Categorization of Human Monkey-pox from Skin Lesion Images based on Transformer and Ensemble Learning using GRADCAM

by

Ibne Hassan
22241121
Raida Mobashshira Tahsin
19101272
Sadia Shara Toma
19101016
Tausif Tazwar Quadria Utchhash
19101022

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 2023

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:

Ibne Hassan
22241121

Raida Mobashshira Tahsin
19101272

Sadia Shara Toma
19101016

Tausif Tazwar Quadria Utchhash
19101022

Approval

The thesis/project titled "Categorization of Human Monkey-pox from Skin Lesion Images based on Transformer-Based Ensemble Learning with GRADCAM" submitted by

1. Sadia Shara Toma (19101016)
2. Raida Mobashshira Tahsin (19101272)
3. Ibne Hassan (22241121)
4. Tausif Tazwar Quadria Utchhash (19101022)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 14, 2023.

Examining Committee:

Supervisor:

Dr. Muhammad Iqbal Hossain
Associate Professor
School of Data and Sciences
BRAC University

Co-Supervisor:

Rafeed Rahman
Lecturer
School of Data and Sciences
BRAC University

Program Coordinator:

Md. Golam Rabiul Alam
Professor
School of Data and Sciences
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson
School of Data and Sciences
BRAC University

Ethics Statement

Diagnosis of a disease purely based on the prediction made by a deep learning model and without any further diagnosis by a medical professional can pique an ethical predicament in people. Thus, in our research work, we have ensured the transparency of our model predictions and we also encourage people to seek confirmation of their diagnosis from a medical expert.

Abstract

As the world keeps healing from the worldwide outbreak of COVID-19, the MPOX virus poses a new risk. The Mpox virus is not as deadly or contagious as COVID-19, but new patient cases are recorded every day from a wide variety of nations.Therefore, it will not come as a surprise if another global pandemic occurs due to a lack of precautionary measures. Therefore, it is essential to detect them before they spread throughout the community. The World Health Organisation (WHO) has issued numerous precautionary warnings regarding MPOX. If MPOX spreads swiftly, it poses a significant threat to public health. The result is a significant increase in hospital wait times. This means that hospitals require additional supplementary or auxiliary systems.Recent advances in machine learning have demonstrated immense promise for diagnosis based on image data, including detection of cancer, identification of tumour cell, and identification of COVID-19 patients. Therefore, identical technology could possibly be used to detect the human skin infection known as MPOX. After acquiring an image it can be utilised for further diagnosis and early identification of MPOX.In this research, we leverage 13 recently developed deep learning (DL) models to suggest a new strategy for enhancing the accuracy of MPOX image detection and classification. Eight of the suggested models are based on transformer, while the remaining five are CNN-based models that have been pre-trained.The publicly available Monkeypox Skin Images Dataset (MSID) is utilized to evaluate the suggested method. Four standard metrics—precision, accuracy, F1-score and recall —were applied to the outcomes after the models were fine-tuned. We ultimately utilised an ensemble approach to enhance overall performance and obtain more precise results. We achieved the best results possible with our approach, which included a 99% F1 score, 99% accuracy, 100% precision, and 99% recall. Based on these promising outcomes, which surpass those of existing methods, we propose applying the suggested method for widespread testing by health practitioners. This model can be used as a supplementary diagnostic system for the early detection of MPOX skin lesions.

Keywords: Covid-19,pathogens,symptoms,monkeypox,CLAHE,GRADCAM,ensemble learning, transformer based models, pre-tained models

Dedication

We devote our work to the accurate detection and identification of Monkeypox virus and to aid the unfortunate people who suffer from these disease. We hope that our small contribution can make a significant difference in the research that will be done in the future by other researchers on these three deadly diseases using deep learning methods.

Acknowledgement

Without the Almighty Allah, it would not have been possible for us to finish our thesis without any serious obstacles. Secondly, we would like to thank our Supervisor, Dr. Muhammad Iqbal Hossain, for his valuable advice and patient support. We would also like to thank our CoSupervisor, Mr. Rafeed Rahman. Their expertise and guidance were invaluable in shaping our research methodology. Their insightful feedback guided us to think from a different direction and brought our research to a higher level. Finally, we wanted to thank our parents for their kind encouragement and guidance during the course of our study.

Table of Contents

Bibliography **70**

List of Figures

List of Tables

Nomenclature

Following is a list of symbols abbreviations that will be applied later inside the document's body.

$Adam$ Adaptive Momentum Estimation

CAM Channel Attention Module

$CBAM$ Convolutional Block Attention Module

CCT Compact Convolutional Transformer

$CLAHE$ Contrast-limited adaptive histogram equalization

CNN Convolutional Neural Networks

$Cumsum$ Cumulative Sum

DRC Democratic Republic of the Congo

$EANet$ External Attention Transformer

$FNet$ Fourier Transformer

GAP global average pooling

$gMLP$ Gated Multilayer Perceptron

$GRADCAM$ Gradient-weighted Class Activation Mapping

HE Histogram Equalization

IDE Integrated Development Environment

$MLP - Mixer$ Multilayer Perceptron

$MPOX$ Monkeypox

$MSID$ Monkeypox Skin Images Dataset

$MSLD$ Monkeypox Skin Lesion Dataset

$ReLU$ Rectified Linear Unit

SAM Spatial Attention Module

$SwinT$ Swin Transformer

UN United Nations

ViT Vision Transformer

WHO World Health Organization

$Xception$ Extreme Inception

Chapter 1

Introduction

1.1 Motivation

The global epidemic of COVID-19 during the year 2020 shook the world, and now there are reports that MPOX may arrive in 2022 [2]. The smallpox virus and the cowpox virus are both linked to this virus. Though transmission from person to person is common, the virus is primarily spread by rodents and monkeys [7]. The MPXV virus, a member of the orthopoxvirus family, is responsible for the transmissible illness known as MPOX. The MPOX virus (MPXV, or MPOX) [35] was discovered in a monkey by scientists in Denmark. A kid exhibiting smallpox-like symptoms was admitted to the hospital in the Republic of the Congo in 1970, marking the first ever confirmed case in a human [4]. Cameroon, the Republic of the Congo, the Central African Republic, the Democratic Republic of the Congo, Gabon, Côte d'Ivoire, Liberia, Sierra Leone, and Nigeria were among the nations that had seen the epidemic in the past [39]. The extremely contagious MPOX virus is well known to affect many people who live in close vicinity to tropical rainforests in West and Central Africa. Even though there has been an uptick in confirmed instances of MPOX, it is not as highly contagious as COVID-19. Fifty people contracted MPOX in 1990 [30] throughout all of Central and West Africa. Yet, by the year 2020, there will already be over 5,000 confirmed cases. In 2022, many countries outside of Africa, including the United States and Europe, reported cases of MPOX that had been previously assumed to be confined to Africa. There has been an outbreak of MPOX not just in Africa but also in Europe, Africa, the Western Pacific, the Americas, and countries in the Eastern Mediterranean. The virus can spread through simple interaction with an infected human, animal, or object. Saliva, nasal discharge, or airborne particles can all play a role in transmission [38]. The virus can also spread through animal bites. Short-term symptoms of the MPOX virus involve fever, headache, back discomfort, poor energy, weariness, swollen lymph nodes, muscle pains, and a persistent rash or red lumps on the skin [18] [8].This rash can manifest anywhere on the body, including the face, hands, throat, feet, mouth, groin, and pelvic and/or anal regions [26]. In most circumstances, anti-inflammatory and antipyretic drugs can help a patient recover from an infection. Infants, children, and those with compromised immune systems are more likely to have severe symptoms, including death. So far, the MPOX virus has been found in two distinct groups: one in West Africa and another in Central Africa. The MPOX virus is currently incurable because no effective treatments exist. The development

of a vaccine is the ultimate solution. The primary means of diagnosing MPOX are an electron microscope blister test and polymerase chain reaction (PCR). Because doctors are in short supply and PCR is not always available, particularly in rural regions [36], supplementary diagnostic techniques are required for the early diagnosis of MPOX skin lesions. Consequently, artificial intelligence (AI)-based techniques could aid in their detection by analyzing virus images. Studies in computer vision, an area of AI research, are particularly important here. Computer vision research includes picture categorization, object identification, and image segmentation. This is why we chose to conduct research on deep learning models to improve computer vision so it can detect and classify MPOX skin lesions at an early stage. And deliver the results as soon as possible so that it can be treated properly. Those who are hesitant to visit the hospital in the early phases can also conduct a test from the convenience of their own homes. They simply need a smartphone's camera to take photos so that the system can analyze them to detect MPOX. This will make identifying MPOX extremely simple and inexpensive. In addition, help control the outbreak of MPOX by keeping this contagious disease under control.

1.2 Research Problem

Despite the fact that MPOX is an ancient disease that has recently resurfaced, people lack the knowledge necessary to recognise the symptoms at an early stage. According to the World Health Organisation, an outbreak poses a hazard to global public health. As they resemble those of other skin-related illnesses (chickenpox, measles), the early symptoms are rarely treated seriously. However, if detected at an early stage, the disease will not spread. Therefore, early detection of MPOX is essential. There are multiple methods for detecting the orthopox virus, but none are specific to MPOX. By using polymerase chain reaction (PCR) and genome sequencing, MPOX can be identified. However, the procedure is time-sensitive and costly, requiring specialised equipment, trained personnel, and bioinformatics for computer analysis. Due to the widespread availability of smartphones, artificial intelligence (AI)-powered dermoscopy image recognition systems might aid in the rapid diagnosis and treatment of MPOX. Therefore, our objective is to develop a system based on deep learning that can rapidly detect MPOX from photographs of the affected area. Thus, it can be identified at an early stage, and appropriate measures can be taken to prevent an outbreak.

1.3 Research Objective

In this research, we present an approach based on deep learning (DL) for identifying MPOX skin lesions in an image. Our objective is to ensure early detection of MPOX so that individuals can receive timely treatment. To develop a harmless, rapid, and cost-effective method for diagnosing MPOX. So in the event of an outbreak, individuals can detect MPOX in the privacy of their own homes. Therefore, there will be no unnecessary crowds in the hospitals. To train on the MPOX Skin Images Dataset (MSID), we selected the most cutting-edge image categorization methods available right now, such as ViT, CCT, MLP-Mixer, gMLP, FNET Swin Transformer, EANet, Perceiver, InceptionResnetV2, EfficientNetB3, MobileNetV3Large

DenseNet201, and Xception. After the models were fine-tuned, the results were analyzed using four well-established metrics: precision, accuracy, F1-score, and recall. Finally, we utilized an ensemble approach to enhance the overall performance.
Our key additions to this paper are outlined below:

- Using image data, introduce a new method for identifying MPOX patients.
- Construct the new system using 13 recent deep learning models (both transformer-based and pre-trained) in order to conduct the experiment.
- Fine-tune and train the models utilizing the MPOX Skin Images Dataset (MSID).
- Compare and contrast the models using four established metrics: precision, accuracy,F1-score and recall.
- Improve the overall performance of recently pre-trained CNN-based models, using the ensemble technique
- Construct a new system to aid in diagnosing MPOX.
- Show the explainability using Grad-CAM for the DL models.
- Provide class wise study for confusion matrix

1.4 Thesis Structure

The remaining sections of this paper are structured as follows:

- The literature review for the MPOX is presented in Section 2.
- The methods and procedures that were followed are outlined in Section 3.
- The experimental plan and implementation are described in Section 4
- The results are analyzed and discussed in Section 5
- The concluding remarks with prospective future research are mentioned in Section 6

Chapter 2

Literature Review

Deep and machine learning aid medical diagnosis and treatment. Machine learning (ML), a branch of AI, offers great potential in decision-making support, industrial applications, medical imaging, and illness diagnosis. ML imaging systems help doctors make informed decisions since they are safe, accurate, and fast. Many machine learning and deep learning models and algorithms predict sickness. Thus, new strategies for early and accurate diagnosis and therapy evaluation are needed. Medical image analysis AI models can detect several virus-related ailments [20],[10]. Madhavan et al. [10] created a deep learning model (Res-COvNet) for COVID-19 viral identification using transfer learning. ResNet-50 [34] extracted characteristics from X-ray pictures and added a classification layer. Their X-ray proposal properly identified normal, bacterial, viral, and COVID-19 patients with 96.2 percent accuracy. [32] also reviewed COVID-19 detection deep learning methods. [24] diagnosed facial skin disorders using deep learning.
Deep learning models detected chicken pox, herpes, and COVID-19. Sandeep et al. [21] detected psoriasis, chicken pox, vitiligo, melanoma, ringworm, acne, lupus, and herpes using deep learning (DL). CNNs classified the skin lesion into eight sickness classes and were compared to the VGG-16 pre-trained model [3].They detected it with 78% precision. [19] presents low-cost image analysis for HZV detection using CNN89.6% of 1,000 photos correctly identified HZV. Glock et al. [22] used transfer learning to identify measles. They achieved 81.7% sensitivity, 97.1% specificity, and 95.2% accuracy using the ResNet-50 model on the varied rash image dataset. In [11], big data ensemble learning was proposed for Ebola virus illness detection. They extracted knowledge from massive data sets using Apache Spark, Kafka, and ANN/GA. MPOX is becoming a global health issue. The WHO classified MPOX as an international medical emergency on July 23, 2022. 75 non-African nations have confirmed instances. MPOX's similarities to chickenpox and measles make early clinical diagnosis challenging. Computer-assisted lesion morphology detection may help monitor and diagnose MPOX-infected patients without PCR test kits. When given enough training examples, deep-learning systems can detect skin lesions. Before the epidemic, medical specialists were unfamiliar with MPOX due to its rarity. Scientists are pleased by the finding of COVID-19 via supervised machine learning. Lack of MPOX skin photos hinders machine learning to detect MPOX from patient skin photographs. Despite the long history of MPOX in humans, few studies have shown that machine learning approaches can diagnose MPOX using image processing. Since the virus has just recently been widely introduced in several countries,

there is no publicly accessible dataset for training and testing image-based MPOX diagnosis. [25] offered the most MPOX skin images. Web scraping provides access to a large image database of healthy and sick skin. Infected skin shows measles, cowpox, varicella, smallpox, and MPOX[1] . created the MPOX Skin Lesion Dataset (MSLD) from photos of measles, chickenpox, and MPOX skin lesions. Websites, news portals, and case reports provided most of these photographs. A three-fold cross-validation test was performed after increasing the sample size. MPOX illness researchers have proposed different data gathering and classification methods. Web scraping was suggested for MPOX cutaneous lesion data collection [1]. Researchers preprocessed the dataset with scaling and data improvement. Thus, researchers made the dataset open source for academics. They planned MPOX classification [16] The data set contains 804 original and 39,396 enhanced photos. The classification assignment employed ResNet50, Inception-V3, DenseNet121, MnasNet-A1, MobileNet-V2, ShuffleNet-V2-1X, and SqueezeNet models. ShuffleNet-V2-1 had the highest accuracy of 0.79. [17] used web mining and expert evaluation to classify MPOX. The data collection includes 43 MPOX, 47 chickenpox, 17 measles, and 54 normal pictures. MPOX classification was their goal [12]. 1915-enhanced images are used. They then assessed transfer learning using the VGG-16 model and two methods [12]. The first technique classified photos into MPOX and chickenpox, whereas the second augmented them. Data augmentation decreased MPOX classification accuracy from 97% to 78%. Transfer learning was used to categorise MPOX using pre-trained deep learning models such as VGG-16, ResNet50, and InceptionV3, as well as their ensembles. 228 original photos and 3192 enhanced images are used. The ResNet50 model had the highest accuracy, precision, recall, and F1 score in the investigation. Most DL investigations on virus-related disease detection have used transfer learning with CNN-based, pre-trained DL techniques. Outdated DL models were deployed. No study detected MPOX epidermal lesions using transformer-based models or ensemble approaches. Since there are few MPOX virus detection studies, we identified an opportunity to innovate and gain confidence among medical professionals during large-scale examinations. These opportunities revealed the need for a novel MPOX image diagnosis method to fill this gap. This research compares 14 transformer-based models to newer CNN-based pre-trained models for MPOX image categorization. Combining the best pre-trained CNN-based models enhances performance. Use precision, recall, F1-score, and accuracy to test the proposed method.[5],[6]

Chapter 3

Methodology

3.1 Dataset

The experiments conducted in this study were based on the publicly available Monkeypox Skin Images Dataset (MSID) in Fig.3.1 on Kaggle [14],[25]. This data set includes four categories: monkeypox, chickenpox, measles, and normal. All image classes are compiled from health-related websites on the Internet. The pictures were PNG files. This dataset contains 279 images of cases of monkeypox, 91 images of measles, 107 images of chickenpox, and 293 images of normal cases. There are a total of 770 images included in Table 3.1. The images included in the dataset were not modified or augmented. The Department of Computer Science and Engineering at the Islamic University in Kusht-7003, Bangladesh, created the entire dataset. We selected this specific dataset because it received a usability score of 7.50. In addition, according to the ratings on Kaggle, its completeness is 100% . In addition, the dataset is utilized in both publications [28], [27]. Our objective is to classify MPOX skin lesions using recent transformers and pre-trained models by applying ensemble learning. Despite the fact that the dataset is new and has just recently been utilized (in 2022), [15] recent transformers and pre-trained models implementing ensemble learning have never been implemented on it. To our knowledge, no previous study has tried to categorize MPOX skin lesions using the state-of-the-art transformers and pre-trained models incorporating ensemble learning that we have selected [28]. Our selected models outperform existing state-of-the-art models such as the VGG-19 and Densenet-169. Therefore, we chose this recent dataset in order to test a new methodology and obtain more precise results.

Category or Class	**Number of Samples**
Monkeypox	279
Chicken pox	107
Measles	91
Normal	293
Total	770

Table 3.1: Number of images within the dataset folder by category

Figure 3.1: Dataset Image Samples

3.2 Pre-processing

3.2.1 CLAHE (Contrastive Limited Adaptive Equalization)

Any image may be described in terms of the intensity or grey level of the provided image at any particular pair of coordinates (x,y) is expressed by the two-dimensional function f(x,y), where x and y are spatial (plane) coordinates. When the intensity values of f and the coordinates (x, y) are all discrete, finite quantities, the resulting image is referred to as a digital image. Mathematical models are frequently used to describe images and other signals. A signal is frequently thought of as a function that depends on various factors. It can be represented as a one-dimensional function that depends on time, a two-dimensional image that depends on two plane coordinates, or a three-dimensional function that traces a volumetric object in space. A scalar function can be used to describe colourless pictures, while a vector function can be used to describe colour images with three component colours.Digital, discrete, or continuous functions can all be employed in image processing applications. While the domain and range of a continuous function are continuous, those of a discrete function have discrete domain sets, and those of a digital function have discrete range sets. The picture is static in character in an image processing activity, meaning that time is constant. Coordinates are natural integers because digital image functions are typically utilised in computerised image processing, which is typically expressed in the form of matrices. The region R in the plane is where the image function's

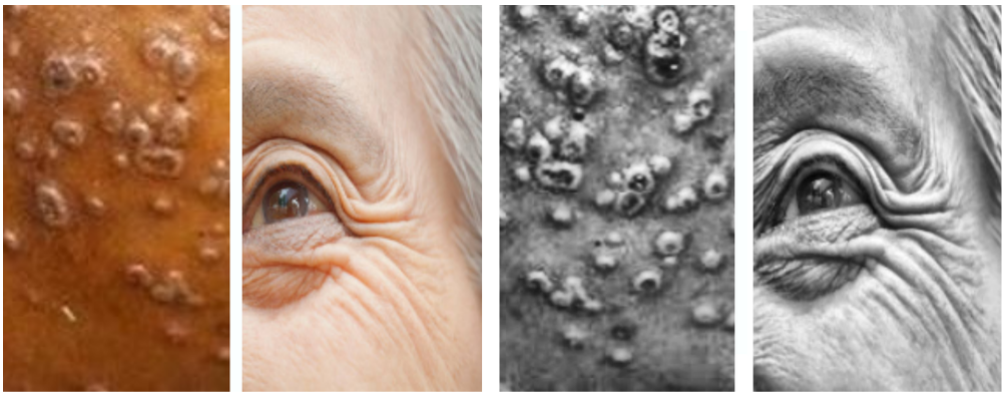

Figure 3.2: Before and after applying CLAHE

domain is defined:

$$R = \{(x, y), 1 \leq x \leq x_m, 1 \leq y \leq y_n\} \tag{3.1}$$

Here, x_m and y_n represents maximum coordinates. The arrangement of colours in a given image in a certain format is known as a colour space. A colour model refers to the method used to portray colour. Images, such as RGB (red, green, and blue), are created using a variety of colour spaces for effective visual representation. YPbPr stands for green, blue, and red cables. XYZ stands for colour in x, y, and z dimensions [37].Here, figure 3.2 shows the changes made to an image both before and after CLAHE was applied to it.

3.3 Deep Learning Models

3.3.1 Vision Transformer

This is how a ViT model works according to [34]: a 1D sequence of input for token embeddings into the conventional transformer. To handle 2D images and transform the image $x \in R^{H \times W \times C}$ into a series of flattened 2D patches, where (H, W) is the resolution of the initial picture, C is the number of channels, (P, P) is the sharpness of each image patch, and N is the number of patches produced. This input length of the sequence additionally acts as the transformer's operational length. We compress the patches and define the transformer's constant latent vector size D for each of its layers via a trainable linear projection. The patch embeddings are the results of this projection.

We add an adaptable embedding to the collection of embedded patches, comparable to BERT's [class] token, whose state at the Transformer encoder's output acts as the visual representation y. For pre-training and fine-tuning, an identification head is installed. During pre-training, an MLP with a single hidden layer is used to set up the classification head, and during fine-tuning, a single linear layer is used. Position embeddings are included to patch embeddings and preserve positional information. We employ more basic 2D-aware position embeddings because we have not seen any appreciable speed advantages from doing so. The series of generated embedding vectors is used as input to the encoder [33] and also visible in figure 3.3. Multihead self attention and MLP blocks are alternately layered in the Transformer encoder [35]. Prior to each block, Layernorm (LN) is applied, and each block is followed by residual connections [36,37].

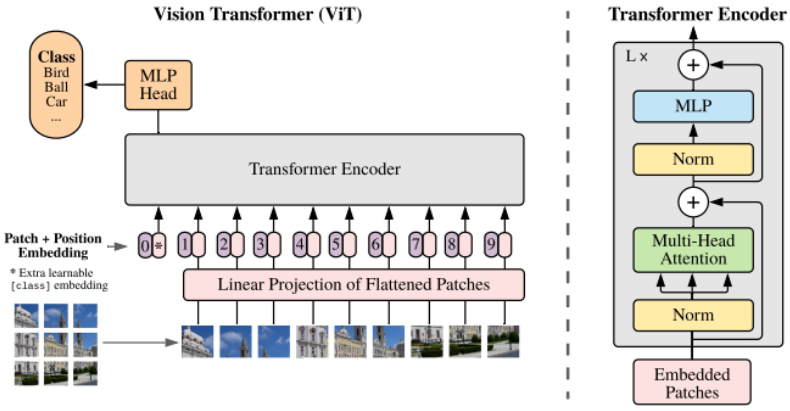

Figure 3.3: Overview of Vision Transformer, [33]

3.3.2 Swin Transformer

Swin transformer first uses a module for patch separation, such as to divide an input picture with RGB values into non-overlapping patches, ViT is applied. Every patch was handled represents a token with their set of attributes to be a result of concatenation of the raw RGB values of the pixels. We utilize a patch measure of 4 *times*4 in our implementation, hence, Each patch's feature measure is $4 \times 4 \times 3 = 48$. This raw merit feature is projected to some arbitrary estimatimation using a layer of linear embedding. On these patch tokens, Several Transformer blocks having customized self-attention computation are used (Swin Transformer modules).This raw-valued feature is projected to an arbitrary size making use of a layer of linear embedding.The token quantity is preserved by the Transformer blocks ($\frac{H}{4}\frac{W}{4}$ When paired alongside the regular embedding, they belong to as Stage 1. As the network grows deeper, patching layers together to produce an ordered model reduces the number of tokens. The initial patch combining layer integrates the distinct features of each pair of adjoining patches. and uses an uninterrupted layer to the concatenated 4-C Dimensional Characteristics. The total amount of tokens is reduced by a factor of $2 \times 2 = 4$,and 2C is the resultant measurements. Swin Transformer components are subsequently utilized to transform features,with regard to the resolution remaining at ($\frac{H}{8}\frac{W}{8}$ The first patch component is termed to as Stage 2 integrating and modifying features. The procedure is repeated twice, with output resolutions of ($\frac{H}{16}\frac{W}{16}$ and $\frac{H}{16}\frac{W}{16}$ respectively, as Stage 3 and Stage 4. These processes combine to form a structure for illustration. Traditional convolutional networks, such as VGG and ResNet, use the same characteristic map resolutions. As a result, the proposed architecture is easily replaceable in existing vision techniques' core networks.
Swin Transformer is created by swapping out a window-shifted module for the usual A Transformer block contains a multi-head self-attention mechanism.The basic Transformer architecture and its picture classification alteration both use global self-attention to compute the associations between tokens. Because the global computation is quadratic in complexity in respect with relation to the quantity of tokens, It is ineffective for many problems with vision that need an excessive amount of tokens for intensive modeling or to display a superior resolution picture.
Global computation of self-attention can be costly in case of a significant amounts

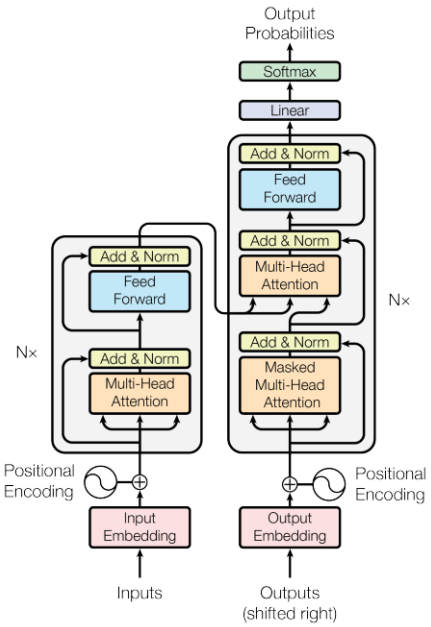

Figure 3.4: Architecture of the Transformer encoder, [40]

of hardware, but window-based self-attention computation is flexible.Because the window-based attention by itself mechanism is missing cross-window links, its simulation power is constrained. The suggested shifted window partitioning approach introduces Connections between into windows while preserving the swift non-overlapping computation windows by switching between two partitioning Swin Transformer blocks in succession. The initial module employs a normal technique for window dividing, beginning with the pixel at the upper left corner, and ending with the bottom-right pixel.8×8 The feature map is distributed uniformly. divided into 2×2 sizes of windows 4×4(M=4).The following components then uses a windowing arrangement that differs from the previous layer by shifting the windows $(\frac{M}{2}, \frac{M}{2})$ pixels taken from the normally Windows shared The splitting of rotatable windows technique produces an array of Swin Transformer blocks. Since it creates links among adjacent windows that do not overlap in the preceding layer by layer, the relocated window partitioning technique proves effective in recognising objects in images and in semantic segmentation of photos. [9]

3.3.3 Compact Convolutional Transformer(CCT)

We substitute a straightforward CVT and ViT-Lite use a convolutional block for patching and embedding in order to Include an inductive distortion in the CCT framework. This block uses a traditional design that includes a max pool, a single convolution, and ReLU activation.Given an image or a map of features $x \in R^{H \times W \times C}$:

$$x_0 = \text{max-pool}(\text{ReLU}(\text{Conv2d}(x))) \tag{3.2}$$

The location of the transformer backbone's embedding dimension is d, and the Conv2d operation comprises d filters. Additionally, the max reservoir and spiral procedures which might cross over that could improveby appearance introducing

prejudices based on induction.Because of that, this kind of system can store local spatial data. Additionally, since the input resolution is no longer strictly divided by the preset patch size, the models benefit from greater flexibility than ViT models, for example. We intend to use convolutions to incorporate the image into a latent representation since we believe they will be more successful and result in richer tokens for the transformer. These blocks are repeated for even more downsampling and are adjustable for downsampling ratio (kernel size, stride, and padding). The entire amount of tokens matches the final value of the feature map used as input, as well as self-attention possesses a quadratic spatial and temporal complexity in terms of token count, therefore, more downsampling yields fewer tokens, which dramatically reduces computation (at the expense of performance).
In addition to the improved performance, we discovered that this tokenization approach also provides additional flexibility for removing the positional embedding from the model because it still achieves excellent performance. Compact Convolutional Transformers are produced via this convolutional tokenizer, SeqPool, and the transformer encoder. With the exception of also indicating the number of convolutional layers, we employ a similar notation for CCT variants. Below figure 3.5 describes the Comparison between the architectures of ViT, CVT, CCT. For example,CCT - $\frac{7}{3} \times 2$ has 7 transformer encoder layers and a 2-convolutional layer tokenizer with a 3×3 kernel size.[39]

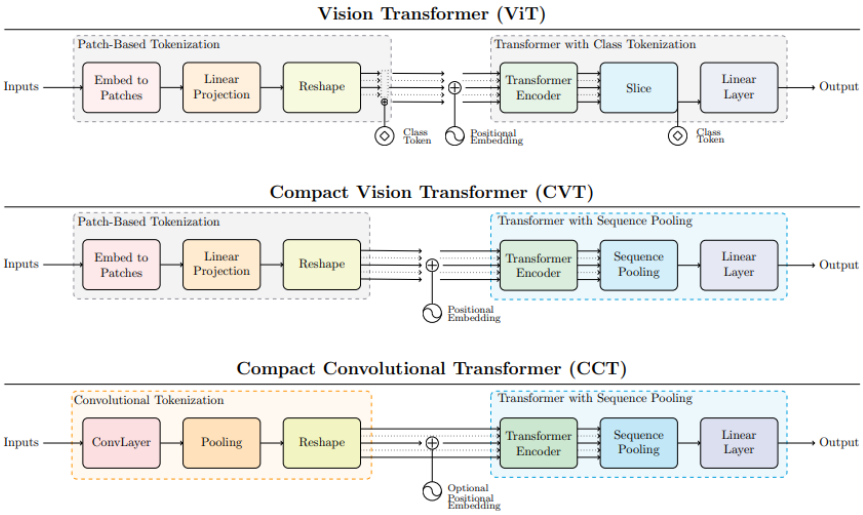

Figure 3.5: Figure: Comparison between ViT, CVT, CCT, [39]

3.3.4 External Attention Transformer(EANet)

A transformer-based model called the External Attention Transformer (EANet) uses two linear layers to calculate external attention. For visual tasks, it is a more straightforward option than self-attention.The encoder and decoder of the EAT paradigm each contain multiple layers of EAT blocks.Two linear layers make up each EAT block: one for computing the key vectors and the other for computing the query vectors.The query vectors and key matrix are put together to generate a dot product that is then used to compute the attention weights. Following that, paying attention to the operation's result is calculated as a weighted average of the value

vectors, in which the weights correspond to the weights of attention. Instead of dot-product focus utilised in conventional self-attention models, the External Attention Transformer (EANet) model computes attention scores using two linear projections. This is how it goes:
Embedded Inputs: The sequence is entered into the model as embeddings (such as an image or a text).After linearly projecting these embeddings, query, key, and value embeddings are created.The model calculates a set of attention ratings for each query by taking the query's dot product embedding as well as each essential embedding.These attention scores are then subjected to a softmax function in order to produce a probability distribution across the values. The output representation for that query is then created using the distribution that results from computing a weighted sum of the value embeddings.
Multi-Head Attention: Each collection of query, key, and value projections used by the EANet model is referred to as a "head". The output representation is created by concatenating and linearly projecting the outputs from these parallel heads.
Non-linearity: Before computing the dot product with the query embedding, the EANet model An activation function that is non-linear (like the ReLU) is used to the output of the linear projection. This adds non-linearity to the attention scores.Prior to feeding the input embeddings into the attention layer, the model employs positional encoding to add positional information.The final output representation is transmitted through a feedforward neural network at the output layer before the final prediction is made.Overall, the EANet model is comparable to the self-attention process seen in conventional transformer models, but it calculates attention scores in a different way. By doing this, it can solve some of the limits of self-attention and produce cutting-edge outcomes on a range of visual tasks.[40]

3.3.5 Fourier Transformation(FNet)

Fnet is a neural network architecture. It combines the Fourier Transform with a self-attention mechanism. Fnet was developed for sequential data processing applications. This includes natural language processing, audio recognition, and video analysis. Instead of the dot product the self-attention mechanism in FNet is modified to apply the (FFT).
FNet uses the Fourier Transform to efficiently compute attention weights, particularly for longer sequences. In transformer-based models, the standard self-attention mechanism includes calculating the dot product of the query and key vectors. This is computationally expensive for longer sequences. Because the number of dot products that need to be computed grows quadratically with sequence length.
In contrast, the Fourier Transform efficiently computes attention weights in the frequency domain, which is practical for longer sequences. The Fourier Transform is a mathematical procedure that converts time-domain signals to frequency-domain signals. Signals in the frequency domain are a mix of sinusoidal waves of varying frequencies and amplitudes. Fnet uses the Fourier Transform to process sequences of different lengths without recurrence or convolution. The Fast Fourier Transform (FFT) is a computationally efficient approach for computing a signal's Fourier Transform. It calculates a sequence's Discrete Fourier Transform (DFT), This is a conversion from the domain of frequency to the duration domain.The FFT is used in FNet to transform the input sequence into the frequency domain. First, a lin-

ear transformation is used to embed the input sequence into a lower-dimensional space. Then the Fast Fourier Transform (FFT) transforms the embedded sequence into the frequency domain. This converts the sequence from the time domain to the frequency domain, allowing FNet to process large sequences quickly. The transformed sequence has three parts: query, key, and value. These parts are created by linearly transforming the Fourier-transformed sequence. The attention weights are calculated using the product of the Fourier-transformed query and the complex conjugate of the Fourier-transformed key. The weighted sum of the Fourier transformed value and the attention weights is used to calculate the attention output, which is then transformed back to the time domain using the inverse FFT. FNet can efficiently compute attention weights for long sequences without the quadratic complexity of dot-product-based attention. Fnet trains faster and performs better on longer sequences than typical neural network architectures. Standard backpropagation techniques can be used to train a Fnet. The Fast Fourier Transform (FFT) is more numerically stable than dot product-based attention, especially for long sequences where numerical errors might develop. This is due to the fact that complex numbers are more numerically stable than real numbers in dot-product-based attention. Fnet can be used independently or with other neural network architectures like convolutional or recurrent.
Overall, FNet's improved self-attention mechanism calculates attention weights using the Fourier Transform rather than the dot product, resulting in more efficient and reliable processing of longer sequences[41].

3.3.6 Multilayer Perceptron (MLP-Mixer)

The most well-known and commonly used neural network model is the Multilayer Perceptron (MLP). In this paradigm, the connection among the inputs and the results is nonlinear. It is a feed-back construction, which implies that the network signals go from input to output. Because that cannot be a loop, the output of each neuron has no influence on the neuron individually. The architecture is composed of up of hidden layers,input, and output(which have no direct connection to the environment). It has the ability to perform any function for activation.The number of layers and neurons within a neural network are known as hyperparameters. Feedback networks may send impulses in either direction due to reaction links in the network. These networks are exceptionally strong and can be highly complex. Increasing layers improves the amount of detail of decision areas. A perceptron having a single layer and a single input generates semi-planar decision zones. By introducing another layer, each neuron functions as a typical perceptron for the outputs from the front layer, allowing the entire network to estimate convex choice regions from the intersection of the neurons' semi axes. Consequently, a three-layer perceptron may produce any decision areas. Data is propagated based on the input layer to the last layer through the architecture. This is the phase of forward gearbox. The inputs are mixed with the starting weights and applied to the activation function in the weighted sum. Each linear combination spreads to the next layer. Each layer transfers the outcome of its computation as well as its internal data representation to the next layer. This goes through the hidden layers to get to the last layer. The output (the difference between the expected and actual outcome) is used to calculate error, which must be kept to a minimum. The Multilayer Perceptron model uses

back propagation to change the network's weights repeatedly with the goal to lower the expenses associated with function. Finding and updating the model's derivative for each weight in the network.Figure 1 depicts the architecture of the MLP-Mixer model. 3.6 below.

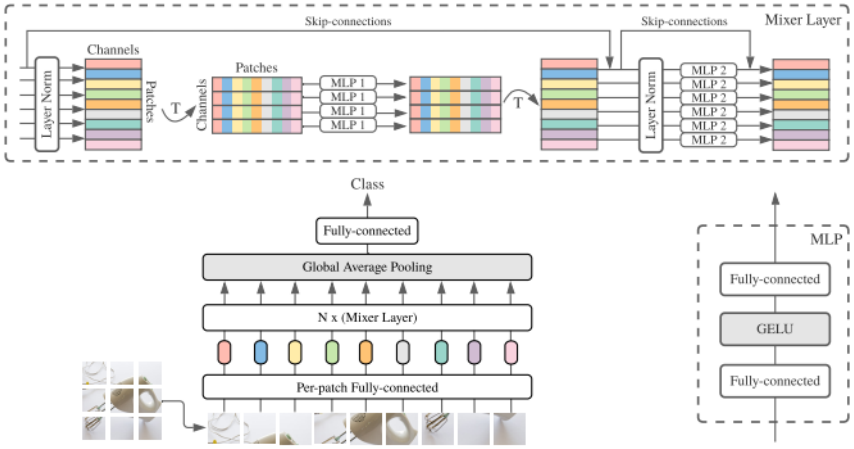

Figure 3.6: MLP-Mixer architecture,[42]

In a neuron,Differentiable functions include the function which incorporates inputs and weights, including the weighted sum, and the threshold function, known as ReLU. Because gradient descent is the optimising function used by multilayer perceptrons, those functions must possess a limited divergence. The standard deviation error gradient is determined throughout every inputs and outputs pairs once the weighted sums have been transmitted through every layer in every iteration that follows. The gradient continues to propagate back by updating the initial hidden layer's weights with the new value. This is how the weights are returned to the neural network's initial location. Gradient Descent iterates until single input-output pair's gradient reaches a point, This represents that the gradient remains the same more radically than the previous stated completion requirements. Finally, the output is put through a threshold function in order to obtain the anticipated class identities. The multilayer perceptron's power originates from non-linear activation functions. Almost any nonlinear function, excluding polynomial functions, can be used for this purpose. As a universal approximator, the multilayer perceptron can approximate any continuous function.

3.3.7 Gated Multilayer Perceptron(gMLP)

The Gated Multilayer Perceptron (gMLP) is a neural network design that combines MLP with attention processes. Attention is used in gMLP to calculate how much each layer of the network contributes to the final output. It is a sort of neural network with a feedforward function that transmits data among layers by selectively employing gating strategies. The Transformer design, a well-known natural language processing (NLP) architecture, serves as the foundation for gMLP's attention mechanism. The transformer's attention mechanism computes the significance of various elements of the input sequence for each output element. It allows the network to concentrate on specific portions of the input data as it processes it. The

basic idea behind the attention mechanism is to use a weighting system to highlight specific sections of the input data during data processing. This weighting technique assigns a weight to each input feature, indicating how important that information is to the output. Attention mechanisms are classified into two types: local attention and global attention (refer to the original study). By focusing on a tiny sample of input data, local attention mechanisms weight the contribution of each neuron in a local network region to the final output. Global attention mechanisms, on the other hand, use the full input data to weight each neuron's contribution to the final output. The attention mechanism works by computing a weighted sum of the data input. By focusing on the most relevant sections of the input data, the attention mechanism can improve neural network accuracy and speed. In gMLP, the attention mechanism is updated to calculate the importance of different layers for the final output. Specifically, gMLP uses global attention to compute the weights of the layers. The attention mechanism takes each layer's output as input and computes a weight for each layer that reflects how important that layer is to the final output. The final output of the network is the weighted sum of the outputs of all layers.
First, the information enters the gMLP's input layer, which is typically a vector. The number of characteristics in the input data determines the number of neurons in the input layer. The information is then selectively passed between network tiers using gating layers. gMLP employs two types of gating layers: channel gating and spatial gating. Channel gating regulates the flow of information along the channel dimension of the input data. It is composed of a series of fully connected layers that process incoming data and generate a gating vector that controls data flow to the following layer. Spatial gating regulates data flow along the spatial dimension of the input data. It has a set of convolutional layers that process the input data and generate a gating map to control the flow of information to the next layer. To perform nonlinear changes on the input data, the MLP layers employ activation functions such as sigmoid or ReLU. The performance of the network can be increased by adjusting the number of MLP layers and neurons in each layer. A gMLP's ultimate output is either a single result (for regression) or a probability distribution across numerous classes (for classification).
gMLPs reduce gating loss during training to encourage gating mechanisms to selectively convey relevant information between layers. Regularisation techniques like dropout or weight decay may be used to avoid overfitting the training data. In gMLP, attention allows the network to focus on different sections of the input. It is particularly useful for activities with long-term dependencies or when certain elements of the input are more significant than others. The selective information transfer between layers of gMLPs improves the network's efficiency and effectiveness. Additionally, utilising attention in gMLP allows the network to accept inputs of varying durations or sizes, which is important in cases where the input data is not always the same size or shape. Overall, gMLP combines the power of MLPs with the adaptability of attention mechanisms to produce a flexible neural network architecture.[43].

3.3.8 Perceiver

Perceiver is a mode-independent neural network design that can handle high-dimensional input data, including audio, video, and pictures. It works smoothly and effectively

on different types of data without relying on too many rules or assumptions for each type. It does not require that the data be processed in a specific manner before being used. When we develop models for different sorts of data, such as photos, sounds, or videos, we generally use "priors," which are assumptions or rules that are specific to that type of data. Models with strong priors function well for that type of data but not for other types. However, perceiver perform as well as models created for each data type (data-specific models). In conventional image models, such as ViT, we must split the image into smaller pieces in order to process it because it is too difficult to process each pixel as an input. But in Perceiver, we can input the entire image as a flattened array of pixels. To accomplish this, positional encodings are used to capture the significance of the pixels' arrangement within an image. It is done by adding the coordinates of each pixel's Fourier features to the end of the data before sending it to the perceiver for processing. This helps the model figure out where each pixel is and how they connect to each other in the image. Positional encodings represent the spatial relationships between pixels. Instead of using learned positional encodings like in ViT, the perceiver model uses coordinate-based Fourier features as positional encodings for expressing the spatial information.
Perceiver's model architecture is simplistic but vast. There are two primary categories of blocks: the cross-attention block with a learnable set function and the latent transformer block. The cross-attention mechanism in Perceiver is similar to the regular self-attention mechanism used in transformer-based models, but there are a few important differences. Instead of computing the dot product of the query and key vectors, the perceiver's cross-attention mechanism computes a more generic similarity measure utilising a trainable linear transformation and a non-linear activation function. The cross-attention block accepts the input vector and the latent representation as Q and K, respectively, and generates a set of attended vectors as V. The Q and K matrices are multiplied to create a matrix of similarity scores, which show how relevant each input element is to each latent representation element. The scores are normalised using a softmax function and multiplied with the V matrix to generate the attended vectors, which comprise information from both the input and the latent representation. In the perceiver, Q is not calculated using the input like it is in a normal transformer encoder. Instead, it is computed from a latent array of the same size ($N \times D$). The cross-attention block in Perceiver uses a learnable set function. This is a crucial feature that enables the perceiver model to process input of arbitrary size and structure. It accepts the cross-attention mechanism's outputs as inputs and generates a fixed-size representation of the input data.
The learnable set function in the perceiver model is responsible for transforming the input, which is a sequence of elements (e.g., pixels in an image), into a set of abstract representations. By applying a learned function to the input sequence, a set of vectors is generated. These vectors present the input in a more abstract and compressed form, and they serve as the input to the cross-attention and latent transformer blocks that follow.The combination of the cross-attention mechanism and the learnable set function enables the perceiver to analyse high-dimensional input data with a fixed number of parameters without demanding any spatial or temporal structure in the data. The perceiver model uses the latent transformer block, which is a type of transformer block, to handle the abstract representations of the input created by the learnable set function. It comprises two sub-blocks: an MSA block and an MLP block. The MSA block helps the model identify re-

lationships between various components in the input, while the MLP block adds a non-linear transformation to the MSA block's outputs. These two subblocks process the input and generate a representation that is appropriate for further processing, like classification and regression.
One of Perceiver's primary strengths is its capacity to interpret high-dimensional input data with a set number of parameters. This makes it valuable for applications with limited memory and computing resources. Another benefit of Perceiver is that it can learn representations of raw data that are unaffected by transformations like rotation, translation, and scaling. This is accomplished by employing the cross-attention mechanism, which enables the network to independently focus on various parts of the input data[44].

3.3.9 InceptionResnetV2

Previous Inception models were trained in a partitioned method, with each replica divided into a number of sub-networks in order to fit the full model in memory. Because the Inception design is extremely flexible, the number of filters in each layer can be altered significantly without affecting the effectiveness of the fully developed model. We used to carefully set layer sizes to balance computation across the many model sub-networks and achieve optimum training speed. Conversely, with the arrival of TensorFlow, our latest models are able to be trained without replicas being segregated. This is due, in part, to the most recent memory advances for reverse propagation, which have been rendered possible by carefully deciding which tensors must be used for gradient computation and organising the computation to use fewer of these tensors. Our studies have been confined to tweaking isolated network components while maintaining network stability, and we have historically been cautious about changing architectural decisions. Networks that did not simplify previous decisions appeared more difficult than they were. Each convolution marked with a V has its whole input patch contained in the preceding layer, and the grid size of the output activation map is modified as a result.
The residual versions of the Inception networks use smaller Inception blocks than those used in the original Inception. The layer of filter expansion (1×1 convolution without activation) that comes after each Inception block is used to scale up the dimensionality of the filter bank before the addition to match the depth of the input. This is necessary to compensate for the dimensionality loss generated by the Inception block. Inception-ResNet-v2 mirrors the unprocessed price of the newest Inception-v4 network. Another small technical contrast between our residual and non-residual Inception versions is that batch normalisation was only implemented on top of the conventional layers for Inception-ResNet, not the summations. Although we believe extensive use of batch normalisation will be beneficial, we wanted to keep the option to train each model replica on a single GPU. Layers with huge activation sizes were discovered to be using an excessive amount of GPU RAM due to their massive memory footprints. By eliminating the batch normalisation layer, we were able to substantially boost the total number of Inception blocks. We predict that with improved computing resource applications, this compromise will be obsolete[45].

3.3.10 EfficientNet B3

Having a strong baseline network is also essential since model scaling does not alter the layer operators F^i in the baseline network. We will test the efficacy of our scaling strategy using current ConvNets, but we have also created a new mobile-size baseline dubbed EfficientNet in order to better illustrate its usefulness.Utilising a multi-objective neural architecture search that maximises FLOPS and accuracy, we build our baseline network. Here, search space is also implemented. Since we are not aiming for any particular hardware device, we are optimising FLOPS rather than latency. We refer to the effective network we find as EfficientNet-B0. The design is comparable to Mnas Net. Net, with the exception that our EfficientNet-B0 is a little bit larger due to the 400M FLOPS target. Squeeze-and-excitation optimization is added to mobile inverted bottleneck MBConv, which serves as the foundation of this system. EfficientNet B0 is the baseline network for all the versions of EfficientNet. A compound scaling approach is used to scale up EfficientNet-B0 in two steps, beginning with the baseline:

• STEP 1: Based on some equations first fix $\varphi = 1$ is needed, assuming there are twice as many resources available, Conduct a quick grid search of α, β, γ .In particular, with the restriction of $\alpha \times \beta^2 \times \gamma^2 \approx 2$ the best values for EfficientNet-B0 are$\alpha = 1.2, \beta = 1.1,$ and $\gamma = 1.15$.

• STEP 2: Using another equation to scale up the baseline network with φ varying parameters, we then fix α, β, γ and use them as constants to achieve EfficientNet-B1 to B7.

Particularly, it is possible to accomplish even higher performance by searching for and physically encompassing a large model, but the search cost climbs to an unsustainable level for bigger models. Our solution avoids this problem by running the search only once on the small foundation network (step 1) and applying the same scaling variables for the models that follow (step 2) [46].

3.3.11 MobileNetV3Large

These layers are used as building blocks to create the most efficient MobileNetV3 models. Swish nonlinearities of various forms also aid layers. The hard sigmoid has been employed rather than the sigmoid, which is used by the squeezing, excitation, and swish nonlinearities but is computationally costly and hard to maintain precision in fixed point arithmetic.

Network search has been shown to be an extremely efficient way of discovering and enhancing network topologies. In order to set up the global network architecture for MobileNetV3, platform-aware NAS can be used to optimise every network block. The NetAdapt approach is then applied when determining the number of filters per layer. Both of these approaches are appropriate and can be paired to find models that are optimised for a particular hardware platform. A platform-aware neural architecture technique is utilised to discover overall network topologies. With a target latency of roughly 80 ms, an RNN-based controller and the same factorised hierarchical search space produce results comparable to huge mobile models. As a result, for the large mobile model, all that is required is to use the original MnasNet-A1 and then add NetAdapt and other optimisations on top of it.

When we find models using architectural search, we see that some of the earlier levels and some of the later layers are more expensive than others. We propose many

architectural modifications to reduce latency while maintaining accuracy in these troublesome layers. These updates are not included in the current search space. The initial change affects how the network's final few tiers interact to deliver the final features more effectively. Current models based on the inverted bottleneck topology of MobileNetV2 and its derivatives use 1×1 as a final layer, convolution is used to expand into a higher dimensional feature space. This layer is crucial for having extensive prediction features. However, there will be an increase in latency as a result. To reduce latency while keeping the high-dimensional features, we advance this layer beyond the final average pooling. This final set of characteristics is now computed at 1×1 1x1 spatial resolution rather than 7×7spatial resolution. This design option has the advantage of practically eliminating delay and computation for feature computing. The preceding bottleneck projection layer is no longer necessary to minimise computation once the cost of this feature-generating layer has been reduced. We may be able to reduce computational complexity even further as a result of this discovery by removing the projection and filtering layers from the preceding bottleneck layer. The final stage effectively reduces the delay by 7 ms, or 11%.It saves operational time and performs 30 million fewer operations with virtually no accuracy loss. The first set of filters is an extra, expensive layer. In latest mobile models, there are 32 filters in all. 3×3 convolution are frequently employed to generate the initial filter banks for edge detection. These filters usually mirror each other. To avoid duplication, we experimented with lowering the number of filters and applying various non-linearities. The hard-swish non-linearity was chosen for this layer since it outperformed the other non-linearities we investigated. We were able to minimise the number of filters from 32 to 16 while retaining the same accuracy by utilising ReLU or Swish. As a result, two milliseconds and ten million MAdds are saved. Swish non-linearity enhances neural network accuracy greatly when used as a drop-in replacement for ReLU. The equation:

$$\text{swish}(x) = x \cdot \sigma(x) \tag{3.3}$$

the non-linearity is described. Although this non-linearity enhances accuracy, it has a non-zero cost in embedded systems due to the higher cost of computing the sigmoid function on mobile devices. We approach this issue in two ways. In place of the sigmoid function, we utilise its piecewise linear hard analog: $\frac{\text{ReLU6}(x+3)}{6}$.The only difference is that ReLU6 is used instead of a distinct clipping constant. Swish in its hard version is given as:

$$\text{h-swish}[x] = \frac{x \cdot \text{ReLU6}(x+3)}{6} \tag{3.4}$$

Constants were chosen since they were straightforward and a suitable match for the original smooth version. We discovered that the hard versions of all of these functions had no noticeable accuracy difference but had several deployment benefits in our experiments. To begin, almost all software and hardware frameworks offer optimal ReLU6 implementations. Second, any potential loss of numerical precision is avoided by using the approximation sigmoid in quantized mode. Finally, h-swish can be used as a piecewise function to significantly reduce the number of memory accesses and hence the delay cost. Because the activation memory of each layer typically reduces by half as the resolution falls, the cost of implementing nonlinearity decreases as we progress deeper into the network. Surprisingly, we discovered that

the majority of the benefits of Swish are only noticed at the deeper levels. As a result, we only use h-swish in the second half of our architecture. Even with these modifications, the h-swish delay penalty persists. The squeeze-and-excite bottleneck was proportionate to the convolutional bottleneck in magnitude. Instead, we repair them all so that the expansion layer is intact.$\frac{1}{4}$ the number of channels. We found that doing so improves accuracy with no apparent delay and only an insignificant rise in the total amount of parameters.[47]

3.3.12 DenseNet201

Convolutional neural networks (CNNs) have demonstrated cutting-edge image classification performance. However, the vanishing gradient problem makes training deep CNNs challenging. This occurs when the gradients of the loss function become progressively narrower as the network deepens, making adaptation difficult.DenseNet, an alternative CNN architecture that addresses the issue of fading gradients, is proposed by the authors. Each network layer in DenseNet is linked to all network levels above and below it. This allows the network to transfer data between layers, keeping gradients from becoming too small. Dense connections have several benefits, such as assisting the network in detecting more complex features from input data and lowering the number of feature maps by exploiting transition layers. As a result, the network can generalise to new data more effectively. It also keeps the network from growing too huge and computationally expensive.DenseNet is a convolutional neural network (CNN) design based on the premise that shorter linkages between layers between input and output allow convolutional networks to be substantially deeper, more precise, and quicker to train. DenseNet201 is an extension of the DenseNet architecture, which is well-known for its cutting-edge performance in a variety of image classification applications. The number and depth of parameters are the primary differences between DenseNet and DenseNet201. DenseNet-201 contains 201 layers compared to 121 layers in DenseNet-121, making it a more powerful network capable of extracting more complicated features from input data. DenseNet201 also has more than twice as many parameters as DenseNet-121, allowing it to learn more complicated representations of the data input. Because of the added layers and parameters, including the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and the COCO detection test, DenseNet201 beat other image classification methods. DenseNet201 is employed when a rather large dataset necessitates a high level of precision.DenseNet201 is a densely connected feedforward deep neural network design that connects each layer to every other layer. Because of this connection arrangement, known as a "Dense" block, DenseNet201 may learn more complex features from the input data. It also enhances information flow between layers and increases feature reuse, allowing for smoother gradient flow and assisting in the learning of more discriminative features. A conventional CNN's layers are only connected to the layers above and below it, whereas DenseNet201's layers are connected to all layers above and below it. Dense connections enable networks to learn more complex properties, avoid the vanishing gradient problem, and outperform on short datasets.DenseNet introduces dense blocks, which are made up of several densely connected layers. The output feature maps of each layer are concatenated with the feature maps of all previous layers in a thick block to create feature maps that fully reflect the input.The authors propose a function consisting

of ReLU activation, batch normalisation, and a 3×3 convolutional layer for dense block layers. This composite function serves as a fundamental DenseNet building block and enables the network to effectively capture complex and hierarchical patterns.

DenseNet201 applies batch normalisation to deep neural network training to stabilise it by normalising the outputs of each layer. As a result, the network is less sensitive to changes in input data. In DenseNet, the growth rate hyper-parameter regulates how many feature mappings each layer contributes to the following layers. It is used to create a compromise between maintaining a complete set of features and minimising the number of parameters. A higher growth rate results in more feature maps and a more complex model.

Transition layers are used in networks to limit the amount of feature maps, manage their development, and prevent overfitting. They consist of a 1×1 convolutional layer, a batch normalisation layer, and an average pooling layer and are positioned between dense blocks. Before forwarding them to the next dense block, they decreased feature maps and minimise the number of channels. This contributes to the network not being too big and computationally costly.

The DenseNet201 network consists of a 7×7 convolutional layer, followed by a max pooling layer with a 3×3 filter. Countless batch normalisation layers, convolutional layers, skip connections, and ReLU activation compose four dense blocks. Following the dense blocks is a transition layer consisting of a 1×1 convolutional layer, a batch normalisation layer, and a 2x2 average pooling layer. The final layers of DenseNet201 consist of a fully connected layer with 1000 units, a global average pooling layer, and a softmax activation layer that generates predicted class probabilities. The effectiveness of DenseNet201 has been demonstrated across a wide range of image classification tasks, encompassing object recognition and segmentation. The dense connectivity of this particular model makes it particularly suitable for applications that involve fine-grained classification or feature detection. This is due to the fact that the model's architecture enables it to extract more specific information from the input images. In contrast to traditional networks, this approach alleviates the issue of vanishing gradients, promotes the reuse of features, and enhances feature propagation. Consequently, it leads to improved parameter efficiency, accuracy, and reduced overfitting.

DenseNets were tested on a range of image classification tasks and shown their cutting-edge performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). They also demonstrated that DenseNets outperformed other deep CNN designs such as CIFAR-100, CIFAR-10, ImageNet, and SVHN in image classification tasks. On the ImageNet dataset, DenseNets, for example, achieved an error rate of 3.57 percent. This was far lower than the error rates of prior deep CNN designs. DenseNet produced comparable results with fewer parameters. ResNet-152, for example, has 60.3 million parameters, while DenseNet-121 has 10.2 million, and Inception-v4 has 144 million.DenseNet201 and DenseNet are two robust convolutional neural network architectures that excel at image classification. It makes use of dense layer connections. DenseNet201 is an improved and more complex version of the original DenseNet architecture, with additional layers and settings for capturing more complex and discriminating data[48].

3.3.13 Extreme Inception(Xception)

The architecture of a convolutional neural network is built on depth-separable convolution layers. In effect, the following hypothesis was proposed: mapping cross-channel correlations and spatial correlations in convolutional neural network feature maps can be completely dissociated. The proposed architecture is known as Xception, which stands for Extreme Inception,because it is a more powerful version of the idea that underpins the Inception design. The Xception architecture's 36 convolutional layers serve as the network's feature extraction foundation. Our convolutional base will be followed by a logistic regression layer because the aforementioned experimental evaluation will only comprise image categorization. Fully connected variables can be introduced before the logistic regression layer, as stated in the experimental assessment section. The 36 convolutional layers are organised into 14 modules, with the exception of the first and end modules, and are surrounded by linear residual connections. Xception is essentially a linear stack of depth-wise separable convolution layers with residual connections. This makes the architecture incredibly simple to build and modify; it just takes 30 to 40 lines of code using a high-level framework like Keras or TensorFlow-Slim, which is close to designs like VGG-16 but significantly more difficult to specify than architectures like Inception V2 or V3. Under the MIT licence, an open-source implementation of Xception using Keras and TensorFlow is available as part of the Keras Applications module 2 [49].

3.3.14 NasNetMobile

In our method, the general structures of the convolutional nets are manually predetermined. They are made up of repeating convolutional cells with the same architecture but varying weights. Given a feature map as input, we need two types of convolutional cells to do two distinct tasks: (1) convolutional cells that return the same dimension, and (2) convolutional cells that return a feature map with a factor of two reduced height and width. Normal cells and reduction cells are the names given to the first and second types of convolutional cells, respectively. We use an initial operation with a stride of two to reduce the height and width of the reduction cell's inputs. Striding is an option for every procedure we consider when building convolutional cells. In convolutional networks, the designs of the normal and reduction cells sought by the controller RNN differ.Each cell in the search space is given two initial hidden states, h_i and h_{i-1} which are either the outputs of two cells in the two lower layers that came before them or the input picture. Given these two starting hidden states, the controller RNN iteratively predicts the remaining structure of the convolutional cell. The controller's predictions for each cell are organised into B blocks, each of which comprises 5 prediction steps made by 5 different softmax classifiers that correspond to discrete choices of the block's elements:
Step 1: Choosing a hidden state from h_i,h_{i-1}, or the collection of hidden states generated in earlier blocks.
Step 2: From the identical options as in the first step, choose a second concealed state.
Step 3. Selecting a procedure to perform on the hidden state chosen in Step 1.
Step 4. Picking a task to perform on the hidden state you chose in Step 2.
Step 5: Determining a technique for combining the results of Steps 3 and 4 to produce a new hidden state.

The newly created hidden state is added to the list of existing hidden states as a potential input in subsequent blocks by the technique. The number of times the controller RNN repeats the five prediction procedures indicated above corresponds to the number of B blocks in a convolutional cell. In our studies, selecting B = 5 yields good results, despite the fact that we have not extensively studied this region due to computer constraints. In phases 3 and 4, the controller, RNN, chooses an operation to perform on the hidden states. In step 5, the controller RNN decides whether to concatenate the two hidden states along the filter dimension or to add them element by element. The final cell output is obtained by concatenating all of the unused hidden states produced in the convolutional cell in depth.We only provide the controller two 5B predictions in total, one for the normal cell and one for the reduction cell, so that the controller RNN can predict both the normal and reduction cells. In our study, we use the NAS proposal for reinforcement learning, although random search can also be used to locate architectures in the NASNet search space. In random search, we can sample from the uniform distribution rather than the softmax classifiers of the controller RNN.

3.3.15 Ensemble Learning(Stacking Method)

An ensemble of individually trained classifiers (such as neural networks or decision trees) combine their predictions when classifying novel situations. According to previous research, an ensemble is frequently more accurate than any of the individual classifiers in the ensemble [51].
The Stacking ensemble learning method was based on the idea of Stacked generalization [50].Defining the level 0 space as the region of the R^{n+1} space that the original learning set θ occupied. The original learning set θ is referred to as a level 0 learning set, and any generalizer that generalises straight off of θ in the level 0 space is referred to as a level 0 generalizer.A set of k numbers produced by (a subset of) the N_{G_j} cooperating with that partition for each of the r partitions of θ,$\{\theta_{i1},\theta_{i2}\}$. The input component of element θ_{i2} (i.e., $G_j(\theta_{i1})$; the input component of θ_{i2}), the input component of element θ_{i2}, or the vector in the input space connecting that input component of θ_{i2} to its nearest neighbor in θ_{i1} can all be examples of what these k numbers can be in practice. Consider each of these sets of k numbers as the input portion of a point in the space R^{k+1}. Each such point's corresponding output value is determined using the output component of the relevant θ_{i2}, maybe in conjunction with $G_j(\theta_{i1})$; the input component of θ_{i2}. This area R^{k+1} is referred to as the level 1 space. There are r points in the level 1 space since there are r partitions of θ. The level 1 or reduced learning set refers to those r points. By using a generalizer in the level 1 space, we want to generalize from. There are numerous ways we may accomplish this. The typical approach is to start with a question in the level 0 space, transform it using the same processes that created the level 1 learning set's input components, and then answer that level 1 question by extrapolating from the level 1 learning set. Then, this level 1 estimate is downgraded to a level 0 guess. The output components of the θ_{i2} are utilized to calculate the output components of the level 1 learning set, which determines how said transformation is carried out. This kind of generalizing procedure is referred to as stacked generalization. Iterating the operation as a whole will produce levels $p > 1$ (many stackings). For the time being, the above-described two levels of layered generalization will suffice. It

is significant to emphasize that many-layered generalization-related components are currently black art.

3.3.16 Gradient-weighted Class Activation Mapping

Gradient-weighted Class Activation Mapping (Grad-CAM) creates a coarse localization map by utilising the gradients of any target idea flowing into the final convolutional layer to highlight crucial locations in the image for concept prediction. Deeper representations in a CNN, according to earlier research, are capable of capturing higher-level visual constructs. Because convolutional layers save spatial information that would otherwise be lost in fully linked layers, the final convolutional layers should provide the optimum combination of high-level semantics and precise spatial information. The neurons in these layers search the visual for information relevant to a semantic class, such as object fragments. Grad-CAM assigns relevance values to each neuron for a particular choice of interest by feeding gradient information into the last convolutional layer of the CNN. Although the method is fairly broad and can be used to explain activations at any deep network layer, in this study it is largely focused on explaining decisions made at the output layer.
Up until the final convolution layer, to which the gradients are propagated, the precise computation consists of successive matrix products of the weight matrices and the gradient with respect to activation functions. When computing neuron significance weights while backpropagating gradients with respect to activations, this is done. As a result, this weight represents a partial linearization of the deep network downstream from A and captures the importance of feature map k for a target class c. To do this, we weight forward activation maps with a ReLU. We apply a ReLU to the linear combination of maps since we are only interested in characteristics that have a positive impact on the class of interest. Negative pixels are most likely found in other image forms. Localization maps that lack this ReLU usually highlight classes other than the ones intended and perform poorly in terms of localization [53].

3.4 Evaluation Metrics

We compared the performance of classification models using four widely used performance metrics: accuracy (Equation 3.5), precision (Equation 3.8), recall (Equation 3.7), and F1-score (Equation 3.6). The accuracy, confusion matrix, training and validation loss (graph), and training and validation accuracy (graph) are the primary metrics used to evaluate a model.True Positive (TP) refers to the number of positive classes that have been correctly classified. True Negative (TN) refers to the number of negative classes that have been correctly classified. False Positive (FP) is the inaccurate classification of a positive class. False Negative (FN) refers to the number of negative classes that have been inaccurately classified. P, R, F, and A stand for precision, recall, F1-score, and accuracy, respectively. The F1 score is a quantifiable metric. It describes the precision of the model for a given dataset. It is used in binary classification methods. The model integrates precision and recall. The accuracy of a model is the proportion of correct predictions relative to the total number of predictions. Frequently, the confusion matrix is referred to as an error matrix. It assists in visualizing a model's performance. It records the number of

correct and incorrect predictions made during testing.

$$Accuracy, A = \frac{TN + TP}{TN + FP + TP + FN} \tag{3.5}$$

$$F1 - score, F = \frac{2 \times R \times P}{R + P} \tag{3.6}$$

$$Recall, R = \frac{TP}{FN + TP} \tag{3.7}$$

$$Precision, P = \frac{TP}{FP + TP} \tag{3.8}$$

3.5 The proposed methodology

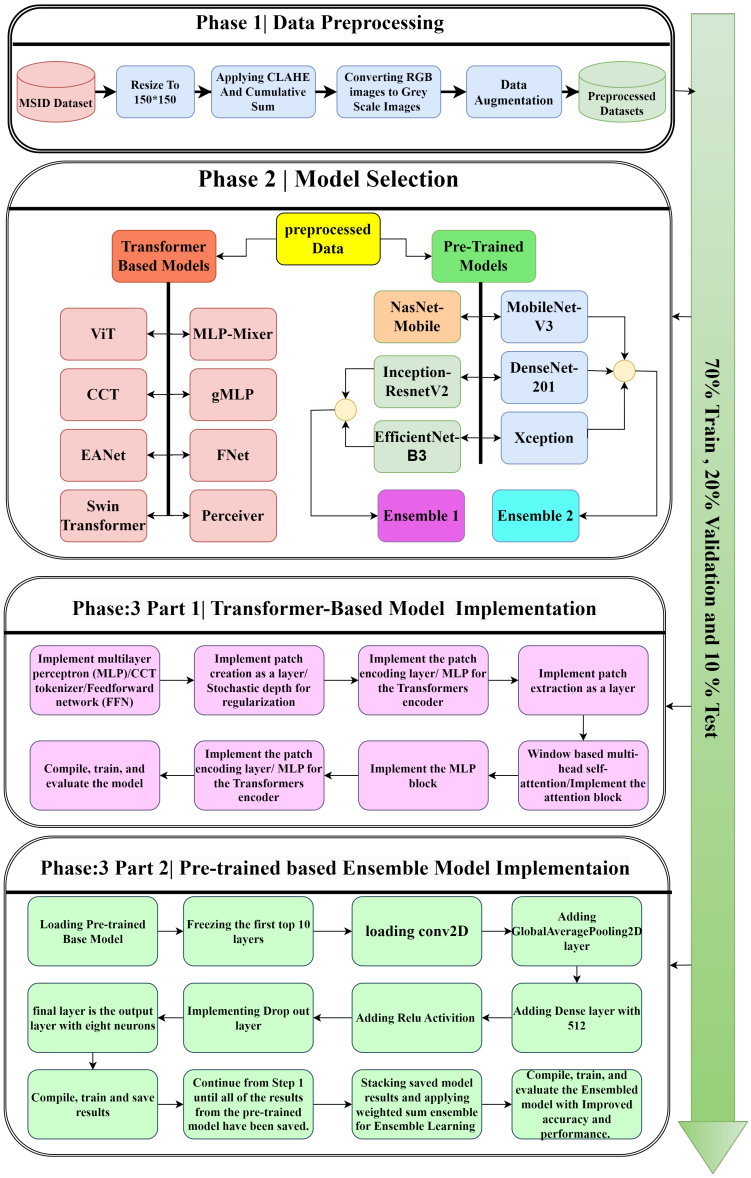

Figure 3.7: The proposed methodology

Fig. 3.7 illustrates the overall system for the detection of monkeypox, which consists of several phases. Raw images were first passed through the preprocessing pipeline.

Data augmentation (resizing, shuffling, and normalization) was done in the preprocessing pipeline. The pre-processed data set was then partitioned into a training set and a testing set, and we trained the selected models in phase 2 using the training data. After each epoch, the training accuracy and loss were determined. At the same time, validation accuracy and loss were also obtained. The performance of the proposed system was measured with the following evaluation metrics: confusion matrix, precision, accuracy, recall, and F1-score.

3.6 Workplan

We started our investigation by looking for available datasets on platforms like Kaggle and others. Finally, we identified the MSID dataset from [25]. It was a hybrid dataset that was gathered. We began pre-processing the dataset as soon as it was collected. A few procedures were included in the preprocessing stage, such as resizing the dataset and performing CLAHE with cumulative sum to improve the contrast of the photos. We divided the dataset into three parts: a 70% training set, a 10% testing set, and a 10% remaining portion for validation before choosing the models to run the dataset. Subsequently,we began the model selection process, which includes steps like identifying the most recent models that have not previously been employed to determine and categorise Mpox. Two distinct groups of the models we chose were: 1) transformer-based models such as the ViT, CCT, EANet, Swin Transformer, MLP-Mixer, gMLP, FNet, and perceiver; and 2) pre-trained models like Xception, NasNetMobile, MobileNetV3Large, EfficientNetB3, and Inception-ResnetV2. The pre-trained models are then split into two groups using ensemble learning, with Ensemble 1 being made up of MobileNetV3Large, DenseNet201, and Xception, and Ensemble 2 being made up of InceptionResnetV2 and EfficientNetB3 groups. Following the model selection phase, we began to set up and execute the models through hyperparameter tuning. Then, in order to tell the difference between photos of Mpox(monkeypox), chickenpox, measles, and normal skin, we assembled, trained, and evaluated the models that were deployed. The accuracy of the models was then ranked from highest to lowest. To ensure that we are receiving the appropriate work, we took advantage of GradCAM to examine the results of our research planning. Figure 3.8 illustrates how our research plan developed over time.

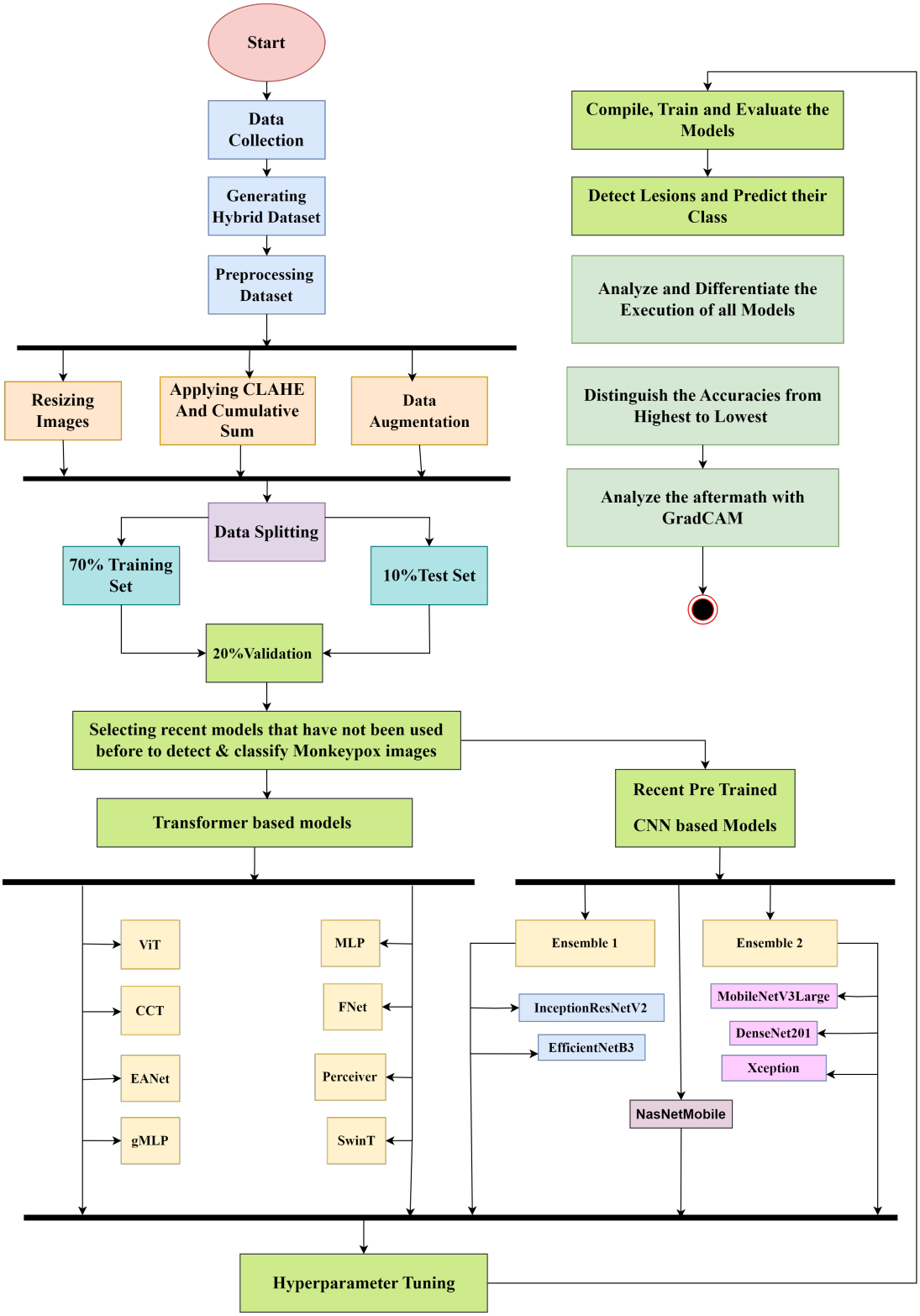

Figure 3.8: Workplan flowchart

Chapter 4

Implementation

4.1 Setup for Experimentation

4.1.1 Computational Software & Training Hardware used in this study

Experiments have been carried out on a device with the hardware characteristics demonstrated here : The CPU of the Intel Core i7 12th Generation, The GPU of the NVIDIA RTX3080 Ti GeForce with 12 GB of memory, and memory of 64GB. In contrast, the software parameters included a Windows 10 Pro platform with CUDA 11.3.1, Cudnn 8.2.1, TensorFlow 2.6.0, and JupyterLab 3.5.3 IDE with Python 3.11.3. The trials were carried out in accordance with these specifications which were $\geq$ 32 phases, allowing the model training procedure to be completed in a short amount of time.

4.1.2 Library List

The libraries used in this research are OpenCV, Tensorflow, Keras, Matplotlib, Scipy, SKLearn, Seaborn, Numpy, Pandas, Pathlib

4.1.3 Structural view of the code skeleton

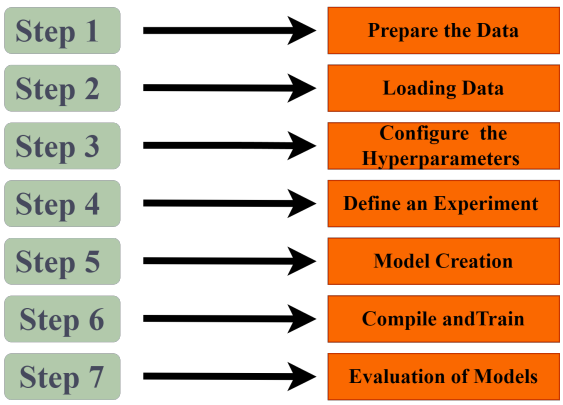

Figure 4.1: Structural view of the code skeleton

4.2 Pre-processing & Augmentation

4.2.1 Pre-processing (CLAHE)

Spreading data images, resizing input datasets, using CLAHE to apply histogram equalization, and enhancing the data all served as preprocessing steps for the data set. We did not need to conduct feature extraction from the dataset. It is conducted when there is a binary or garbage value in the data. As we do not have any garbage or binary values, we can use the whole dataset. First, all of the picture data was downsized to 150*150. The pictures were converted from PNG to JPG. JPG data

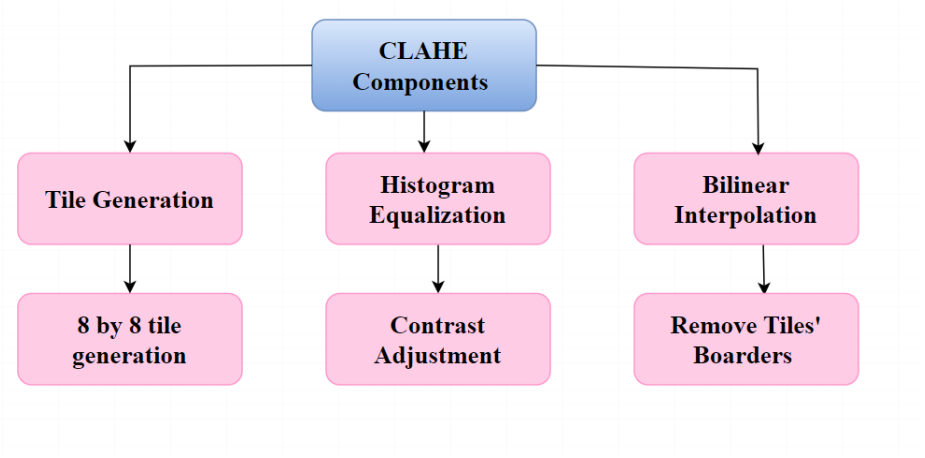

Figure 4.2: Componenets of CLAHE

is easier to train. JPG optimizes space usage and file size. Then most histogram images were 0–255. It fluctuated. We imported OpenCV and Matplotlib to plot the graph and read the picture from a NumPy array to verify the issue. Too high range bar point. The extreme left and right portions also lack pixel intensity values. Our test picture lacks contrast. Histogram equalization (HE) spread pixel intensity across the range to fix this. Constrained histogram equalization works well. In large areas with bright and dark pixels, the histogram won't work properly [39]. Histogram equalization (HE) did not fix it. Next, we used Contrast Limited Adaptive Histogram Equalization (CLAHE) [Fig.4.2], which addresses both the insufficient contrast and the too-bright or too-dark HE issues [2]. Balanced luminance channel values are much higher than BGR image values with equal channel values. CLAHE implementation focused on two parameters: Limit contrast with clip limit. 3.0 is tile standard. GridSize determines column and row tile counts. 8 x 8. Tiled pictures for CLAHE use it. Two datasets were created. One with cumulative sum and CLAHE, the other without. After CLAHE, we grayscaled all photos. Grayscale images make description easier and require less computer power than color photos. Grayscale provides perfect skin tones, softer shadows and gradients, clear small- and knock-out designs, and fine detail. More grayscale levels can improve color transitions and detail. Gray-scale images were more accurate than colorful ones. Grayscale photos are recommended since color does not affect image categorization.

4.2.2 Augmentation of datasets

We used dataset augmentation to add images to our small dataset. Pipeline image improvement methods are shown in Fig.4.3. Before augmenting, all photos in each class's training and validation sets were scaled to 224 by 224 pixels.30 enhancements were made. If we augment, bias will occur. Bias occurs when data is enhanced 50 times. The Keras image processing framework's ImageDataGenerator package expands the dataset. Rotation, inversion, width, and height adjustments are available in ImageDataGenerator. ImageDataGenerator is included. This parameter is added to the dataset in Table 4.1. Thus, facility and generator types are randomly selected [31]. Keras, a Python library, was used to build our model. The parameters have been adjusted: Online data enrichment. After resizing each image to 150*150 for augmentation, we use these settings:
rescale = 1/255, rotating limits = 45 degrees, width shift limit = 0.2, height shift limit = 0.2, shear limit = 0.2, zoom limit = 0.2, horizontal turnaround=true, vertical turnaround=true, and filling mode=reflective. 70%, 20%, and 10% of the data set are reserved for training, validation, and testing purposes, respectively. The ensemble requires no dataset partitioning. The model is divided when it is executed.

Generator Type	**Facility**
Rotation Range	Randomly 0 to 45 degree
Height Shift Range	till 20%
Width Shift Range	till 20%
Shear limit	20%
Zoom limit	20%
vertical turnaround	Positive
Horizontal turnaround	Positive
Staffing function	Reflective

Table 4.1: Parameters used in data augmentation

Category or Class	**Samples before augmentation**	**Samples after augmentation**
Monkeypox	279	8928
Chicken pox	107	3424
Measles	91	2912
Normal	293	9376

Table 4.2: Number of samples before & after augmentation

4.3 Model selection

To train on the Monkeypox Skin Images Dataset (MSID), we selected the latest state-of-the Art image classification architectures such as ViT, CCT, MLP-Mixer, gMLP, FNET Swin Transformer, EANet, Perceiver, InceptionResnetV2, EfficientNetB3, MobileNetV3Large DenseNet201, and Xception. We did not need to extract

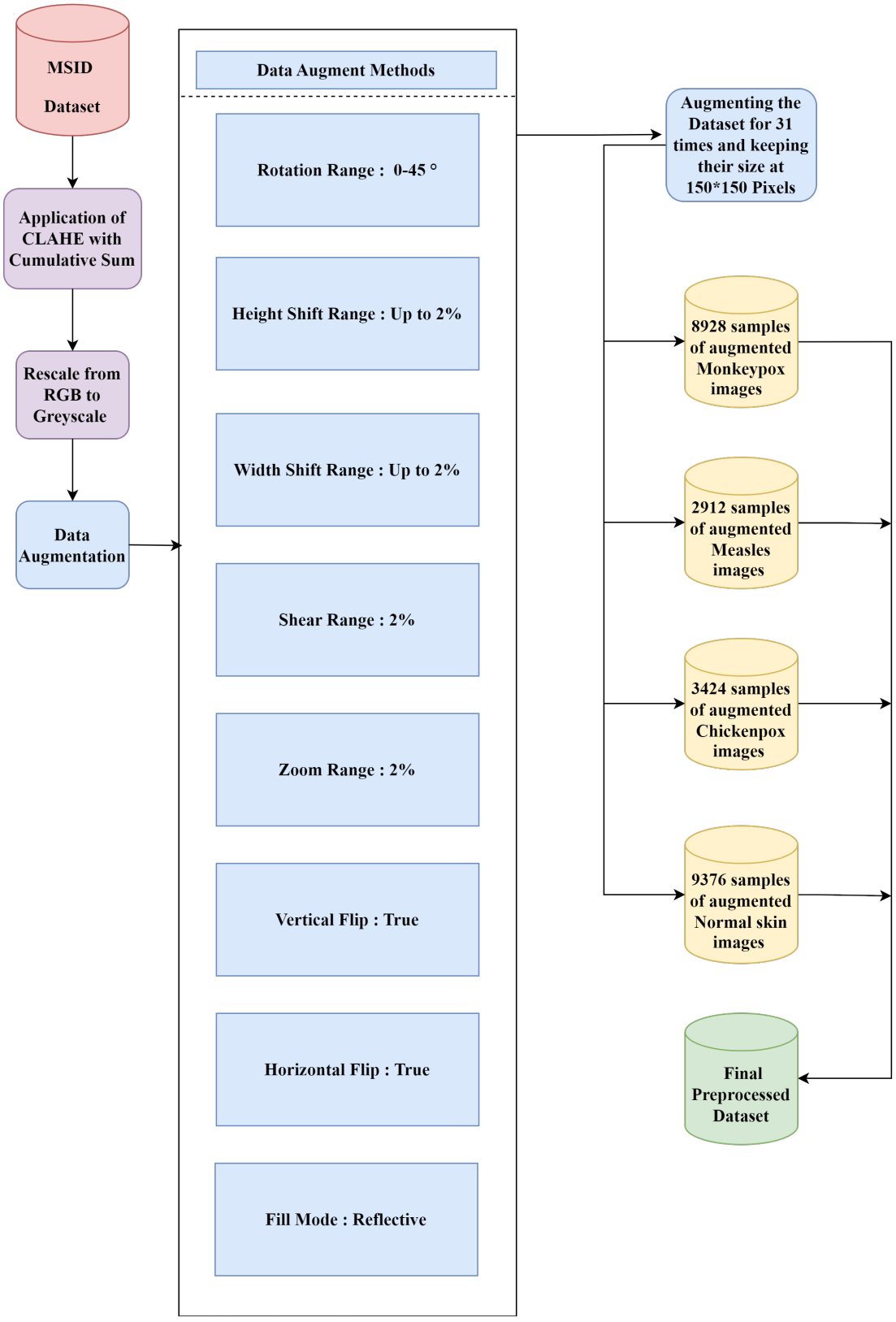

Figure 4.3: Data Pre-processing & Augmentation Diagram.

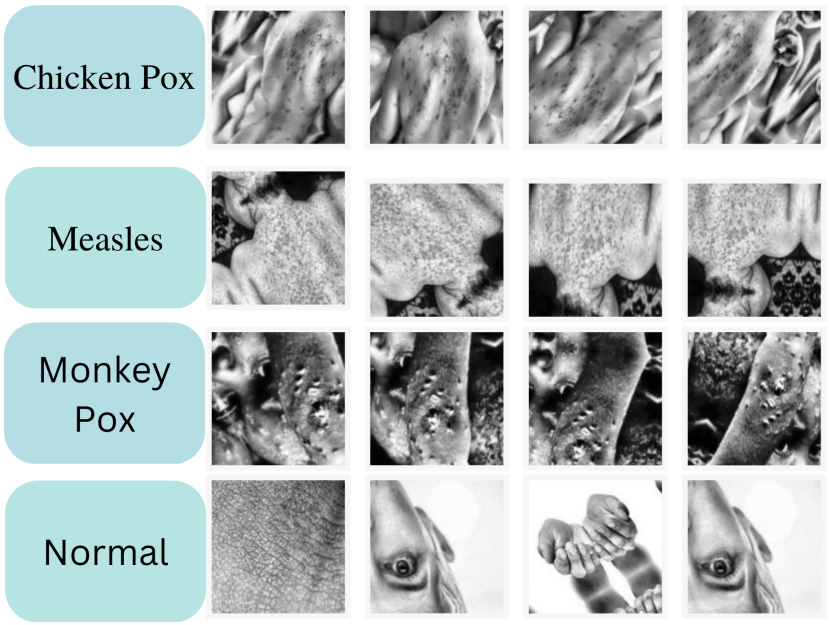

Figure 4.4: Dataset Images after applying CLAHE and data Augmentation

any features from the dataset. During training, the selected models learn to extract features. When it came to model selection, we placed a strong emphasis on both uniqueness and robustness. To that end, we took great effort in selecting Transformer-based models, which had never been applied to this particular subject or datasets previously. In addition, we used pre-trained models to implement the Ensemble technique in order to get the highest possible level of performance. Not only that, but we also attempted to experiment with the effects that transformer-based models have on our datasets. This is because a transformer-based self-attention mechanism allows any dataset to compute in parallel, which may be a benefit and can be a tough rival to previously employed pre-trained models.

4.4 Hyperparameter Tuning

Hyperparameters are parameters with adjustable values. An algorithm can perform well if the hyperparameters are appropriately chosen. Hyperparameters have a direct impact on model layout, functionality, and performance. Choosing the proper hyperparameter values is a crucial aspect of the effectiveness of machine learning techniques. Thus, the accuracy score of a neural network model is enhanced. To obtain the best results with each of our models, we had to manually tune all of the hyperparameters. The learning rate, weight decay, batch size, input shape, epochs number, image size, patch size, patch number, projection dimension, head numbers, transformer layer, stochastic depth rate, dropout rate, attention dropout, projection dropout rate, number of transformer blocks, embedding dimension, MLP dimension, and dimension coefficient were the hyperparameters that we had to manually tune to achieve the best results.
In Table 4.3, the initial hyperparameters for ViT (Vision Transformer) were (32,

32, 3), The rate of learning = 0.001, decay of weight = 0.0001, size of batches = 256, epoch number = 200, size of images = 32, size of patches = 6, dimension of projection = 64, head numbers = 4, layers of transformer = 8, head units of MLP = [2048, 1024]. With these values, our accuracy for 200 epochs was 81%. Thus, it was clear that more manual adjustment could result in greater precision and superior matrices. On the second attempt, the image size was changed to 72 and the input shape was changed to (72, 72, 3), which increased the result for 200 epochs to 86%. On the third attempt, the image size was changed to 128 and the input shape was changed to (128, 128, 3), which increased the result for 200 epochs to 88%. On the fourth attempt, the image size was changed to 256 and the input shape was changed to (256, 256, 3), but the result for 200 epochs was reduced to 84%. As a result, we determined the picture size and input shape to be 128 by 128 (128, 128, 3). Next, the rate of learning was reduced to 0.01, and the decay of weight was also reduced to 0.01, but the expected result of 82% was not achieved. Therefore, preserving the default learning rate and weight decay provided the best possible outcome. Following that, we decided on a batch size that ranges from 16 to 512 and is a power of two. In general, a size 32 is a reasonable starting point. However, in our experiment with ViT, 32-batch size produced 84%, while 500-batch size produced 92%, the highest result to date. Thus, we decided to maintain a batch size of 500. However, the revolutionary aspect of the hyperparter matrix was the dropout rate. The greatest accuracy for 200 epochs was achieved when the dropout rate was increased from 0.01 to 0.05. The dropout rate was then decreased from 0.05 and increased from 0.05, but the accuracy did not exceed 94%. In addition, all other hyperparameters were adjusted manually to optimize the result, which eventually attained 94% accuracy and 96% precision for ViT.

DR	**IS**	**LR**	**WD**	**BS**	**A(%)**	**P(%)**
0.01	32	0.001	0.0001	256	81	84
0.01	72	0.001	0.0001	256	86	88
0.01	128	0.001	0.0001	256	88	90
0.01	256	0.001	0.0001	256	84	85
0.01	128	0.01	0.001	256	76	80
0.01	128	0.001	0.0001	256	80	83
0.03	128	0.001	0.0001	256	90	93
0.05	128	0.001	0.0001	256	92	94
0.05	128	0.001	0.0001	512	94	95

Table 4.3: Step by step manual Hyperparameter Tuning for ViT. Here DR=Dropout Rate, IS=Image size, LR=Learning Rate, WD=Weight Decay, BS=Batch Size, A=Accuracy, P= Precision

Again, in Table 4.4 the initial hyperparameters for CCT (Compact Convolutional Transformers) input shape were (32, 32, 3), rate of learning = 0.001, decay of weight = 0.0001, size of batch = 128, layer of transformer = 2, epochs number = 200, and size of images = 32. With these values, the accuracy for 200 epochs was 84%. On the second attempt, the image sizes and input shape were changed to 72 and (72, 72, 3), respectively, and it produced 86%, a better result than the initial result of 84%. Then, after adjusting it to 128 and (128, 128, 3), the accuracy for 200 epochs

increased to 89%, indicating that increasing image size and input shape produced better outcomes. However, after increasing it to 256 and (256, 256, 3), the accuracy for 200 epochs dropped to 87%. Therefore, it was evident that maintaining 128 and the input shape (128, 128, 3) was the optimal choice. Next, decreasing the batch size for CCT to 64 increased the accuracy for 200 epochs to 94%, while increasing the dropout rate from 0.01 to 0.05 increased the accuracy to 95%. The best results for 200 epochs were achieved with a rate of dropout of 0.1 and a batch size of 32, which resulted in 97% accuracy and 99% precision.
Again, the initial hyperparameters for CCT (Compact Convolutional Transformers) input shape were (32, 32, 3), rate of learning = 0.001, decay of weight = 0.0001, size of batch = 128, Layer of transformer = 2, epochs number = 200, and size of images = 32. With these values, the accuracy for 200 epochs was 84%. On the second attempt, the image sizes and input shape were changed to 72 and (72, 72, 3), respectively, and it produced 86%, a better result than the initial result of 84%. Then, after adjusting it to 128 and (128, 128, 3), the accuracy for 200 epochs increased to 89%, indicating that increasing image size and input shape produced better outcomes. However, after increasing it to 256 and (256, 256, 3), the accuracy for 200 epochs dropped to 87%. Therefore, it was evident that maintaining 128 and the input shape (128, 128, 3) was the optimal choice. Next, decreasing the batch size for CCT to 64 increased the accuracy for 200 epochs to 94%, while increasing the dropout rate from 0.01 to 0.05 increased the accuracy to 95%. The best results for 200 epochs were achieved with a rate of dropout at 0.1 and a size of the batch at 32, which resulted in 97% accuracy and 99% precision.

DR	**IS**	**LR**	**WD**	**BS**	**A(%)**	**P(%)**
0.01	32	0.001	0.0001	128	84	86
0.01	72	0.001	0.0001	128	86	88
0.01	128	0.001	0.0001	128	89	90
0.01	256	0.001	0.0001	128	87	88
0.01	128	0.01	0.001	256	86	88
0.01	128	0.001	0.0001	64	93	96
0.03	128	0.001	0.0001	64	94	96
0.05	128	0.001	0.0001	64	95	97
0.05	128	0.001	0.0001	32	97	97

Table 4.4: Step-by-step manual Hyperparameter Tuning for CCT. Here DR=Dropout Rate, IS=size of Image, LR=Rate of Learning, WD=Decay of Weight, BS=Size of Batch, A=Accuracy, P= Precision

4.5 Baseline Evaluation

We were able to improve the accuracy, precision, and remainder of all the matrices by manually changing the hyperparameters for each model to reach the best results possible. As with Vision Transformer Hyperparameter Tuning and Compact Convolutional Transformers, we implemented the hyperparameters and constantly tuned them for all models and architectures until we found the optimal ones for each architecture, shown in Table 4.5. In addition, each accuracy and precision matrix

Model	**DR**	**IS**	**LR**	**WD**	**BS**	**A(%)**	**P(%)**
ViT	0.05	128	0.001	0.0001	500	94	95
CCT	0.1	128	0.001	0.05	32	97	97
MLP-Mixer	0.5	72	0.001	0.0001	64	92	93
gMLP	0.5	72	0.001	0.0001	64	93	93
Fnet	0.03	72	0.001	0.0001	32	87	83
Swin transformer	0.03	72	0.001	0.0001	32	72	75
EAnet	0.02	72	0.001	0.0001	32	65	87
Perceiver	0.01	72	0.001	0.0001	16	65	89
NASNetMobile	0.04	140	0.001	0.0001	128	95	94
InceptionResnetV2	0.4	140	0.001	0.0001	128	96	96
EfficientNetB3	0.4	140	0.001	0.0001	128	98	98
Ensemble 1(InceptionResnetV2 + EfficientNetB3)	0.4	140	0.001	0.0001	128	99	99
MobileNetV3Large	0.4	140	0.001	0.0001	128	84	82
DenseNet201	0.4	140	0.001	0.0001	128	97	98
Xception	0.4	140	0.001	0.0001	128	98	99
Ensemble 2(MobileNetV3Large+ DenseNet201+ Xception)	0.4	140	0.001	0.0001	128	99	99

Table 4.5: Baseline Evaluation (after hyperparameter tuning for each model multiple times). Here DR=Dropout Rate, IS=size of Image, LR=Rate of Learning, WD=Decay of Weight, BS=Size of Batch, A=Accuracy, P= Precision

was included to show that the hyperparameter setting was correct. On the final attempt, the image size was changed to 72 and the input shape was changed to (72, 72, 3), which increased the result for 200 epochs to 86%. On another attempt, the image size was changed to 128 and the input shape was changed to (128, 128, 3), which increased the result for 200 epochs to 88–97%. On the fourth attempt, the image size was changed to 256 and the input shape was changed to (256, 256, 3), but the result for 200 epochs was reduced to 80–87% for most of the models. As a result, we determined the picture size and input shape to be 128 by 128 (128, 128, 3). Hyperparameters have direct influence over the layout, functionality, and performance of models.
By adjusting hyperparameters, data scientists may improve the performance of their models. The achievement of machine learning techniques is contingent on selecting the appropriate hyperparameter quantities, which is a crucial stage in the efficacy of this technique. As a result, the accuracy score for a neural network model is improved.

4.6 Training Models

4.6.1 Vision Transformer (ViT)

ViT (vision transformer) models surpass CNNs in terms of computing effectiveness and accuracy by a factor of roughly four. This breakthrough in deep learning architectures is groundbreaking because it eliminates the requirement for recurrent connections and convolutions. ViT has a more consistent representation of features across all tiers. Each layer's representation is said to be identical. ViT may compile global data ahead of time due to self-attention. ViT efficiently reproduces representations at all levels, from the most fundamental to the most advanced. However, training the Transformer from scratch on small datasets can be difficult, so we improved it and applied it to large datasets. Essential libraries like TensorFlow and Numpy were called first in our implementation. After more than 8 attempts of hyperparameter tweaking, it was adjusted to a 128*128 picture size and input shape of (128, 128, 3) with a batch size of 500, the rate of learning at 0.001, a patch size of 16, and a decay of weight at 0.0001 for dataset preparation. Across 200 epochs, the ViT (Vision Transformer) achieved 94% accuracy and 96% precision in Fig.4.4. The multilayer perceptron (MLP) was then introduced. The top layer of the two-layer classification network known as MLP is a Gaussian Error Linear Unit. The power that came out of the converter was supplied by the very last MLP block, which is also known as the MLP head. Following that, patch production layer implementations and patch encoding layer implementations were utilized. Finally, the code for the ViT model was written, with each transformer block divided into two independent layers. The code was then written, trained, and evaluated on test data, yielding a top-5 accuracy of 94%, a precision of 96%, a recall of 100%, and an f1-score of 94% shown in Table 4.6. The sequence's input length also serves as the transformer's operational length. Using a trainable linear projection, we compress the patches and determine the transformer's constant latent vector size D for each of its layers. This projection yielded the patch embeddings.

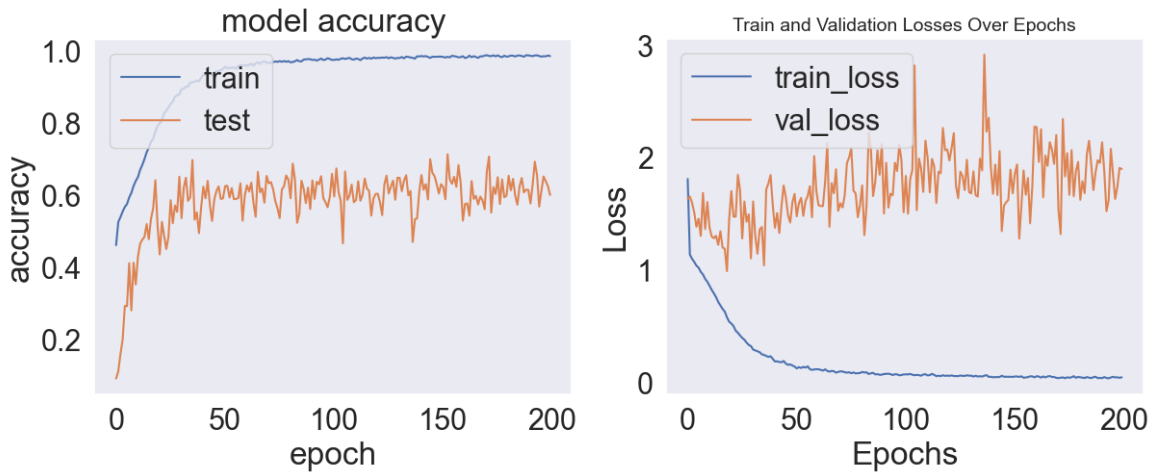

Figure 4.5: Line graphs illustrating the Train as well as Validation Accuracy and Loss values of ViT during the course of 200 training epochs

	P	**R**	**F**	**S**
Chickenpox	0.89	1.00	0.94	649
Measles	0.96	1.00	0.98	602
Monkeypox	0.90	1.00	0.95	1807
Normal	1.00	0.85	0.92	1980
Accuracy			0.94	5038
macro average	0.94	0.96	0.95	5038
weighted average	0.94	0.94	0.94	5038

Table 4.6: Classification Report of Vision Transformer (ViT), where P = Precision, R= Recall, F = f1-score , S = support

4.6.2 Compact Convolutional Transformers(CCT)

Compact Convolutional Transformers use sequential pooling to replace the patch embedding with a convolutional embedding, improving explanatory bias and reducing the demand for position embeddings. CCT, in addition to expanding the number of input variables, provides greater precision than ViT-Lite (which employs smaller ViTs). CCT's hyperparameters and constraints were stochastic depth rate of 0.1, transformer layers of 2, batch size of 32, image size of 128, rate of learning at 0.001, decay of weight at 0.05, convolutional layers of 2, and dropout rate of 0.1, which were tuned numerous times. The CCT writers' initial approach for image handling was the tokenizer. Following CCT tokenization, the layer-dropping stochastic depth for regularisation technique was applied. Inference has layers. Dropout is similar, except it operates on a block of layers rather than a single node. Transformer encoder residual blocks appear before stochastic depth in CCT. The MLP for a transformer encoder was added. The final CCT model was then incorporated. Compact Convolutional Transformers use process pooling to replace patch insertions with convolutional embeddings, which improves inductive bias and eliminates the need for positional embeddings. After 200 iterations of gathering and training, the test data with parameters of 407,365 yielded results of 97% accuracy, 99% precision, 98% recall, 97% f1-score, and 100% top-5 accuracy shown in Fig.4.5 and Table 4.7.

It outperformed other transformer-based models.

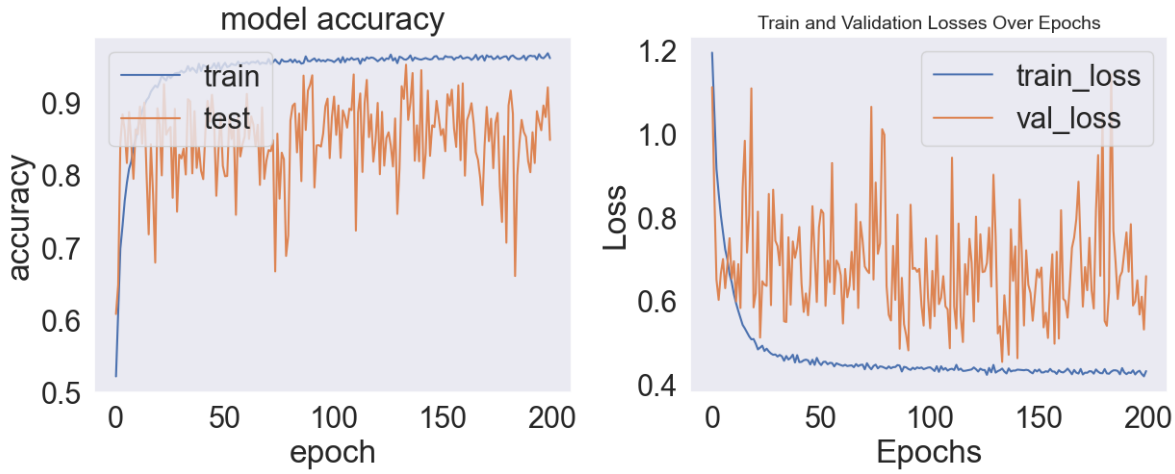

Figure 4.6: Line plots of the Train as well as Validation Accuracy and Loss values of CCT over 200 training epochs

	P	**R**	**F**	**S**
Chickenpox	0.96	0.95	0.96	649
Measles	0.99	0.91	0.95	602
Monkeypox	0.97	0.97	0.97	1807
Normal	0.96	0.98	0.97	1980
Accuracy			**0.97**	5038
weighted average	0.97	0.97	0.97	5038
macro average	0.97	0.95	0.96	5038

Table 4.7: Classification Report of Compact Convolutional Transformers (CCT), where P = Precision, R= Recall, F = f1-score, S = support

4.6.3 MLP-Mixer

The picture size was set to 72, the batch size to 64, the learning rate to 0.001, the patch size to 4, the number of blocks to consist of 4, the weight decay to 0.0001, the embedding dimension to 256, and the dropout rate to 0.5 for the custom hyper-parameter tuning. Patch extraction was then put into practice. The MLP-Mixer model was then used. The architecture known as the MLP-Mixer is made completely of multi-layer perceptrons (MLPs). It has two different types of MLP layers: one that has been applied to picture patches and combines the global features separately from the per-location characteristics. The alternative technique for blending location data along channels and across patches The MLP-Mixer model achieved 92% accuracy, 96% precision, 98% recall, 94% f1-score, and 1,068,299 parameters after being built, trained, and evaluated across 200 epochs shown in Fig.4.6 and Table 4.8.

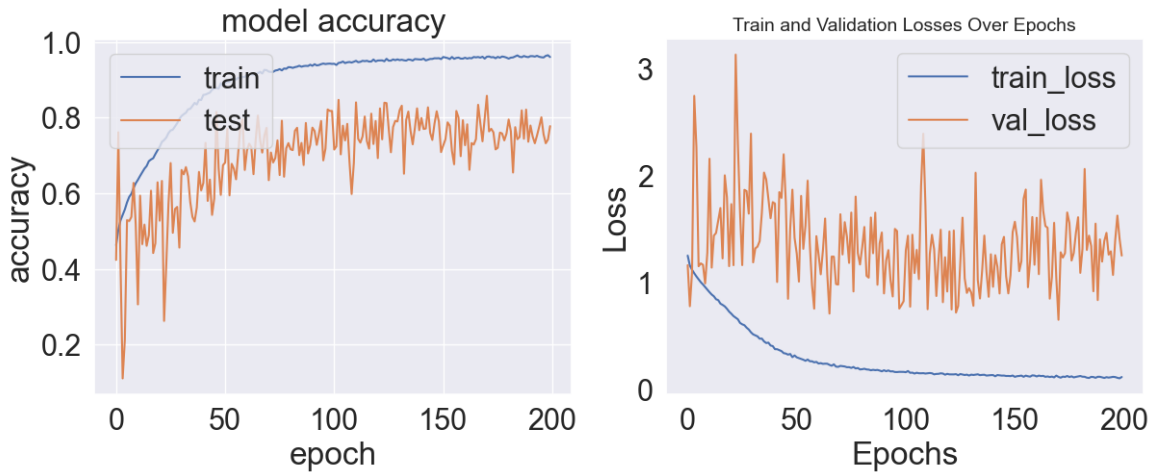

Figure 4.7: Line plots of the Train as well as Validation Accuracy and Loss values of MLP-Mixer over 200 training epochs

	P	**R**	**F**	**S**
Chickenpox	0.83	0.96	0.89	649
Measles	0.85	0.98	0.91	602
Monkeypox	0.95	0.89	0.92	1807
Normal	0.96	0.92	0.94	1980
Accuracy			0.92	5038
weighted average	0.93	0.92	0.92	5038
macro average	0.90	0.94	0.92	5038

Table 4.8: Classification Report of MLP-Mixer, where P = Precision, R= Recall, F = f1-score , S = support

4.6.4 gMLP

The gMLP architecture combines an SGU (Spatial Gating Unit) with an MLP design. By geographically modifying the input via linear projections over a patch (along a channel), the SGU permits cross-patch communications in the spatial (channel) domain. gMLPs reduce gating loss during training to encourage gating mechanisms to selectively convey relevant information between layers. Regularisation techniques like dropout or weight decay may be used to avoid overfitting the training data. In gMLP, attention allows the network to focus on different sections of the input. It is particularly useful for activities with long-term dependencies or when certain elements of the input are more significant than others. combining the input with spatial transformation and element-wise multiplication. The picture size was set to 72, the size of the batch to 64, the rate of learning to 0.001, the patch size to 4, the number of blocks to consist of 4, the weight decay to 0.0001, the embedding dimension to 256, and the dropout rate to 0.5 for the custom hyperparameter tuning. The MLP-Mixer model displayed 93% accuracy, 98% precision, 99% recall, a 94% f1-score, and 823,819 parameters after being built, trained, and validated across 200 epochs shown in Fig.4.7 and Table 4.9.

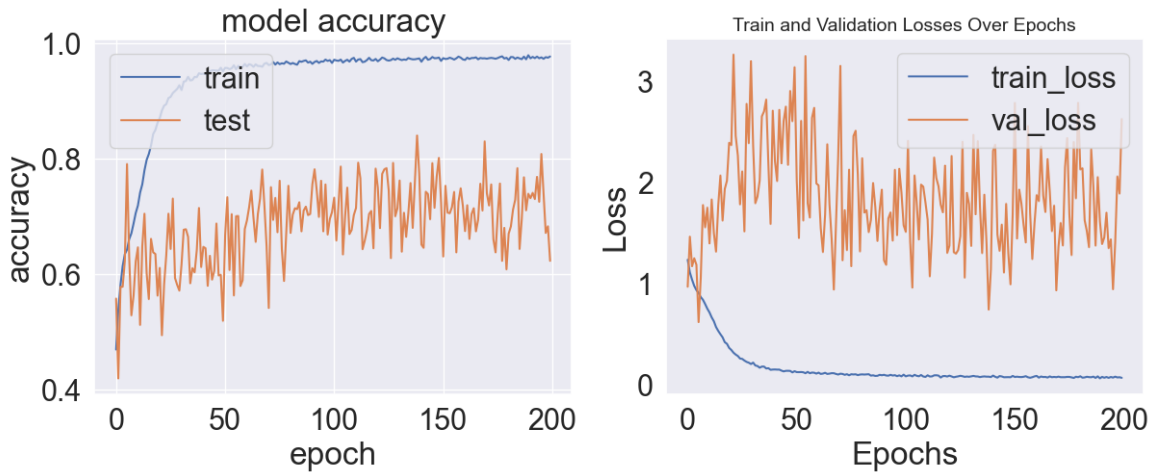

Figure 4.8: Line plots of the Train as well as Validation Accuracy and Loss values of gMLP over 200 training epochs

	P	**R**	**F**	**S**
Chickenpox	0.91	0.95	0.93	649
Measles	0.78	0.99	0.87	602
Monkeypox	0.94	0.95	0.94	1807
Normal	0.98	0.88	0.93	1980
Accuracy			0.93	5038
weighted average	0.93	0.93	0.93	5038
macro average	0.90	0.94	0.92	5038

Table 4.9: Classification Report of gMLP,, where P = Precision, R= Recall, F = f1-score , S = support

4.6.5 FNet

The Fnet architecture is a neural network architecture. The Fourier Transform is combined with a self-attention mechanism. Fnet was designed for applications requiring sequential data processing. Natural language processing, voice recognition, and video analysis are all examples of this. The self-attention method in FNet has been changed to use the Fast Fourier Transform (FFT) instead of the dot product. FNet employs the Fourier Transform to compute attention weights effectively, especially for longer sequences. The standard self-attention mechanism in transformer-based models includes calculating the dot product of the query and key vectors. For longer sequences, this is computationally expensive. Because the number of dot products that must be computed grows quadratically with the length of the sequence. The FNet employs a block like a Transformer block. Nonetheless, in the Transformer block, FNet substitutes a 2D Fourier transformation layer without parameters for the self-attention layer: With the patches, a single 1-dimensional Fourier transform is performed, and with the channels, a 1-dimensional Fourier transform is performed. It was the same as MLP-Mixer and gMLP for hyperparameter adjustment. After being developed, trained, and tested across 200 epochs, the MLP-Mixer model achieved 87% accuracy, 96% precision, 88% recall, 88% f1-score, and 823,819 parameters shown in Fig.4.8 and Table 4.10.

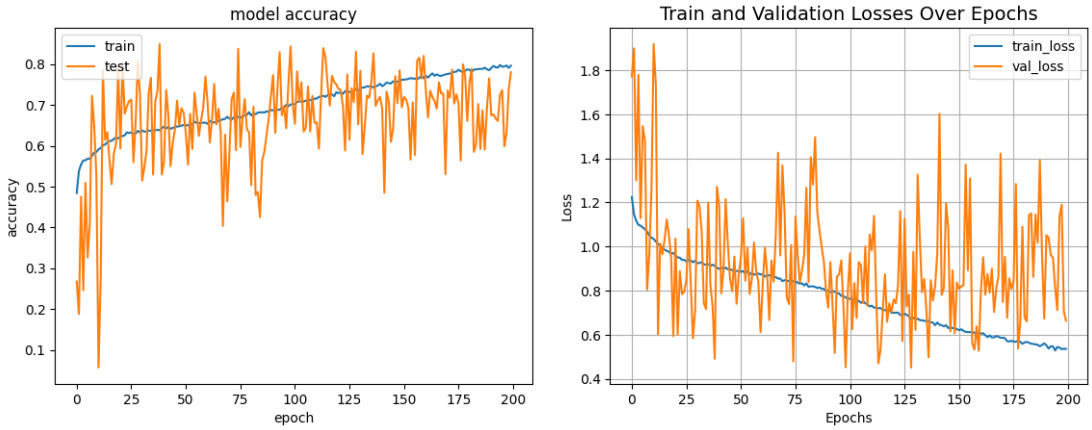

Figure 4.9: Line plots of the Train as well as Validation Accuracy and Loss values of FNet over 200 training epochs

	P	**R**	**F**	**S**
Chickenpox	0.75	0.66	0.70	649
Measles	0.74	0.81	0.77	602
Monkeypox	0.86	0.84	0.85	1807
Normal	0.86	0.89	0.87	1980
Accuracy			0.83	5038
weighted average	0.83	0.83	0.83	5038
macro average	0.80	0.80	0.80	5038

Table 4.10: Classification Report of FNet , where P = Precision, R= Recall, F = f1-score , S = support

4.6.6 Swin Transformer

Swin Transformers outperform other vision models in terms of speed-accuracy compromise. Swin includes inductive biases including locality, hierarchical feature representation, and translation invariance, allowing it to be used as an all -purpose foundation for a variety of image identification use cases. Swin Transformer's hyperparameters were examined, and the optimum hyperparameters were (72, 72, 3), batch size of 32, window size of 2, rate of learning 0.001, decay of weight 0.0001, and dropout rate of 0.03. Then, to assist us with extracting a set of patches from an image, combining patches, and implementing dropouts, we wrote two helper functions named window partition and window reverse. Then there was window-based, multi-head self-attention. Transformers typically employ global self-attention, which determines how each token connects to the others. In terms of token quantity, the global estimation is quadratic and difficult. Finally, we completed the Swim Transformer by replacing typical multi-head attention with moving window attention. After 200 iterations of collection and training, the test data with 222,388 parameters provided results of 72% accuracy, 88% precision, 93% recall, 79% f1-score, and 100% top-5 accuracy shown in Fig.4.9 and Table 4.11. More fine-tuning and hyperparameter tweaking can improve the result.

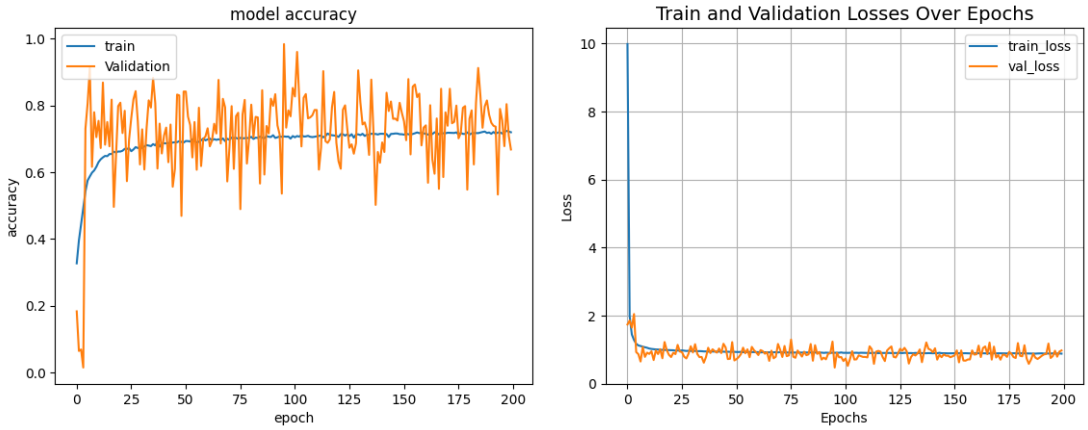

Figure 4.10: Line plots of the Train as well as Validation Accuracy and Loss values of Swin Transformer over 200 training epochs

	P	**R**	**F**	**S**
Chickenpox	0.64	0.37	0.47	649
Measles	0.80	0.43	0.56	602
Monkeypox	0.63	0.93	0.75	1807
Normal	0.88	0.72	0.79	1980
Accuracy			0.72	5038
weighted average	0.75	0.72	0.71	5038
macro average	0.74	0.62	0.64	5038

Table 4.11: Classification Report of Swin Transformer, where P = Precision, R= Recall, F = f1-score , S = support

4.6.7 Perceiver

The perceiver model, like ConvNets, scales to hundreds of thousands of inputs and is built on transformers without being bound by architectural requirements for its inputs. The model can handle enormous input volumes by continuously condensing inputs into a small latent bottleneck via an asymmetric attention technique. Because it is too difficult to process each pixel as an input in standard image models, such as ViT, we must split the image into smaller portions in order to process it. However, we can feed the complete image as a flattened array of pixels into the perceiver. The following parameters were used to calibrate the hyperparameter: picture size 72, batch size 16, rate of learning at 0.001, decay of weight 0.01, rate of dropout 0.01, patch size 2, latent dimension and projection dimension 256, and a number of transformer blocks 4. According to the official Keras paper, the feed-forward network, patch creation, and patch encoding were then implemented. Finally, the perceiver model was enhanced with a cross-attention module and a self-attention transformer. The test data with parameters of 9,742,591 generated results of 65% accuracy, 86% precision, 90% recall, 71% f1-score, and 100% top-5 accuracy after 200 iterations of gathering and training shown in Fig.4.10 and Table 4.12. Additional fine-tuning and hyperparameter tweaking improve results.

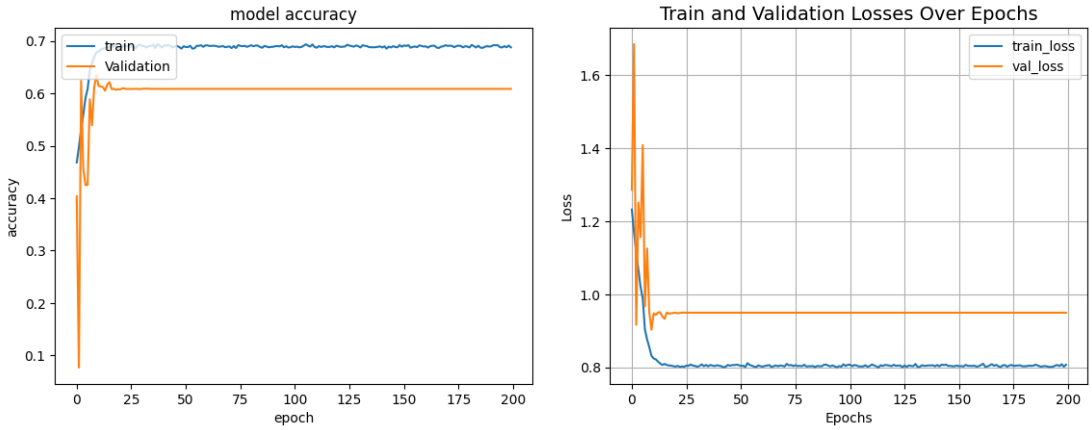

Figure 4.11: Line plots of the Train as well as Validation Accuracy and Loss values of Perceiver over 200 training epochs

	P	R	F	S
Chickenpox	0.61	0.31	0.41	649
Measles	0.53	0.45	0.48	602
Monkeypox	0.58	0.90	0.71	1807
Normal	0.86	0.61	0.71	1980
Accuracy			0.65	5038
weighted average	0.69	0.65	0.64	5038
macro average	0.64	0.56	0.58	5038

Table 4.12: Classification Report of Perceiver, where P = Precision, R= Recall, F = f1-score , S = support

4.6.8 External Attention Transformer (EANet)

External attention is a new EANet-provided attention mechanism that is built on two external, compact, accessible, and common recalls. Self-attention has been replaced.External attention examines all sample correlations automatically, resulting in linear complexity. It is a more straightforward choice than self-attention for visual tasks.The EAT paradigm's encoder and decoder each have numerous layers of EAT blocks.Each EAT block is made up of two linear layers: one for computing the key vectors and the other for computing the query vectors.The hyperparameter was tuned with image size 72, batch size 32, learning rate 0.001, weight decay 0.01, dropout rate 0.01, patch size 2, latent dimension and projection dimension 256, number of transformer blocks 8, patch size 2, label something of 0.1, and MLP dimension 64. Following that is the patch extraction and encoding layer, which is followed by the external attention block. The MLP block and transformer block were also added, with 64 dimensions and 8 additional dimensions. The accuracy was 65%, precision was 77%, recall was 82%, the f1-score was 71%, and the top-5 accuracy was 100% after constructing, training for 200 epochs shown in Fig.4.11 and Table 4.13, and validating the findings on test data with parameters of 356,979. More hyperparameter modification and fine tuning improves results.

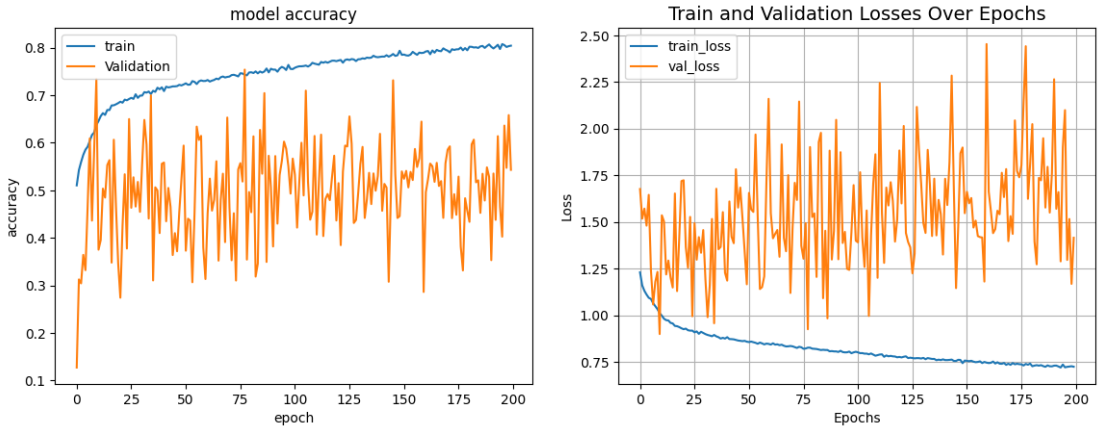

Figure 4.12: Line plots of the Train as well as Validation Accuracy and Loss values of External Attention Transformer over 200 training epochs

	P	**R**	**F**	**S**
Chickenpox	0.60	0.40	0.48	649
Measles	0.51	0.55	0.53	602
Monkeypox	0.63	0.82	0.71	1807
Normal	0.77	0.62	0.68	1980
Accuracy			0.65	5038
weighted average	0.67	0.65	0.65	5038
macro average	0.63	0.60	0.60	5038

Table 4.13: Classification Report of External Attention Transformer, where P = Precision, R= Recall, F = f1-score , S = support

4.6.9 NASNetMobile

NasNetMobile was utilized separately for training in our research.NASNet-Mobile is a convolutional neural network trained on images from the ImageNet database. MobileNets are low-latency, low-power models that may be tailored to meet the resource constraints of a wide range of use cases. They may be used to create classification, recognition, embedding, and separation systems.
In the course of our research, we utilized NasNetMobile for individual training. The convolutional neural network known as NASNet-Mobile was educated using the ImageNet database's photographs as training data. Mobile networks are models that have low latency and low power consumption, and they can be customized to meet the resource constraints of a variety of different use cases. It is possible to build systems with them that are capable of separating, embedding, recognizing, and categorizing information. ImageNet was used as the basis for NasNetMobile's weights, and Softmax was used for the classifier activation. The GlobalAveragePooling2D layer, the dense layer with 512 and RELU activation, the dropout layer with a rate of 0.2, and the output layer with eight neurons are all examples of novel layers. After that, Adam Optimizer is used to fit the models, save them, and build them. After going through 200 iterations of data collection and training, the test data with 472,299 parameters had an accuracy of 95%, a precision of 96%, a recall of 97%, and a f1-score of 96%, as shown in Fig. 4.12 and Table 4.14. In addition, the top-5

accuracy was a perfect 100

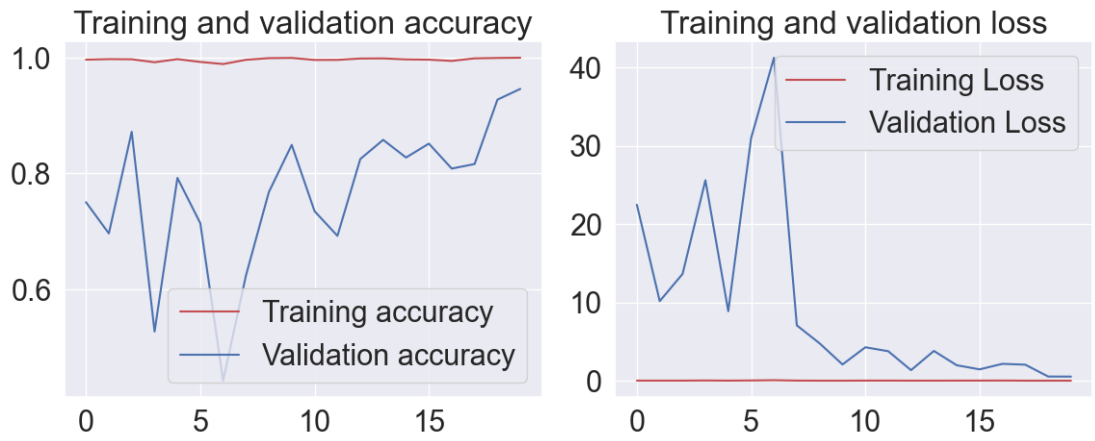

Figure 4.13: Line plots of the Train as well as Validation Accuracy and Loss values of NASNetMobile over 200 training epochs

	P	**R**	**F**	**S**
Chickenpox	0.82	0.95	0.88	659
Measles	0.91	0.93	0.92	612
Monkeypox	0.96	0.97	0.96	1811
Normal	1.00	0.93	0.96	2005
Accuracy			0.95	5087
weighted average	0.95	0.95	0.95	5087
macro average	0.92	0.95	0.93	5087

Table 4.14: Classification Report of NASNetMobile, where P = Precision, R= Recall, F = f1-score , S = support

4.6.10 Ensemble Approach

In an ensemble model, we assign more weight to classifiers or models with better accuracies (for example, InceptionResNetV2 and EfficientNetB3). We used multiple models in this case to enable the selection of an improved model or classifier while limiting the risk associated with incorrect model selection. We chose stacking, formerly known as stacked generalisation, as the strategy for our investigation. Stacking is based on the concept of combining multiple models in order to improve prediction accuracy in comparison to individual models. Thus, ensemble learning requires a minimum of two models for the stacking technique, and for our first ensemble learning strategy, we chose InceptionResNetV2 and EfficientNetB3.
The fact that these two models had never been employed for ensemble techniques was one of the primary grounds for their selection. This ensemble group was never applied to our dataset or utilised to detect skin lesions such as MPOX. They were the most current and least used pre-trained models available.That is why we selected to use them in our experiment.
Model averaging is an ensemble method in which lots of sub-models make up the total predicted equally.The predictions of numerous trained models are averaged

in a model-averaging ensemble. Because each model has its own set of benefits and drawbacks, it is usually preferable to create numerous models that can extract various features and then combine their predictions.
For the ensemble approach, there were numerous techniques such as simple average, weighted average, and weighted sum. A disadvantage of the average strategy is that regardless of how well a model performs, each model provides the exact piece to the ensemble projection, and the implementation does not improve and may even decline in the worst-case scenario. Therefore, a weighted sum ensemble is an optimal method for enhancing overall performance. It essentially combines estimations from numerous models, with each model's contribution weighted according to its proficiency or quality, and can provide superior performance in every manner compared to the performance of the individual models.
The first trial of our ensemble model was a huge success, with 99% accuracy. Two models were utilised, the first being InceptionResNetV2 and the second being EfficientNetB3. There are 25436 files in 4 classifications. In this case, we used 20349 files for training and 5087 files for validation in this case.First, the two models are trained using our datasets. Individually while freezing their initial top ten layers or not taking the top ten models for training. Furthermore, new layers were added, including a , a dense layer with 512 and relu activation, a dropout layer along with a rate of 0.2, and an output layer along with eight neurons. Following that, the models are put together with Adam Optimizer, fitted, and stored. Their greatest performances from each epoch are kept in a.HDF5 file one by one throughout time. Individually, model 1 (InceptionResNetV2) achieved 97% accuracy in only 20 epochs, while model 2 (EfficientNetB3) achieved 98% accuracy in only 20 epochs. Then, using a weighted sum ensemble, combine the best saved results from each model, giving more weight to the best-performing model. Because our EfficientNetB3 model outperformed InceptionResNetV2 by 98%, we merged it with 0.9 weight for EfficientNetB3 and 0.7 weight for InceptionResNetV2. We did not give InceptionResNetV2 a low weight because it performed well on its own. Even though both models performed admirably on their own, we desired even greater results. Thus, using 0.7 and 0.9 weights, we ensemble the model for 40 epochs and obtain 99% accuracy, which exceeded our expectations and was more than satisfactory.
In the second experiment of our ensemble model, we attempted to expand our exploration with three models using the stacking ensemble approach. Overall, we achieved 99% accuracy. MobileNetV3Large, DenseNet201, and Xception are the three models. Again, we used 20349 files for training and 5087 files for validation in the datasets. The top 10 layers of each model were then frozen, as with the first ensemble approach. the dense layer along with 512 and relu activation, the dropout layer along with a rate of 0.2, and the final output layer along with eight neurons were all included here. Separately, MobileNetV3Large provided 84% accuracy, DenseNet201 provided 97% accuracy, and Xception provided 98% accuracy. Then, using the weighted sum ensemble technique, we ensembled the outcomes of all three saved models and obtained 99% accuracy, which was an improvement over the individual accuracy. Finally, we can see that the ensemble approach not only decreased the spread or dispersion of the predictions and model performance but also improved the average prediction performance over every contributing member in the ensemble. There are two primary, interrelated reasons to prefer an ensemble over a singular model: Performance and Robustness.An ensemble is able to

make more accurate predictions and achieve greater performance than any individual model. Moreover, an ensemble reduces the dispersion or spread of predictions and model performance. The predictive accuracy of ensemble methods is greater than the accuracy of individual models.This increases the likelihood of high variance, low accuracy, noise, and bias if a single model is incapable of producing the correct prediction for a particular data set. By integrating multiple models, we have greater chances of increasing the level of accuracy.

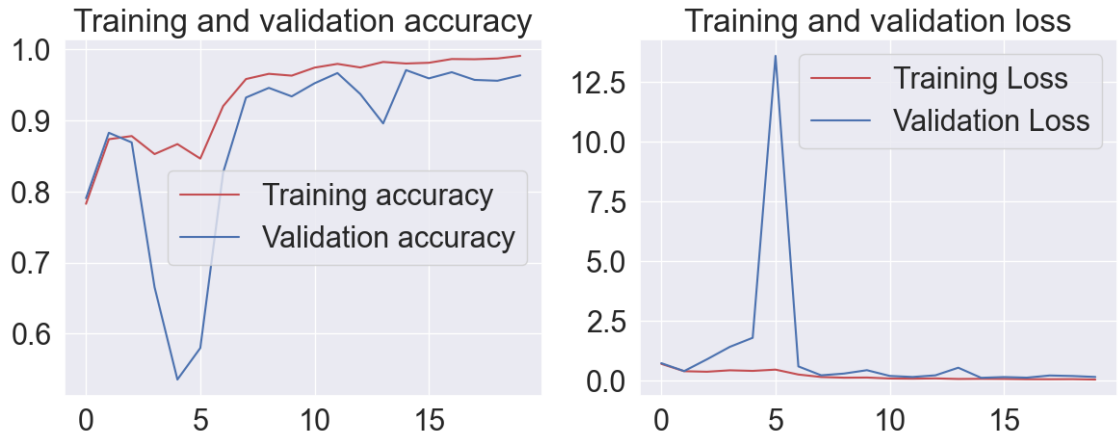

Figure 4.14: Line plots of the Train as well as Validation Accuracy and Loss values of InceptionResNetV2 over 200 training epochs

Figure 4.15: Line plots of the Train as well as Validation Accuracy and Loss values of EfficientNetB3 over 200 training epochs

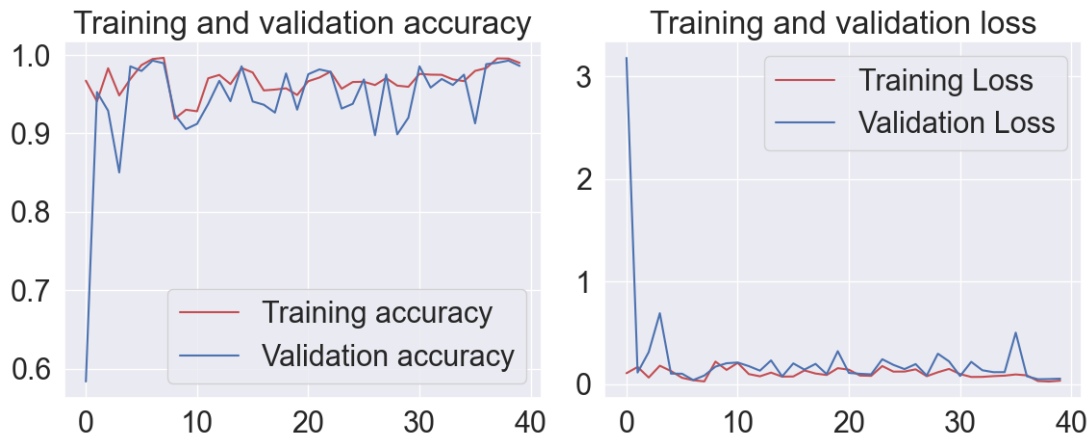

Figure 4.16: Line plots of the Train as well as Validation Accuracy and Loss values of Ensemble 1 over 200 training epochs

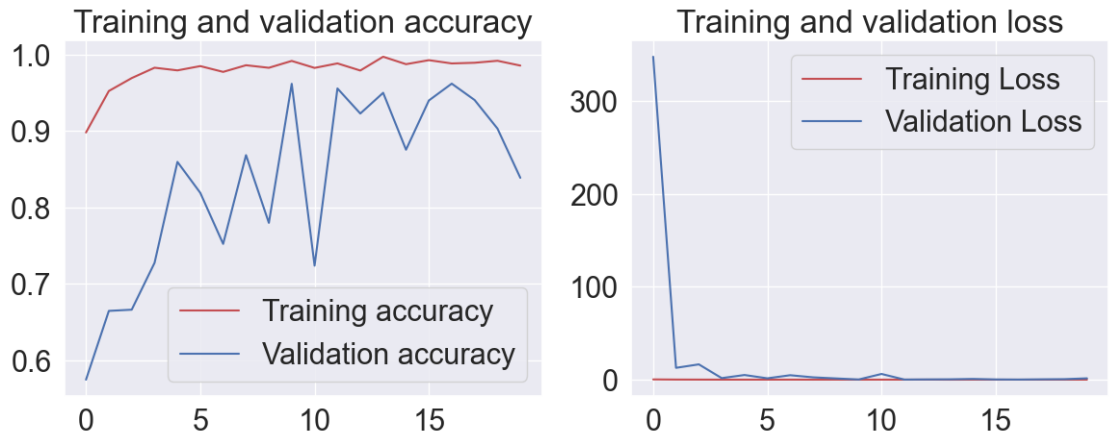

Figure 4.17: Line plots of the Train as well as Validation Accuracy and Loss values of MobileNetV3Large over 200 training epochs

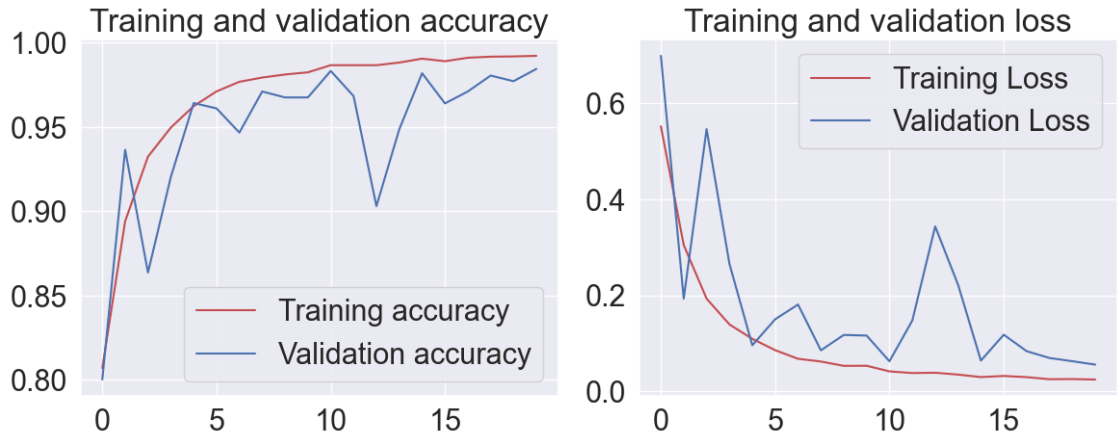

Figure 4.18: Line plots of the Train as well as Validation Accuracy and Loss values of DenseNet201 over 200 training epochs

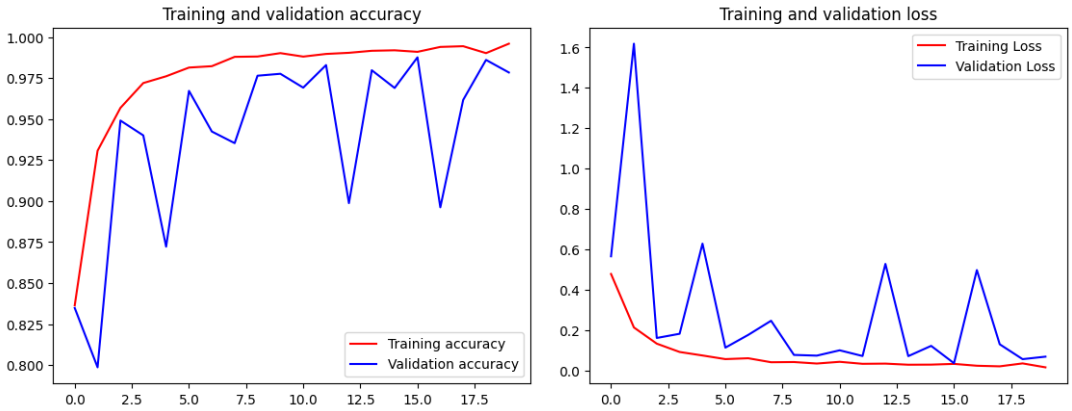

Figure 4.19: Line plots of the Train as well as Validation Accuracy and Loss values of Xception over 200 training epochs

Figure 4.20: Line plots of the Train as well as Validation Accuracy and Loss values of Ensemble 2 over 200 training epochs

Chapter 5

Result Analysis

5.1 A comparison of Transformer-based and Pretrained Ensemble models

Using precision, accuracy, recall, and f1-score, we evaluate the suggested approach to contemporary pretrained deep learning models on the MSID dataset. The results are shown in Table 5.1 and Table 5.2.

Models	**A(%)**	**P(%)**	**R(%)**	**F(%)**	**Para**
ViT	94	95	94	94	3,343,819
CCT	97	97	97	97	407,365
MLP-Mixer	92	93	92	93	1,068,299
gMLP	93	93	93	93	823,819
Fnet	87	83	83	83	544,011
Swin Transformer	72	75	91	71	222,388
Eanet	65	87	85	85	356,979
Perceiver	65	89	86	84	9,742,591
NasNetmobile	95	94	94	95	5,556,508
InceptionResnetV2	96	96	96	96	56,107,368
EfficientNetB3	98	98	98	98	12,623,287
MobileNetV3Large	84	82	84	81	4,391,432
DenseNet201	97	98	98	98	18,638,024
Xception	98	99	99	99	21,193,904

Table 5.1: Comparison of transformer based & pretrained CNN based DL models using Accuracy, Precision, Recall, F1-score and Parameters. Here, A= Accuracy, P=Precision, R=Recall, F=F1-score, Para=Parameters

Table 5.1 and Fig. 5.1 show that among the eight transformer-based models, Compact Convolutional Transformers (CCT) had the highest accuracy (97%, Precision: 99%, Recall: 98%, and F1-score: 97%). The three best classification performances were achieved by EfficientNetB3 (Accuracy: 98%, Precision: 99%, Recall: 99%, and F1-score: 99%), Xception (Accuracy: 98%, Precision: 99%, Recall: 98%, and F1-score: 98%), and DenseNet201 (Accuracy: 97%, Precision: 98%, Recall: 98%, and F1-score: 97%).

Models	A(%)	P(%)	R(%)	F(%)	Para
Ensemble 1 (InceptionResnetV2 + EfficientNetB3)	99	99	99	99	68,730,655
Ensemble 2 (MobileNetV3Large +DenseNet201 +Xception)	99	99	99	99	44,223,360

Table 5.2: Performance comparison of different pre trained CNN based models combination for ensemble approach using Accuracy, Precision, Recall, F1-score and Parameters. Here, A= Accuracy, P=Precision, R=Recall, F=F1-score, Para=Parameters

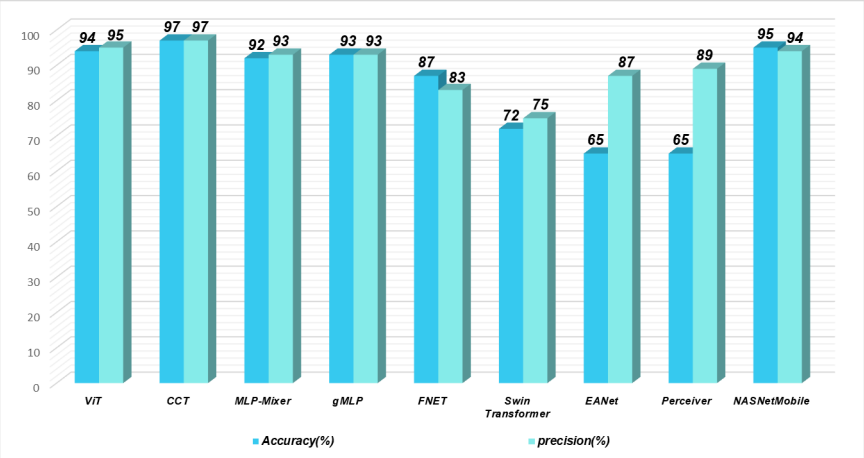

Figure 5.1: Result Analysis of Individual Models

The overall results are presented in Table 5.2 and Figure 5.2. They demonstrate that both of the proposed ensemble techniques come out on top, triumphing over all of the other individual models. Ensemble 1 has an accuracy of 99.9 percent and employs two models (InceptionResnetV2 and EfficientNetB3) in its computations. Both the recall rate, which is 99.9 percent, and the precision score, which is 100 percent, are very high. In addition to having a recall rate of 99 percent and a score of 100 percent on the F1-scale, Ensemble 2, which is comprised of three models (MobileNetV3Large, DenseNet201, and Xception), has an accuracy rate of 99 percent and provides 100 percent precision. It also has a recall rate of 99 percent. Both of the ensemble models have a greater accuracy of 2%, a greater precision of 1%, a greater accuracy of 1% in recall, and a greater accuracy of 2% in F1-score, also known as CCT. When compared to the Transformer-based model (FNet) with the worst performance, both ensemble models are 16 percentage points more accurate, 14 percentage points more precise, 10 percentage points more accurate in the recall, and 12 percentage points more accurate in F1-score.

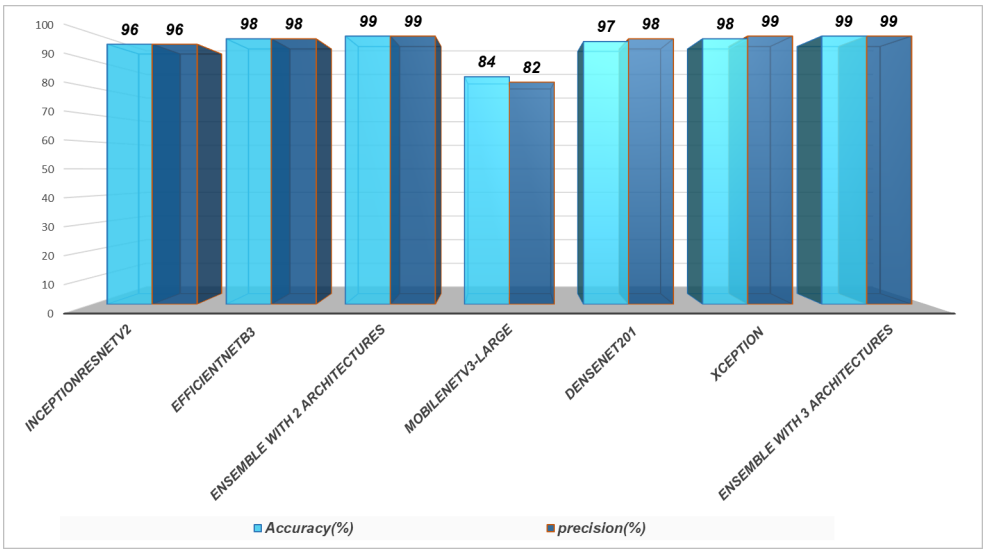

Figure 5.2: Result Analysis of The Ensemble Models

5.2 Raw vs pre-processed datasets comparison study

To highlight the value of preprocessing our datasets, we developed a comparison table. The stark contrast between the before and after pre-processing in Table 5.3 and Fig. 5.3 demonstrates the significance of the improvement for better performance. Here, we presented and trained every model with our raw datasets, where no data preprocessing took place.

There were a total of 770 raw datasets where neither histogram equalization nor data augmentation had been implemented. As a result, most of the models' accuracy and precision were close to or below 70% or 60%. However, after implementing CLAHE (Contrast Limited Adaptive Histogram Equalization) and data augmentation, accuracy and precision increased to 99.9%. Therefore, the preprocessing of the dataset increased accuracy and precision by 20% to 40% across all matrices.

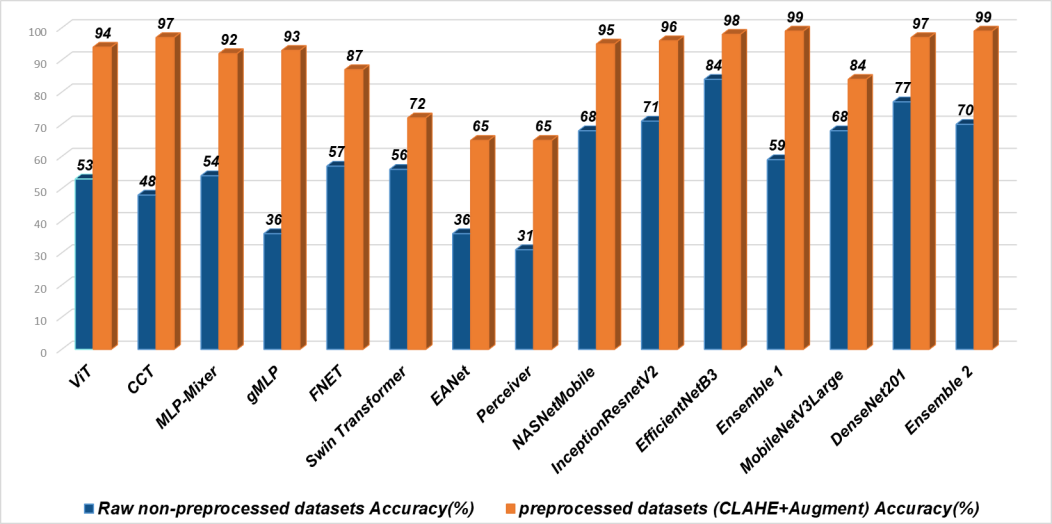

Figure 5.3: Result Analysis on Raw non-preprocessed datasets Vs Preprocessd datasets

Models	**RA(%)**	**RP(%)**	**RR(%)**	**PA(%)**	**PP**	**PR**
ViT	53	55	45	94	95	94
CCT	48	41	27	97	97	97
MLP-Mixer	54	55	58	92	93	92
gMLP	36	39	42	93	93	93
Fnet	57	57	63	87	83	83
Swin Transformer	56	47	54	72	75	91
Eanet	36	24	32	65	87	85
Perceiver	31	10	31	65	89	86
NasNetmobile	68	65	65	95	94	94
InceptionResnetV2	71	34	66	96	96	96
EfficientNetB3	84	65	67	98	98	98
Ensemble 1 (InceptionResnetV2 + EfficientNetB3)	59	47	46	99	99	99
MobileNetV3Large	68	46	50	84	82	84
DenseNet201	77	72	69	97	98	98
Xception	62	45	44	98	99	99
Ensemble 2 (MobileNetV3Large +DenseNet201 +Xception)	70	73	70	99	99	99

Table 5.3: Comparison of Raw vs Preprocessed datasets. Here RA= Accuracy of Raw data, P=Precision of Raw data, R=Recall of Raw data, PA= Accuracy of Preprocessed data, PP=Precision of Preprocessed data, PR=Recall of Preprocessed data

5.3 Ensemble model selection for contenders

We chose the most recent pre-trained models for the ensemble to improve accuracy and performance in our research. For decision fusion, we pick the top five models and their assortments. The outcomes are shown in detail in Table 2. According to Table 2, the combination of both ensemble models provides the greatest performance when compared to other transfer-based or CNN-based models in Table 1. We chose InceptionResnet50 as Model 1 and EfficientNetB03 as Model 2 for our initial ensemble experiment. InceptionResnet50 and EfficientNetB03 had respective accuracy levels of 94% and 97%. However, after using Ensemble, the accuracy increased to 99%. We chose MobileNetV3Large as Model 1, DenseNet201 as Model 2, and Xception as Model 3 for our second ensemble experiment. The accuracy of MobileNetV3Large, DenseNet201, and Xception was 84%, 97%, and 98%, respectively. However, after using Ensemble, the accuracy increased to 99%. Thus, it enhanced the ensemble's average prediction performance and reduced the spread or dispersion of predictions and model performance.

5.4 Comparative study with state of the art methods

We evaluated our suggested model with a selection of closely related image detection and classification methods for detecting MPOX skin lesions. In the paper [13]", it was proposed to develop the first publicly available MPOX.image dataset by accumulating images from various sources (e.g., news media and websites). Then, a low-modified VGG16 model was introduced for detecting MPOX patients using image data. However, there are some limitations. First, their models are limited to binary classification. Second, they only considered the VGG-16 DL model for transfer learning, which does not identify the top-performing pre-trained DL approaches and their optimum combinations to achieve optimal performance. Third, their models' interpretability is inadequate. Therefore, it is challenging to establish credibility among health practitioners during mass screenings. Fourth, they applied LIME as XAI to generate a heatmap surrounding the tumors that can be seen on the images.

The authors of the paper "cite66" utilized the same MSID dataset that we did for our research. On the other hand, they utilized the preliminary version of the MSID dataset, which comprised a total of 572 images divided into two categories. The images were scaled down to 224 by 224 pixels without any additional processing or augmentation of the dataset being done beforehand. The month of September 2022 saw the release of this publication. During the course of the experimental analysis, a total of twelve image classification networks that utilized transfer learning were used. They are ResNet-18, ResNet-50, VGG-16, Densenet-161, EfficientNet B7, EfficientNet V2, GoogLeNet, MobileNet V2, MobileNet V3, ResNeXt-50, ShuffleNet V2, and ConvNeXt. They selected only networks affiliated with CNN. In addition to this, a heatmap was generated around each of the lesions on the images with the help of the Pytorch framework and XAI.A MPOX image dataset [64] was used in the research presented in the article "cite67." The month of October 2022 saw the publication of this article. They proposed utilizing a common architecture for all 13 pretrained DL models (MobileNetV2, VGG-19, ResNet-101, ResNet-50, IncepResNetv2, InceptionV3, VGG-16, Xception, EffcientNet-B0, EfficientNet-B1, EffcientNet-B2, DenseNet-121, and DenseNet-169) for MPOX detection and classification. This was done in order to improve accuracy. Then, in order to improve performance all around, they combined the models that had the best results. Despite this, their rate of accuracy was 87.13 percent.

In each of these publications, the machine learning models that were actually implemented were quite dated. To In order to solve this problem, we chose to conduct an experiment using both recent CNN-based pre-trained models and 14 transformer-based models. The goal of the experiment was to determine which model is superior when it comes to classifying MPOX images on an individual basis. Precision, recall, F1-score, and accuracy were the metrics that were used to assess each model's performance. In the end, we improved the overall performance by combining the best pre-trained CNN-based models that we could find. The heatmaps that surrounded the lesions on the images were produced with the help of GradCAM. We spared no effort in order to pick the hyperparameters that worked best. The overall accuracy of our assessment was 99 percent. As a result of its higher level of accuracy, the method that we have suggested can be relied upon by medical professionals for the

Methods	**PA(%)**	**OA(%)**	**Paper Limitations**	**Our Contributions**
Modified VGG-16, 2022, [13]	88% overall	99% overall	Binary Classification. Only considered the VGG- 16 DL model for transfer learning	Classified with 4 classes and used latest 16 models. Applied transformer based for the first time in this field
Explainable Artificial Intelligence Assisted CNN, 2022, [29]	75.44% Mobile Net V3	84% Mobile Net V3	Used older CNN models such as :Densenet-161, EfficientNetB7, EfficientNetV2, MobileNetV2, MobileNetV3	Used recent CNN models such as: Densenet-201 & EfficientNetB3
Pre-trained DL models, 2022 [23]	87.13% ensemble	99% for both ensemble	Used pre-trained DL models: VGG-16, VGG-19, ResNet-50, ResNet-101, IncepResNetv2, MobileNetV2, InceptionV3, Xception, EffcientNet-B0, EfficientNet-B1 , EffcientNet-B2, DenseNet-121, DenseNet-169	Used recent DL classification models ViT, CCT, MLP-Mixer, gMLP, FNET, Swin Transformer, EANet, Perceiver, InceptionResnetV2, EfficientNetB3, Mobile NetV3Large, DenseNet201, Xception

Table 5.4: Comparative Study with State-of-the-Art Methods

Here, PA= Cited Paper's Accuracy, OA = Our Accuracy

purpose of conducting mass screening, as can be seen in Table 5.4.

5.5 Grad-CAM approach

We selected GradCAM for the XAI method since it helps in locating the network's final convolutional layer. Then examine the gradient data entering that layer which has shown in Fig. 5.4 . After training and evaluating the pre-trained models and transformer-based models, we implemented GradCam, which leveraged two functions: get image to array and heatmap. This heatmap enables us to visually confirm the area of the image on which our model is focused.

In Fig 5.5, In the heatmap function, the factors are the image array, the model, the last convolutional layer, and the output forecasts. This means that we build a model that first connects the activations to the input picture. We will additionally normalize the heatmap among none and one for display reasons.

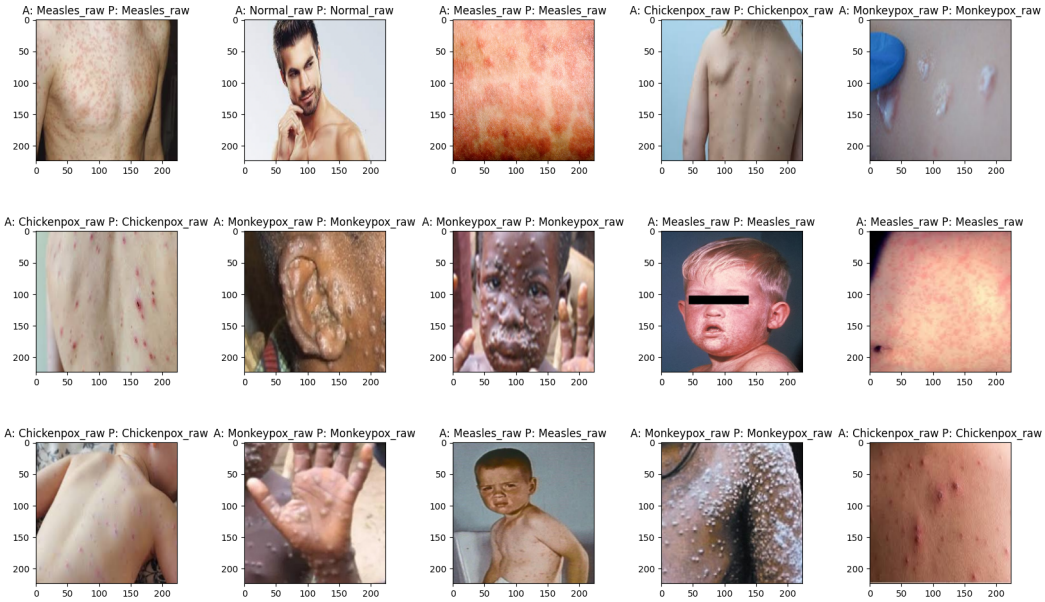

Figure 5.4: Correct GradCam prediction representations on our datasets

We made use of another function that saves and displays the grad cam in order to achieve the superimposed rendering seen in Figure 5.7. This function now accepts image path, heatmap, cam path, and alpha equal to 0.4 as arguments. Directed Grad-CAM has the capacity to investigate and explain categorization mistakes, which is one of the many notable advantages it has over alternatives. However, when there are multiple instances of the same class present in an image, Grad-CAM is unable to correctly locate objects within the image.
Calculating the color gradients of the classification value in comparison to the final layered map of features is done in order to determine which parts of the provided picture have the greatest influence on the ability to recognize rating.

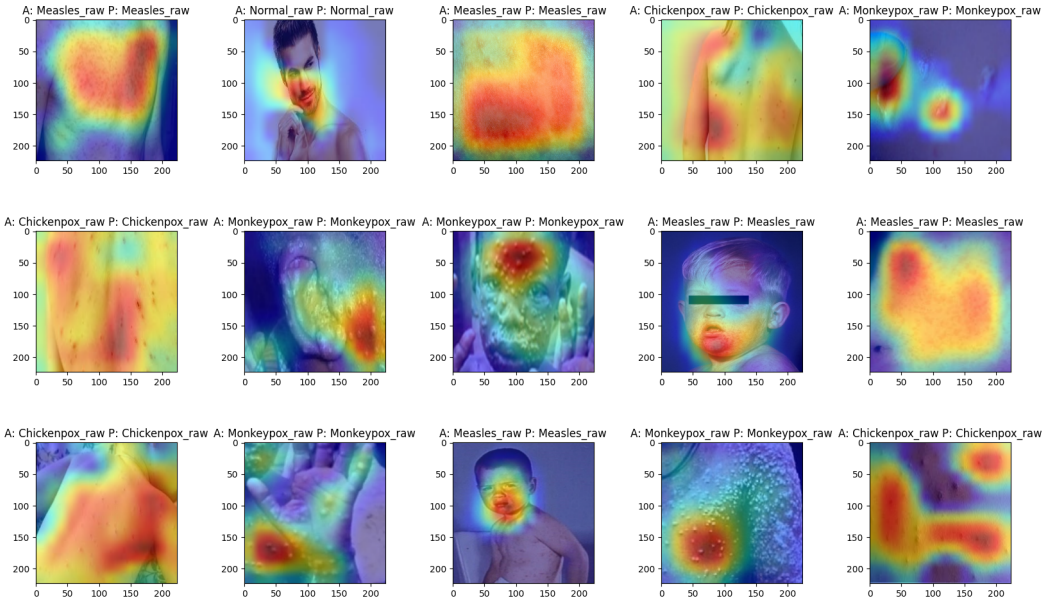

Figure 5.5: GradCam's superimposed visualization explanation

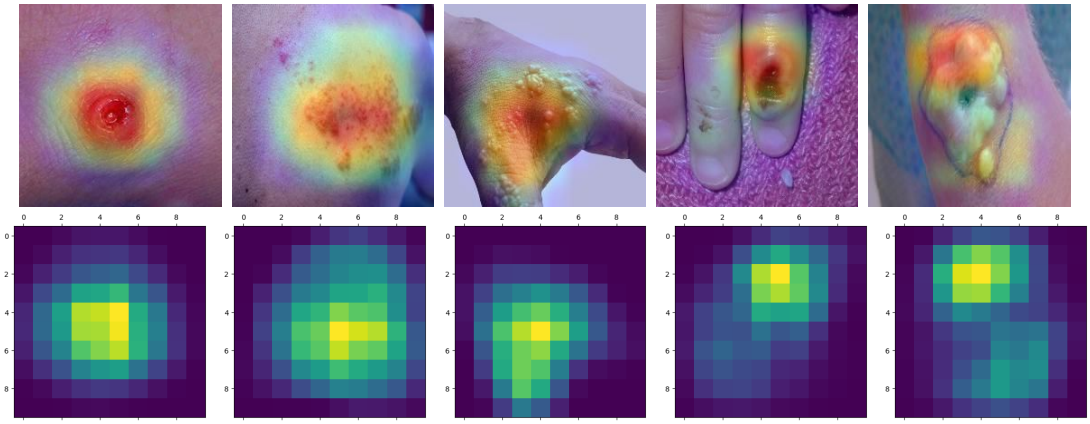

Figure 5.6: GradCam's superimposed visualization explanation on our implemented datasets and their heatmap visualization

5.6 Analyzing Vision Transformer (ViT)

The primary goal that we have set for ourselves is to shed light on the mechanisms that make it possible for ViTs to gain knowledge from image data. The utilization of our datasets will be utilized in order to accomplish this goal. This is the most important objective that we hope to accomplish with the help of our datasets, and we hope to do it as soon as possible. An exhaustive investigation into the implementations of a wide variety of different ViT analysis tools is shown here as an example of how such an investigation might be carried out. This investigation is shown here as an example of how such an investigation might be carried out. It was necessary to scale the input images in such a way that they fell somewhere within the range [-1, 1] in order to make use of the earliest ViT models. In order for us to be able to process the data for the other model families, we are required to normalize the images by utilizing the mean and standard deviation of ImageNet-1k's channels. Only then will we be able to move on to the processing of the data. Within the code cell,

we will be displaying three unique images that we have selected. These images were selected from the various manifestations of MPox (monkeypox), chickenpox, and measles that are shown in Figure 5.8. The model was loaded once that activity had been completed successfully. The training of this model began with ImageNet-21k, and later on, ImageNet-1k was used to provide assistance in the development of the model. ImageNet-21k was used initially. ImageNet-1k was used subsequently. The next step is to determine how local and global information is transmitted into vision transformers by computing the Mean Attention Distance from each attention head of a different transformer block. By carrying out this step, you will be able to learn the process by which information is transmitted into vision transformers. In the context of this article, the term "mean attention distance" refers to the amount of time that elapses between query tokens and the attention weights of other tokens. Specifically, this term refers to the amount of time that elapses between query tokens and the attention weights of other tokens. Calculating the geometric distance between image regions (tokens) is the first step in the process of analyzing each individual image, which is then followed by the multiplication of that distance by the attention ratings for each region of the image. The first thing that needs to be done in order to complete this procedure is to calculate the amount of geometric distance that separates the various image regions. As can be seen in Figure 5.9, this was comprised of a total of twelve individual heads.

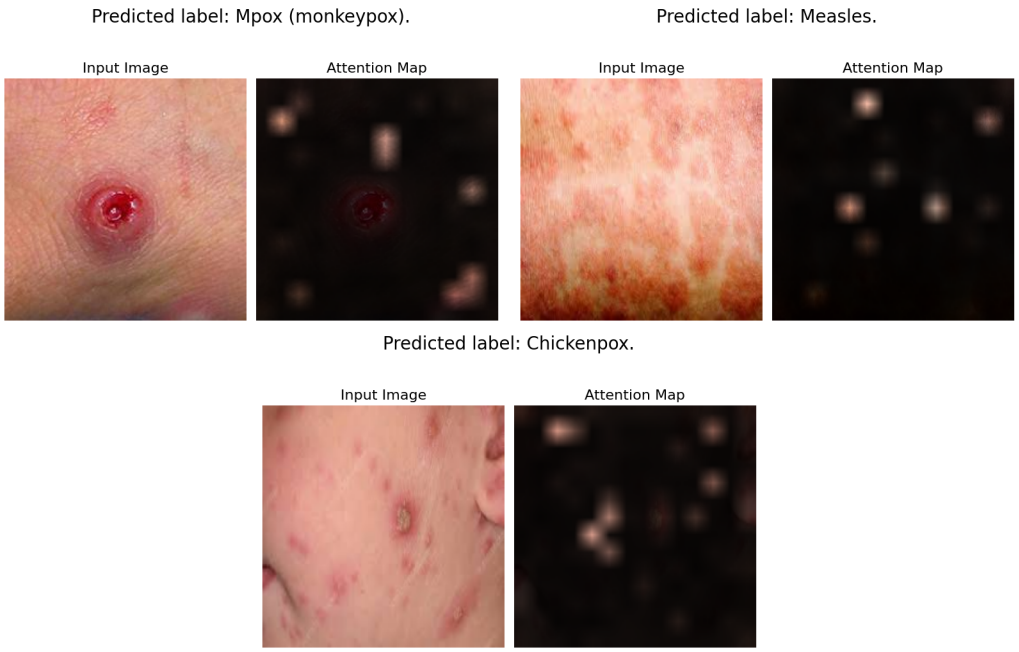

Figure 5.7: Input image and Its attention map

The attention rollout method was the second strategy that was considered for the dilemma that was being faced. When referring to a method for measuring data flow that makes use of Transformer block self-attention layers, the term "attention rollout" is the one that is intended to be used to describe the technique. When someone mentions the "Attention rollout" method, they are referring to this particular approach. The developers of the first version of the ViT software decided to make use of this approach in order to carry out their analysis of the patterns that they had acquired.

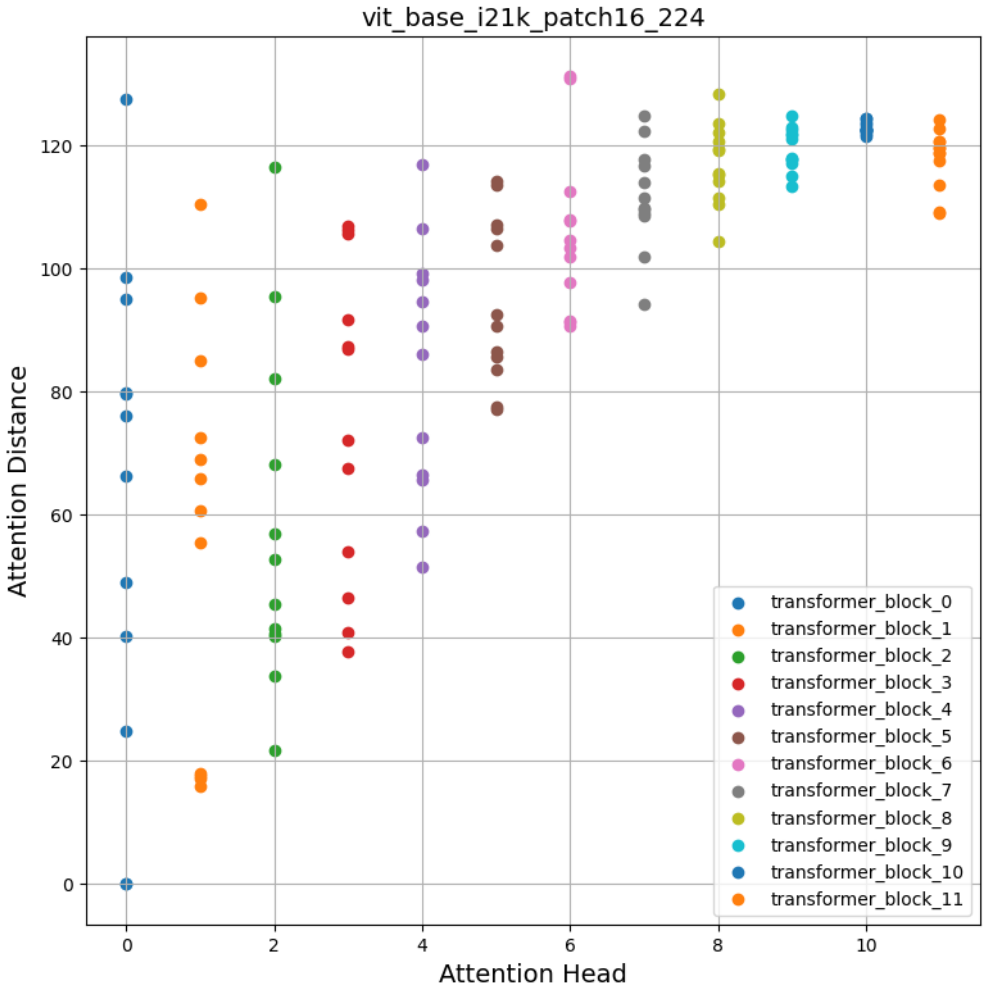

Figure 5.8: Attention Head of 12 items

The utilization of attention heatmaps was the third approach, and Figure 5.10 provides an illustration of one of these maps. The comprehension of the representation of the vision transformer can be made simple and simple in its simplicity by superimposing attention maps on top of the input visuals. This makes the comprehension of the representation simple in its simplicity. This role has the potential to be of significant assistance to others. On the basis of this, one is able to determine the primary focus that the model is intended to illustrate. Each and every one of the Transformer blocks has the ability to transform into a different head in a variety of different configurations. The information that is gathered from the heads of the transformer blocks is transferred to the appropriate subspaces so that it can be processed. Because of this, each head is able to direct its attention to the particular visual location that possesses the information that is most pertinent to it. This enables the heads to work more efficiently. It is necessary to view all of the maps associated with the attention heads in question at the same time in order to acquire a comprehensive comprehension of the information that is being received by each attention head.

The fourth strategy involved the visualization of the obtained projection filters, and an example of this can be seen in Figure 5.10. Transformer blocks are in charge of calculating the attention weights that are allotted to the key and the query

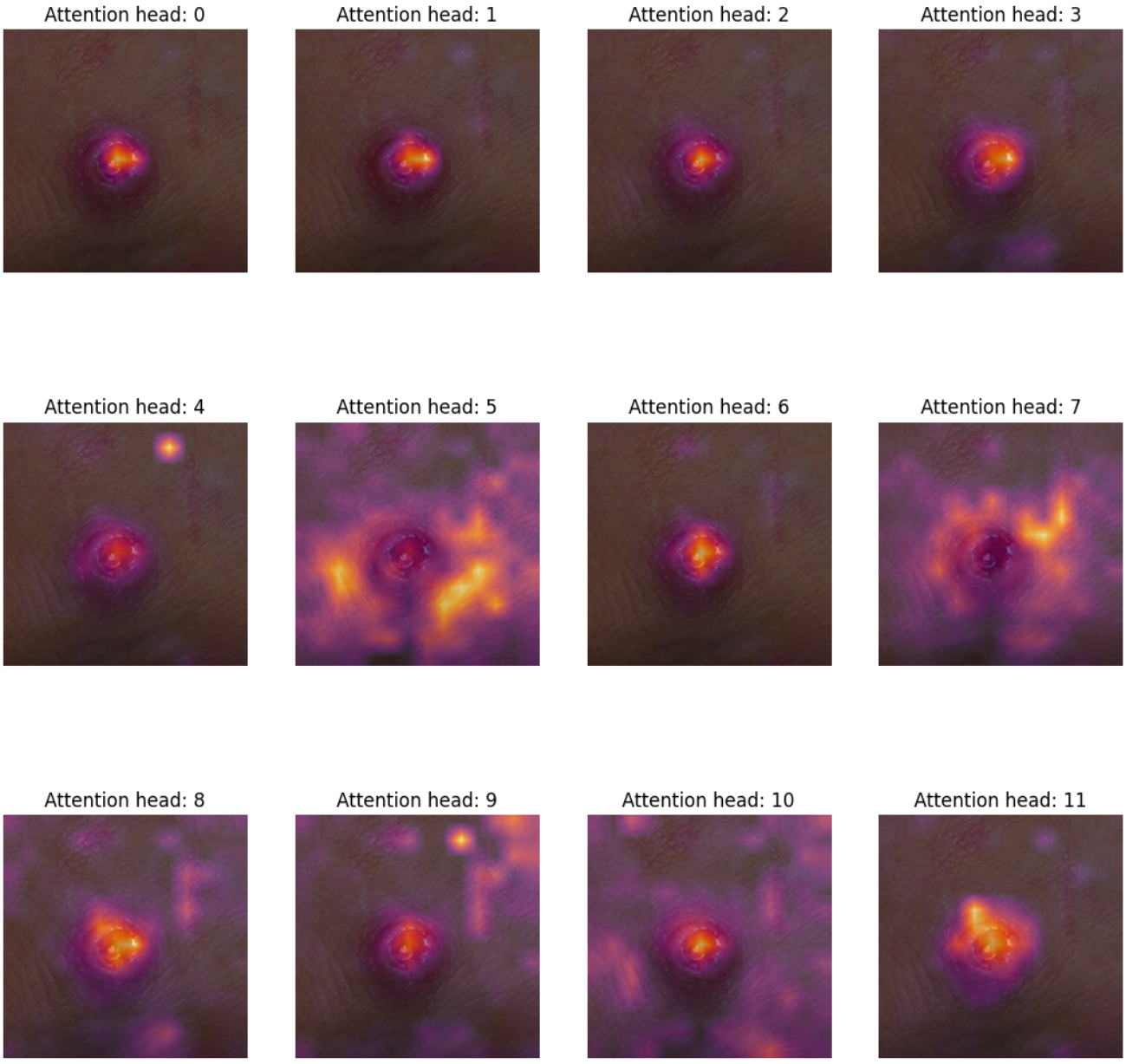

Figure 5.9: Attention heatmaps

respectively. The purpose of the weights is to provide a graphical representation of the significance of the query key. Because the key and the inquiry are both extracted from the same image using ViTs, the weights are what determine which part of the image is the most important. This is the case because the weights are what determine which part of the image is extracted first. This is the case because the weights are what decide which part of the image is extracted first. Consequently, this is why this is the case. With the assistance of ViTs, patches that do not overlap one another can be linearly projected and flattened out into a single layer. The ViT model has produced the projections that are presented in the following table.
As part of the fifth approach, a visual representation of the positional embeddings shown was created. There is no way in which permutations can have an effect on transformers. This suggests that the spatial position of the input token is ignored in the processing of the transaction. By making use of the positional information

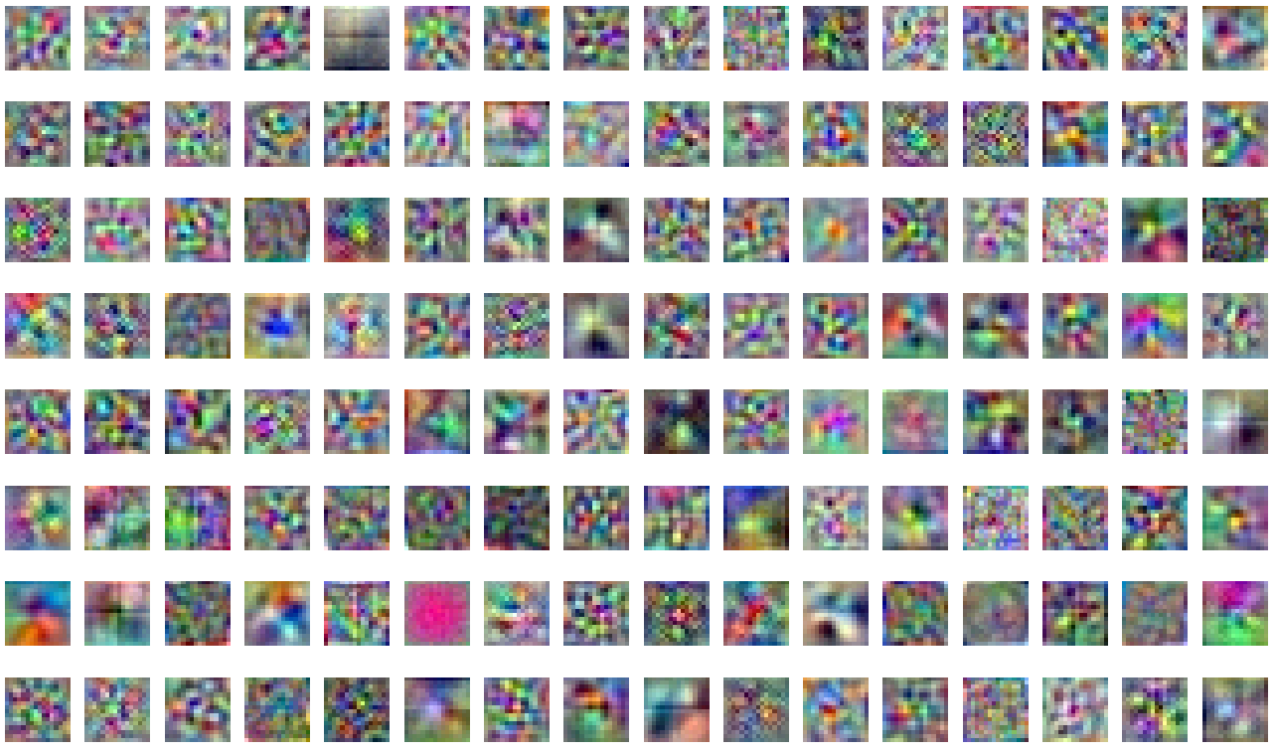

Figure 5.10: Visualization of the learned projection filters

that is contained on the input tokens, we are able to get around this limitation. It is possible to acquire positional data, but it is also possible to manually create it. We have been successful in obtaining positional embeddings for each of our three unique ViTs. We are going to evaluate the learned positional embeddings by comparing them to themselves throughout the entirety of this section so that we can determine how effective they are. Evidence of positional embedding similarity can be found by examining the dot product of the model.

5.7 Class-wise study for Confusion matrix

Using the confusion matrix in Fig.5.11 to Fig.5.18, we analyze the class-by-class performance of the ensemble technique. Our first ensemble approach with two architectures (InceptionResNetV2 and EfficientNetB3) is able to classify images into four distinct categories. Specifically, our method can distinguish measles and monkeypox virus images from normal images and chickenpox virus images with high accuracy. Furthermore, the suggested model properly identifies all cases of chickenpox (641) and measles (604) in the test set. The Grad-CAM image reveals that the backbone CNN detected comparable features between the measles and monkeypox viruses. In our second ensemble learning for three architectures (MobileNetV3Large, DenseNet201, and Xception), we utilized MobileNetV3Large, DenseNet201, and Xception. Chickenpox was accurately detected 651 times out of 659 times, measles 593 times out of 612 times, monkeypox 1777 times out of 1811 times, and normal pictures 2002 times out of 2005 times. Both ensemble learning methods outperformed all previous models used on our datasets, putting our results ahead of any other existing state-of-the-art techniques.

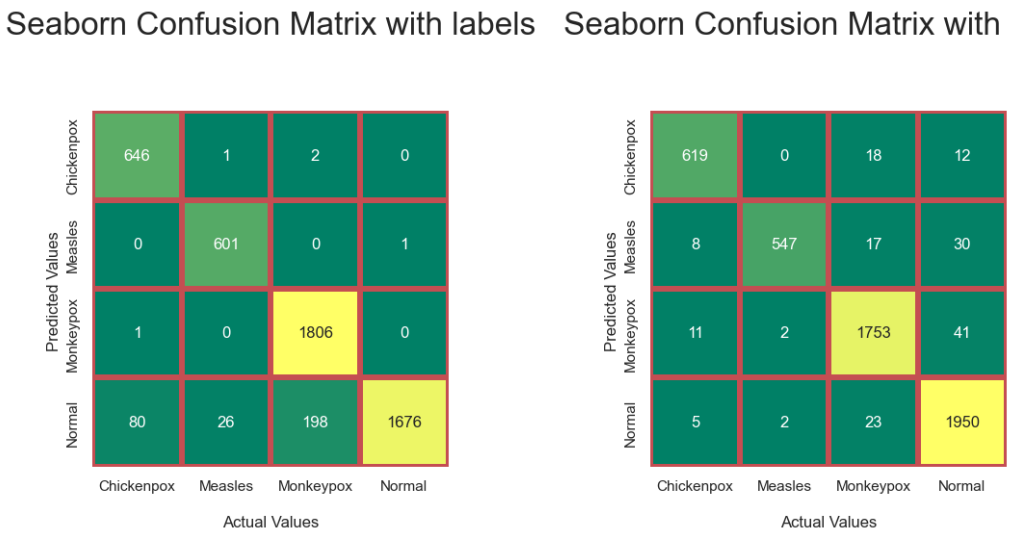

Figure 5.11: Confusion Matrix of ViT (Left) & CCT (Right)

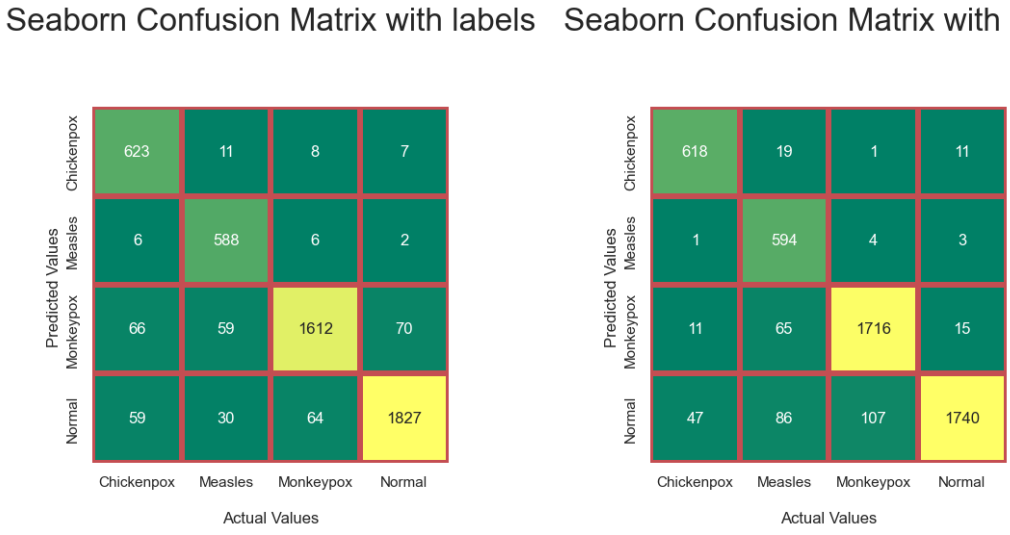

Figure 5.12: Confusion Matrix of MLP-Mixer (Left) & gMLP (Right)

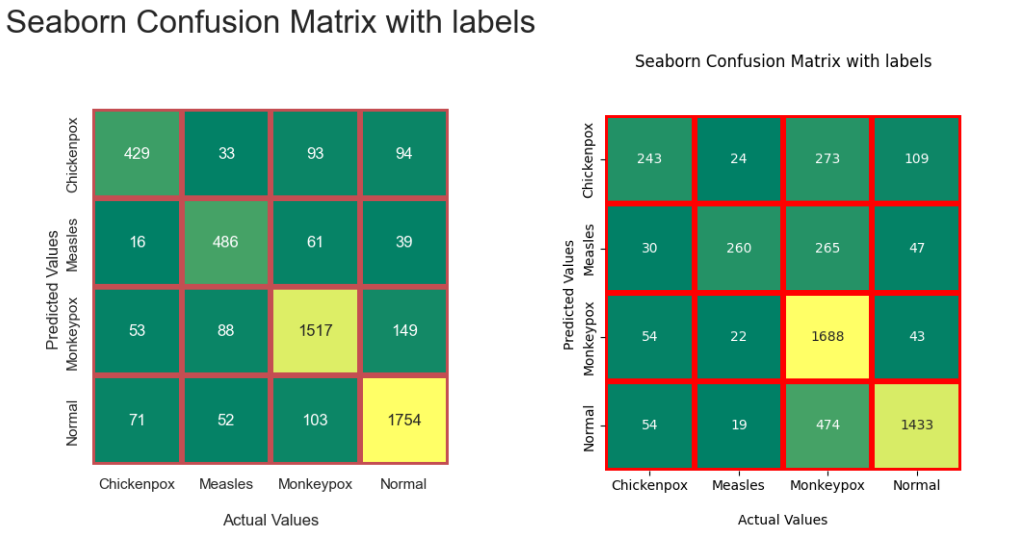

Figure 5.13: Confusion Matrix of FNet (Left) & Swin Transformer (Right)

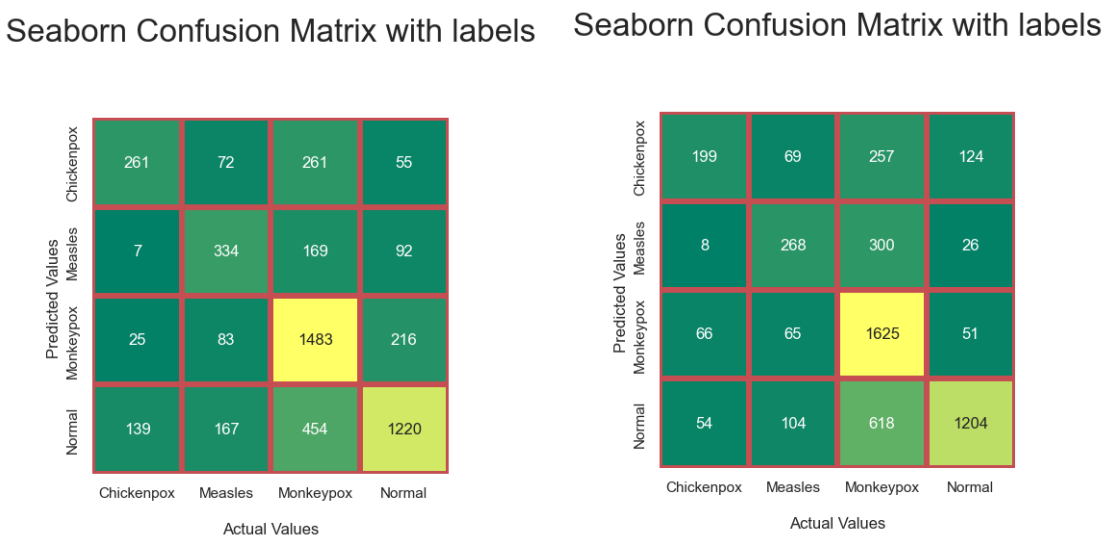

Figure 5.14: Confusion Matrix of EAnet (Left) & Perceiver (Right)

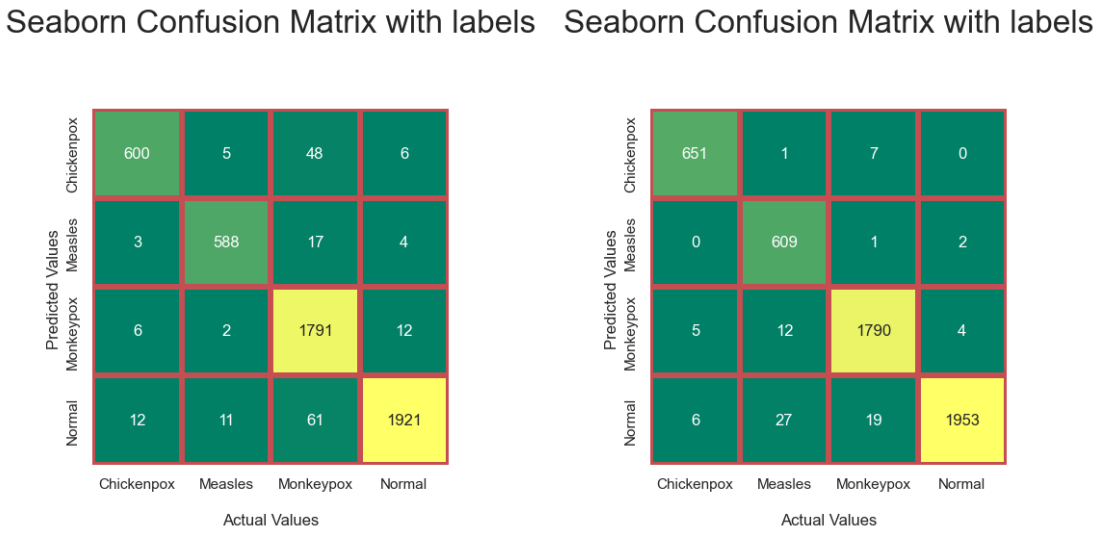

Figure 5.15: Confusion Matrix of InceptionResNetV2 (Left) & EfficientNetB3 (Right)

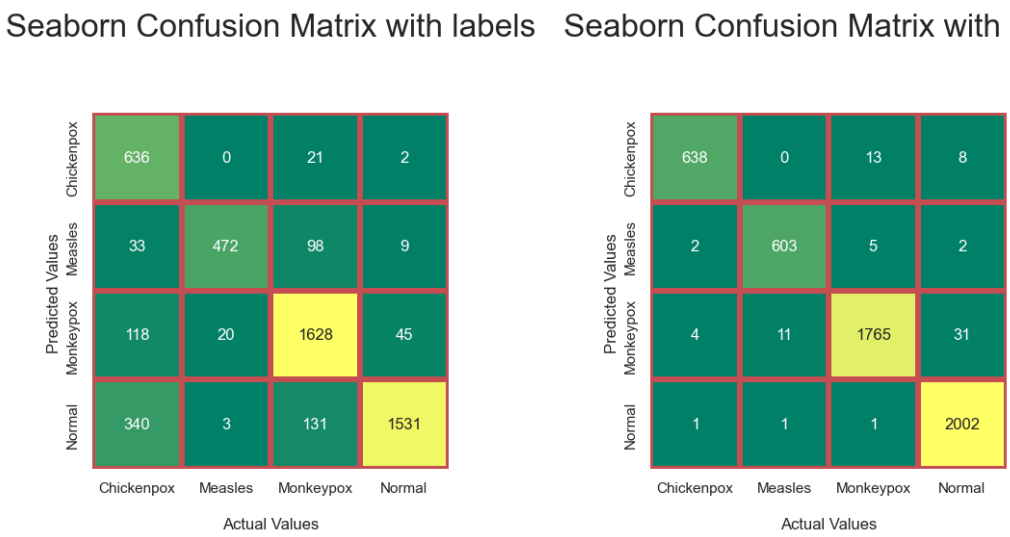

Figure 5.16: Confusion Matrix of MobileNetV3Large (Left) & DenseNet201 (Right)

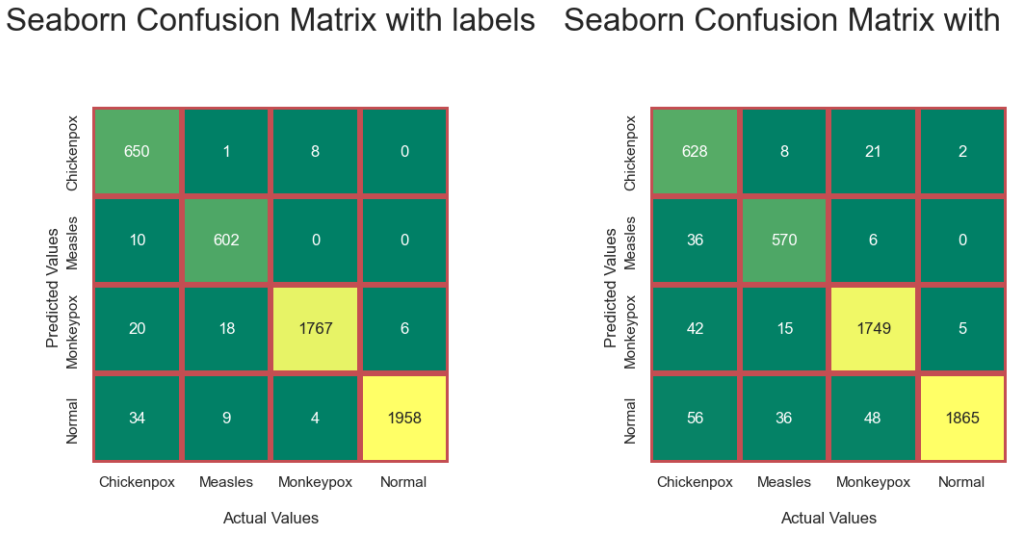

Figure 5.17: Confusion Matrix of Xception (Left) & NASNETMobile (Right)

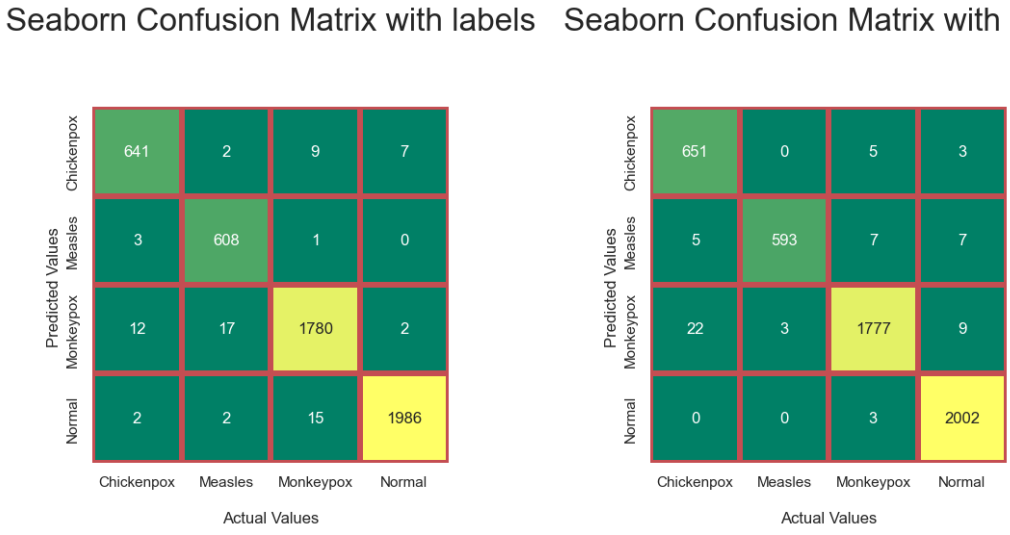

Figure 5.18: Confusion Matrix of Ensemble 1 (Left) & Ensemble 2 (Right)

Chapter 6

Conclusion and Future work

6.1 Conclusion

The objective of the research we conducted was to aid in preventing another pandemic comparable to COVID-19. Our contributions are shown in Fig 6.1.

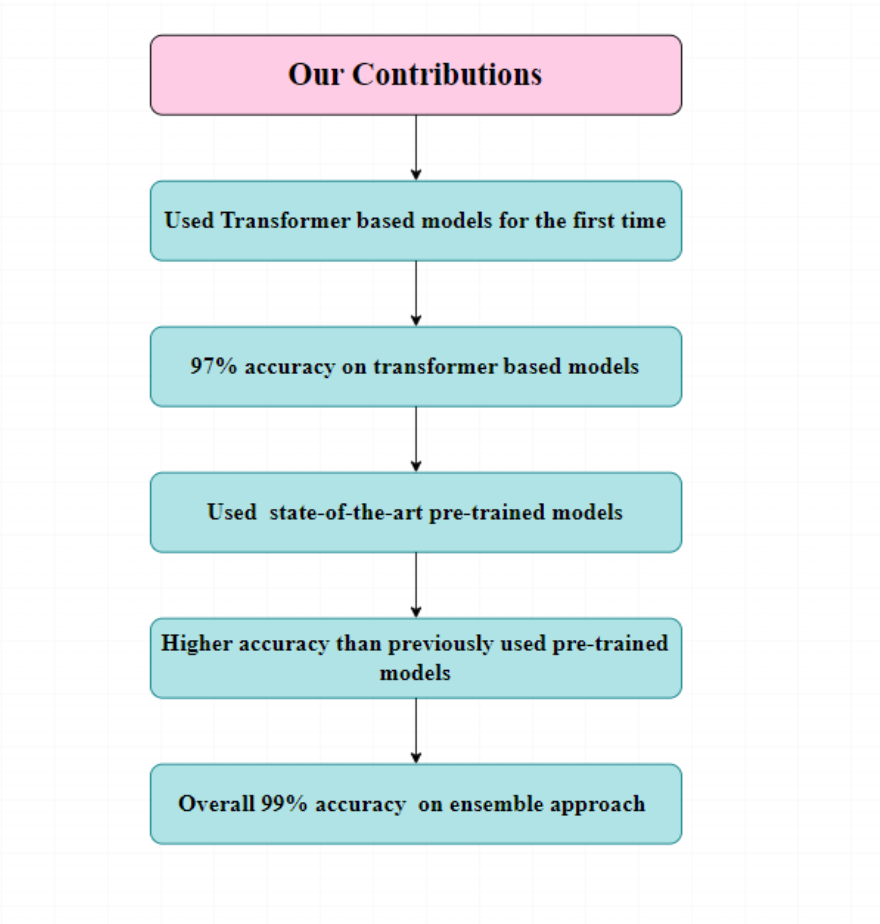

Figure 6.1: Our Research Contributions

Few advancements have been made in detecting Mpox because many people are unaware that it exists. In order to implement and compare recent transformer-based and pre-trained deep learning models, we employed deep learning. In the first chapter of our paper, we discussed how the MPOX situation poses a threat to humanity,

which is why we chose to focus on this particular topic. In addition, in Chapter 2, we described associated studies in the field of zoonotic disease identification from images. Then, we performed an extensive analysis to figure out that this sector should receive more attention, as this type of detection system can also be developed for other skin lesions. Next, in Chapter 3 of our thesis, we provided a brief description of the models we developed to detect and classify images of Mpox-infected skin from three other classifications. The most interesting parts of our writing are chapters 4 and 5, as they comprise our experimental setup, proposed model implementation, results, and analysis of the gathered data. Thus, our thesis concluded by identifying our limitations and future goals.

6.2 Limitations and Future Work

We hope to advance our research by compiling an extensive collection of data that includes numerous additional skin conditions. At first, we tried to collect datasets from the hospitals in our community and from other institutions via email, but we were only able to get MSID. The difficulty we ran into was the potential breach of patient confidentiality that can occur when sharing a patient's photo. As there are numerous patients and our country has not yet had many cases of Mpox (monkeypox), obtaining the patients' authorization to share their photographs would take time. Therefore, traveling would be necessary for us to obtain the patient's consent, which would take time and money. Even if we were to obtain clearance from some universities, we would still need a medical expert's skills to categorize the dataset we had gathered, and accessing them would be exceedingly challenging because they were conducting their own study. We will look for more datasets in the future to use for our research. In order to expand the class of photographs we already have and enhance our present dataset for more precisely identifying skin with Mpox (monkeypox) infection, we are looking into the collection of additional skin illnesses. Finally, there are incredibly few pre-trained models. More models will be trained using the images we currently have. Last but not least, we will try to apply our findings to a system that will help doctors provide primary care for all skin conditions, not just Mpox (monkeypox). Last but not least, we will attempt to create an app for the patients' immediate and primary diagnosis and care.

Bibliography

[1] G. HAYWARD, "Remarks on smallpox, cowpox and varioloid," *The Boston Medical and Surgical Journal*, vol. 62, no. 9, pp. 173–177, 1860. DOI: 10.1056/NEJM186003290620901. eprint: https://doi.org/10.1056/NEJM186003290620901. [Online]. Available: https://doi.org/10.1056/NEJM186003290620901.

[2] A. M. McCollum and I. K. Damon, "Human Monkeypox," *Clinical Infectious Diseases*, vol. 58, no. 2, pp. 260–267, Oct. 2013, ISSN: 1058-4838. DOI: 10.1093/cid/cit703. eprint: https://academic.oup.com/cid/article-pdf/58/2/260/961034/cit703.pdf. [Online]. Available: https://doi.org/10.1093/cid/cit703.

[3] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, Apr. 2015. [Online]. Available: https://arxiv.org/abs/1409.1556.

[4] L. D. Nolen, L. Osadebe, J. Katomba, *et al.*, *Extended human-to-human transmission during a monkeypox outbreak in the democratic republic of the congo*, Jun. 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4880088/.

[5] K. Roy, S. Chaudhuri, S. Ghosh, S. Dutta, P. Chakraborty, and R. Sarkar, "Skin disease detection based on different segmentation techniques," Mar. 2019, pp. 1–5. DOI: 10.1109/OPTRONIX.2019.8862403.

[6] K. Roy, S. S. Chaudhuri, S. Ghosh, S. K. Dutta, P. Chakraborty, and R. Sarkar, "Skin disease detection based on different segmentation techniques," in *2019 International Conference on Opto-Electronics and Applied Optics (Optronix)*, 2019, pp. 1–5. DOI: 10.1109/OPTRONIX.2019.8862403.

[7] E. Alakunle, U. Moens, G. Nchinda, and M. I. Okeke, "Monkeypox virus in nigeria: Infection biology, epidemiology, and evolution," *Viruses*, vol. 12, no. 11, 2020, ISSN: 1999-4915. DOI: 10.3390/v12111257. [Online]. Available: https://www.mdpi.com/1999-4915/12/11/1257.

[8] I. A. Abdullah Ali Salamai El-Sayed M. El-kenawy, "Dynamic voting classifier for risk identification in supply chain 4.0," *Computers, Materials & Continua*, vol. 69, no. 3, pp. 3749–3766, 2021, ISSN: 1546-2226. DOI: 10.32604/cmc.2021.018179. [Online]. Available: http://www.techscience.com/cmc/v69n3/44146.

[9] Z. Liu, Y. Lin, Y. Cao, *et al.*, *Swin transformer: Hierarchical vision transformer using shifted windows*, Aug. 2021. [Online]. Available: https://arxiv.org/abs/2103.14030.

[10] M. V. Madhavan, A. Khamparia, D. Gupta, S. Pande, P. Tiwari, and M. S. Hossain, *Res-covnet: An internet of medical health things driven covid-19 framework using transfer learning - neural computing and applications*, Jun. 2021. [Online]. Available: https://link.springer.com/article/10.1007/s00521-021-06171-8#citeas.

[11] O. Sarumi, "Machine learning-based big data analytics framework for ebola outbreak surveillance," in Jun. 2021, pp. 580–589, ISBN: 978-3-030-71186-3. DOI: 10.1007/978-3-030-71187-0_53.

[12] M. M. Ahsan, M. R. Uddin, M. Farjana, A. N. Sakib, K. A. Momin, and S. A. Luna, *Image data collection and implementation of deep learning-based model in detecting monkeypox disease using modified vgg16*, Jun. 2022. [Online]. Available: https://arxiv.org/abs/2206.01862.

[13] M. M. Ahsan, M. R. Uddin, M. Farjana, A. N. Sakib, K. A. Momin, and S. A. Luna, *Image data collection and implementation of deep learning-based model in detecting monkeypox disease using modified vgg16*, Jun. 2022. [Online]. Available: https://arxiv.org/abs/2206.01862.

[14] M. M. Ahsan, M. R. Uddin, and S. A. Luna, *Monkeypox image data collection*, Jun. 2022. [Online]. Available: https://arxiv.org/abs/2206.01774.

[15] S. N. Ali, M. T. Ahmed, J. Paul, *et al.*, *Monkeypox skin lesion detection using deep learning models: A feasibility study*, Jul. 2022. [Online]. Available: https://arxiv.org/abs/2207.03342v1.

[16] T. Islam, M. A. Hussain, F. U. H. Chowdhury, and B. M. R. Islam, "A web-scraped skin image database of monkeypox, chickenpox, smallpox, cowpox, and measles," *bioRxiv*, 2022. DOI: 10.1101/2022.08.01.502199. eprint: https://www.biorxiv.org/content/early/2022/08/09/2022.08.01.502199.full.pdf. [Online]. Available: https://www.biorxiv.org/content/early/2022/08/09/2022.08.01.502199.

[17] T. Islam, M. A. Hussain, F. U. H. Chowdhury, and B. R. Islam, *Can artificial intelligence detect monkeypox from digital skin images?* Jan. 2022. [Online]. Available: https://www.biorxiv.org/content/10.1101/2022.08.08.503193v2.

[18] E.-S. M. El-kenawy, F. Albalawi, S. A. Ward, *et al.*, "Feature selection and classification of transformer faults based on novel meta-heuristic algorithm," *Mathematics*, vol. 10, no. 17, 2022, ISSN: 2227-7390. DOI: 10.3390/math10173144. [Online]. Available: https://www.mdpi.com/2227-7390/10/17/3144.

[19] J. V. M. Lara and R. M. A. Velásquez, "Low-cost image analysis with convolutional neural network for herpes zoster," *Biomedical Signal Processing and Control*, 2022.

[20] U. M, G. H. Lakshman, KhwairakpamBidyanandaSingh, and S. Pande, "Detection of covid from chest x-rays using gan," *EPRA International Journal of Research and Development (IJRD)*, vol. 7, no. 5, pp. 166–175, May 2022. [Online]. Available: http://www.eprajournals.net/index.php/IJRD/article/view/453.

[21] C. Sitaula and T. B. Shahi, *Monkeypox virus detection using pre-trained deep learning-based approaches - journal of medical systems*, Oct. 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10916-022-01868-2.

[22] C. Sitaula and T. B. Shahi, *Monkeypox virus detection using pre-trained deep learning-based approaches - journal of medical systems*, Oct. 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10916-022-01868-2.

[23] C. Sitaula and T. B. Shahi, *Monkeypox virus detection using pre-trained deep learning-based approaches - journal of medical systems*, Oct. 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10916-022-01868-2.

[24] N. Yadav, S. Alfayeed, A. Khamparia, B. Pandey, D. Thanh, and S. Pande, "Hsv model-based segmentation driven facial acne detection using deep learning," *Expert Systems*, vol. 39, Mar. 2022. DOI: 10.1111/exsy.12760.

[25] D. Bala, *Monkeypox skin images dataset (msid)*, Feb. 2023. [Online]. Available: https://data.mendeley.com/datasets/r9bfpnvyxr.

[26] [Online]. Available: https://www.who.int/emergencies/situations/monkeypox-oubreak-2022.

[27] A. A. Abdelhamid, *Article versions notes*. [Online]. Available: https://www.mdpi.com/2227-7390/10/19/3614/notes.

[28] K. D. Akin. [Online]. Available: https://dergipark.org.tr/tr/download/article-file/2636020.

[29] K. D. Akin. [Online]. Available: https://dergipark.org.tr/tr/download/article-file/2636020.

[30] S. N. Ali and M. T. Ahmed, *Monkeypox skin lesion detection using deep learning models: A feasibility study*. [Online]. Available: https://arxiv.org/format/2207.03342.

[31] F. H. Athina. [Online]. Available: http://dspace.bracu.ac.bd:8080/xmlui/bitstream/handle/10361/14995/16136004_MNS.pdf?sequence=1.

[32] T. Bhatt, *A review on covid-19*. [Online]. Available: https://www.springerprofessional.de/en/a-review-on-covid-19/18882544.

[33] A. Dosovitskiy. [Online]. Available: https://arxiv.org/pdf/2010.11929.pdf.

[34] K. He. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

[35] B. J.-R. N. M. E. A. I; *Human monkeypox, 1970-79*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/6249508/.

[36] A.-S. M; *Addressing the physicians' shortage in developing countries by accelerating and reforming the medical education: Is it possible?* [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/28979916/.

[37] A. Mishra1. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/2109/2109.00886.pdf.

[38] P.-Y. Nguyen and W. S. Ajisegiri, *Reemergence of human monkeypox and declining population immunity in the context of urbanization, nigeria, 2017–2020 - volume 27, number 4-april 2021 - emerging infectious diseases journal - cdc*. [Online]. Available: https://wwwnc.cdc.gov/eid/article/27/4/20-3569_article.

[39] K. N. A. H. SN; *The 2022 outbreak and the pathobiology of the monkeypox virus*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/35760647/.

[40] A. Vaswani. [Online]. Available: https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.