ACC (%)	PRE	REC	F1-Score	AUC-ROC	Conv. Epoch
DeiT-Base Distilled Patch16-224	Transformer-based	99.38	100	0.9939	0.9938	0.9938	0.9938	22
Swin Transformer	Transformer-based	99.45	99.97	0.9946	0.9942	0.9944	0.9998	25
PVT v2	Transformer-based	99.64	99.97	0.0.9964	0.9971	0.9967	0.9999	45
ViT_b_16	Transformer-based	99.77	99.77	0.9973	0.9976	0.9975	1	46
Resnet18	CNN	99.10	99.97	0.9896	0.9911	0.9903	0.0000	32
ResNet-50	CNN	82.72	97.80	0.7332	0.6974	0.6860	0.9912	50
VGG16	CNN	97.71	99.70	0.9765	0.9761	0.9759	0.9997	11
InceptionV3	CNN	99.26	99.70	0.9921	0.9932	0.9926	0.999	32
Xception	CNN	99.52	99.97	0.9951	0.9954	0.9953	0.9998	38
MobileNetV2	CNN	94.35	99.87	0.9444	0.9435	0.9432	0.9555	26
ConvNext Base	Modern CNN	99.77	100	0.9982	0.9973	0.9977	1	27
ConvNextv2	Modern CNN	99.84	99.97	0.9987	0.9986	0.9986	0.9999	15
EfficientNet B0	CNN	90.31	99.77	0.8353	0.8361	0.8301	0.9952	12
EfficientNet B7	CNN	91.67	99.45	0.9283	0.9167	0.9173	0.9967	10
VisionMamba-tiny-patch16	SSMs	98.71	99.97	0.9872	0.9871	0.9871	0.9999	33
VisionMamba-small-patch16	SSMs	99.45	100	0.9946	0.9945	0.9945	0.9999	22
VisionMamba-base-patch16	SSMs	99.65	100	0.9965	0.9965	0.9965	0.9999	14

Table 7.2: Performance comparison of various models for PlantVillage Dataset

Table 7.2 lists the performance metrics of all the models we trained in the controlled environment for the plant leaf disease classification task which is a multi-class classification task. Please note that our primary metric for tracking model performance during training was the training loss. Additionally, the AUC-ROC score is the macro-average of the one-versus-rest (ovr) score for each of the fifteen classes in the data set.

Overall Best Performers

Almost all models performed exceptionally well at this multi-class classification task. However, in terms of the Top1Accuracy score, ViT and large ConvNext models outperformed other models by a slight margin. However, DeiT and the two larger VisionMamba models resulted in perfect scores in terms of Top5Accuracy while all other models except one scored more than 99. This indicates little delta between all the various models as all of them fit very well with the training data and perform well on the test set.

Transformer-based Models

Transformer-base DeiT, ViT, Swin Transformer, and PVTv2 models, all scored more than 99 in the Top1 and Top5 accuracy metrics. In addition, their precision, recall, f-1, and auc-roc scores were also more than 0.99 which leaves little room for improvement. The models trained over more epochs than the wildfire dataset and we observed a steady decline in training and evaluation loss. Figure 7.4 is the final confusion matrix for the best-performing transformer-based model which is ViT.

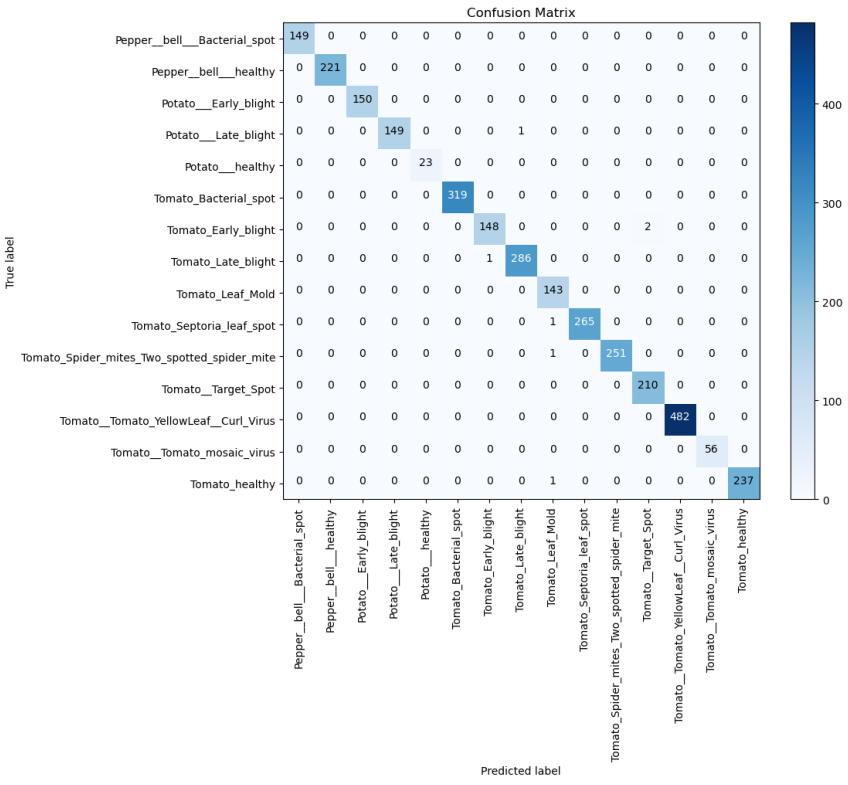


Figure 7.4: Confusion Matrix of ViT

CNN Models

ConvNext Base and ConvNextv2 were the clear frontrunners in terms of performance. ConvNext Base being the larger model took a few more epochs to converge while ConvNextv2 converged earlier and performed marginally better. Interestingly, the Resnet-50 model failed to converge within the allotted 50 epochs and it turned out to be the worst-performing model for this dataset. However, Resnet-18 converged within 32 epochs and performed on par with other models trained on this dataset. In figure 7.5, we observed an erratic grad norm but a steady reduction in training loss. Among the other CNN-based models, VGG16 was the fastest to converge while InceptionV3 took the longest. Furthermore, MobileNet, which is a small model designed to work in resource-constrained environments, also produced a Top1Accuracy score of 90+ which is on par with much larger models. Figure 7.6 is the confusion matrix for the ConvNext base model.

Figure 7.5: Grad Norm and Loss for Resnet-50 over 50 epochs

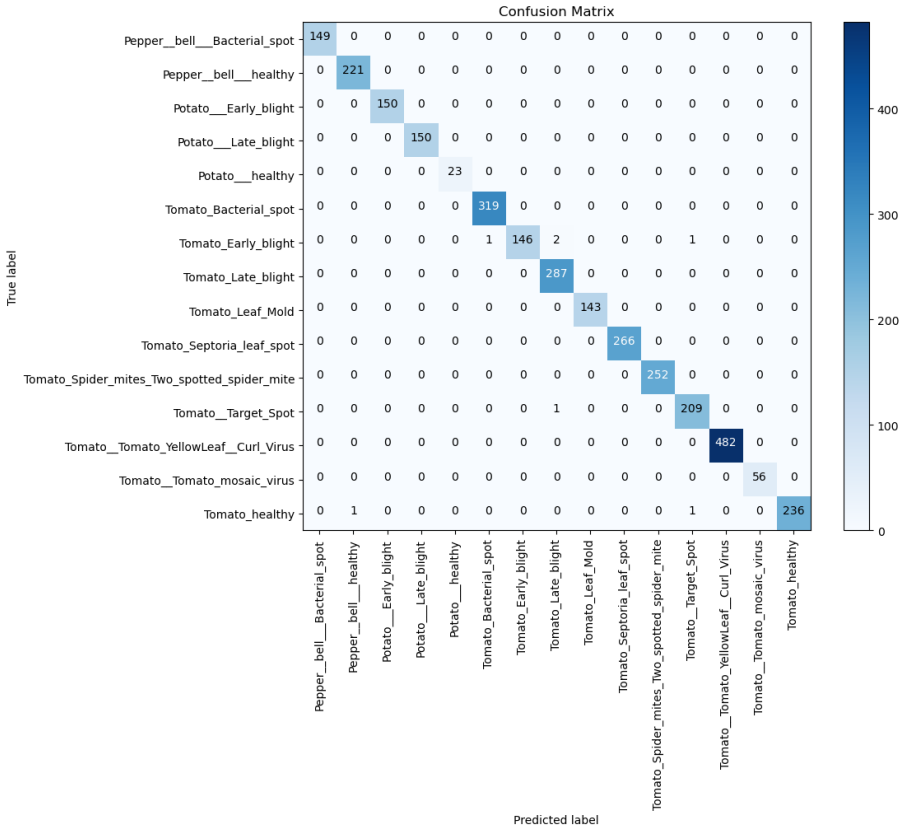


Figure 7.6: Confusion Matrix of ConvNext Base

SSMs

It is encouraging to see that all three models with SSM-based vision backbones performed excellently and on par with other CNN and transformer-based models. Moreover, the smallest models of the three, ViM-Tiny, performed better than comparatively smaller CNN-based models scoring an impressive Top1Accuracy score of 98.71% which is higher than MobileNet. In addition, the base ViM model achieved similar performance to the base ViT model indicating that ViM performs on par with ViT without the need for attention in multi-class classification tasks. Figure 7.7 is the confusion matrix for ViM base which is the best-performing SSM model for the plant village dataset.

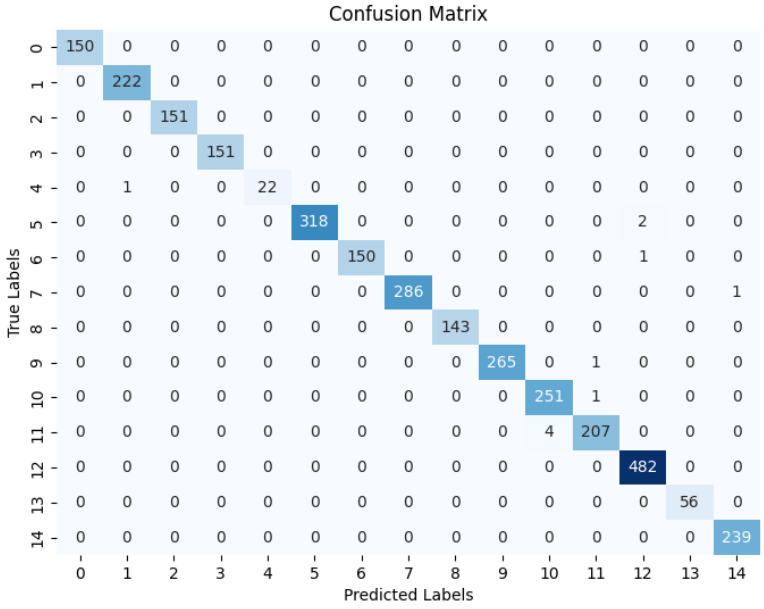


Figure 7.7: Confusion Matrix for ViM Base model

Efficiency and Convergence

In general, all models took longer to train and converged after significantly more epochs when compared to the wildfire dataset. One model, ResNet-50, failed to converge within the allotted 50 epochs due to unstable gradients. SSM and transformer-based models took a little longer to train but exhibited marginally better performance.

Result Overview

Overall, all models that converged within 50 epochs demonstrated good performance on common image classification metrics. However, this left little room to explore avenues for improvement, particularly, because the delta between the performance of smaller models compared to larger models was fairly small. In addition, all three ViM models scored very similar accuracy scores. This meant that our goal of transferring knowledge from larger to smaller models would have been worthwhile.

7.3 Performance Comparison of Models for Skin Cancer Classification

Model	Category	ACC (%)	PRE	REC	F1-Score	AUC-ROC	Conv. Epoch
DeiT-Base Distilled Patch16-224	Transformer-based	89.85	0.8585	0.93	0.8928	0.9751	41
Swin Transformer	Transformer-based	90.15	0.8778	0.91	0.8936	0.9702	50
PVT v2	Transformer-based	90.76	0.8918	0.9067	0.8992	0.9716	27
ViT_b_16	Transformer-based	90.91	0.8822	0.9233	0.9023	0.9694	20
Resnet18	CNN	90.91	0.8797	0.9267	0.9026	0.972	35
ResNet-50	CNN	90.15	0.9094	0.87	0.8893	0.967	38
VGG16	CNN	88.79	0.8951	0.8533	0.8737	0.9646	50
InceptionV3	CNN	89.09	0.8826	0.8767	0.8796	0.9652	20
Xception	CNN	91.67	0.9359	0.8767	0.9053	0.9777	36
MobileNetV2	CNN	91.36	0.8857	0.93	0.9073	0.9752	40
ConvNext Base	Modern CNN	92.42	0.9252	0.9067	0.9158	0.9781	29
ConvNextv2	Modern CNN	83.79	0.8083	0.8433	0.8254	0.9273	50
EfficientNet B0	CNN	92.12	0.8899	0.9433	0.9159	0.9777	36
EfficientNet B7	CNN	90.91	0.9	0.9	0.9	0.9659	24
VisionMamba-tiny-patch16	SSMs	87.03	0.7855	0.9679	0.8672	0.9566	16
VisionMamba-small-patch16	SSMs	89.22	0.8226	0.9607	0.8863	0.9707	16
VisionMamba-base-patch16	SSMs	90.16	0.8534	0.9357	0.8927	0.9621	28

Table 7.3: Performance comparison of models for Skin Cancer Dataset

Table 7.3 presents the performance comparison of the deep-learning models for binary classification of skin cancer diagnosis using the same evaluation metrics as the wildfire classification task. The convergence epochs (Conv. Epoch) are also recorded in the table. The skin cancer classification task reveals subtle differences in model performance across various architectures.

Overall Best Performers

The ConvNext Base model achieved the highest overall performance with an Accuracy of 92.42%, Precision of 0.9252, Recall of 0.9067, F1-Score of 0.9158, and AUC-ROC of 0.9781. The EfficientNet B0 model follows closely achieving nearly identical results with an accuracy of 92.12%, Precision of 0.8889, Recall of 0.9433, F1-Score of 0.9159, and AUC-ROC of 0.9777. The Receiver Operating character(ROC) curve 7.8 shows that the model can distinguish between the two classes extremely well. These results highlight the models' ability to produce accurate predictions while minimizing false negatives. The ConvNext base model converges in 29 epochs while the EfficientNet B0 requires an additional 7 epochs (36 epochs in total). However, EfficientNet B0 also has a significantly lower parameter count, which creates scope for comparison of their computational efficiencies and their deployability in real-world applications in the medical domain.

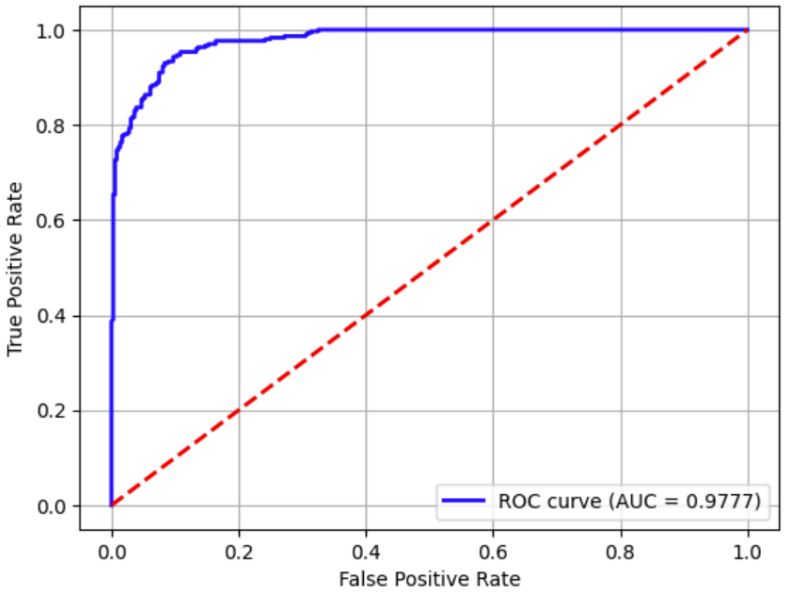


Figure 7.8: ROC Curve for EfficientNet B0

Transformer-Based Models

Among transformer-based models, ViT b_16 demonstrated the strongest performance with an accuracy of 90.91%, F1-Score of 0.9023, and AUC-ROC of 0.9694. It effectively balances precision (0.8822) and recall (0.9233), while requiring 20 epochs for convergence. This convergence time is notably less than most of the CNN models and Transformer-based models as well. PVT v2 and Swin Transformer also achieved competitive performances with their accuracies being 90.76% and 90.15% respectively. However, these transformer-based models required longer convergence times (with one model reaching 50 epochs) compared to SSM-based models, limiting their efficiency for large-scale training scenarios.

CNN Models

Similar to the Flame dataset, the CNNs showed consistently good results for this binary classification task as well. Both the best-performing models are CNN-

based. Furthermore, MobileNetV2 achieved high accuracy (91.36%), Recall(0.93), and strong F1-score (0.9073), making it a lightweight yet effective option for deployment in resource-constrained environments. Xception is also mentionable due to its excellence in accuracy(91.67%), precision (0.9359), and AUC-ROC (0.9777), demonstrating its suitability for applications prioritizing precision over speed. However, it must be noted that all the CNN models required a greater number of epochs to complete training compared to the SSM-based models making them computationally less efficient.

SSMs

State Space Models (SSMs) showed competitive performance to many of the other SOTA models. The VisionMamba-base-patch16 was leading among SSMs with an accuracy of 90.16%, F1-Score of 0.8927, and AUC-ROC of 0.9621. While their overall metrics were slightly lower than ConvNext Base and EfficientNet B0, SSMs were notably efficient in convergence. This efficiency makes them appealing for scenarios where rapid training is critical.

Efficiency and Convergence

For this dataset, most of the models took a significantly larger number of epochs than the other two datasets discussed above, with many of the models training for the complete 50-epoch cycle. Among all the architectures, SSMs took the least number of epochs to train (16-28 epochs), achieving a quick convergence, even though Vision Mamba models slightly lagged in performance. The accuracy/training graphs, 7.10 and 7.9, for this binary classification task, help to visualize this difference in convergence. While the other models kept on training for (30-50) steps/epochs, the vision mamba model variants converged at a much lower epoch count.

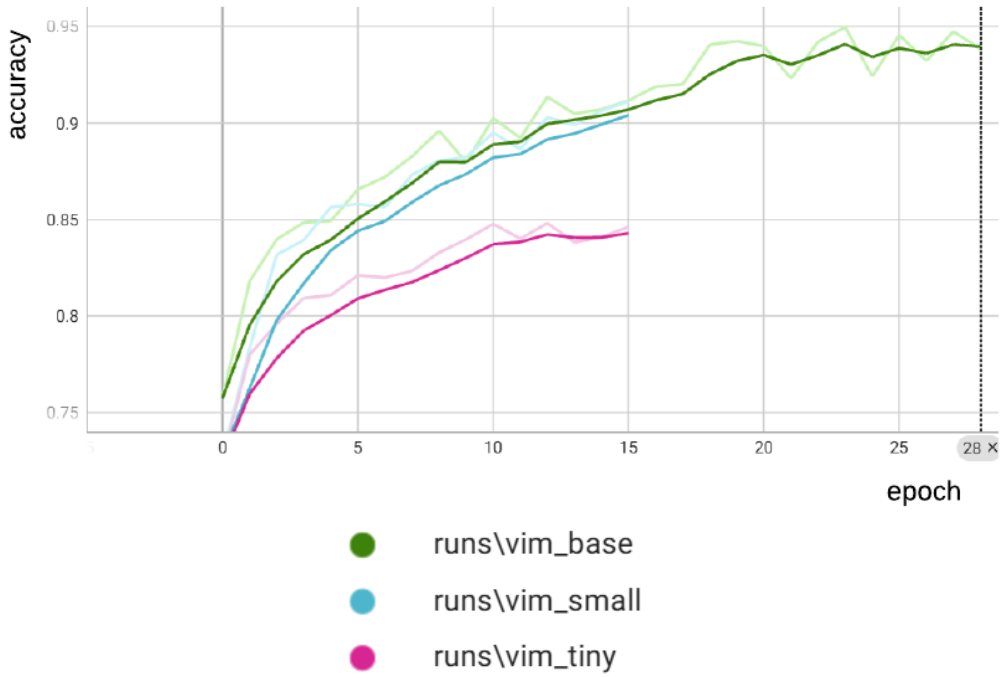


Figure 7.9: Accuracy/Train Graph for SSM-based models

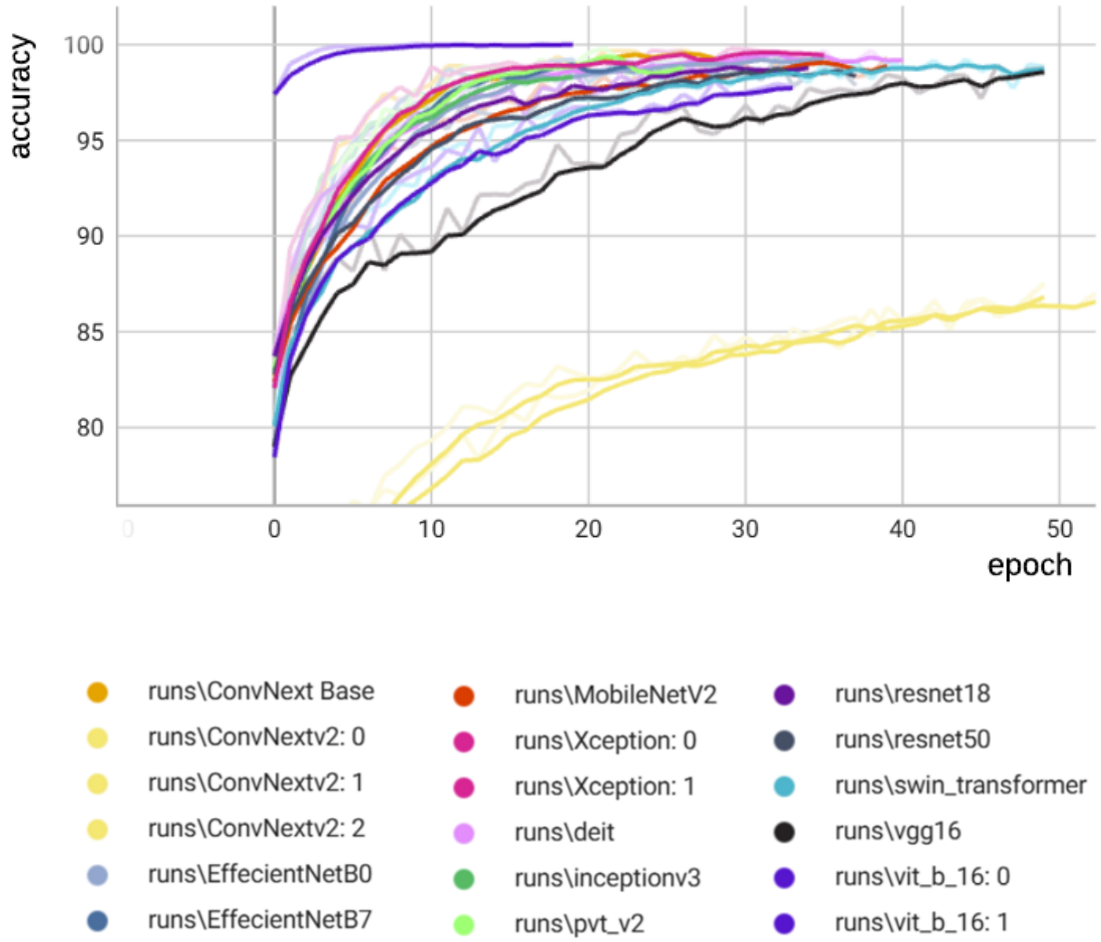


Figure 7.10: Accuracy/Train Graph for Transformer-Based and CNN models

Result Overview

The skin cancer classification task highlights ConvNext Base as the best-performing model due to its exceptional accuracy, F1-score, and AUC-ROC, paired with efficient convergence. EfficientNet B0 closely follows, excelling in AUC-ROC and convergence speed. It can be deduced that the EfficientNet models show exceptional performance in binary classification tasks as seen from the results from the Flame and Skin Cancer datasets. In Efficiency and Convergence, Vision Mamba took the lead with the fastest convergence rate. SSMs' performance is on par with the Transformer-based models even for this dataset, suggesting Vision Mamba as a promising and efficient replacement to the Transformer-based models.

7.4 Performance of distilled ViM Tiny

Our methodology for picking candidate models and datasets is detailed in section 6.3. We performed distillation on the ViM Tiny 7.06M parameter model using the ViM base and EfficientNet-B7 models as teacher models. We trained the student model by varying the α, softmax temperature, τ, and alternating between 'soft' and 'hard' distillation. Our final experimental run included two previously pre-trained and fine-tuned teacher and student models where the ViM Tiny student model scored accuracy metrics on par with the much larger EfficientNet-B7 model. We performed all distillation experiments on the IEEE FLAME wildfire detection dataset.

Experiment 1: Vim Base to Vim Tiny

We perform distillation from a Vim Base model which was previously pre-trained on the ImageNet-1k classification dataset and fine-tuned on the FLAME wildfire dataset. The student model, Vim-Tiny had its weights randomly initialized with no prior training. Figure 7.11 displays the output of the first run.

Distillation Type	Soft
Temperature, τ	3
Loss Ratio, α	0.75
Train Time	1:44:17
Epochs	20
Accuracy	67.30

(a) First Run

Distillation Type	Soft
Temperature, τ	5
Loss Ratio, α	0.5
Train Time	1:43:34
Epochs	20
Accuracy	67.2

(b) Second Run

Distillation Type	Soft
Temperature, τ	5
Loss Ratio, α	0.75
Train Time	1:43:33
Epochs	20
Accuracy	66.2

(c) Third Run

Distillation Type	Hard
Loss Ratio, α	0.75
Train Time	1:38:04
Epochs	18
Accuracy	69.92

(d) Fourth Run

Distillation Type	Soft
Temperature, τ	20
Loss Ratio, α	0.75
Train Time	1:43:33
Epochs	22
Accuracy	66.91

(e) Fifth Run

Figure 7.11: Experiment 1 Results

Experiment 2: EfficientNet-B7 to ViM Tiny

EfficientNet-B7 was the best-performing model on the FLAME dataset. We perform the first run with the same conditions as experiment 1. However, for the second

run, we use a ViM Tiny model pre-trained on ImageNet-1k to achieve significantly improved performance. The results are recorded in figure 7.12.

Distillation Type	Hard
Student Model	Random weights
Loss Ratio, α	0.75
Train Time	1:40:05
Epochs	22
Accuracy	70.31

(a) First Run

Distillation Type	Hard
Student Model	Pre-trained IN1K
Loss Ratio, α	0.75
Train Time	1:09:21
Epochs	15
Accuracy	**85.32**

(b) Second Run

Figure 7.12: Experiment 2 Results

In summary, our first experiment with distillation yielded strong base models that performed similarly to their pre-trained and fine-tuned counterparts which as access to a much larger pool of image data. In addition, we observed sufficient knowledge transfer between SSM. Our subsequent experiment's first run also yielded promising results with 'hard' distillation. Moreover, in the next run, we were able to make significant improvements to the ViM Tiny model on the FLAME wildfire dataset by improving the accuracy of the binary classifier from 70.6% to 85.32%. This is a 21% improvement over our previous ViM Tiny model for this binary classification task. In addition, the accuracy of the distilled model is very close to that of our best-performing model for this dataset, which was the EfficientNet-B7 model, which scored 88.84% on the top 1 accuracy metric. However, the EfficientNet-B7 model is 9× larger in terms of parameter count when compared to the Vim Tiny model which is a very impressive outcome for any distilled model.

Chapter 8

Discussion

8.1 Analysis of Results

In this study, we train and evaluate various popular vision models for binary and multi-class classification tasks across three different domains. Our intentions with this work were to study and analyze the capabilities of the new generation of SSMs to act as efficient vision backbones and compare them on multi-domain image classification tasks against existing models. Broadly speaking, we applied three classes of models to our datasets: CNNs, transformers with different variants of attention, and SSM with zero percent attention. All experiments were performed under the control environment described in previous chapters.

Wildfire Detection

The IEEE FLAME wildfire detection dataset comes with previously defined separate train and test sets. We observed very early on that this dataset contained frame-by-frame images of wildfires. This made our models prone to over-fitting on the training set and under-performing on the test set. This posed a unique challenge for our models, the models had to generalize over this data while not overfitting to the training set. Consequently, the best-performing models could be considered to be able to learn the general features of the images well enough to perform on a wide range of settings. We observed that most models were quick to converge and trigger our Early Stopping mechanism but did not learn the general features well enough to classify with a high degree of confidence.

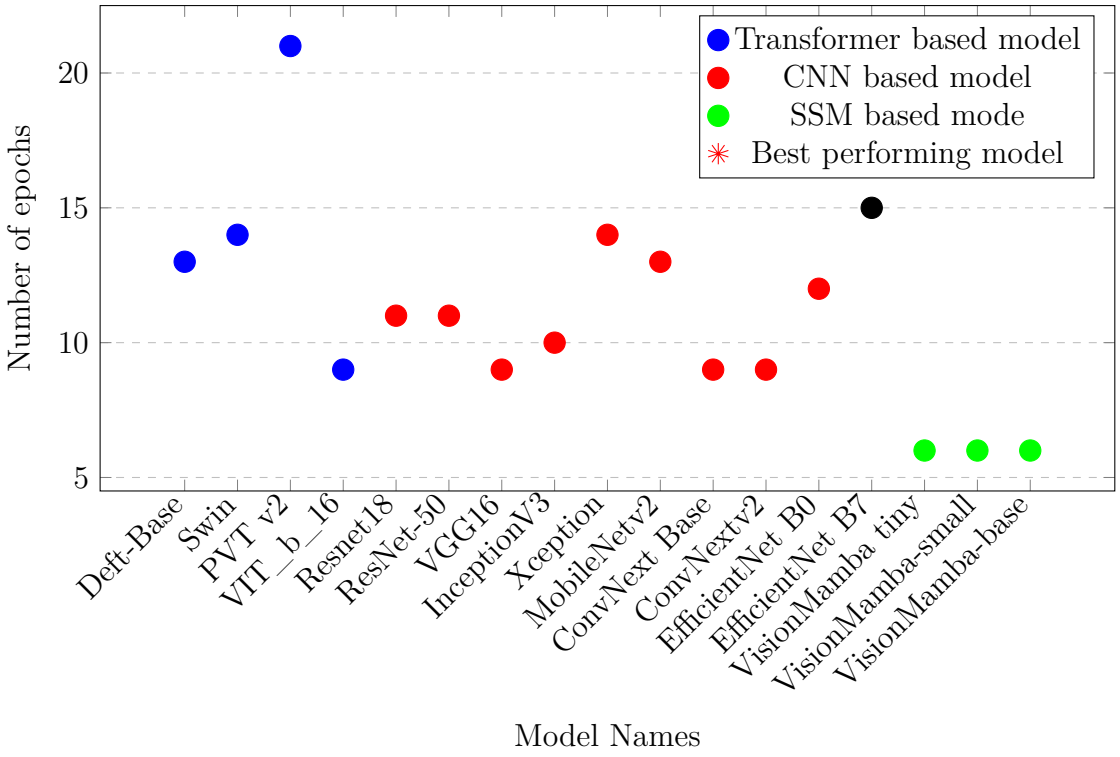


Figure 8.1: Number of epochs taken to trigger early stopping

Figure 8.1 plots the number of epochs taken by each model to converge for the FLAME wildfire dataset with early stopping patience set to 5. Models that converged within 10 epochs generally performed worse than those that continued beyond the 10 epoch marks. This corroborates our hypothesis that the dataset is prone to over-fitting and is therefore a good benchmark to compare the model's ability to learn generalizable information from the training set.

Notably, CNN models performed comparatively better than sequential models for this dataset due to their inherent advantage with inductive bias. The resolution of images in this dataset was also not high enough for sequence modeling. Overall, we observed that models that process images as patch sequences like ViT and ViM could not perform generalization as well as CNN-based models and were more prone to over-fitting. Unsurprisingly, larger models performed better than smaller models due to their ability to learn more fine-grained features from the dataset.

Plant Leaf Disease Classification

The leaf disease classification is a multi-class classification task. Almost all models performed exceptionally well on this dataset barring one, ResNet-50, which failed to converge within the allotted 50 epochs. Figure 7.5 is the trace of the training grad norm for ResNet-50 where we observe sporadic and unexpected changes. However, there was a steady reduction in loss, and training the model over 50 epochs may have resulted in better performance.

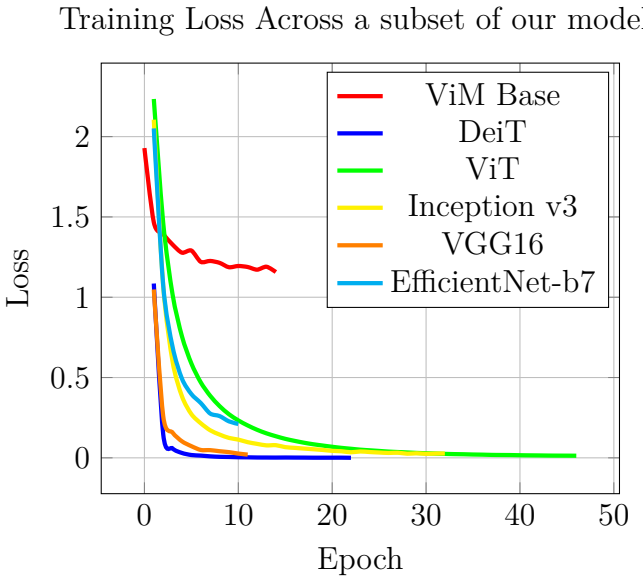


Figure 8.1 plots the traces of training loss for comparably sized ViT, ViM Base, DeiT, InceptionV3, VGG16, and EfficientNet-B7 models. The first three sequential models score 0.9977, 0.9965, and 0.9938 respectively on the Top1Accuracy score. This is an excellent performance and leaves little room for improvement. From the loss trace, we can however observe that ViM Base was the first to converge while ViT took the longest. The CNN-based models all performed well and scored more than 98. Early stopping was triggered for all models with the patience value set to 3.

To conclude our discussion about this multi-class classification task, we observed no real point of discrimination among the CNN, transformer, or SSM-based models. All models performed exceptionally well on the test set and in general ran for a greater number of epochs when compared to the FLAME wildfire detection dataset.

Skin Cancer Detection

EfficientNet, like wildfire detection, emerged as the best-performing model for skin cancer classification, making it the ideal instructor model for knowledge distillation. Its capacity to capture fine-grained visual features efficiently makes it ideal for medical imaging applications. Because of its better precision, it provides the finest foundation for enhancing smaller models such as Mamba via distillation.

One significant finding was that Mamba models converged the fastest, highlighting their strength in rapid learning. This means that they can swiftly adapt to new datasets, making them useful in applications that require frequent updates or continuous learning. Despite this, the total accuracy differences between architectures were minimal, implying that CNNs, Transformers, and SSM-based models performed similarly. Skin cancer categorization is based on localized patterns rather than long-range dependencies, which could explain why self-attention models could not outperform CNNs.

Mamba was by far the most computationally efficient model, even if it did not perform as well in terms of accuracy. Its small memory footprint and high inference speed make it an excellent choice for low-cost edge deployment in healthcare environments. These findings emphasize the balance of accuracy and efficiency, stressing Mamba's strength as deployability and computational cost-effectiveness rather than pure performance supremacy.

8.2 Computational Efficiency

Model	Category	FLOPs (GFLOPs)	PARAMS (M)	Inference Time (ms)	Throughput (FPS)	GPU Reserved (GB)	GPU Allocated (GB)
DeiT-Base Distilled Patch16-224	Transformer-based	16.87	85.80	0.0083	120.83	4.84	1.30
Swin Transformer	Transformer-based	4.51	27.52	0.0113	88.50	3.84	0.44
PVT v2	Transformer-based	3.90	24.85	0.0126	79.10	5.54	0.40
ViT_b_16	Transformer-based	16.87	85.80	0.0083	120.74	5.08	1.30
Resnet18	CNN	1.82	11.18	0.0024	423.76	1.18	0.18
ResNet-50	CNN	4.11	23.51	0.0060	165.85	7.57	0.38
VGG16	CNN	15.47	134.27	0.0014	701.20	6.69	2.02
InceptionV3	CNN	5.73	25.12	0.0121	82.48	4.02	0.36
Xception	CNN	4.57	20.81	0.0058	171.17	6.95	0.33
MobileNetV2	CNN	0.31	2.22	0.0055	183.03	2.77	0.05
ConvNext Base	Modern CNN	15.38	87.57	0.0166	60.31	8.82	1.35
ConvNextv2	Modern CNN	4.47	27.82	0.0091	110.22	4.23	0.44
EfficientNet B0	EfficientNet	0.40	4.01	0.0086	116.35	3.18	0.08
EfficientNet B7	EfficientNet	5.26	63.79	0.0906	11.04	9.82	1.00
VisionMamba-tiny-patch16	SSMs	-	7.06	0.1889	5.29	0.51	0.03
VisionMamba-small-patch16	SSMs	-	25.4	0.1857	5.39	0.36	0.11
VisionMamba-base-patch16	SSMs	-	96.8	0.1494	0.69	0.66	0.37

Table 8.1: Comparison of Computational Efficiency

In table 8.1, CNN models like Resnet18 and EfficientNetB0 show significantly lower FLOPs (40 times smaller) and parameter counts compared to Transformer-based models like DeiT-Base and ViT-b_16, demonstrating their higher computational efficiency. Furthermore, the CNN models in the discussion, Resnet18 and EfficientNetB0 have low inference times (0.0024 ms and 0.0086 ms), while having the

highest throughputs among all models (423.76, 116.35), making them ideal for real-time applications. Transformer-based models, including ViT and Swin Transformer, have higher inference times (around 0.0083–0.0126 ms) and lower throughput, which might limit their utility in scenarios requiring rapid predictions. As we have seen in Chapter 7, these lightweight CNN models did not trade off accuracy for efficiency as they have performed consistently better than the Transformer-based models.

The VisionMamba-tiny also has a very small parameter count(7.06), being one of the most lightweight models in our selection. This model achieved a similar accuracy to DeiT, which has a high parameter count(85.80), for the wildfire classification task. Furthermore, VisionMamba-base outperformed DeiT in the same task and achieved parallel results in skin cancer detection, confirming the paper [78]'s theoretical assertions that SSM-based models can match or even outperform Transformer-based models for visual representation tasks when properly modified. This demonstrates SSM-based models' capability as a lightweight and efficient alternative to Transformer-based architectures.

Vision Mamba Tiny uses the least GPU memory consuming only 0.03–0.37 GB of GPU-reserved memory. This makes SSM-based models suitable for tasks with constrained computational resources. As shown in paper [78], SSM models such as Mamba excel at computation and memory efficiency. These findings were echoed in our experiments, where Mamba models required much fewer computational resources while maintaining equivalent accuracy, demonstrating the promise of SSMs for jobs requiring high-resolution imagery or resource constraints.

Even though CNNs outperformed transformers in terms of accuracy, speed, and memory efficiency in the three tasks, they may lag in capturing complex spatial relationships, which transformers handle better due to their attention mechanisms. However, SSMs showed great potential in overcoming transformer inefficiencies while maintaining good long-range dependence modeling.

Mamba accuracy is not yet optimal [78], which is evident through our results. SSMs like Mamba struggle with position-sensitive visual data and capturing global spatial relationships, but self-attention mechanisms in transformers like DeiT do well. While Mamba's bidirectional state space models approximate global context, they may not be as sophisticated as self-attention. Furthermore, Mamba's relatively new architecture necessitates additional tuning, such as hyperparameter tweaks and task-specific modifications.

8.3 Real-World Applicability

As seen in Chapter 7, incorporating knowledge distillation significantly boosted Mamba-tiny's performance in wildfire detection while maintaining its lightweight nature. Mamba-tiny's compact architecture, along with minimal memory requirements and computational efficiency, make it ideal for use on edge devices. The model can run on devices such as drones, IoT sensors, and mobile hardware because of its limited GPU memory allocation and low hardware costs. This enables on-site wildfire detection and real-time decision-making in remote or resource-constrained

places where cloud computing infrastructure is limited or unavailable. The ability to deploy Mamba-tiny on edge devices highlights the research's scalability since hundreds or thousands of edge devices might run autonomously and provide real-time data over huge geographical areas. The model's memory and computational efficiency save energy, making it an environmentally benign and sustainable option for large-scale environmental monitoring.

With excellent performance and efficiency, VisionMamba-tiny and VisionMamba-small could also be easily integrated within resource-constrained devices for plant disease identification. Many farmers in developing regions lack access to high-cost hardware or cloud services, making this model an affordable choice for agricultural applications without sacrificing performance.

Hospitals and clinics are increasingly seeking sustainable AI solutions that use less energy. Many low-income and disadvantaged areas have limited access to professional dermatologists and modern AI diagnostic tools. Mamba-tiny, because of its efficiency and low cost, can be integrated into mobile health applications and point-of-care medical devices to provide skin cancer screening to these communities. The model's minimal GPU memory utilization and fast inference times make it an appealing choice for AI-powered medical diagnostics that must work on limited hardware. While it may not replace high-accuracy models used in hospital-grade AI systems, it acts as a first-level screening tool (like in mobile diagnostic units or telemedicine apps), helping prioritize high-risk situations for further investigation by experts.

8.4 Future Directions

While Vision Mamba has been extensively explored, a few refinements could enhance its performance. One refinement could be done in optimizing accuracy for Vision Mamba to match the performance of CNNs when processing lower-resolution images for resource-constrained environments without compromising its exceptional efficiency. Secondly, ViM can be used to create segmentation solutions for multi-domains to test its applicability further. The performance evaluation of ViM can then be tested against different CNNs and transformer-based segmentation models.

Lastly, ViM can be deployed on resource-constrained edge devices such as drones, IoT sensors, and embedded systems as they have great potential for real-world applications. Given its computational efficiency, it is well-suited for environments with limited processing power and low latency requirements, but this can only be confirmed through real-life testing.

Chapter 9

Conclusion

This research establishes that SSMs, particularly Vision Mamba offer critical real-time capabilities for object recognition and classification systems. Vision Mamba with its memory efficiency, quick convergence, and sub-quadratic time complexity, stands as an effective alternative deep learning solution in essential domains like disaster management, healthcare, and agriculture. Vision Mamba delivers promising accuracy while managing efficiency at levels that surpass Transformers due to their attention-based limitations. When real-world systems need speedy decision-making while operating under computing resource constraints, Vision Mamba demonstrates unparalleled value as a fundamental tool.

One of the primary findings of this work describes how knowledge distillation increased the performance of Vision Mamba Tiny while keeping the architecture lightweight and computationally efficient. The wildfire detection capabilities of the system rose significantly (14.72% point increase), demonstrating its potential for real-time environmental surveillance and disaster management operations. The novel advancement shows promise to guide research that will develop lightweight scalable AI models to reshape high-risk mission-critical solution development.

Even though Vision Mamba has not reached the same performance level as CNNs and in many cases Transformers, they can improve through research and practice. The adoption of Vision Mamba, as well as related architectures into upcoming technologies, enables new possibilities for breakthrough applications in edge computing and robotics alongside medical imaging and environmental monitoring systems and more. As the technology expands SSM-based models can transform the boundaries of AI-driven solutions while delivering smart high-performance computing capabilities for various applications.

Bibliography

[1] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, ser. MCS '00, Berlin, Heidelberg: Springer-Verlag, 2000, pp. 1–15, ISBN: 3540677046.

[2] A. Honkela, *Nonlinear Switching State-Space Models*, May 2001. [Online]. Available: http://users.ics.aalto.fi/ahonkela/papers/dippa.pdf.

[3] K. Amoura, P. Wira, and S. Djennoune, "A state-space neural network for modeling dynamical nonlinear systems," *NCTA 2011 - Proceedings of the International Conference on Neural Computation Theory and Applications*, pp. 369–376, Jan. 2011.

[4] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11, Bellevue, Washington, USA: Omnipress, 2011, pp. 1017–1024, ISBN: 9781450306195.

[5] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1512.03385.

[6] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, 2015. arXiv: 1503.02531 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1503.02531.

[7] D. P. Hughes and M. Salathé, "An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing," *CoRR*, vol. abs/1511.08060, 2015. arXiv: 1511.08060. [Online]. Available: http://arxiv.org/abs/1511.08060.

[8] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin, "On Particle Methods for Parameter Estimation in State-Space Models," *Statistical Science*, vol. 30, no. 3, pp. 328–351, 2015. DOI: 10.1214/14-STS511. [Online]. Available: https://doi.org/10.1214/14-STS511.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv preprint arXiv:1512.00567*, 2015. DOI: 10.48550/arXiv.1512.00567. [Online]. Available: https://arxiv.org/abs/1512.00567.

[10] K. Ahmed, N. S. Keskar, and R. Socher, *Weighted transformer network for machine translation*, 2017. arXiv: 1711.02132 [cs.AI]. [Online]. Available: https://arxiv.org/abs/1711.02132.

[11] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, 2017. DOI: 10.1109/TGRS.2016.2636241.

[12] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2018, pp. 169–175. DOI: 10.1109/CCWC.2018.8301729. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8301729.

[13] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf.

[14] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[15] L. Mou, L. Bruzzone, and X. X. Zhu, "Learning spectral-spatial-temporal features via a recurrent convolutional neural network for change detection in multispectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 924–935, 2019. DOI: 10.1109/TGRS.2018.2863224.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *Mobilenetv2: Inverted residuals and linear bottlenecks*, 2019. arXiv: 1801.04381 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1801.04381.

[17] R. M. Schmidt, *Recurrent neural networks (rnns): A gentle introduction and overview*, 2019. arXiv: 1912.05911 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1912.05911.

[18] D. R. So, C. Liang, and Q. V. Le, "The evolved transformer," in *International Conference on Machine Learning*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:59523610.

[19] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Re, *Hippo: Recurrent memory with optimal polynomial projections*, 2020. arXiv: 2008.07669 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2008.07669.

[20] N. M. Rezk, M. Purnaprajna, T. Nordström, and Z. Ul-Abdin, "Recurrent neural networks: An embedded computing perspective," *IEEE Access*, vol. 8, pp. 57 967–57 996, 2020. DOI: 10.1109/ACCESS.2020.2982416.

[21] Rismiyati, S. N. Endah, Khadijah, and I. N. Shiddiq, "Xception architecture transfer learning for garbage classification," in *2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*, IEEE, 2020, pp. 1–5. DOI: 10.1109/ICICoS51170.2020.9299017. [Online]. Available: https://ieeexplore.ieee.org/document/9299017.

[22] M. Tan and Q. V. Le, *Efficientnet: Rethinking model scaling for convolutional neural networks*, 2020. arXiv: 1905.11946 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1905.11946.

[23] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers distillation through attention," *arXiv preprint arXiv:2012.12877*, 2020. [Online]. Available: https://arxiv.org/abs/2012.12877.

[24] J. Xiao and Z. Zhou, "Research progress of rnn language model," in *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2020, pp. 1285–1288. DOI: 10.1109/ICAICA50127.2020.9182390.

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2010.11929.

[26] A. Radford, J. W. Kim, C. Hallacy, *et al.*, *Learning transferable visual models from natural language supervision*, 2021. arXiv: 2103.00020 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.00020.

[27] W. Wang, E. Xie, X. Li, *et al.*, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," *arXiv preprint arXiv:2102.12122*, 2021. [Online]. Available: https://arxiv.org/abs/2102.12122.

[28] F. E. Fernandes, L. G. Nonato, and J. Ueyama, "A river flooding detection system based on deep learning and computer vision," *Multimedia Tools and Applications*, vol. 81, no. 28, pp. 40 231–40 251, May 2022. DOI: 10.1007/s11042-022-12813-3. [Online]. Available: https://doi.org/10.1007/s11042-022-12813-3.

[29] R. Ghali, M. A. Akhloufi, and W. S. Mseddi, "Deep learning and transformer approaches for uav-based wildfire detection and segmentation," *Sensors (Basel, Switzerland)*, vol. 22, no. 5, p. 1977, 2022. DOI: 10.3390/s22051977.

[30] A. Gu, K. Goel, and C. Ré, *Efficiently modeling long sequences with structured state spaces*, 2022. arXiv: 2111.00396 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.00396.

[31] J. Huang, Y. Fang, Y. Wu, *et al.*, "Swin transformer for fast mri," *Neurocomputing*, vol. 493, pp. 281–304, 2022. DOI: 10.1016/j.neucom.2022.04.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222004179.

[32] E. M. Huynh, "Vision transformers in 2022: An update on tiny imagenet," *arXiv preprint arXiv:2205.10660*, 2022. [Online]. Available: https://arxiv.org/abs/2205.10660.

[33] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, *A convnet for the 2020s*, 2022. arXiv: 2201.03545 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2201.03545.

[34] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, *Efficient transformers: A survey*, 2022. arXiv: 2009.06732 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2009.06732.

[35] W. Wang, E. Xie, X. Li, *et al.*, "Pvt v2: Improved baselines with pyramid vision transformer," *arXiv preprint arXiv:2106.13797*, 2022. [Online]. Available: https://arxiv.org/abs/2106.13797.

[36] D. Zhang, J. Yang, F. Li, S. Han, L. Qin, and Q. Li, "Landslide risk prediction model using an attention-based temporal convolutional network connected to a recurrent neural network," *IEEE Access*, vol. 10, pp. 37 635–37 645, 2022. DOI: 10.1109/ACCESS.2022.3165051.

[37] S. Zuo, X. Liu, J. Jiao, *et al.*, *Efficient long sequence modeling via state space augmented transformer*, 2022. arXiv: 2212.08136 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2212.08136.

[38] K. Han, Y. Wang, H. Chen, *et al.*, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 87–110, 2023. DOI: 10.1109/TPAMI.2022.3152247.

[39] Z. Hong, E. Hamdan, Y. Zhao, T. Ye, H. Pan, and A. E. Cetin, "Wildfire detection via transfer learning: a survey," *Signal Image and Video Processing*, vol. 18, no. 1, pp. 207–214, Aug. 2023. DOI: 10.1007/s11760-023-02728-3. [Online]. Available: https://doi.org/10.1007/s11760-023-02728-3.

[40] J. Jackson, S. B. Yussif, R. A. Patamia, K. Sarpong, and Z. Qin, "Flood or Non-Flooded: A Comparative Study of State-of-the-Art Models for Flood Image Classification Using the FloodNet Dataset with Uncertainty Offset Analysis," *Water*, vol. 15, no. 5, p. 875, Feb. 2023. DOI: 10.3390/w15050875. [Online]. Available: https://doi.org/10.3390/w15050875.

[41] X. Li and B.-B. Zhang, "Fv-vit: Vision transformer for finger vein recognition," *IEEE Access*, vol. 11, pp. 75 451–75 461, 2023. DOI: 10.1109/ACCESS.2023.3297212.

[42] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," en, *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1–28, Jul. 2023.

[43] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762.

[44] J. Wensel, H. Ullah, and A. Munir, "Vit-ret: Vision and recurrent transformer neural networks for human activity recognition in videos," *IEEE Access*, vol. 11, pp. 72 227–72 249, 2023. DOI: 10.1109/ACCESS.2023.3293813.

[45] S. Woo, S. Debnath, R. Hu, *et al.*, *Convnext v2: Co-designing and scaling convnets with masked autoencoders*, 2023. arXiv: 2301.00808 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2301.00808.

[46] C. A. Alonso, J. Sieber, and M. N. Zeilinger, "State space models as foundation models: A control theoretic overview," *ArXiv*, vol. abs/2403.16899, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:268681121.

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

[48] H. Ezoe and K. Sato, "Model compression method for s4 with diagonal state space layers using balanced truncation," *IEEE Access*, vol. 12, pp. 116 415–116 427, 2024. DOI: 10.1109/ACCESS.2024.3446642.

[49] L. Götz, M. Kollovieh, S. Günnemann, and L. Schwinn, *Efficient time series processing for transformers and state-space models through token merging*, 2024. arXiv: 2405.17951 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2405.17951.

[50] A. Gu and T. Dao, *Mamba: Linear-time sequence modeling with selective state spaces*, 2024. arXiv: 2312.00752 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2312.00752.

[51] A. Gu and T. Dao, *Mamba: Linear-time sequence modeling with selective state spaces*, 2024. [Online]. Available: https://openreview.net/forum?id=AL1fq05o7H.

[52] A. Gu, I. Johnson, K. Goel, *et al.*, "Combining recurrent, convolutional, and continuous-time models with linear state-space layers," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21, Red Hook, NY, USA: Curran Associates Inc., 2024, ISBN: 9781713845393.

[53] A. Hatamizadeh and J. Kautz, *Mambavision: A hybrid mamba-transformer vision backbone*, 2024. arXiv: 2407.08083 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2407.08083.

[54] H. He, Y. Bai, J. Zhang, *et al.*, "Mambaad: Exploring state space models for multi-class unsupervised anomaly detection," *ArXiv*, vol. abs/2404.06564, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:269033047.

[55] Y. He, B. Tu, B. Liu, J. Li, and A. Plaza, *3dss-mamba: 3d-spectral-spatial mamba for hyperspectral image classification*, 2024. arXiv: 2405.12487 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2405.12487.

[56] L. Hollard, L. Mohimont, and L. A. Steffenel, *Designing lightweight CNN for Images: Architectural components and techniques*. unknown, Feb. 2024, pp. 105–129. DOI: 10.1201/9781003478713-5. [Online]. Available: https://doi.org/10.1201/9781003478713-5.

[57] J. Huang, L. Yang, F. Wang, *et al.*, *Mambamir: An arbitrary-masked mamba for joint medical image reconstruction and uncertainty estimation*, 2024. arXiv: 2402.18451 [eess.IV]. [Online]. Available: https://arxiv.org/abs/2402.18451.

[58] Y. Huang, T. Miyazaki, X. Liu, and S. Omachi, *Irsrmamba: Infrared image super-resolution via mamba-based wavelet transform feature modulation model*, 2024. arXiv: 2405.09873 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2405.09873.

[59] F. R. Jafari, G. Montavon, K.-R. Müller, and O. Eberle, *Mambalrp: Explaining selective state space sequence models*, 2024. arXiv: 2406.07592 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2406.07592.

[60] S. Li, H. Singh, and A. Grover, "Mamba-nd: Selective state space modeling for multi-dimensional data," *ArXiv*, vol. abs/2402.05892, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267547860.

[61] H. Lu, A. A. Salah, and R. Poppe, *Videomambapro: A leap forward for mamba in video understanding*, 2024. arXiv: 2406.19006 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2406.19006.

[62] C. Ma and Z. Wang, *Semi-mamba-unet: Pixel-level contrastive and pixel-level cross-supervised visual mamba-based unet for semi-supervised medical image segmentation*, 2024. arXiv: 2402.07245 [eess.IV]. [Online]. Available: https://arxiv.org/abs/2402.07245.

[63] J. Ma, F. Li, and B. Wang, *U-mamba: Enhancing long-range dependency for biomedical image segmentation*, 2024. arXiv: 2401.04722 [eess.IV]. [Online]. Available: https://arxiv.org/abs/2401.04722.

[64] Z. Ma, H. Zhang, and J. Liu, "Db-rnn: An rnn for precipitation nowcasting deblurring," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 5026–5041, 2024. DOI: 10.1109/JSTARS.2024.3365612.

[65] J. Park, H.-S. Kim, K. Ko, M. Kim, and C. Kim, *Videomamba: Spatio-temporal selective state space model*, 2024. arXiv: 2407.08476 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2407.08476.

[66] B. N. Patro and V. S. Agneeswaran, *Mamba-360: Survey of state space models as transformer alternative for long sequence modelling: Methods, applications, and challenges*, 2024. arXiv: 2404.16112 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2404.16112.

[67] H. Qu, L. Ning, R. An, *et al.*, *A survey of mamba*, 2024. arXiv: 2408.01129 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2408.01129.

[68] M. M. Rahman, A. A. Tutul, A. Nath, L. Laishram, S. K. Jung, and T. Hammond, *Mamba in vision: A comprehensive survey of techniques and applications*, 2024. arXiv: 2410.03105 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2410.03105.

[69] L. Ramos, E. Casas, E. Bendek, *et al.*, "Computer vision for wildfire detection: A critical brief review," *Multimedia Tools and Applications*, vol. 83, pp. 83 427–83 470, 2024.

[70] J. Sieber, C. Alonso, A. Didier, M. Zeilinger, and A. Orvieto, *Understanding the differences in foundation models: Attention, state space models, and recurrent neural networks*, May 2024. DOI: 10.48550/arXiv.2405.15731.

[71] Q. Wang, L. Zhou, P. Jin, *et al.*, "Trackingmamba: Visual state space model for object tracking," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 16 744–16 754, 2024. DOI: 10.1109/JSTARS.2024.3458938.

[72] X. Wang, S. Wang, Y. Ding, *et al.*, *State space model for new-generation network alternative to transformers: A survey*, 2024. arXiv: 2404.09516 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2404.09516.

[73] Z. Wang, C. Li, H. Xu, and X. Zhu, *Mamba yolo: Ssms-based yolo for object detection*, 2024. arXiv: 2406.05835 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2406.05835.

[74] R. Xu, S. Yang, Y. Wang, Y. Cai, B. Du, and H. Chen, *Visual mamba: A survey and new outlooks*, 2024. arXiv: 2404.18861 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2404.18861.

[75] M. Yim and J.-C. Chiang, "Mamba-pcgc: Mamba-based point cloud geometry compression," in *2024 IEEE International Conference on Image Processing (ICIP)*, 2024, pp. 3554–3560. DOI: 10.1109/ICIP51287.2024.10647269.

[76] G. Zhao, H. Wu, D. Luo, X. Ou, and Y. Zhang, "Spatial spectral interaction super-resolution cnn-mamba network for fusion of satellite hyperspectral and multispectral image," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pp. 1–12, 2024. DOI: 10.1109/JSTARS.2024.3469184.

[77] Z. Zheng and C. Wu, *U-shaped vision mamba for single image dehazing*, 2024. arXiv: 2402.04139 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2402.04139.

[78] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, *Vision mamba: Efficient visual representation learning with bidirectional state space model*, 2024. arXiv: 2401.09417 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2401.09417.

[79] DeepSeek-AI, D. Guo, D. Yang, *et al.*, *Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning*, 2025. arXiv: 2501.12948 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2501.12948.

[80] M. Imani, *Estimation, Inference and Learning in Non-Linear State-Space*.

[81] N. Kantas, *Sequential Decision Making in General State Space models*.

[82] I. Sutskever, *Training Recurrent Neural Networks*.

[83] F. Tobar, *Kernel-based Adaptive Estimation: Mutli-dimensional and State-Space Approaches*.

Appendix: Training and Evaluation Setup of Vision Mamba

1. System requirements: Windows Subsystem Linux (WSL) or Ubuntu. Works best on Ubuntu 22.04.
2. Version 11.8 of cuda must be installed.
3. Appropriate Nvidia drivers must be installed.
4. Check if everything was installed properly using these two commands:

   ```
   nvidia-smi
   nvcc --version
   ```

5. GCC Compiler version 11 must be installed.
6. Anaconda or Miniconda must be installed.
7. A conda environment must be created.
8. Run two commands:

   ```
   conda create -n your_env_name python=3.10.13

   pip install torch==2.1.1 torchvision==0.16.1 torchaudio==2.1.1 --
   index-url https://download.pytorch.org/whl/cu118
   ```

9. Install vim dependencies using pip and vim/requirements.txt file from Vim GitHub Repository.
10. WSL ONLY: Navigate to causal-conv1d/setup.py, replace os.rename() with shutil.move(). Import shutil on top.
11. Navigate to causal-conv1d and run ‘pip install -e .‘
12. Navigate to mamba-1p1p1 and run ‘pip install -e .‘
13. Navigate to vim/scripts to view our pre-written scripts. To train or finetune on other datasets, go through the main.py and datasets.py files.‘
14. You can write your own training scripts. However, be sure to import models_mamba.py and load the model using the specified version of timm in the requirements file.