international Journal of Intelligent Systems

Stratification of Human Monkey-pox from Skin Lesion Images based on Transformer-Based Ensemble Learning using GRAD-CAM

Ibne Hassan,[1] Raida Mobashshira Tahsin,[1] Sadia Shara Toma,[1] Tausif Tazwar Quadria Utchhash, [1] Dr. Muhammad Iqbal Hossain, [1] Rafeed Rahman, [1]

[1] School of Data and Sciences, BRAC University, Dhaka, 1212, Bangladesh.

Correspondence should be addressed to ibne.hassan@g.bracu.ac.bd, raida.mobashshira.tahsin@g.bracu.ac.bd, sadia.shara.toma@g.bracu.ac.bd, tausif.tazwar.quadria.utchhash@g.bracu.ac.bd, iqbal.hossain@bracu.ac.bd, rahman.rafeed@bracu.ac.bd

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.

Introduction

The global epidemic of COVID-19 during the year 2020 shook the world, and now there are reports that MPOX may arrive in 2022 [1]. 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 [2]. The MPXV virus, a member of the orthopox virus family, is responsible for the transmissible illness known as MPOX. The MPOX virus (MPXV, or MPOX) [3] 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, Cˆote d'Ivoire, Liberia, Sierra Leone, and Nigeria were among the nations that had seen the epidemic in the past [5]. 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 [6] 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 [7]. 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 [8,9]. This rash can manifest anywhere on the body, including the face, hands, throat, feet, mouth, groin, and pelvic and/or anal regions [10]. 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 [11], 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. Despite the fact that MPOX is an ancient disease that has recently resurfaced, people lack the knowledge necessary to recognize the symptoms at an early stage. According to the World Health Organization, 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 specialized 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. 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 main contributions to this paper are outlined below:

- Introducing a new approach for identifying MPOX patients.
- Construct the new system using 14 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

Materials and methods :

In this section, we describe our proposed methodology, data pre-processing, data augmentation and experimental setup.

A. The Proposed Methodology : Fig. 1 illustrates the overall system for the detection of monkeypox, which consists of several phases. Raw images were first passed through the pre-processing pipeline. Data augmentation (resizing, shuffling, and normalization) was done in the pre-processing 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.

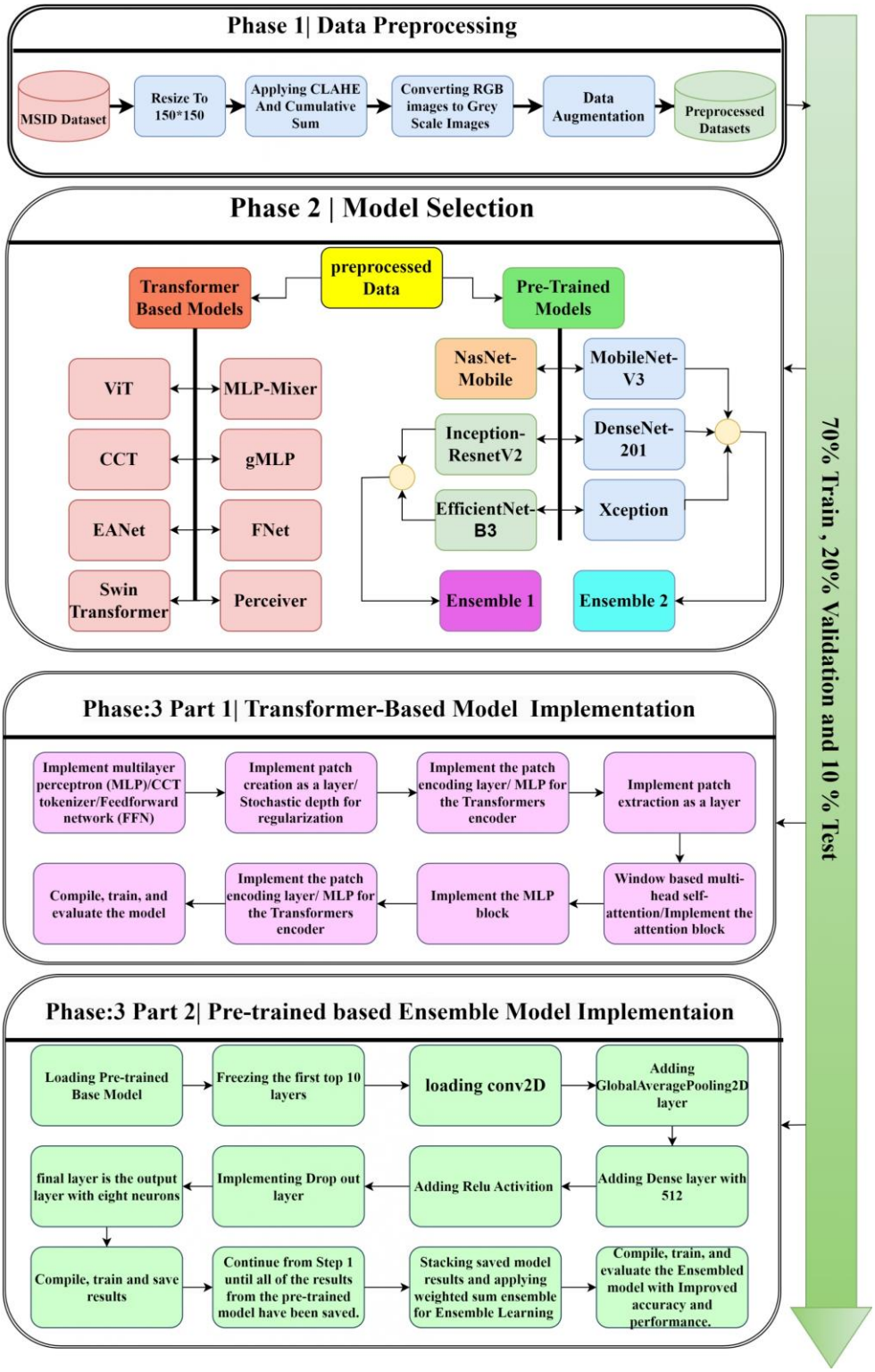


Figure 1: The Proposed Methodology

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.

B. Dataset : The experiments conducted in this study were based on the publicly available Monkeypox Skin Images Dataset (MSID) in Fig. 2 on Kaggle [12]. 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 shown in TABLE I. 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.

Figure 2: Dataset Image Samples.

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 [13,14]. 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), 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. 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.

C. Evaluation Metrics : We compared the performance of classification models using four widely used performance metrics: accuracy (Eq. (1)), F1-score (Eq. (2)), recall (Eq. (3)) and precision (Eq. (4)). 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.

TABLE I :
Dataset Statistics

Category or Class	Number of Samples
Monkeypox	279
Chickenpox	107
Measles	91
Normal	293
Total	770

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.

D. Experimental Setup: Experiments have been carried out on a device with the hardware characteristics demonstrated here: The CPU is the Intel Core i7 12th Generation, and the GPU is the NVIDIA RTX3080 Ti GeForce with 12 GB of memory and 64 GB of memory. 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 greater than or equal to 32 phases, allowing the model training procedure to be completed in a short amount of time. The libraries used in this research are OpenCV, Tensorflow, Keras, Matplotlib, Scipy, SKLearn, Seaborn, Numpy, Pandas, and Pathlib.

E. Pre-processing of Dataset:

Spreading data images, resizing input datasets, using CLAHE to apply histogram equalization, and enhancing the data all served as pre-processing 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. As it is easier to train image data in JPG format. To maximize space usage and keep the file size as small as possible, JPG is the best option. Then it was determined that virtually all of the histogram images spanned from 0 to 255. But the bar was not evenly distributed and fluctuated. In order to verify the issue, we imported the OpenCV library and Matplotlib to plot the graph and read the picture from a NumPy array. A point in the range bar was noted to be too high. Additionally, it was discovered that the extreme left and right portions lack pixel intensity values. This displays how our test picture lacks contrast. To resolve this issue, histogram equalization (HE) was utilized to disperse pixel intensity over the range. Histogram equalization is effective when the image’s histogram is constrained to a particular region. When the histogram covers a large area and there are both bright and dark pixels, it will not function properly in some areas. Moreover, histogram equalization (HE) alone was insufficient to repair it. Next, we used Contrast Limited Adaptive Histogram Equalization (CLAHE), which addresses both the problem of

insufficient contrast and the too-bright or too-dark issue of histogram equalization (HE). This is done for the luminance channel, and the values obtained by balancing a picture's luminance channel are significantly higher than the values obtained by having equal values for all channels in a BGR image. When implementing CLAHE, there are two factors or parameters we focused on: Clip limit to specify the contrast limiting criterion. The standard tile value is 3.0. And GridSize, which sets the number of tiles per column and per row. The possible value is 8 by 8. It is utilized when the picture is tiled for CLAHE implementation. The dataset was categorized into two sets. One with cumulative sum and CLAHE, and the other one without cumulative sum and CLAHE. After CLAHE, we chose to convert all photos to grayscale. The main benefit of using grayscale images to get descriptions instead of working directly with color photos is that the methods are easier and less computer power is needed. Perfect skin tones, softer shadows and gradients, sharp, clear small- and knock-out designs, and fine detail are all provided by grayscale. Smoother color transitions and finer detail may be seen with more levels of grayscale that are accessible. Moreover, better accuracy was found in gray-scale images than in colorful images. Since color does not affect image categorization, grayscale photos are recommended.

F. Data Augmentation : As our dataset was small, we used dataset augmentation to increase the number of images in our dataset. Fig. 3 shows the methods used for improving images to create a pipeline. Before augmenting, all photographs in the training and validation sets for each class were scaled to 224 by 224 pixels. We chose to enhance the dataset 30 times. We will encounter bias if we augment more. For instance, enhancing the data 50 times results in bias [15]. The dataset is initially enlarged using the ImageDataGenerator package from the Keras image processing framework. The ImageDataGenerator function includes numerous choices, including rotation, inversion, and width and height adjustment. ImageDataGenerator's comprehensive utility is included. TABLE II shows how this parameter is added to the dataset. The facility type and generator type are thus chosen at random [16]. Our proposed model is built with Python-based Keras [17,18]. The parameters have been adjusted as follows: We make use of online data enrichment. After resizing each image to 150*150 for augmentation, we use the following settings:

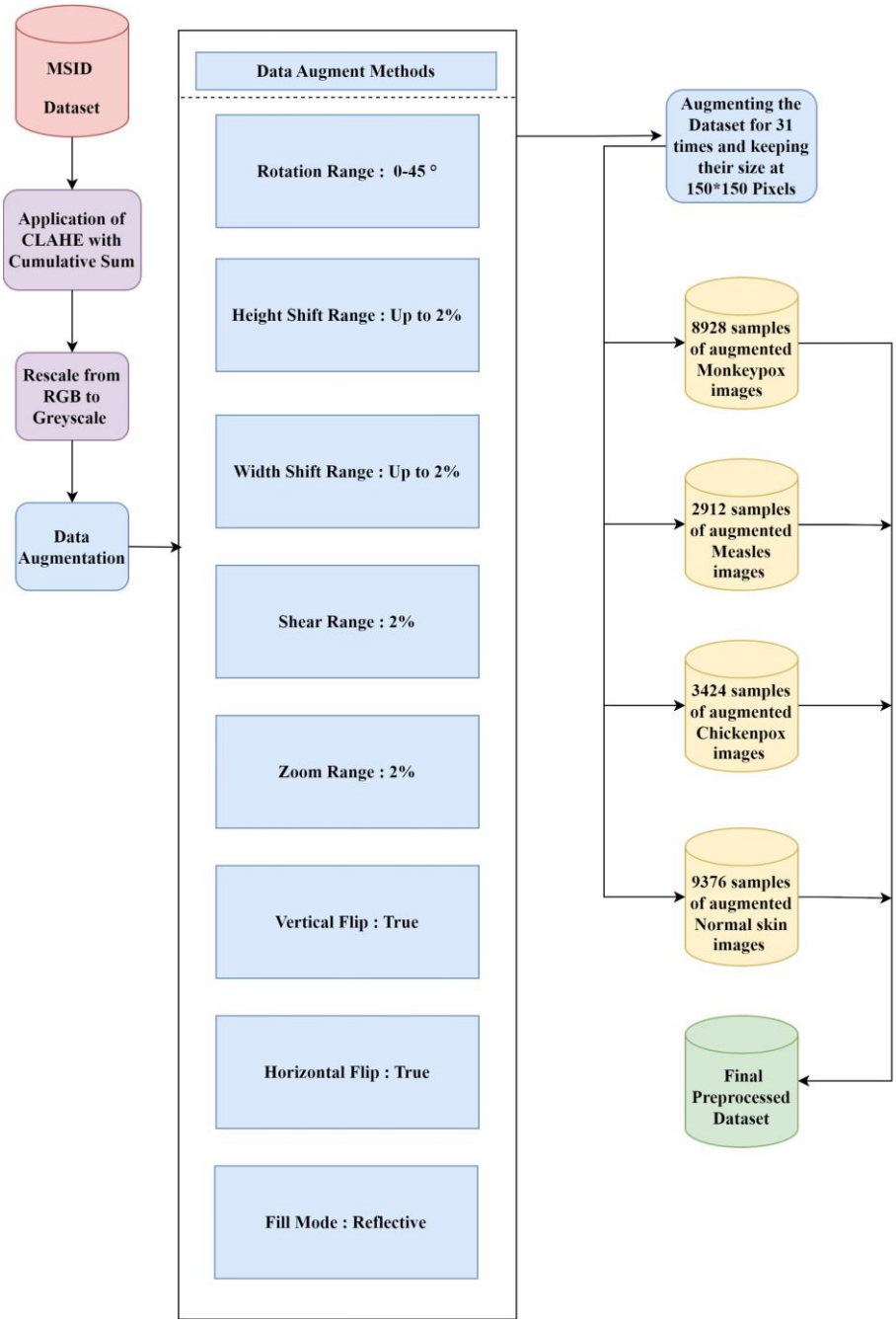


Figure 3: Data Pre-processing & Augmentation Diagram.

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.

TABLE II :
Parameters used in data augmentation

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 III :
Number of Samples Before & after augmentation

Category or Class	Samples Before Augmentation	Samples After Augmentation
Monkeypox	279	8928
Chicken pox	107	3424
Measles	91	2912
Normal	293	9376

G. 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 any features from the dataset. During training, the selected models learn to extract features.

H. Hyper-parameter 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.

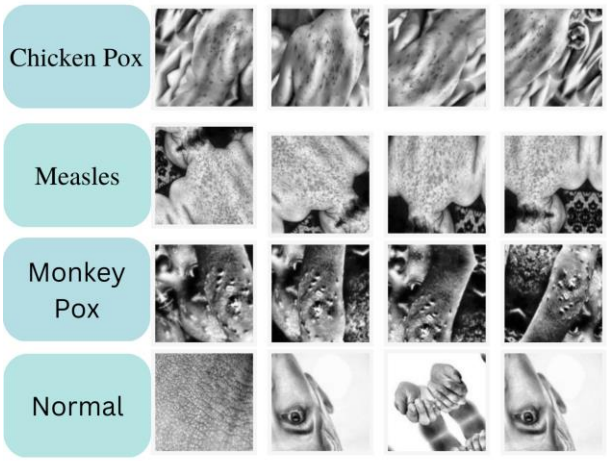


Figure 4 : Dataset Images After Applying CLAHE and Data Augmentation

In TABLE IV, 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.

Again, in TABLE V 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,

TABLE IV :
Step by step manual Hyper-parameter Tuning 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	96

Here, DR=Dropout Rate, IS=Image size, LR=Learning Rate, WD=Weight Decay, BS=Batch Size, A=Accuracy, P= Precision

TABLE V :
Step by step manual Hyper-parameter Tuning for CCT

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	99

Here, DR=Dropout Rate, IS=Image size, LR=Learning Rate, WD=Weight Decay, BS=Batch Size, A=Accuracy, P= Precision

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.

I. 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 VI. In addition, each accuracy and precision matrix 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.

TABLE VI :
Baseline Evaluation (after hyper-parameter tuning for each model multiple times).

Model	DR	IS	LR	WD	BS	A (%)	P (%)
ViT	0.05	128	0.001	0.0001	500	94	96
CCT	0.1	128	0.001	0.05	32	97	99
MLP-Mixer	0.5	72	0.001	0.0001	64	92	96
gMLP	0.5	72	0.001	0.0001	64	93	98
Fnet	0.03	72	0.001	0.0001	32	87	96
Swin Transformer	0.03	72	0.001	0.0001	32	72	88
EAnet	0.02	72	0.001	0.0001	16	65	77
Perceiver	0.01	72	0.001	0.0001	128	65	86
NASNet Mobile	0.04	140	0.001	0.0001	128	95	96
InceptionResnetV2	0.4	140	0.001	0.0001	128	96	99
EfficientNetB3	0.4	140	0.001	0.0001	128	96	99
Ensemble1 (Inception ResnetV2 + EfficientNetB3)	0.4	140	0.001	0.0001	128	99	100
MobileNetV3Large	0.4	140	0.001	0.0001	128	84	95
DenseNet201	0.4	140	0.001	0.0001	128	97	98
Xception	0.4	140	0.001	0.0001	128	98	99
Ensemble2 (Mobile NetV3Large + DenseNet201+Xception)	0.4	140	0.001	0.0001	128	99	100

Here, DR=Dropout Rate, IS=Image size, LR=Learning Rate, WD=Weight Decay, BS=Batch Size, A=Accuracy, P= Precision

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.

Model Training Implementation

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 ground breaking 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.

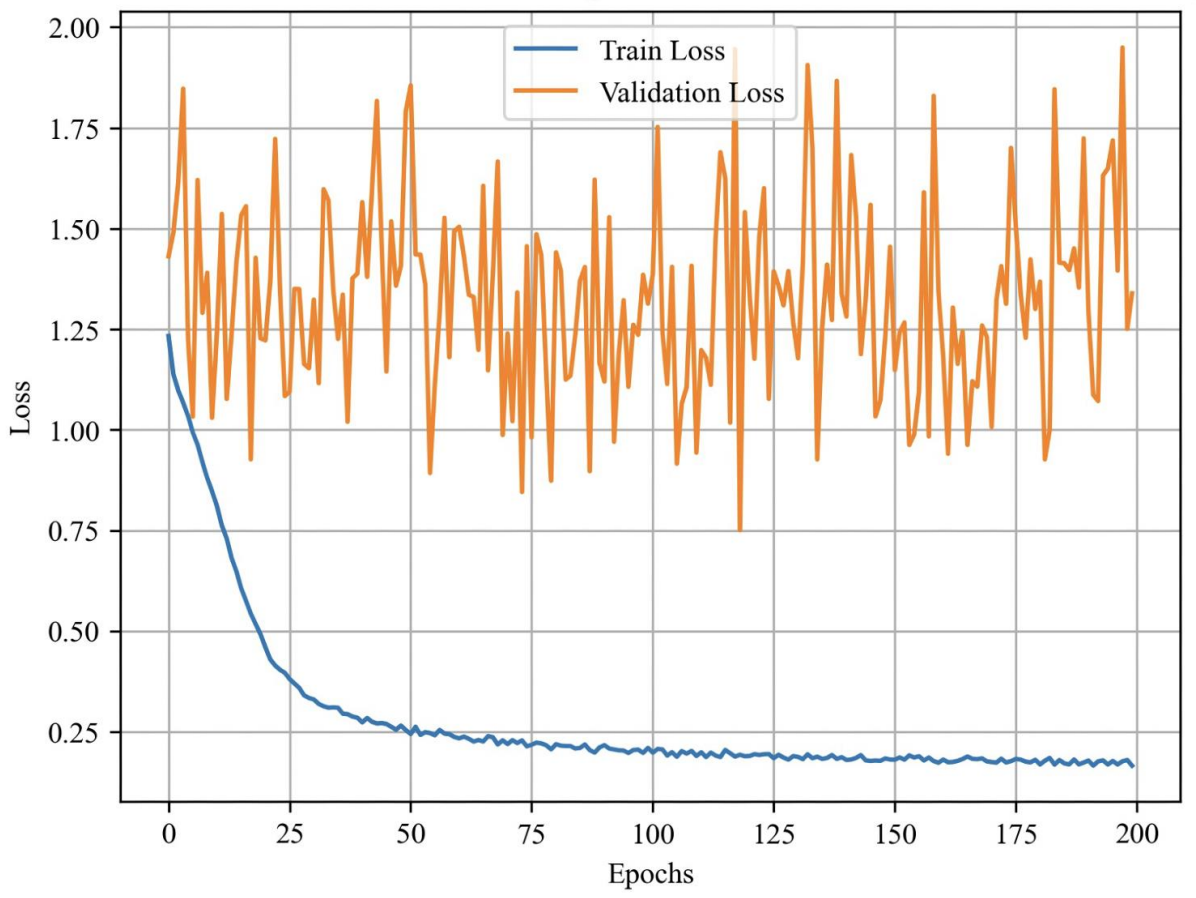


Figure 5: Line graphs depicting the training and validation loss of ViT over 200 training epochs.

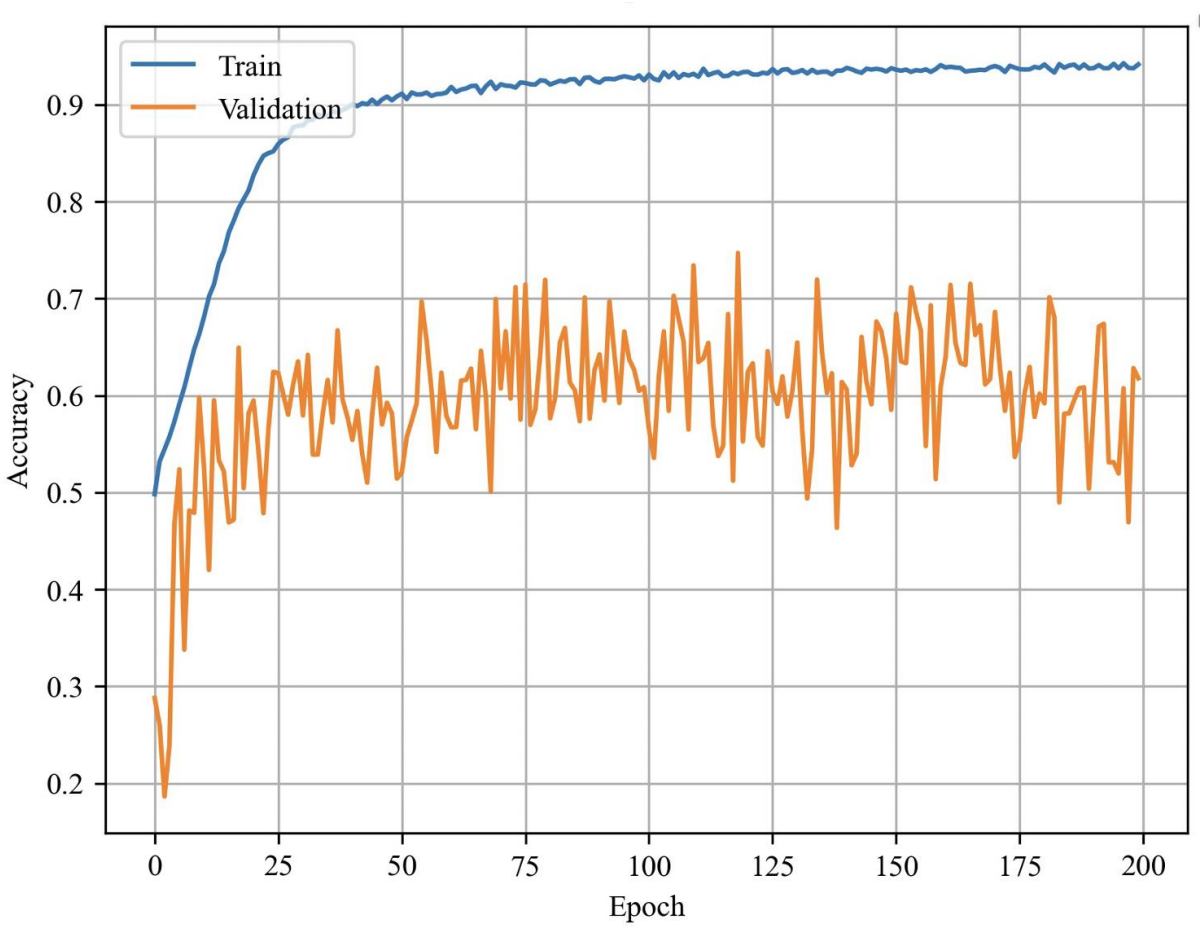


Figure 6: Line graphs depicting the training and validation accuracy of ViT over 200 training epochs.

TABLE VII :
Classification Report of Vision Transformer (ViT) :

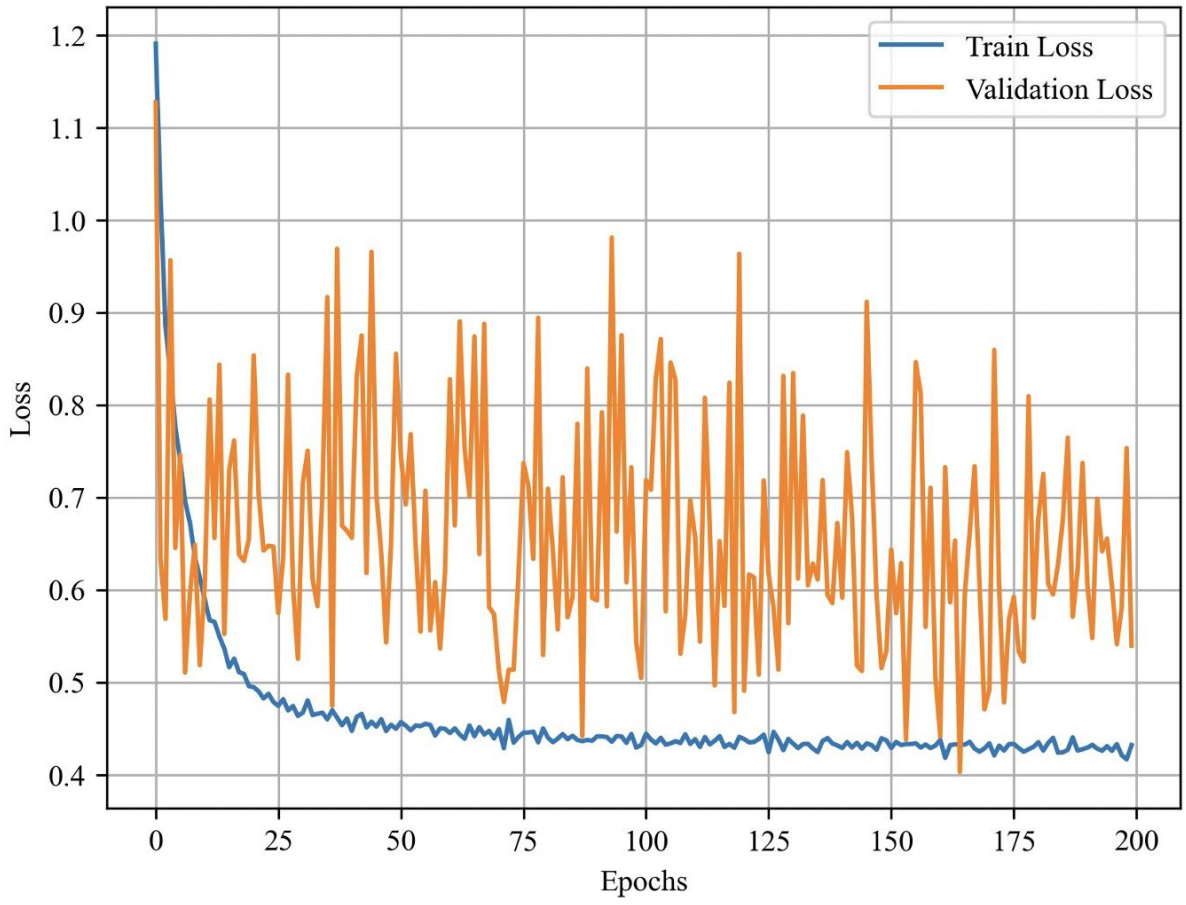
	P	**R**	**F**	**S**
Chickenpox	0.89	1.00	0.94	649
Measles	0.96	1.00	0.98	602
Monkeypox	0.90	1.00	0.95	1807
Normal	1.00	0.85	0.92	1980
Accuracy			0.94	5038
Macro average	0.94	0.96	0.95	5038
weighted average	0.94	0.94	0.94	5038

Here, P = Precision, R= Recall, F = f1-score , S = Support

Across 200 epochs, the ViT (Vision Transformer) achieved 94% accuracy and 96% precision in Fig.5. 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 utilised. 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 94%, a recall of 94%, and a f1-score of 94% shown in TABLE VII. 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.

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.



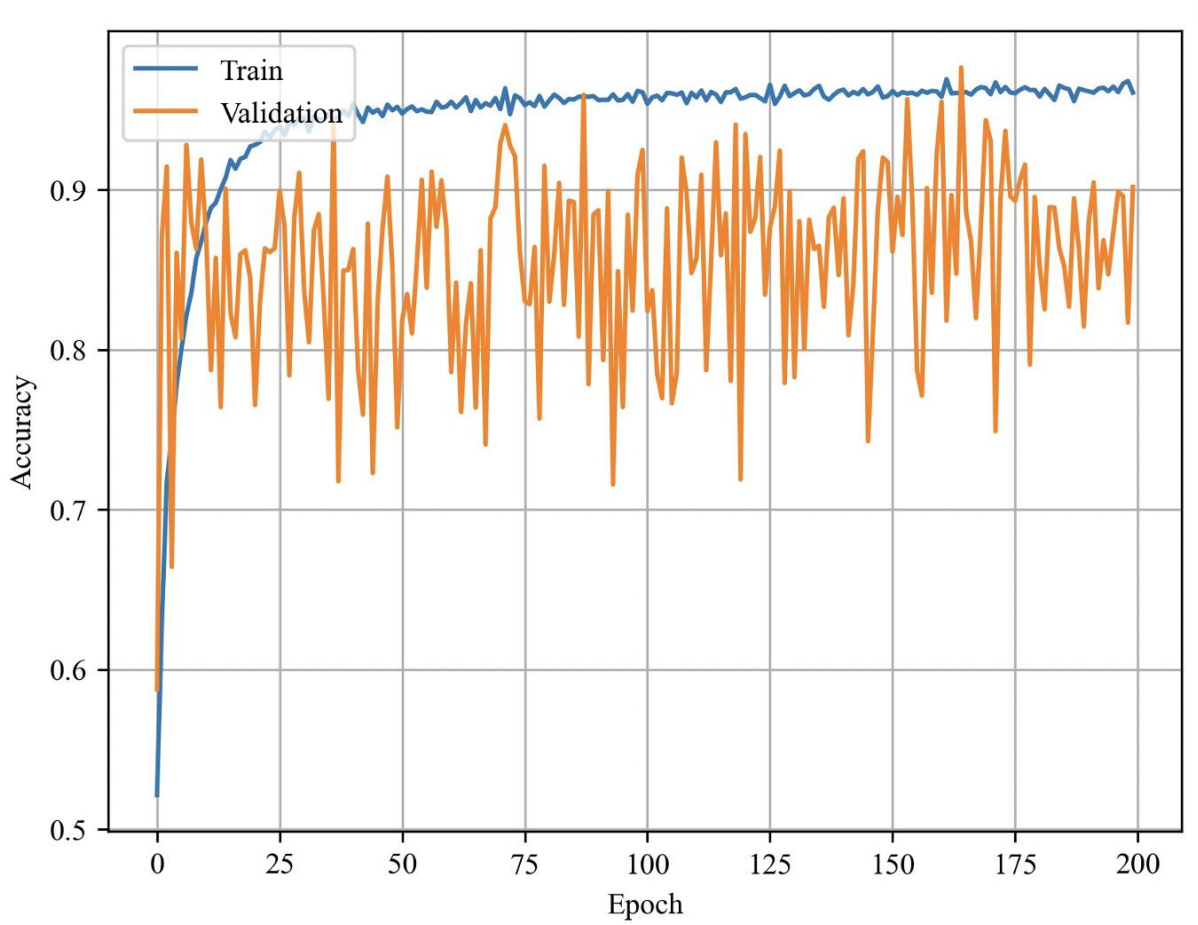


Figure 7 : Line graphs depicting (a) the training and validation loss and (b) training and validation accuracy of CCT over 200 training epochs.

TABLE VIII:
Classification Report of Compact Convolutional Transformers (CCT):

	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
Macro average	0.97	0.95	0.96	5038
weighted average	0.97	0.97	0.97	5038

Here, P = Precision, R= Recall, F = f1-score , S = Support

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, 97% precision, 97% recall, 97% f1-score, and 100% top-5 accuracy shown in Fig.6 and TABLE VIII. It outperformed other transformer-based models.

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 hyperparameter 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.

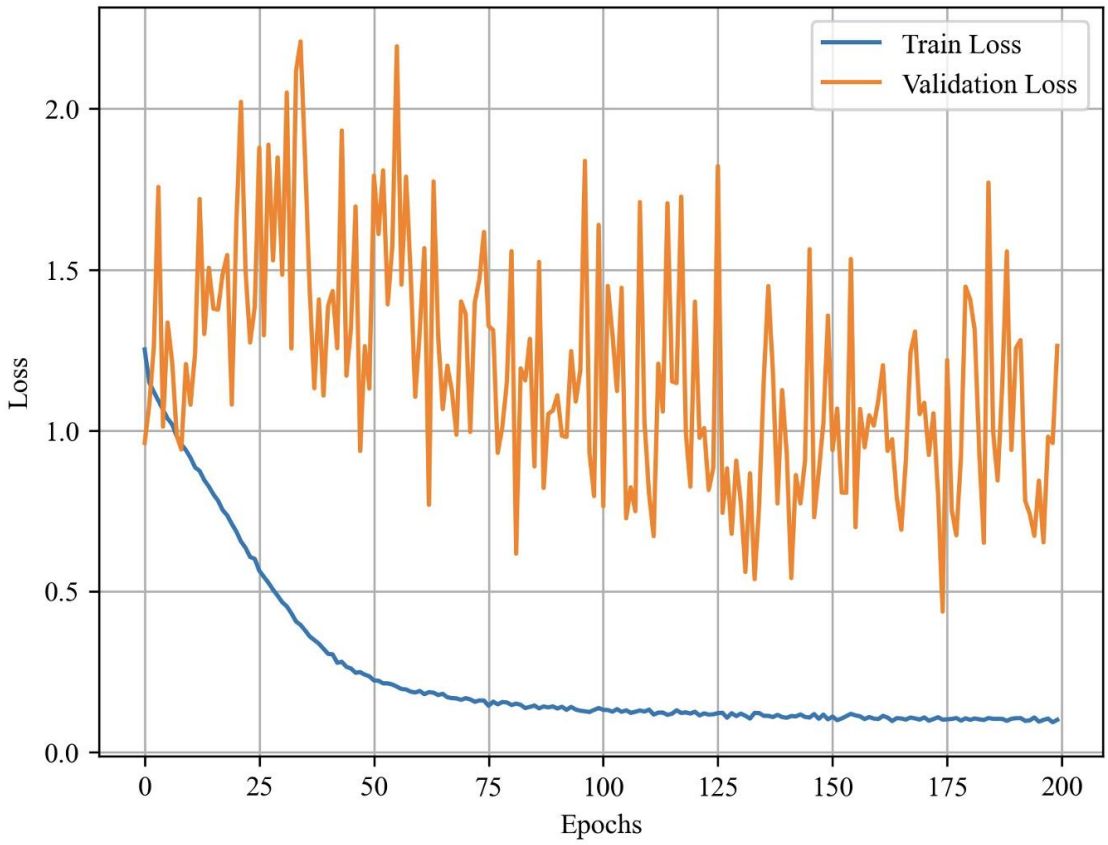


Figure 8: Line graphs depicting (a) the training and validation loss of MLP over 200 training epochs.

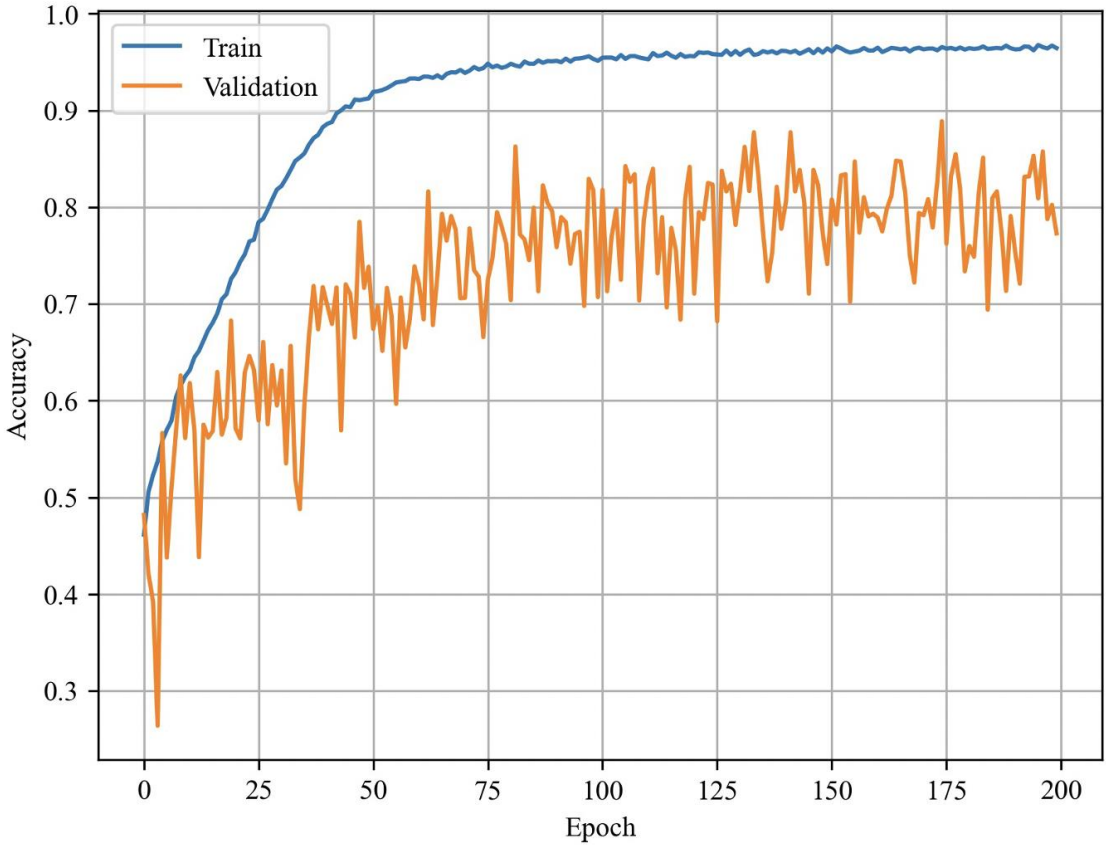


Figure 9: Line graphs depicting the training and validation accuracy (2) of MLP over 200 training epochs.

TABLE IX:
Classification Report of MLP-Mixer :

	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
Macro average	0.93	0.92	0.92	5038
weighted average	0.90	0.94	0.92	5038

Here, P = Precision, R= Recall, F = f1-score , S = Support

The alternative technique for blending location data along channels and across patches The MLP-Mixer model achieved 92% accuracy, 90% precision, 92% recall, 92% f1-score, and 1,068,299 parameters after being built, trained, and evaluated across 200 epochs shown in Fig.7 and TABLE IX.

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.

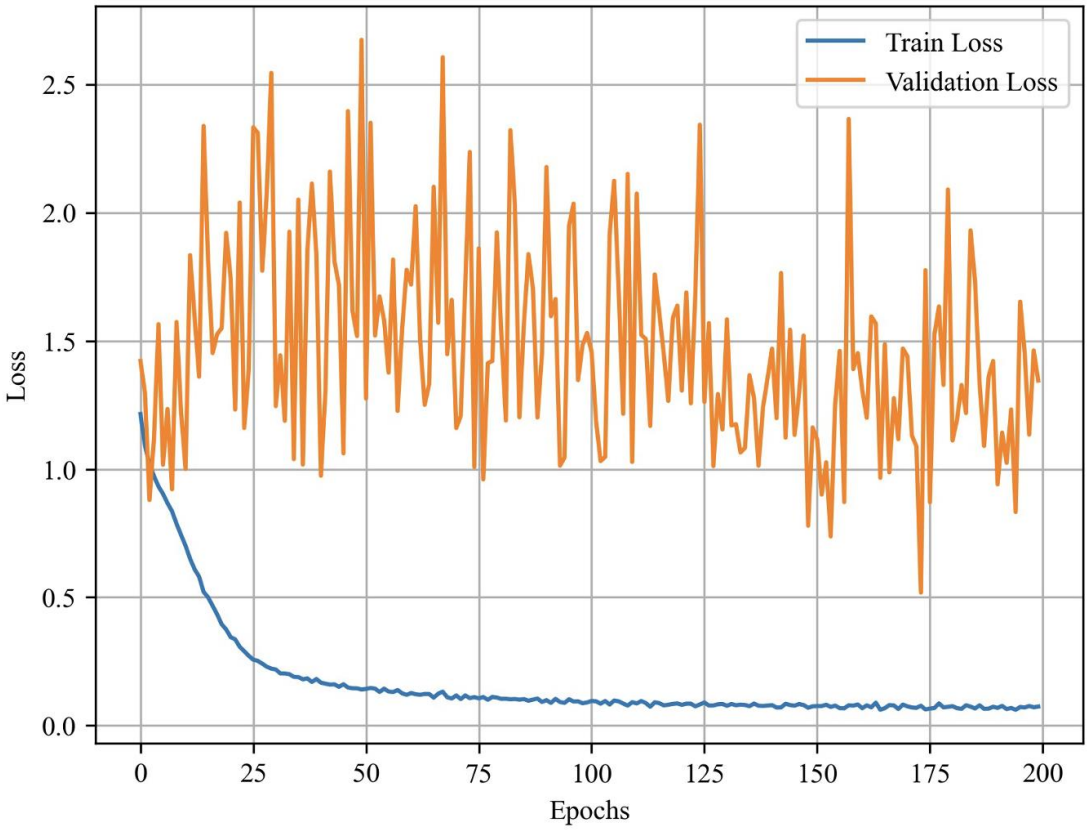


Figure 10 : Line graphs depicting the training and validation loss of gMLP over 200 training epochs.

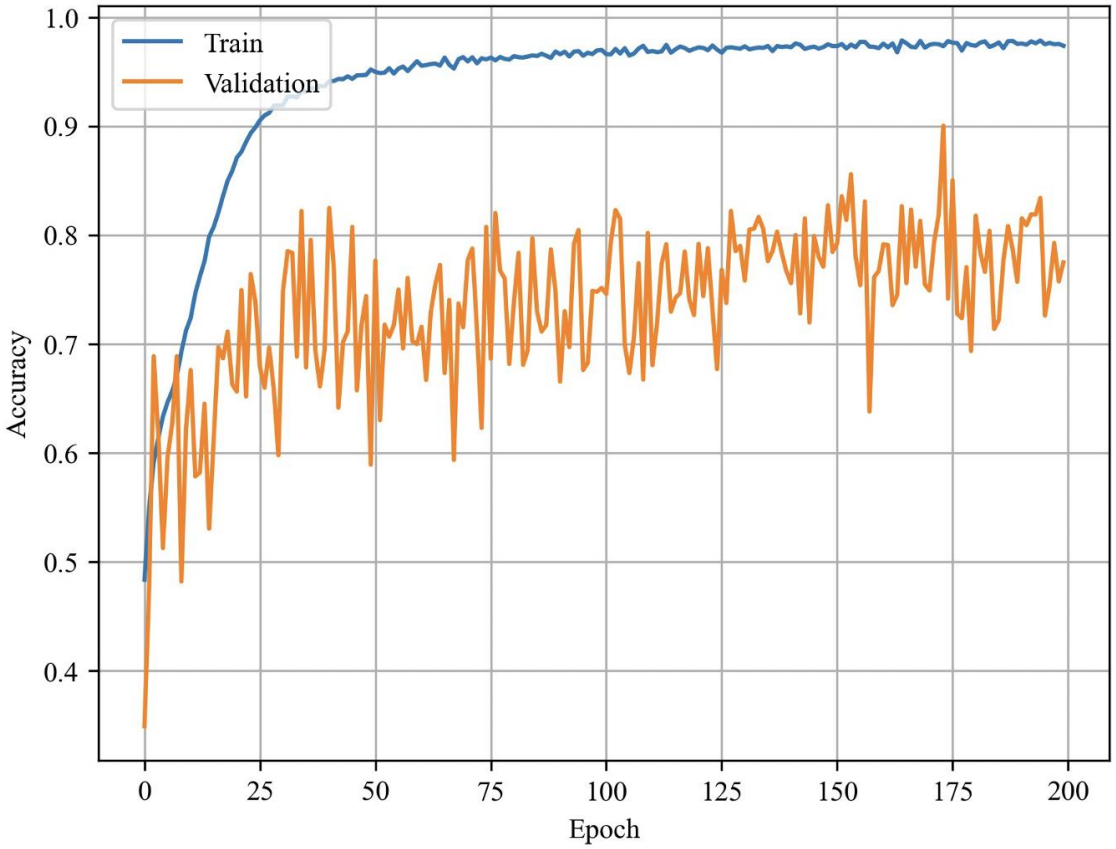


Figure 11 : Line graphs depicting the training and validation accuracy of gMLP over 200 training epochs.

TABLE X:
Classification Report of gMLP :

		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
Macro average		0.90	0.94	0.92	5038
weighted average		0.93	0.93	0.93	5038

Here, P = Precision, R= Recall, F = f1-score, S = Support

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, 93% precision, 94% recall, a 92% f1-score, and 823,819 parameters after being built, trained, and validated across 200 epochs shown in Fig.8 and TABLE X.

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.

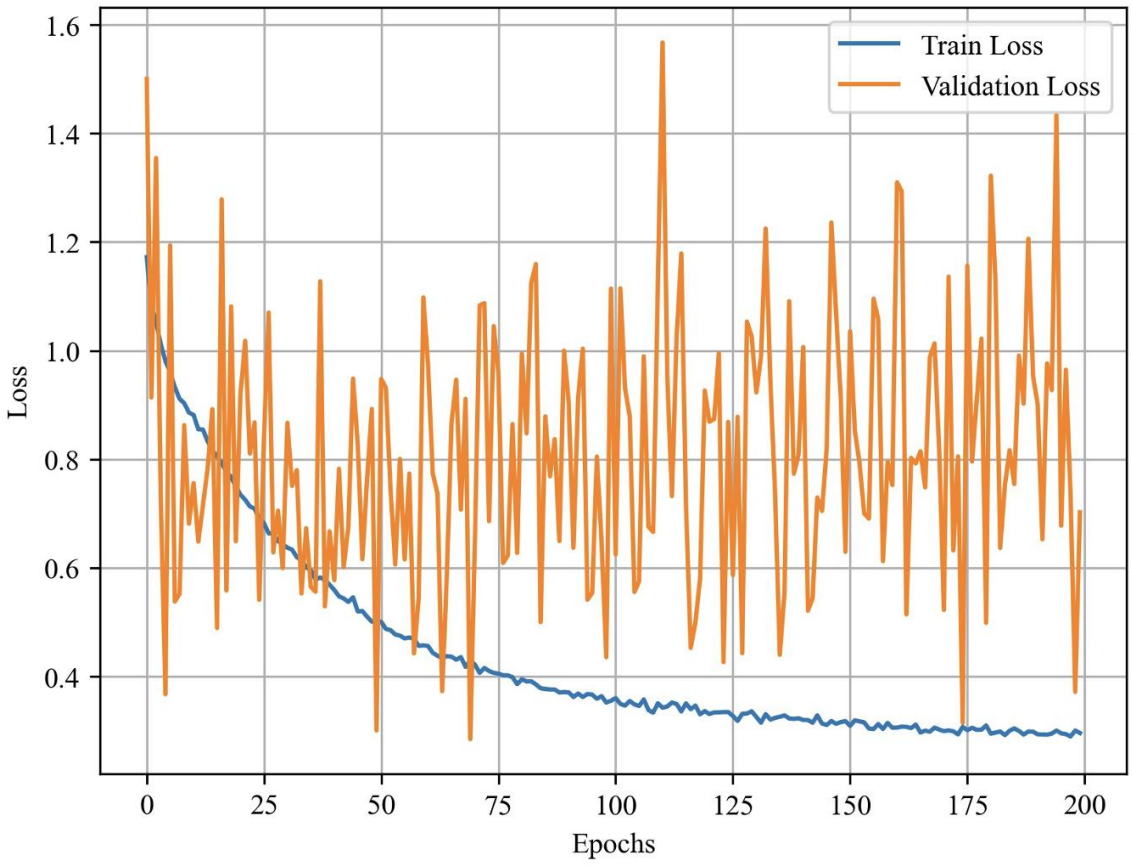


Figure 12: Line graphs depicting the training and validation loss of FNET over 200 training epochs.

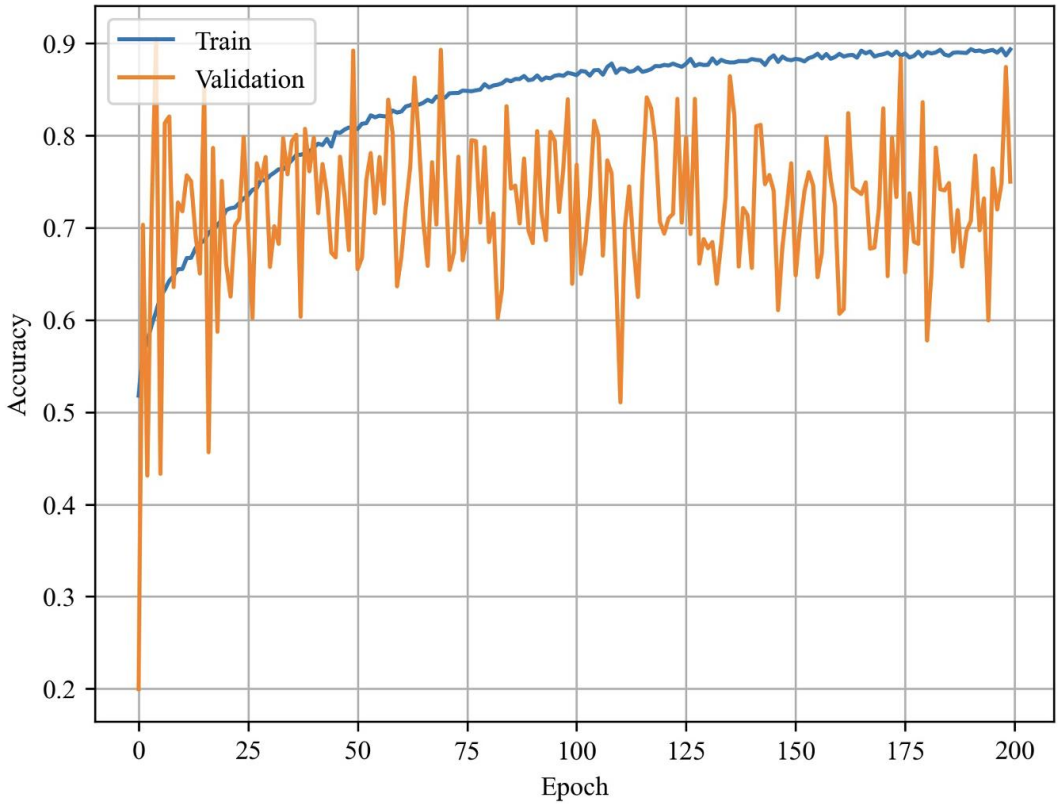


Figure 13: Line graphs depicting the training and validation accuracy of FNET over 200 training epochs.

TABLE XI:
Classification Report of FNET:

		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
Macro average		0.83	0.83	0.83	5038
weighted average		0.80	0.80	0.80	5038

Here, P = Precision, R= Recall, F = f1-score, S = Support

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 83% accuracy, 80% precision, 80% recall, 80% f1-score, and 823,819 parameters shown in Fig.9 and TABLE XI.

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.

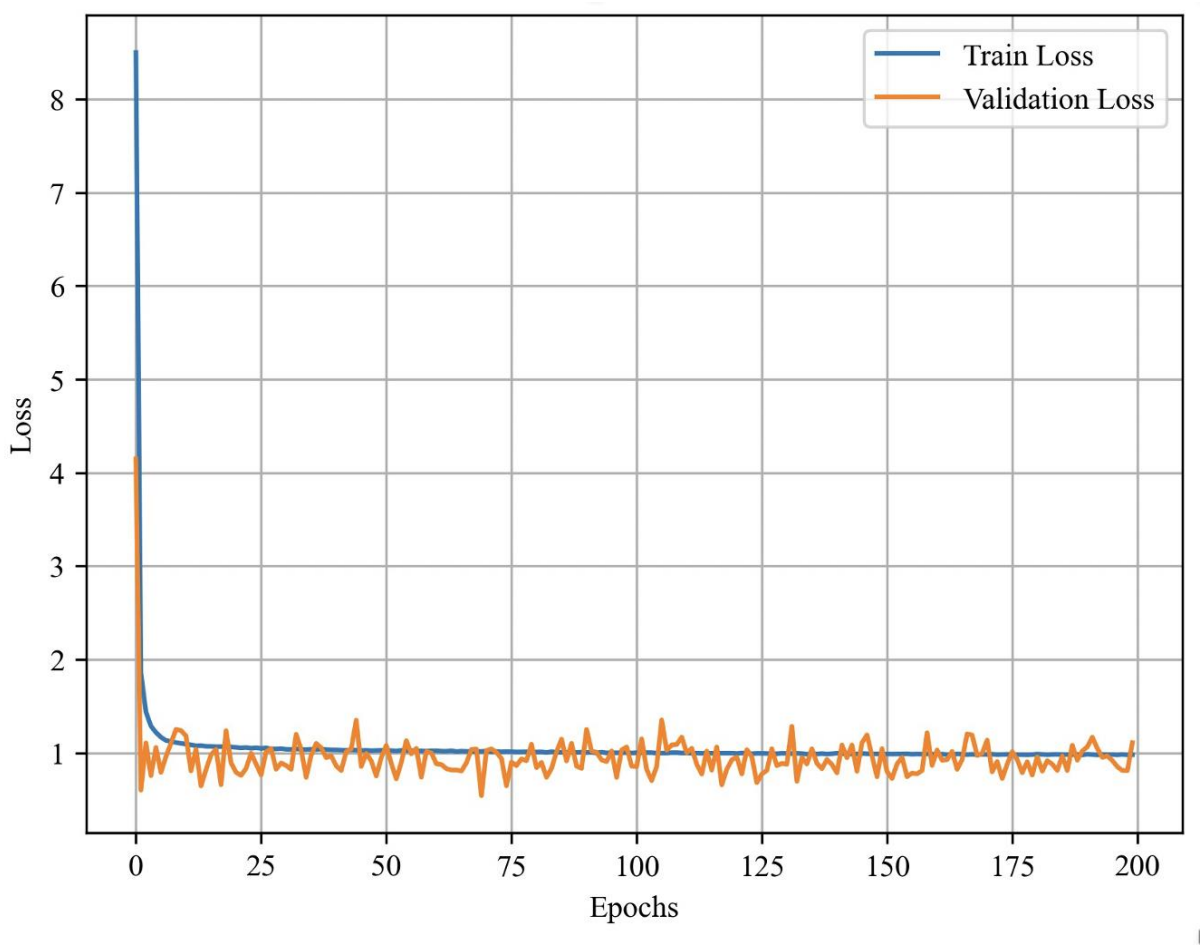


Figure 14 : Line graphs depicting the training and validation loss of Swin Transformer over 200 training epochs

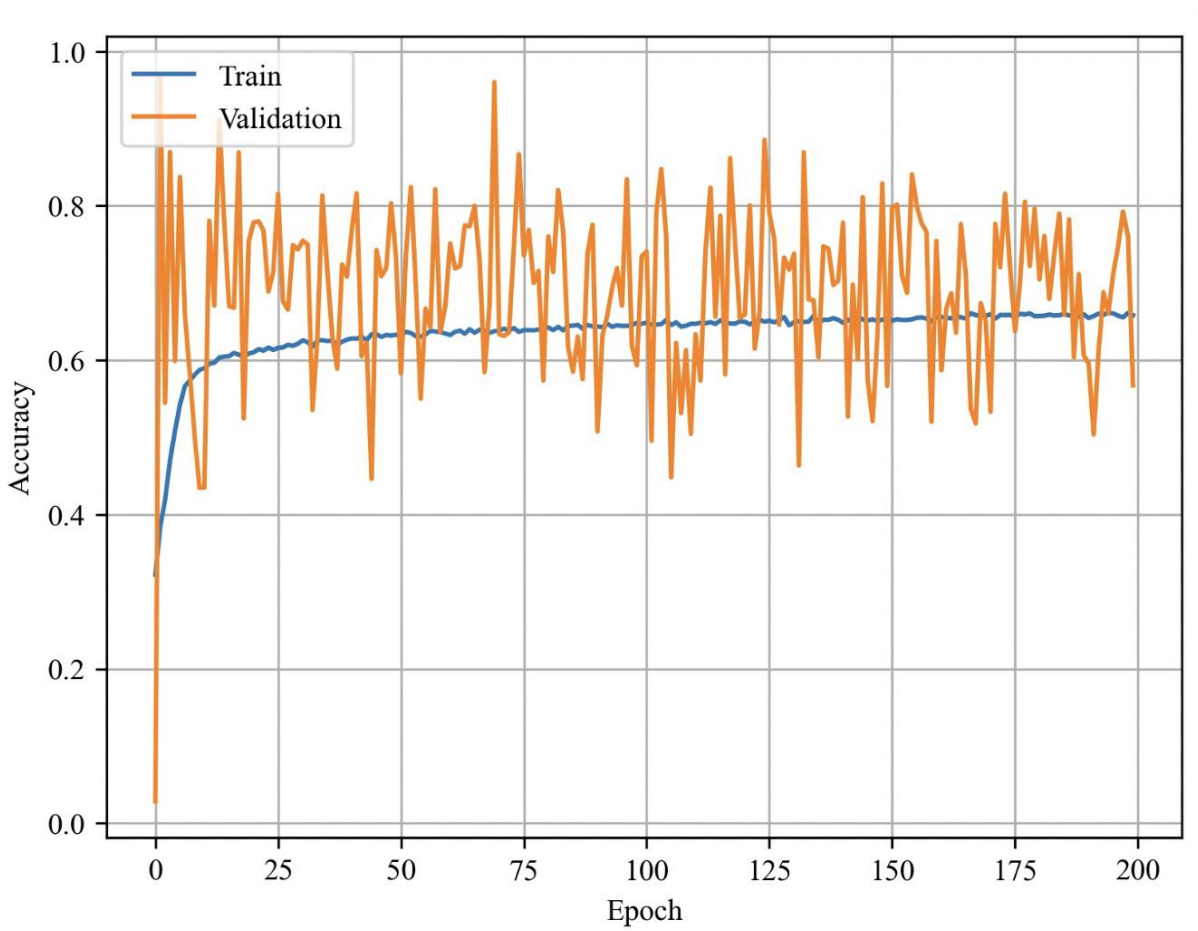


Figure 15 : Line graphs depicting the training and validation accuracy of Swin Transformer over 200 training epochs.

TABLE XII:
Classification Report of Swin Transformer:

		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
Macro average		0.75	0.72	0.71	5038
weighted average		0.74	0.62	0.64	5038

Here, P = Precision, R= Recall, F = f1-score , S = Support

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, 74% precision, 62% recall, 64% f1-score, and 100% top-5 accuracy shown in Fig.10 and TABLE XII. More fine-tuning and hyperparameter tweaking can improve the result.

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 transformers blocks 4.

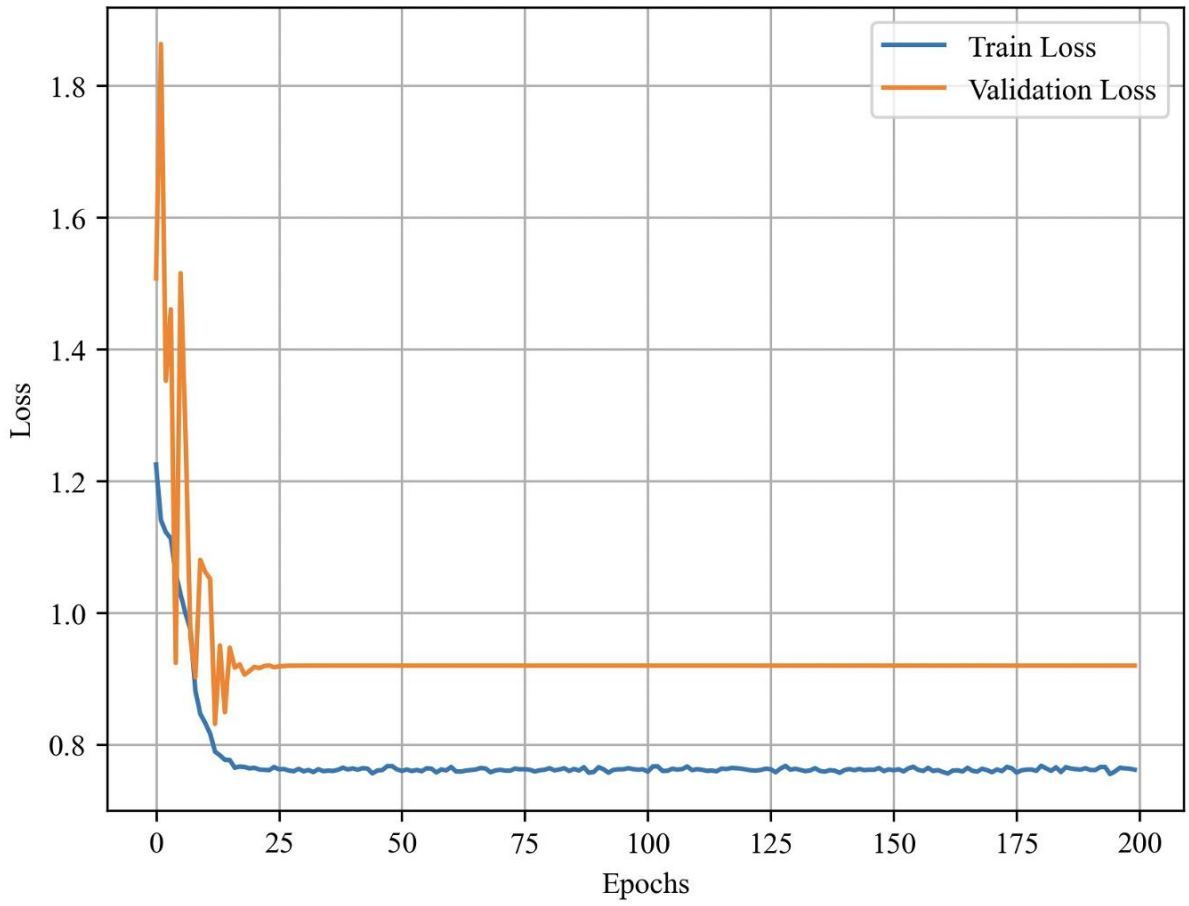


Figure 16 : Line graphs depicting the training and validation loss of perceiver over 200 training epochs.

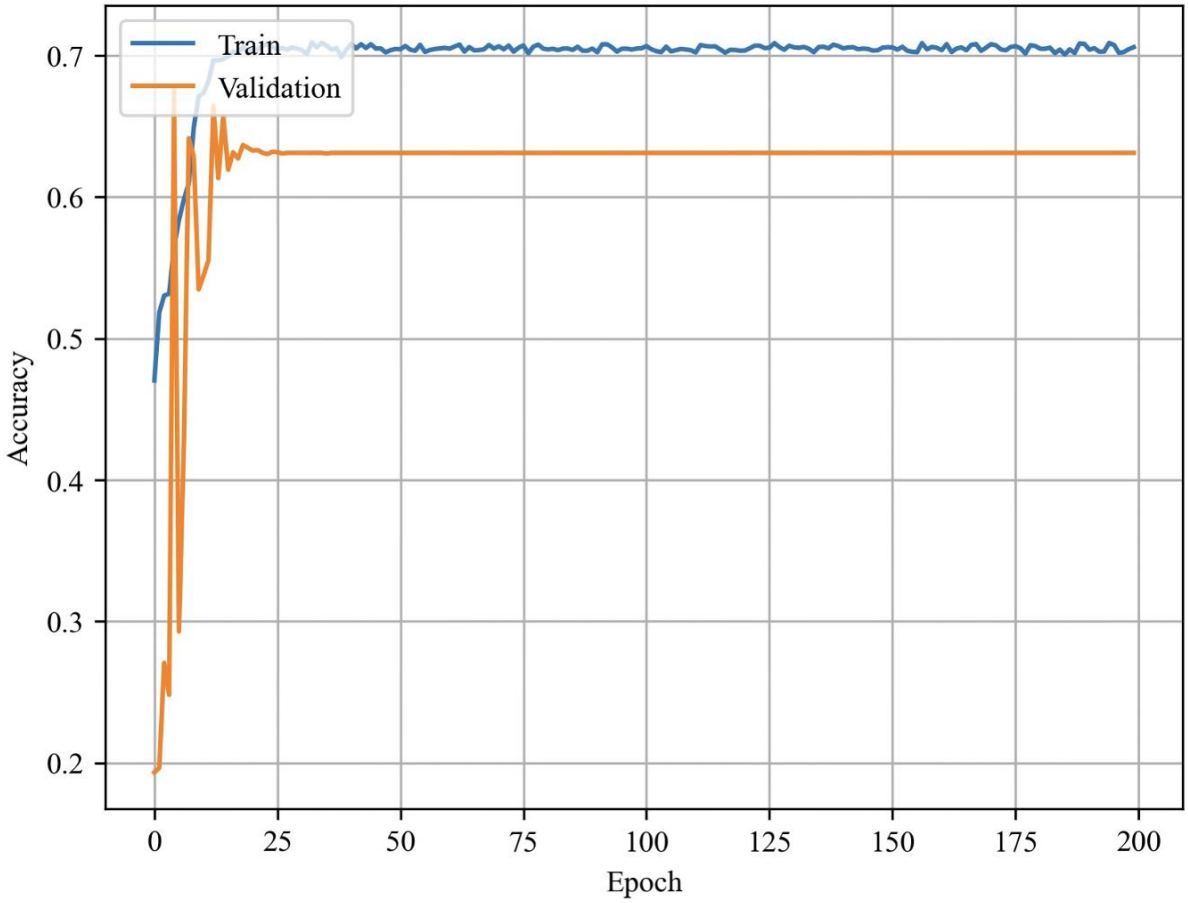


Figure 17 : Line graphs depicting the training and validation accuracy of perceiver over 200 training epochs.

TABLE XIII:
Classification Report of perceiver :

		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
Macro average		0.69	0.65	0.64	5038
weighted average		0.64	0.56	0.58	5038

Here, P = Precision, R= Recall, F = f1-score , S = Support

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, 64% precision, 56% recall, 58% f1-score, and 100% top-5 accuracy after 200 iterations of gathering and training

shown in Fig.11 and TABLE XIII. Additional fine-tuning and hyperparameter tweaking improve results.

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. 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.

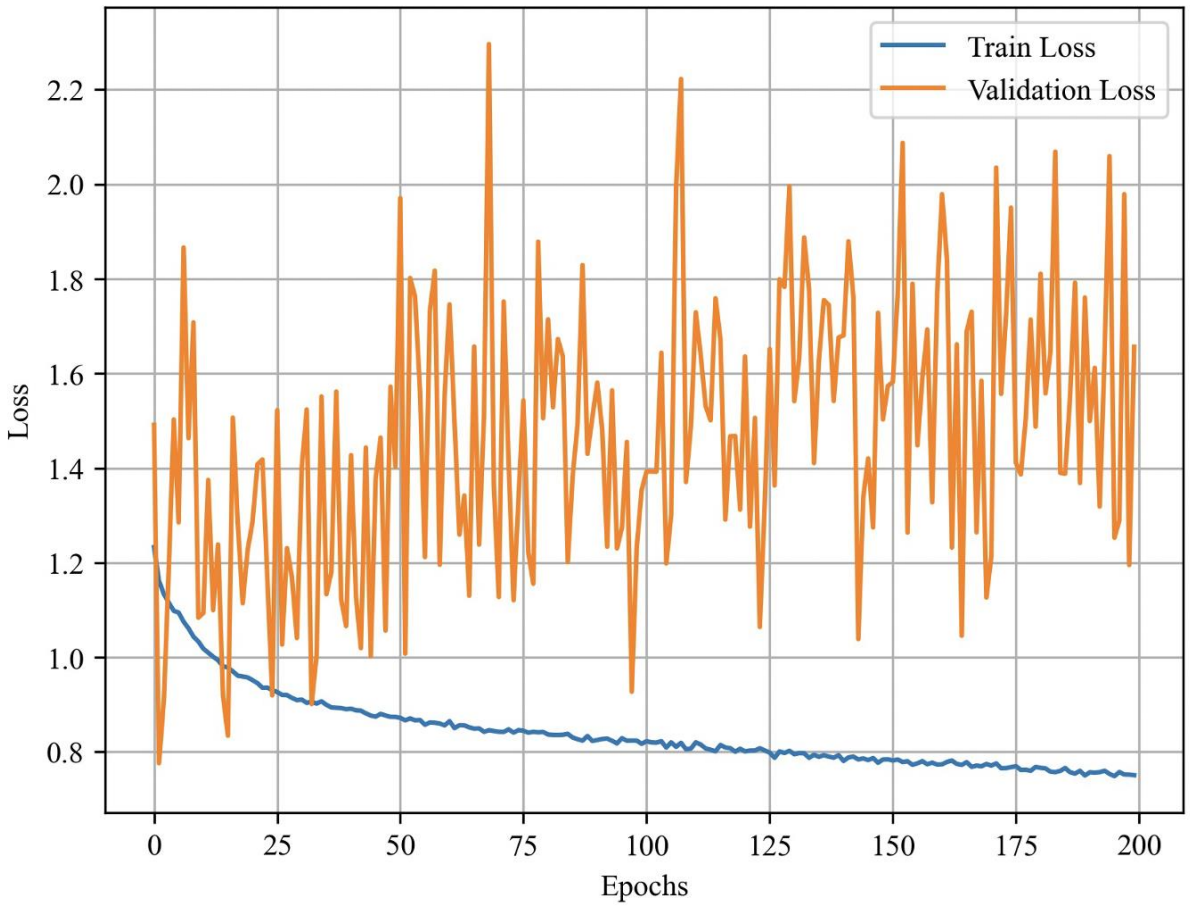


Figure 18 : Line graphs depicting (a) the training and validation loss of EANet over 200 training epochs.

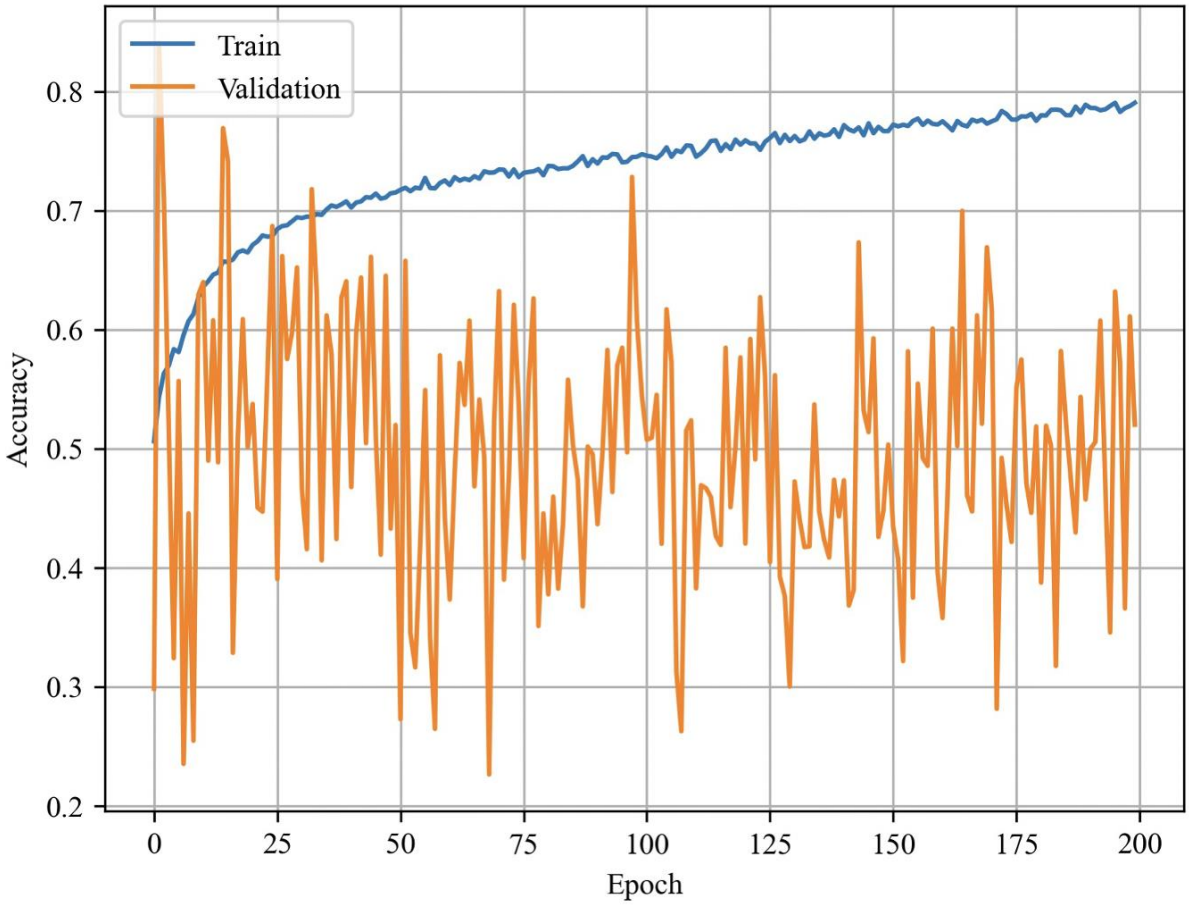


Figure 19 : Line graphs depicting the training and validation accuracy of EANet over 200 training epochs.

TABLE XIV:
Classification Report of External Attention Transformer (EANet) :

		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
Macro average		0.67	0.65	0.65	5038
weighted average		0.63	0.60	0.60	5038

Here, P = Precision, R= Recall, F = f1-score , S = Support

The MLP block and transformer block were also added, with 64 dimensions and 8 additional dimensions. The accuracy was 65%, precision was 63%, recall was 60%, the f1-score was 60%, and the top-5 accuracy was 100% after constructing, training for 200 epochs shown in Fig.12 and TABLE XIV and validating the findings on test data with

parameters of 356,979. More hyperparameter modification and fine tuning improves results.

NASNetMobile : NasNetMobile was employed for individual training in our study. NASNetMobile is a convolutional neural network trained on images from the ImageNet database. Mobile networks are low-latency, low-power models that may be customised to meet the resource constraints of a variety of use cases. They can be used to create systems for categorization, recognition, embedding, and separation. NasNetMobile's weights were ImageNet, and the classifier activation was SoftMax. Some of the novel layers are 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. Adam Optimizer is then used to fit, save, and build the models.

Figure 20: Line graphs depicting (a) the training and validation loss and (b) the training and validation accuracy of NASNetMobile over 20 epochs

TABLE XV:
Classification Report of NASNetMobile :

		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
Macro average		0.95	0.95	0.95	5087
weighted average		0.92	0.95	0.93	5087

Here, P = Precision, R = Recall, F = f1-score, S = Support

After 20 epochs of compiling and training, the test data with 472,299 parameters had 100% top-5 accuracy, accuracy of 95%, precision of 92%, recall of 95%, and a f1-score of 93% shown in Fig.13 and TABLE XV.

Ensemble Approach : We give more weight to classifiers or models with higher accuracies in an ensemble model (for example, InceptionResNetV2 and EfficientNetB3). In this situation, we used many models to enable the selection of a better model or classifier while minimising the risk associated with the wrong model selection.

We chose stacking, formerly known as stacked generalisation, as our study strategy. Stacking is based on the idea of merging many models to improve forecast accuracy over individual models. Thus, ensemble learning requires a minimum of two models for the stacking technique, and we chose InceptionResNetV2 and EfficientNetB3 for our first ensemble learning strategy. One of the key reasons for choosing these two models was that they had never been used for ensemble approaches. This ensemble group was never used in our dataset or to detect skin lesions like MPOX. They are the most recent and least-used pre-trained models at present.
That is why we chose to include them in our experiment. Model averaging is an ensemble method in which many sub-models contribute equally to the total anticipated. A model-averaging ensemble averages the predictions of multiple trained models. Because each model has its own set of advantages and disadvantages, it is usually preferable to develop multiple models that can extract different features and then aggregate their predictions. There were several strategies available for the ensemble approach, including simple average, weighted average, and weighted sum.
The average technique has the problem of providing the exact piece to the ensemble projection regardless of how well a model performs, and the implementation does not improve and may even degrade in the worst-case scenario. As a result, a weighted sum ensemble is the best strategy for improving overall performance. It essentially aggregates estimations from multiple models, with each model's contribution weighted based on its competence or quality and can deliver higher performance in every way when compared to the individual models' performance. Our ensemble model's first experiment was a big success, with 99% accuracy.

Figure 21: Line graphs depicting (a) the training and validation loss and (b) the training and validation accuracy of InceptionResNetV2 over 20 epochs.

Figure 22: Line graphs depicting (a) the training and validation loss and (b) the training and validation accuracy of EfficientNetB3 over 20 epochs.

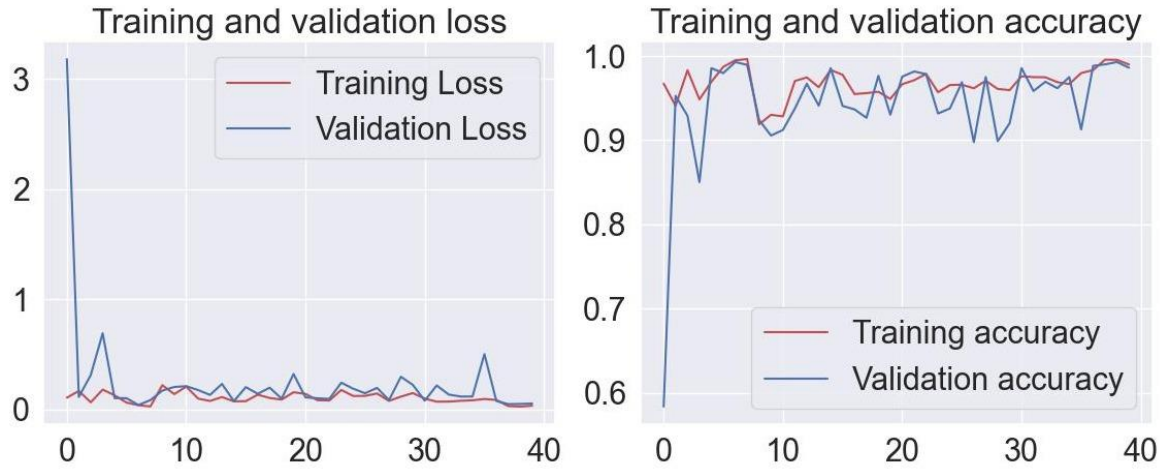


Figure 23: Line graphs depicting (a) the training and validation loss and (b) the training and validation accuracy of Ensemble 1 over 20 epochs.

The first model used was InceptionResNetV2, and the second was EfficientNetB3. There are 25436 files in four categories. In this instance, we used 20349 files for training and 5087 files for validation. Two models are first trained using our datasets. Individually, while freezing their initial top ten layers or not training with the top ten models. New layers were also added, including a dense layer with 512 neurons with RELU activation, a dropout layer with a rate of 0.2, and an output layer with eight neurons. Following that, the models are assembled using Adam Optimizer, fitted, and saved. Their best performances from each era are saved one by one in a HDF5 file over time. Model 1 (InceptionResNetV2) obtained 97% accuracy in 20 epochs, whereas model 2 (EfficientNetB3) achieved 98% accuracy in 20 epochs. The best saved results from each model are then combined using a weighted sum ensemble, with more weight given to the best-performing model. We combined it with 0.9 weight for EfficientNetB3 and 0.7 weight for InceptionResNetV2 because our EfficientNetB3 model outperformed InceptionResNetV2 by 98%. We did not assign a low weight to InceptionResNetV2 because it performed well on its own. Despite the fact that both

models performed brilliantly on their own, we hoped for even better results. Thus, we ensemble the model for 40 epochs using 0.7 and 0.9 weights and reach 99% accuracy shown in Fig.16, which exceeded our expectations and was more than sufficient.

Figure 24: Line graphs depicting (a) the training and validation loss and (b) the training and validation accuracy of MobileNetV3Large over 20 epochs.

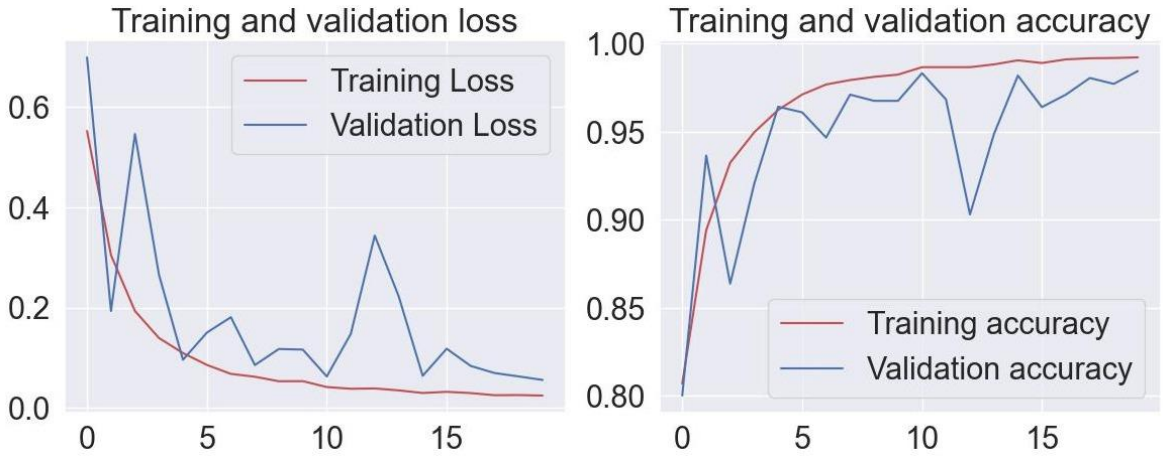


Figure 25: Line graphs depicting (a) the training and validation loss and (b) the training and validation accuracy of DenseNet201 over 20 epochs.

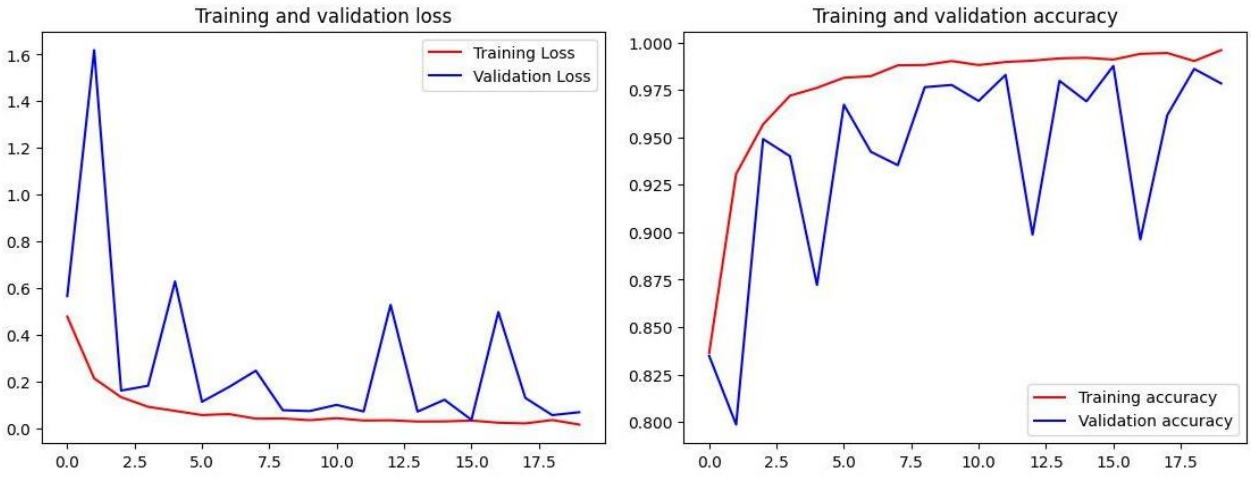


Figure 26: Line graphs depicting (a) the training and validation loss and (b) the training and validation accuracy of Xception over 20 epochs.

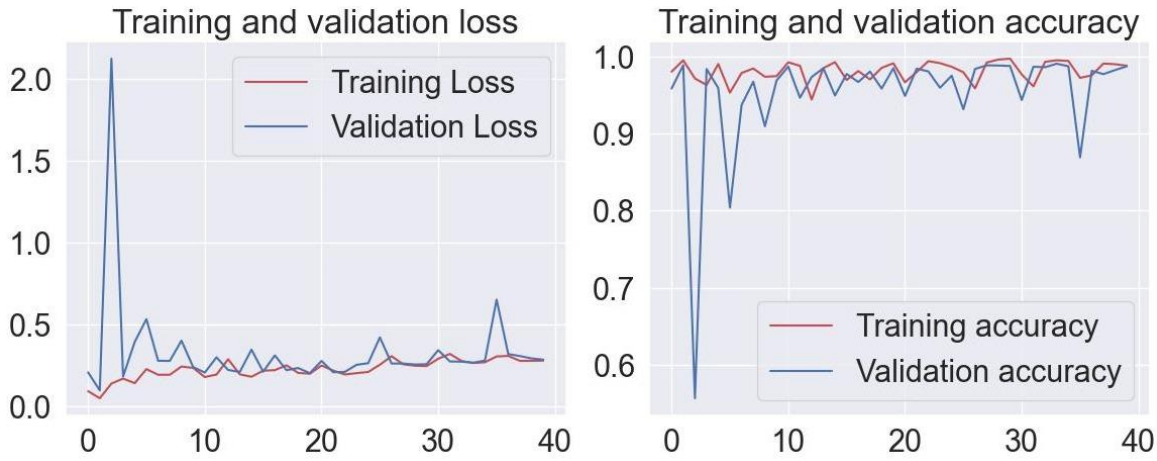


Figure 27: Line graphs depicting (a) the training and validation loss and (b) the training and validation accuracy of Ensemble 2 over 20 epochs.

In the second experiment of our ensemble model, we used the stacking ensemble strategy to widen our study to three models. We achieved 99% accuracy overall. The three models are MobileNetV3Large, DenseNet201, and Xception. In the datasets, we used 20349 files for training and 5087 files for validation. As with the initial ensemble technique, the top 10 layers of each model were then frozen. This includes the dense layer with 512 neurons and RELU activation, the dropout layer with a rate of 0.2, and the final output layer with eight neurons. Separately, MobileNetV3Large had an accuracy of 84%, DenseNet201 had an accuracy of 97%, and Xception had an accuracy of 98%. Then, using the weighted sum ensemble technique, we ensembled the results of all three saved models and obtained 99% accuracy shown in Fig.20, which was better than the individual accuracy. Finally, we can observe that the ensemble technique not only reduced the spread or dispersion of predictions and model performance, but it also increased the average prediction performance across all contributing members of the ensemble. There are two major, interconnected reasons to prefer an ensemble model over a single model: Robustness and performance. An ensemble is capable of making more accurate forecasts and outperforming any single model. Furthermore, an

ensemble reduces prediction dispersion and model performance. Ensemble approaches outperform individual models in terms of predicted accuracy. If a single model is unable to produce the proper forecast for a given data set, the chance of high variation, low accuracy, noise, and bias increases. We have a better possibility of enhancing the level of accuracy by merging multiple models.

Results and Discussion

Result Discussion :
Using precision, accuracy, recall, and f1-score, we evaluate the suggested approach to contemporary pre trained deep learning models on the MSID dataset. The results are shown in TABLE XVI and TABLE XVII. TABLE XVI and Fig.21 shows 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%). In TABLE XVII and Fig.22, overall, both of the proposed ensemble techniques beat all other individual models. Ensemble 1 has a 99.9% accuracy with two models (InceptionResnetV2 and EfficientNetB3). It has 100% Precision, 99% Recall, and 100% F1-score. Ensemble 2 with three models (MobileNetV3Large+ DenseNet201+ Xception) has a 99% accuracy and provides 100% Precision, 99% Recall, and 100% F1-score. Both ensemble models are 2% more accurate, 1% more precise, 1% more accurate in recall, and 2% more accurate in F1-score than the top transformer-based model (Compact Convolutional Transformers, or CCT). Compared to the least performing Transformer-based model (FNet), both ensemble models are 16% more accurate, 14% more precise, 10% more accurate in recall, and 12% more accurate in F1- score.

TABLE XVI: Performance Comparison of Transformer Based and Pretrained CNN Based DL Models.

Models	A(%)	P(%)	R(%)	F(%)	Parameter
ViT	94	96	100	94	3,343,819
CCT	97	99	98	97	407,365
MLP-Mixer	92	96	98	94	1,068,299
gMLP	93	98	99	94	823,819
Fnet	87	96	88	88	544,011
Swin Transformer	72	88	93	79	222,388
Eanet	65	77	82	71	356,979
Perceiver	65	86	90	71	9,742,591

NasNetmobile	95	96	97	96	5,556,508
InceptionResnetV2	96	99	99	97	56,107,368
EfficientNetB3	98	99	99	99	12,623,287
MobileNetV3Large	84	95	90	85	4,391,432
DenseNet201	97	98	98	97	18,638,024
Xception	98	99	98	98	21,193,904

Here, A=Accuracy, P=Precision, R=Recall, F=F1-score

TABLE XVII: Performance Comparison of Different Pre-Trained CNN Based Models Combination For Ensemble Approach.

Models	A(%)	P(%)	R(%)	F(%)	Parameters
Ensemble1 (InceptionResnet V2+ EfficientNet B3)	99	100	99	99	68,730,655
Ensemble2 (MobileNetV3 Large+DenseNet 201 +Xception)	99	100	99	99	44,223,360

Here, A=Accuracy, P=Precision, R=Recall, F=F1-score

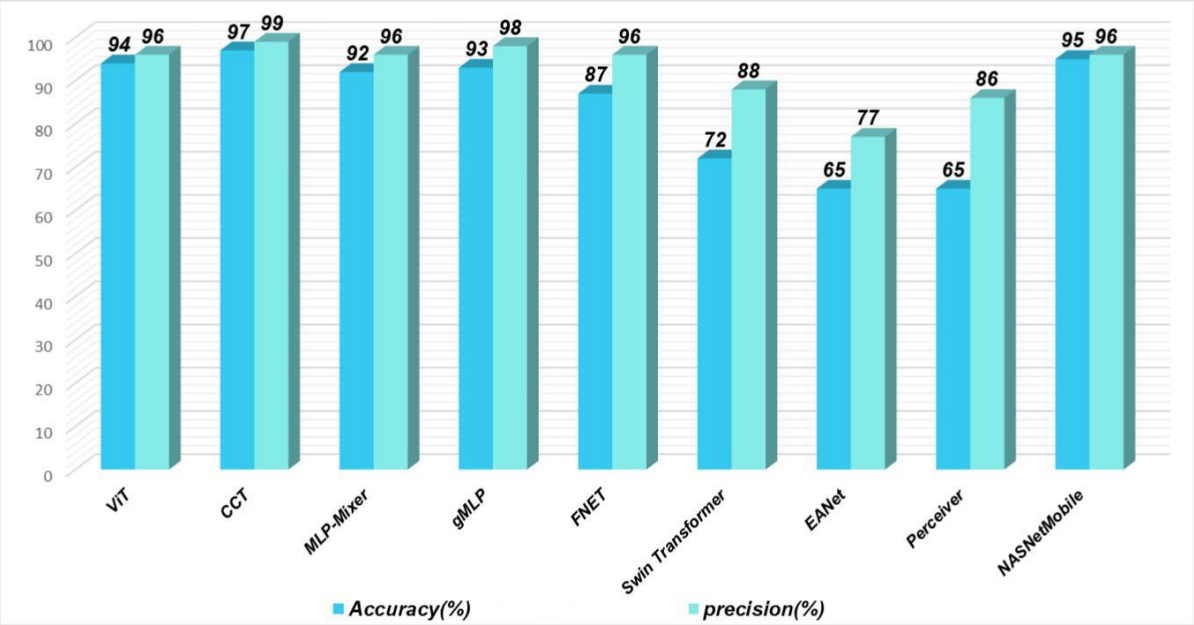


Figure 28: Result Analysis on Individual Models

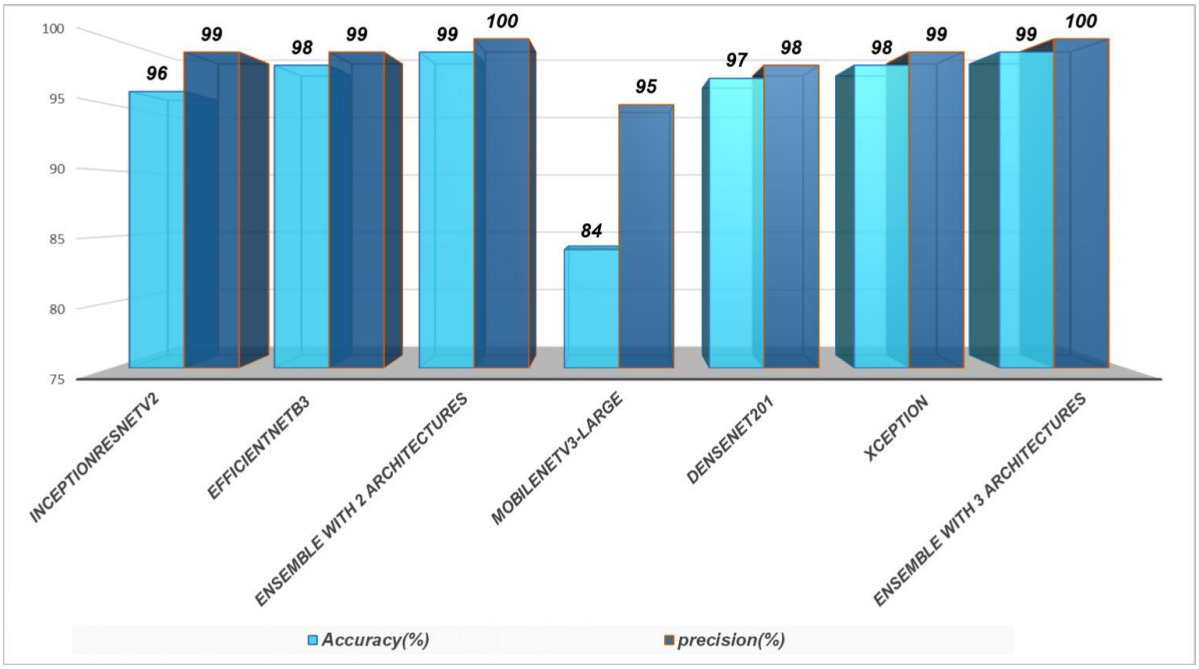


Figure 29: Result Analysis of Ensemble Models

Raw vs Pre-processed Data Sets Comparison Study :

To highlight the value of pre-processing our datasets, we developed a comparison table. The stark contrast between the before and after pre-processing in TABLE XVIII and Fig.23 demonstrates the significance of the improvement for better performance. Here, we presented and trained every model with our raw datasets, where no data pre-processing took place. There was 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 pre-processing of the dataset increased accuracy and precision by 20% to 40% across all matrices.

TABLE XVIII: Comparison of Raw Vs Pre-processed Datasets

Models	RA(%)	RP(%)	RR(%)	PA(%)	PP(%)	PR(%)
ViT	53	46	56	94	96	100
CCT	48	78	21	97	99	98
MLP-Mixer	54	60	41	92	96	98
gMLP	36	33	23	93	98	99
Fnet	57	56	55	87	96	88
Swin Transformer	56	67	70	72	88	93
Eanet	36	50	15	65	77	82

Perceiver	31	32	55	65	86	90
NasNet mobile	68	65	65	95	96	97
Inception ResnetV2	71	73	69	96	99	99
Efficient NetB3	84	76	81	98	99	99
Ensemble1 (Inception ResnetV2+ EfficientNetB3)	59	73	50	99	100	99
MobileNet V3Large	68	74	65	84	95	90
DenseNet 201	77	73	72	97	98	98
Xception	62	59	56	98	99	98
Ensemble2 (Mobile NetV3Large +DenseNet 201+Xception)	70	73	70	99	100	99

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(%)

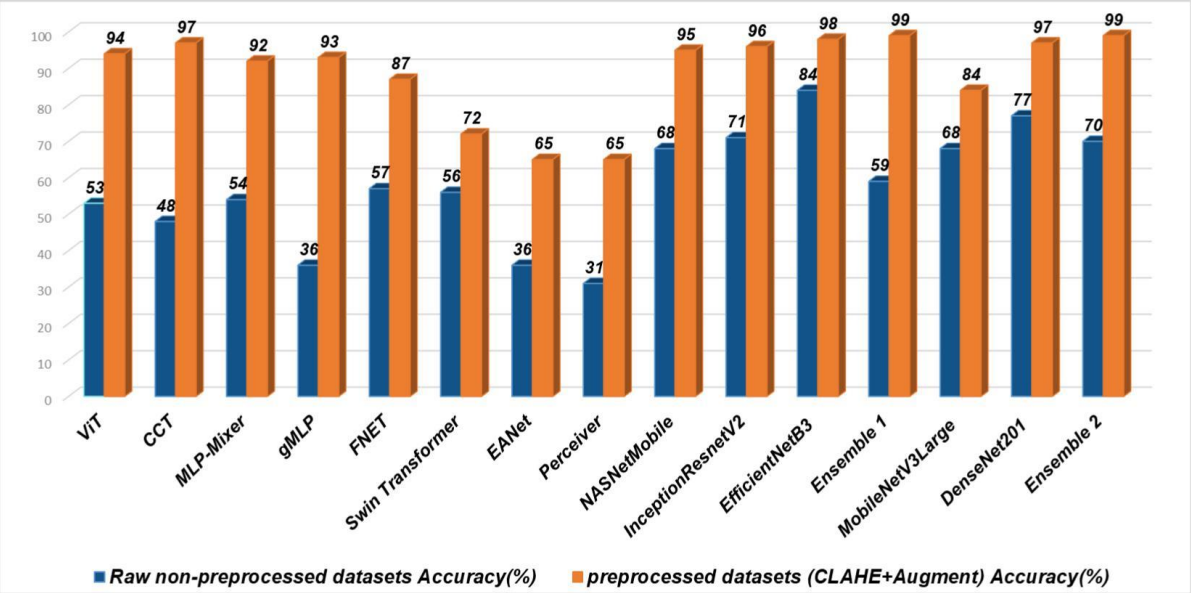


Figure 30: Result Analysis of Raw vs Pre-processed Datasets

Ensemble Model Analysis

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.

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. Image gathering and deep learning-based MPOX diagnosis of illnesses using modified VGG-16 [19]: Explainable AI-Assisted Convolutional Neural Network for labelling the Photos of Monkeypox Skin Lesion [20]; MPOX Virus Detection Using pre-trained deep learning-based techniques [21]. In the paper titled "Image data collection and implementation of a deep learning-based model for detecting monkeypox disease using modified vgg16 [19]", 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 lesions on the images. In the paper titled "Classification of Monkeypox Skin Lesion Using Explainable Artificial Intelligence-Assisted Convolutional Neural Networks" [20], the authors used the same MSID dataset as we did. However, they used the initial version of the MSID dataset, which included 572 images and two classes. The images were resized to 224 by 224 pixels without any augmentation or pre-processing of the dataset. This publication was released in September 2022. In the experimental analysis, twelve image classification networks implementing transfer learning were utilized. 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 only chose CNN-based networks. In addition, a Pytorch framework and XAI were used to generate a heatmap around the lesions on the images. In the article titled "Monkeypox Virus Detection Using Pre-Trained Deep Learning-Based Approaches" [21], a MPOX image dataset [22] was utilized. This article was released in October 2022. They proposed using a common architecture for all 13 pretrained DL models (MobileNetV2, VGG-19, ResNet101, ResNet-50, IncepResNetv2, InceptionV3, VGG-16, Xception, EffcientNet-B0, EfficientNet-B1, EffcientNet-B2, DenseNet-121, and DenseNet-169) for MPOX detection and classification. Then, they combined the best-performing models to enhance performance overall. However, their accuracy was 87.13%. In each of these publications, the deployed DL models were very outdated. To address this issue,

we selected 14 transformer-based models and recent CNN-based pre-trained models for the experiment to determine which model is superior for classifying MPOX images on an individual basis. Each model's performance was evaluated using precision, recall, F1-score, and accuracy. Finally, we combined the finest pre-trained CNN-based models to enhance the overall performance. GradCAM was used to generate heatmaps surrounding lesions on the images. We made every effort to select the optimal hyperparameters. Our total accuracy was 99 percent. Our proposed method is more precise; consequently, health professionals can rely on it for mass screening shown in TABLE XIX.

TABLE XIX: Comparison Study with The State-of-the-Art Methods

Methods	**PA(%)**	**OA(%)**	**Paper Limitations**	**Our Contributions**
Modified VGG16, 2022, [65]	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
Explain able Artificial Intelligence Assisted CNN, 2022, [66]	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: Densenet201 & EfficientNetB3
Pretrained DL models, 2022 [67]	87.13% ensemble	99% for both ensemble	Used pre-trained DL models: VGG-16, VGG19, 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, MLPMixer, gMLP, FNET, Swin Transformer, EANet, Perceiver, InceptionResnetV2, EfficientNetB3, Mobile NetV3Large, DenseNet201, Xception

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

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. 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 heat map enables us to visually confirm the area of the image on which our model is focused. 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 between none and one for display reasons. For superimposed rendering, we used another function that saves and shows the GradCAM. Image path, heatmap, cam path, and alpha = 0.4 were added as arguments to this function. One notable advantage of directed Grad-CAM over alternatives is its capacity to more effectively investigate and explain categorization mistakes. However, Grad-CAM fails to correctly locate objects in an image when there are several instances of the same class. To discover which portions of the supplied, picture have the greatest influence on the ability to recognize ratings, the color gradients of the classification value compared to the ultimate layered map of features are calculated.

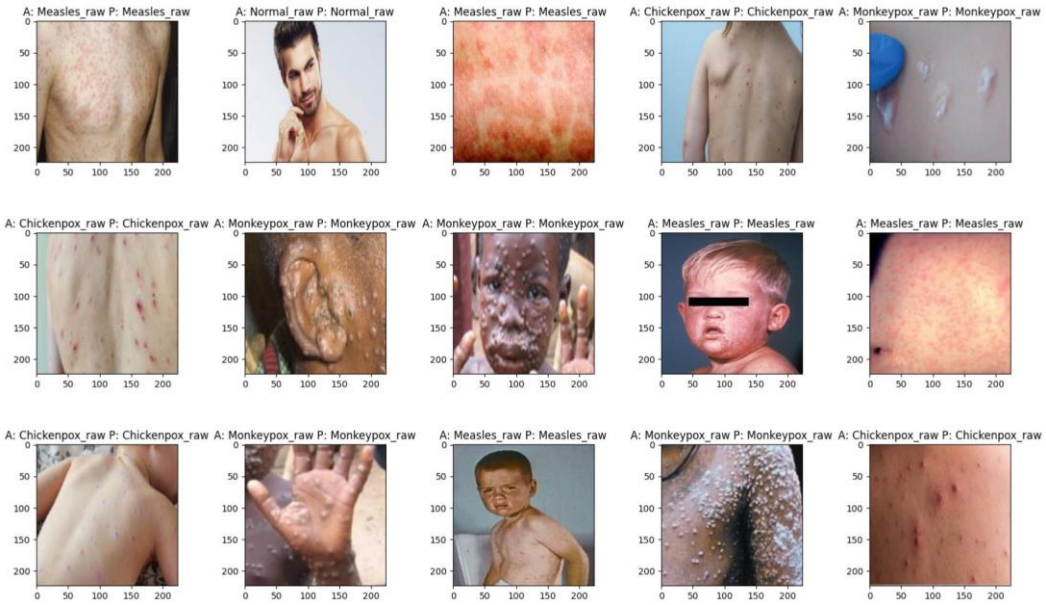


Figure 31: Correct Grad-Cam Prediction Representations on Our Dataset

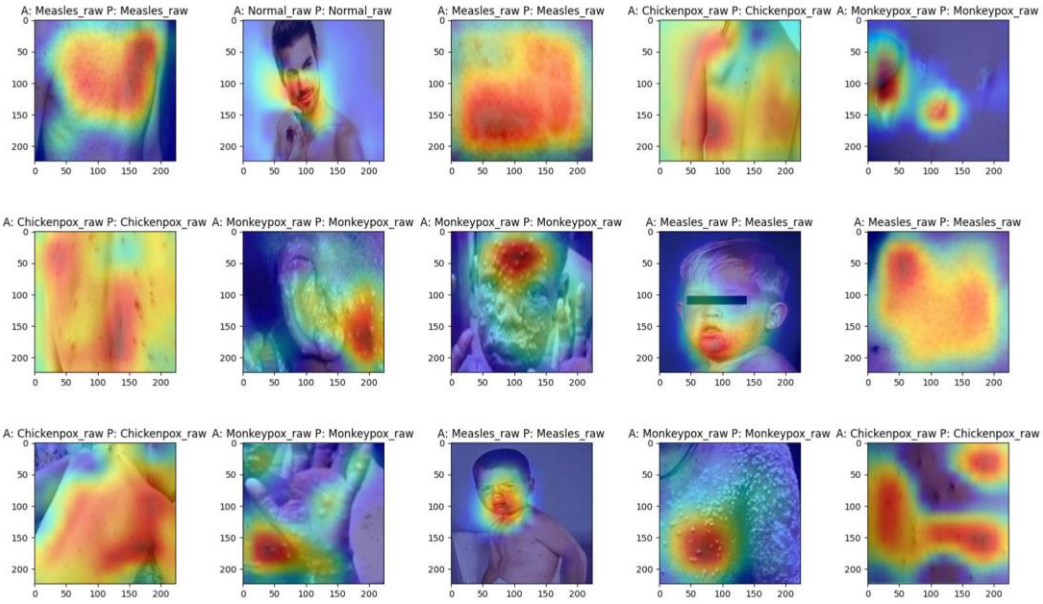


Figure 32: Grad-Cam's Superimposed Visualization Explanation

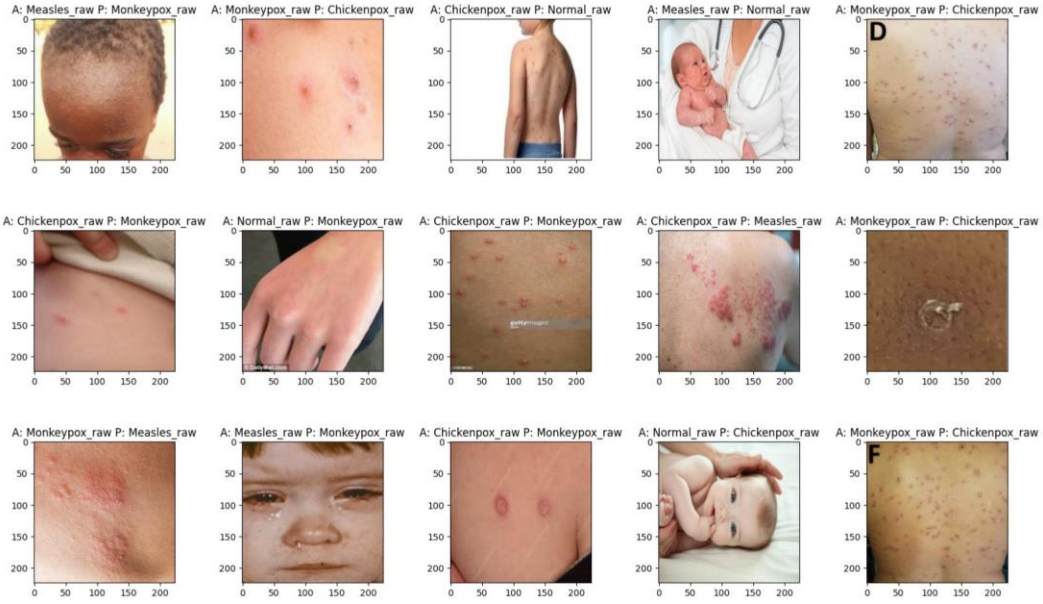


Figure 33: Wrong Grad-Cam Prediction Representations on Our Dataset

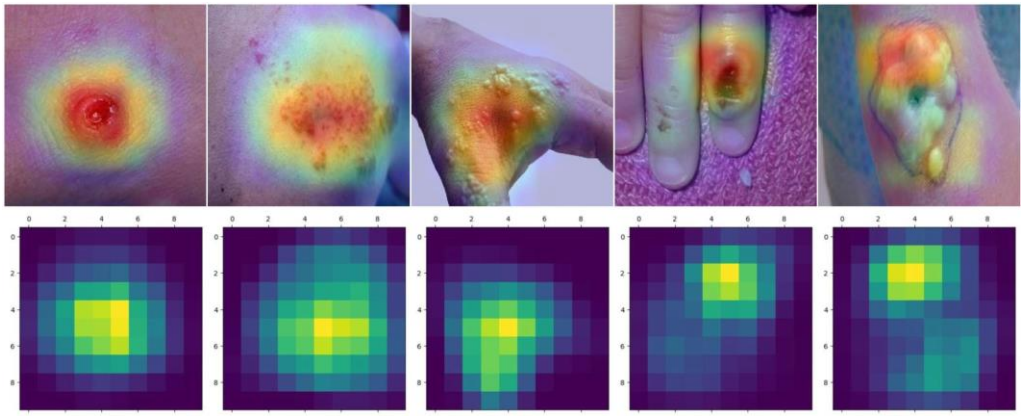


Figure 34: Grad-Cam's superimposed visualization explanation on our implemented datasets and their heatmap visualization

Analyzing Vision Transformer (ViT)

Our primary goal with our datasets is to shed light on the mechanisms that allow ViTs to derive knowledge from image data. The example investigates in depth the implementations of numerous distinct ViT analysis tools. The earliest ViT models require input images to be scaled to [-1, 1]. We must normalize images using the mean and standard deviation of ImageNet1k's channels for the other model families. We selected three images from the three classes of MPOX (monkeypox), measles, and chickenpox shown in Fig.28 and displayed them in the code cell. The model was then loaded. This model was initially trained on ImageNet-21k and then refined using ImageNet1k. Then, compute the "Mean Attention Distance" from each attention head of different transformer blocks to determine how local and global information is sent into vision transformers. The mean attention distance is the duration between query tokens and other tokens' attention weights. For each image, we calculate the geometric distance between image regions (tokens) and multiply it by the attention ratings. This consisted of a total of 12 heads in Fig.29. The second technique was the attention rollout method. "Attention rollout" measures data flow using Transformer block self-attention layers. Using this method, original ViT authors examine acquired patterns. In the third method, attention heatmaps in Fig.30 were utilized. Visualizing attention maps over input visuals is a simple but useful position for understanding vision transformer representation. This establishes the model's focus. Transformer blocks feature several heads. Transformer block heads direct input data to certain subspaces. This allows each head to focus on its own distinct visual location. Each attention head map must be viewed simultaneously in order to fathom what each head sees. The fourth approach was visualizing the acquired projection filters in Fig.31. Transformer blocks establish the attention weights for key and query. Weights indicate the significance of the query key. In ViTs, both the key and the inquiry are extracted from the same image; consequently, the weights determine which segment is crucial. ViTs flatten and linearly project non-overlapping patches. We present ViT model projections below. Visualizing the positional embeddings in Fig.32 was used in the fifth method. Transformers are resistant to permutation. This signifies disregard for the spatial position of the input token. Positional information on input tokens allows us to get around this limitation. Positional data can be acquired or handcrafted. Our three unique ViTs have acquired positional embeddings. Throughout this part, we will compare the learned positional embeddings to themselves. The model's dot product demonstrates positional embedding similarity.

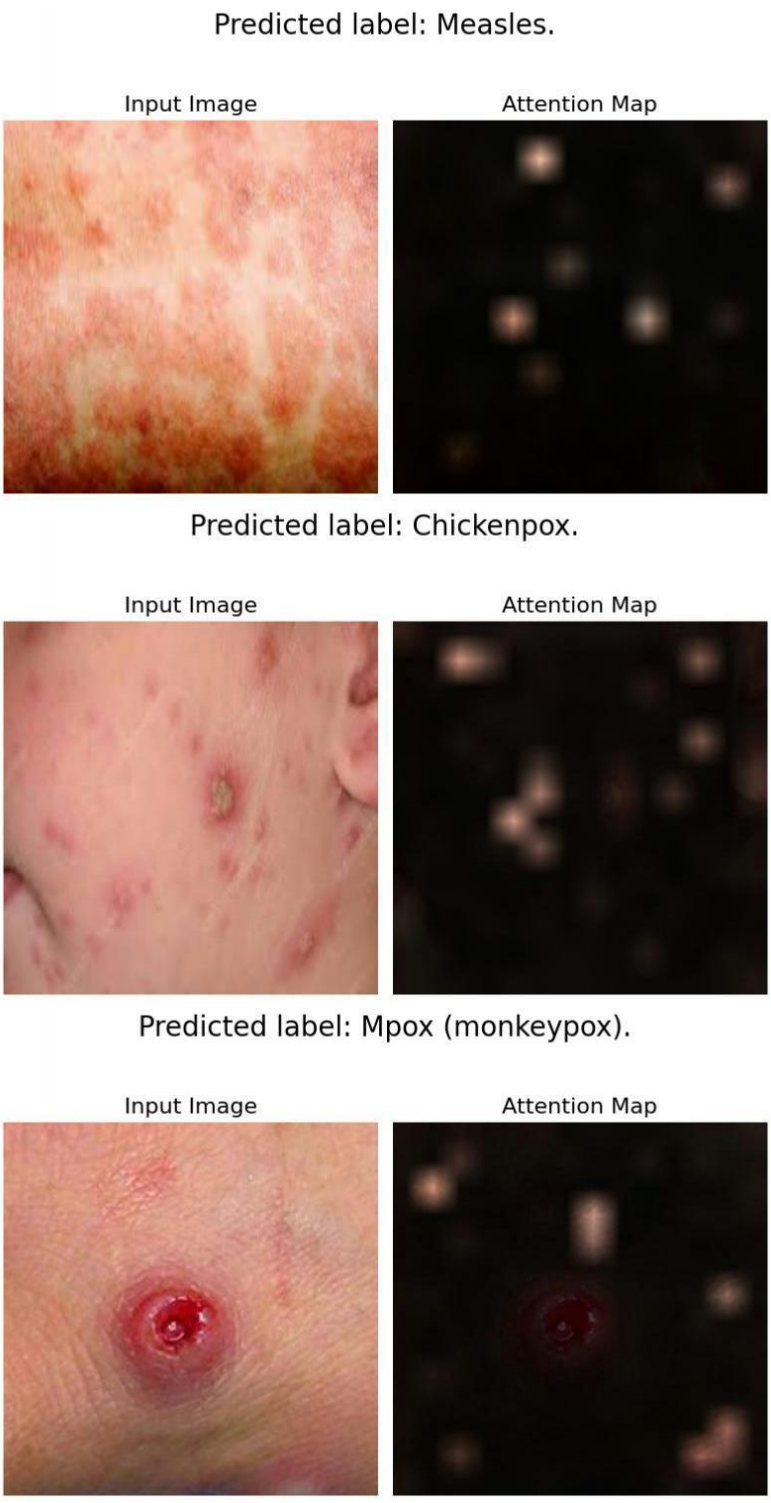


Figure 35: Input Images and its Attention Map

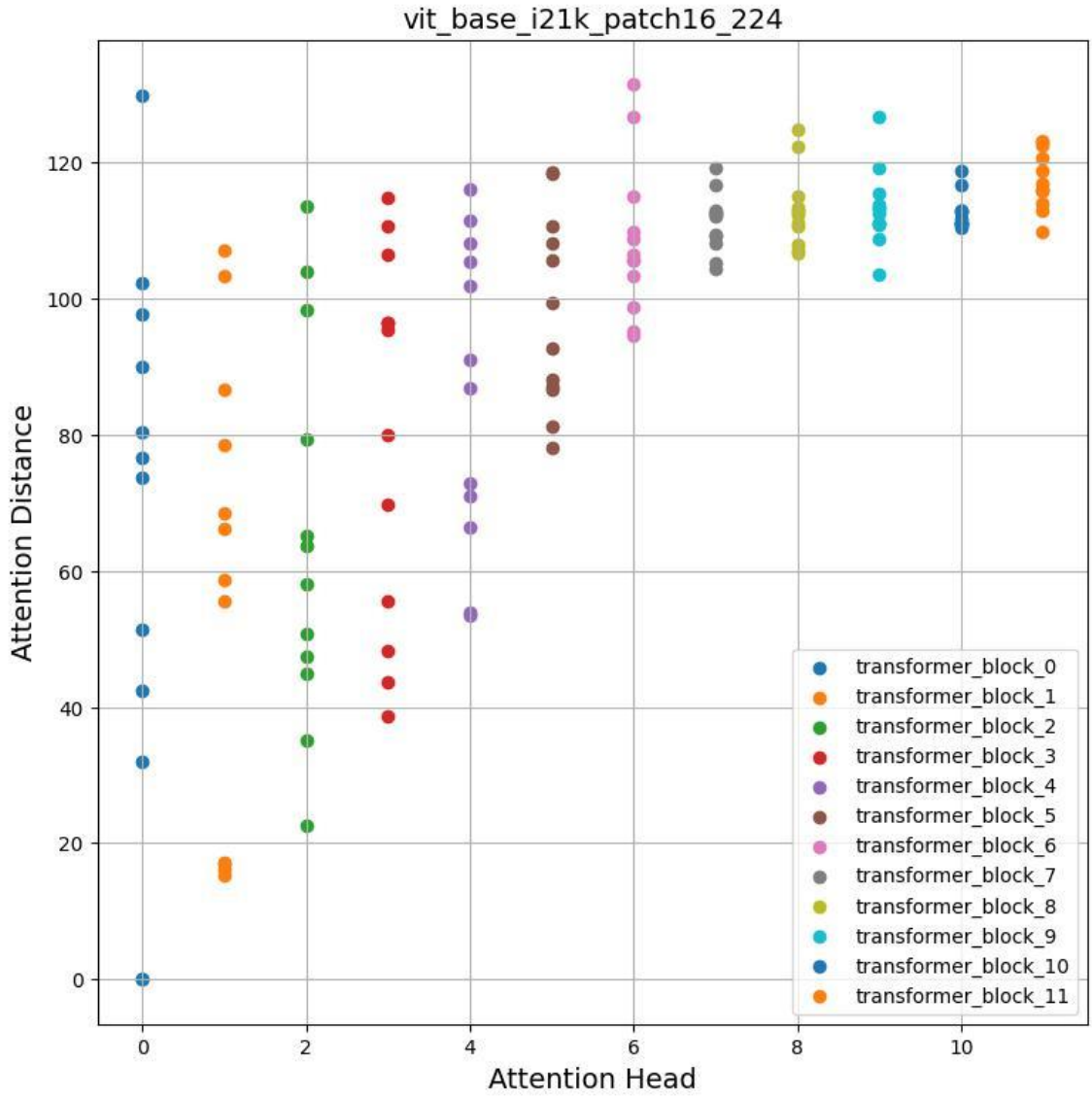


Figure 36: Attention Head of 12 items

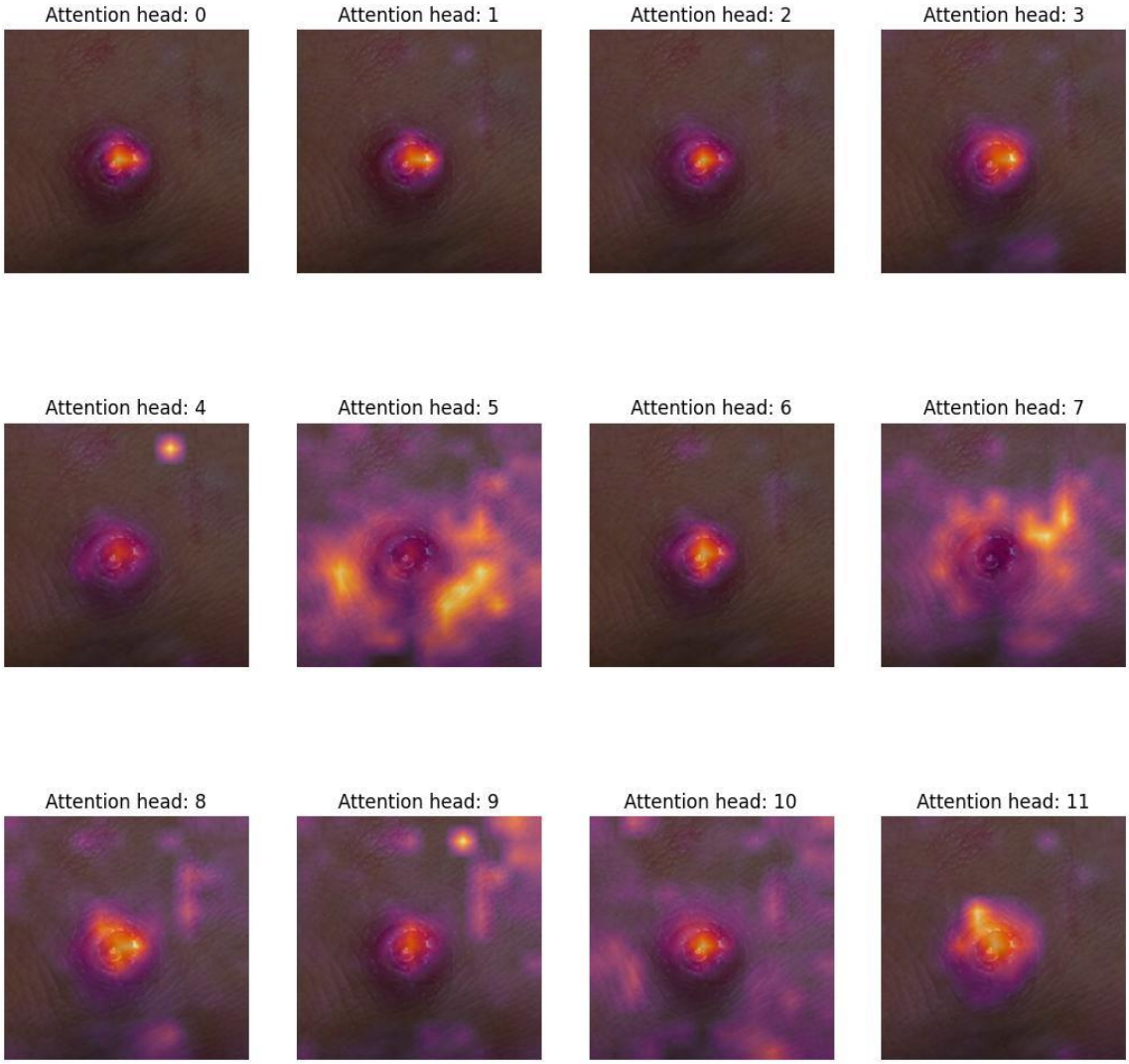


Figure 37: Attention Heatmaps of MPOX

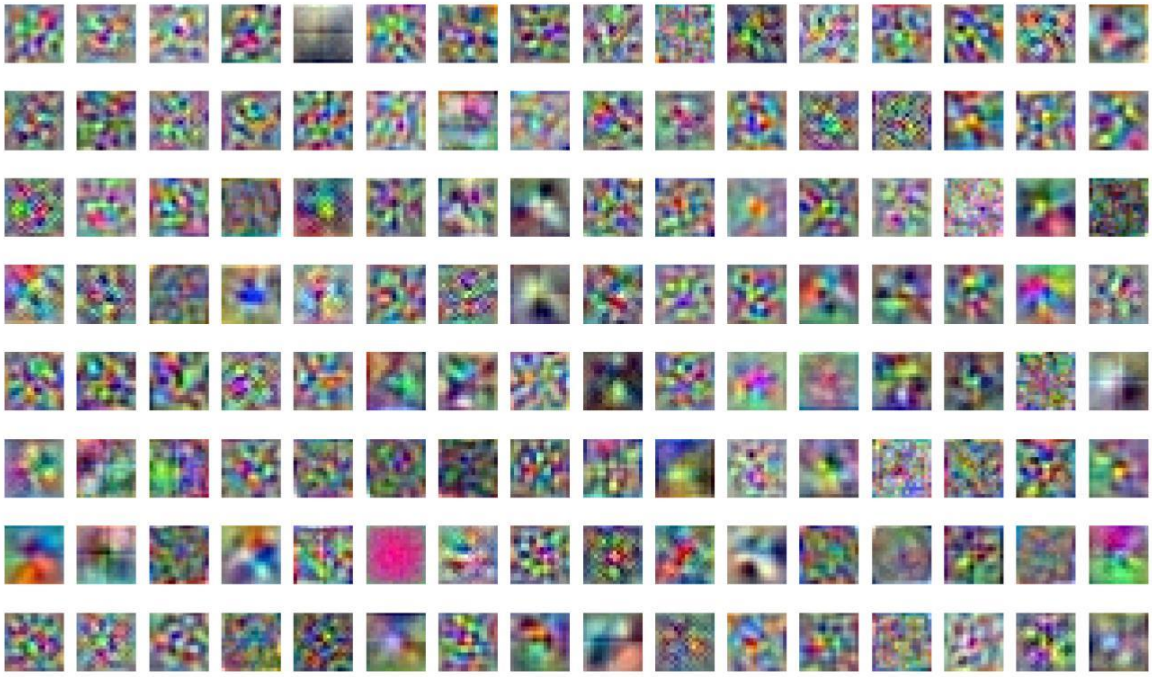


Figure 38: Visualization of The Learned Projection Filters

Class-wise study for Confusion Matrix

Using the confusion matrix in Fig.33 to Fig.48, 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.

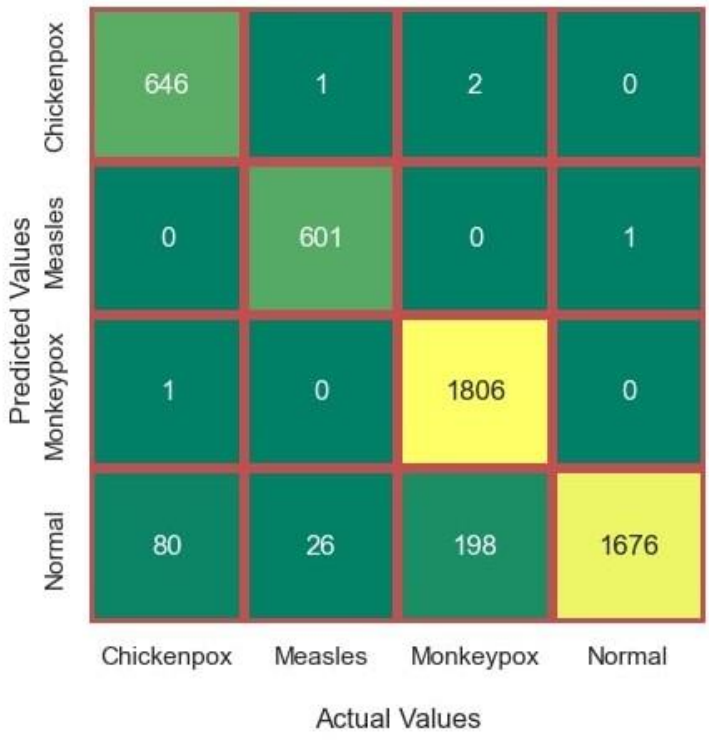


Figure 39: Confusion Matrix of ViT

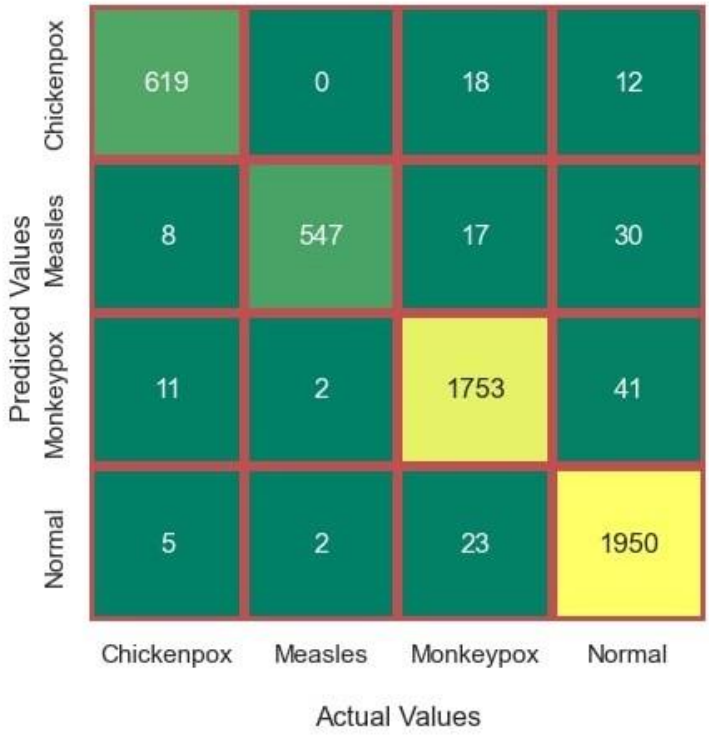


Figure 40: Confusion Matrix of CCT

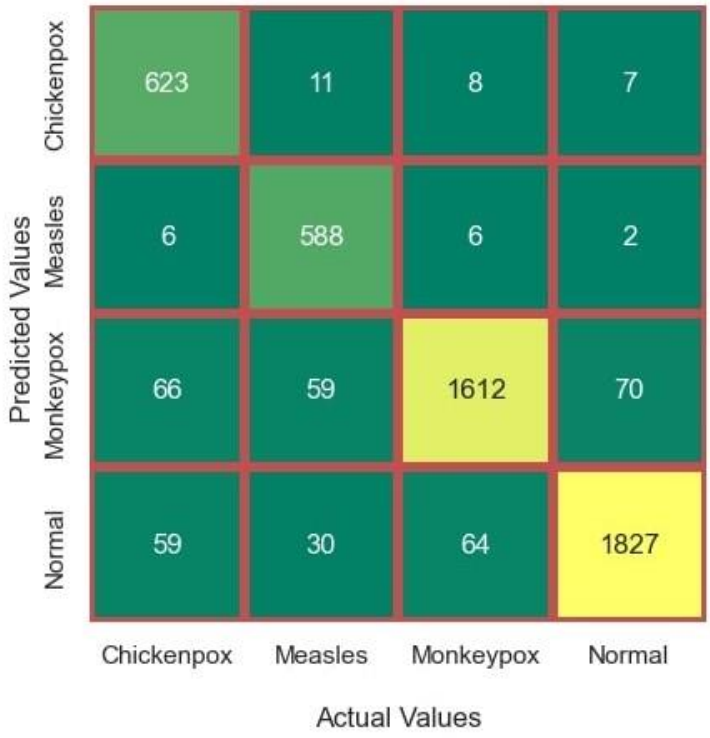


Figure 41: Confusion Matrix of MLP-Mixer

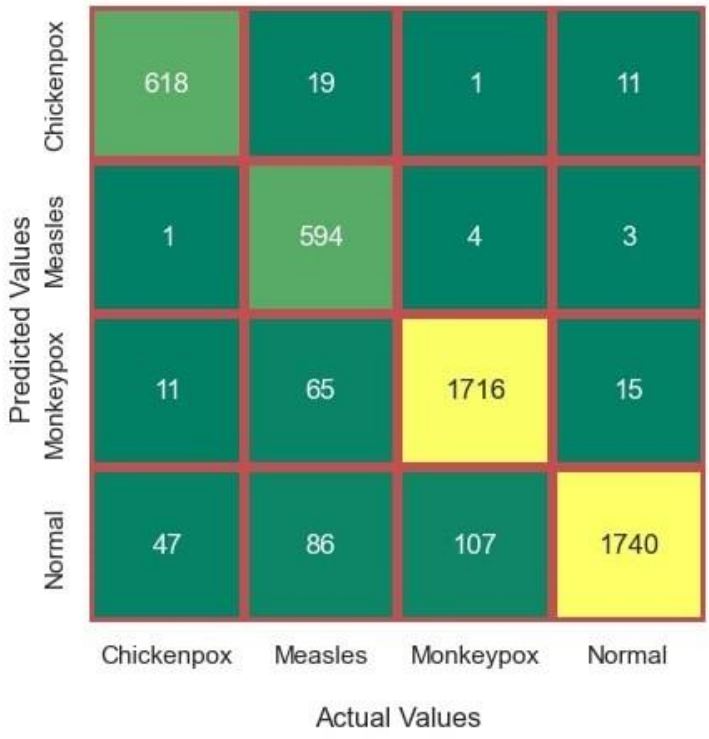


Figure 42; Confusion Matrix of gMLP

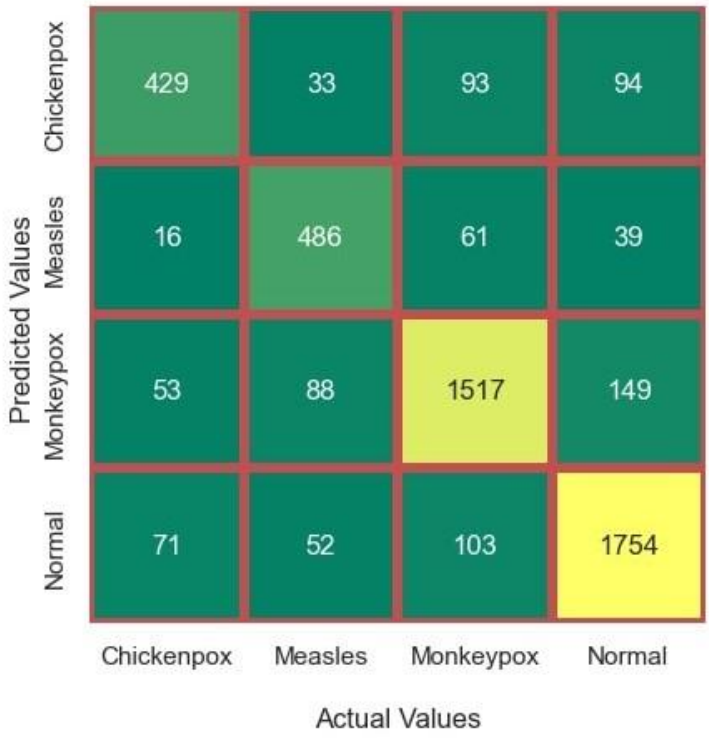


Figure 43: Confusion Matrix of FNET

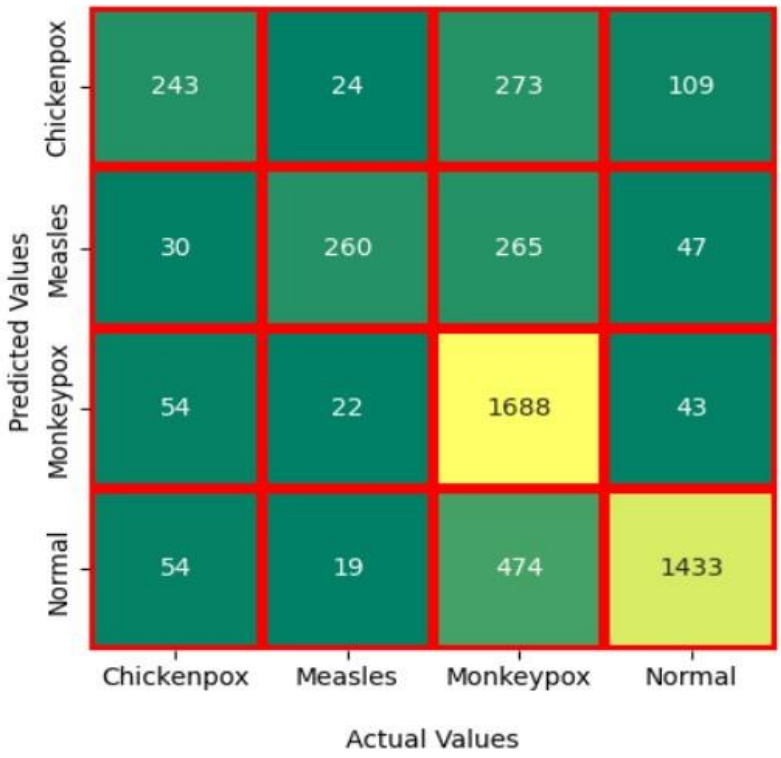


Figure 44: Confusion Matrix of Swin Transformer

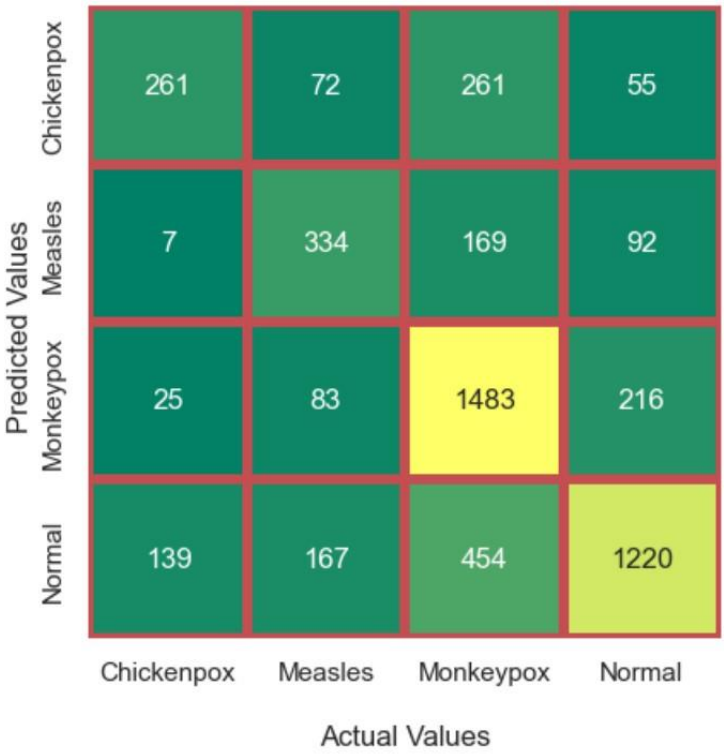


Figure 45: Confusion Matrix of EANet

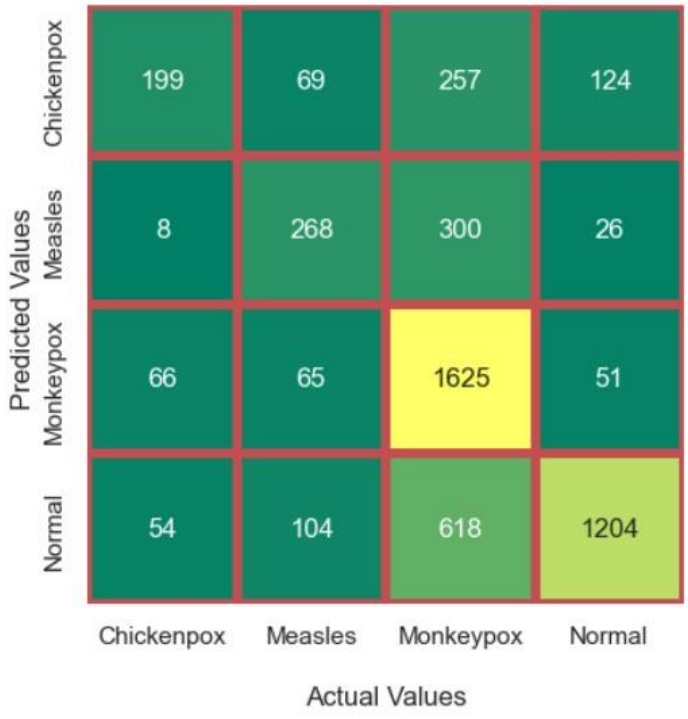


Figure 46:Confusion Matrix of Perceiver

Figure 47: Confusion Matrix of NasNet

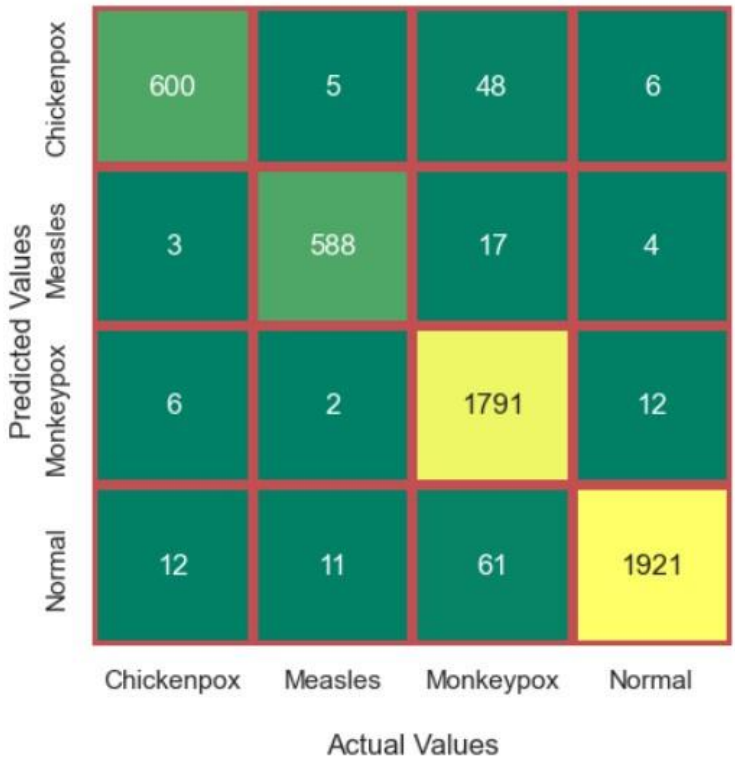


Figure 48: Confusion Matrix of InceptionResNetV2

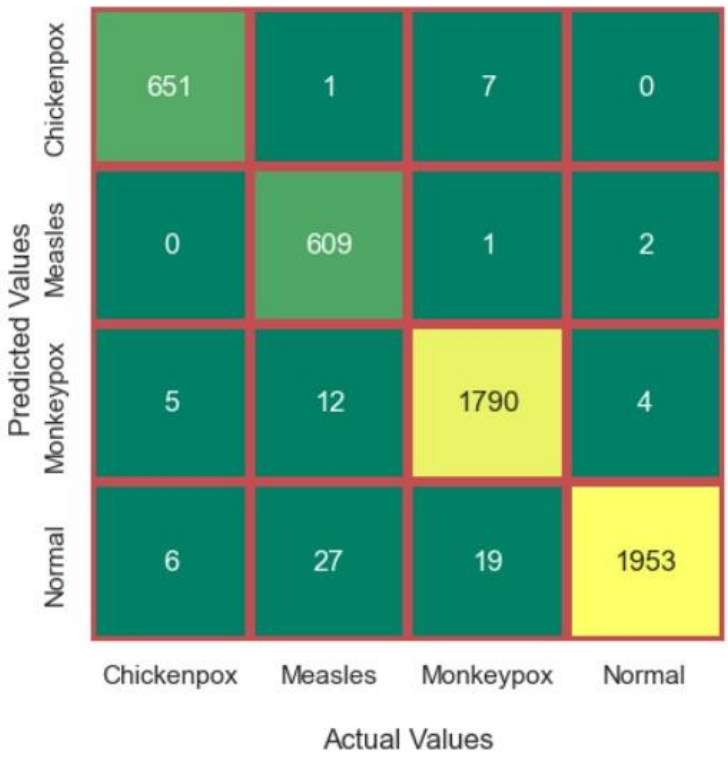


Figure 49: Confusion Matrix of EfficientNetB3

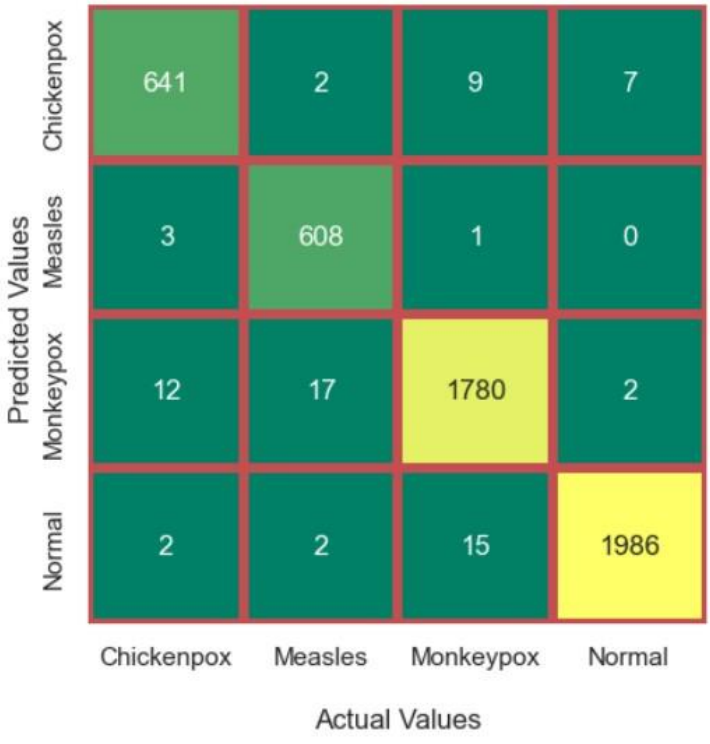


Figure 50:Confusion Matrix of Ensemble 1 (InceptionResNetV2 + EfficientNetb3)

Figure 51: Confusion Matrix of MobileNetV3Large

Figure 52: Confusion Matrix of DenseNet201

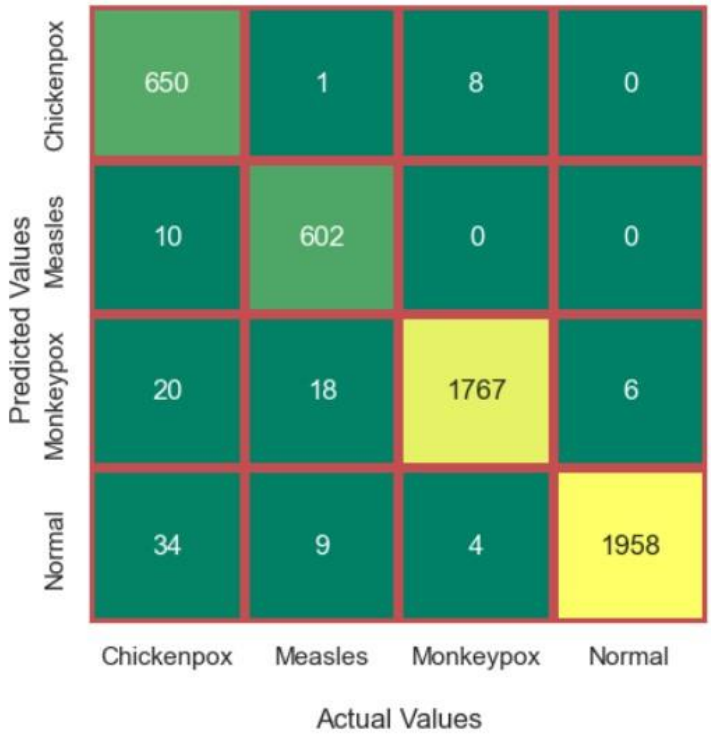


Figure 53: Confusion Matrix of Xception

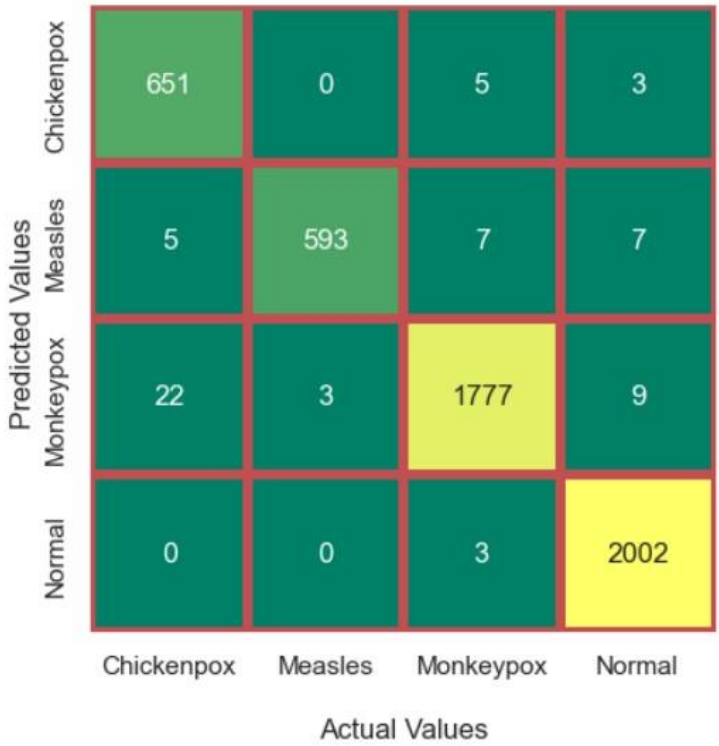


Figure 54: Confusion Matrix of Ensemble 2 (MobileNetV3Large + DenseNet201 + Xception)

Conclusions

The objective of the research we conducted was to aid in preventing another pandemic comparable to COVID-19. 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. 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. Our goal was to make sure early identification of MPOX is possible so waiting time in hospital decreases rapidly.

We leveraged 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 transformers, while the remaining five are CNN-based models that have been pre-trained. In our paper, we used transformer-based models to detect MPOX for the first time. The accuracy was 97%. Again, we utilized pre-trained models and achieved a higher accuracy than ever before. We ultimately utilized 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.

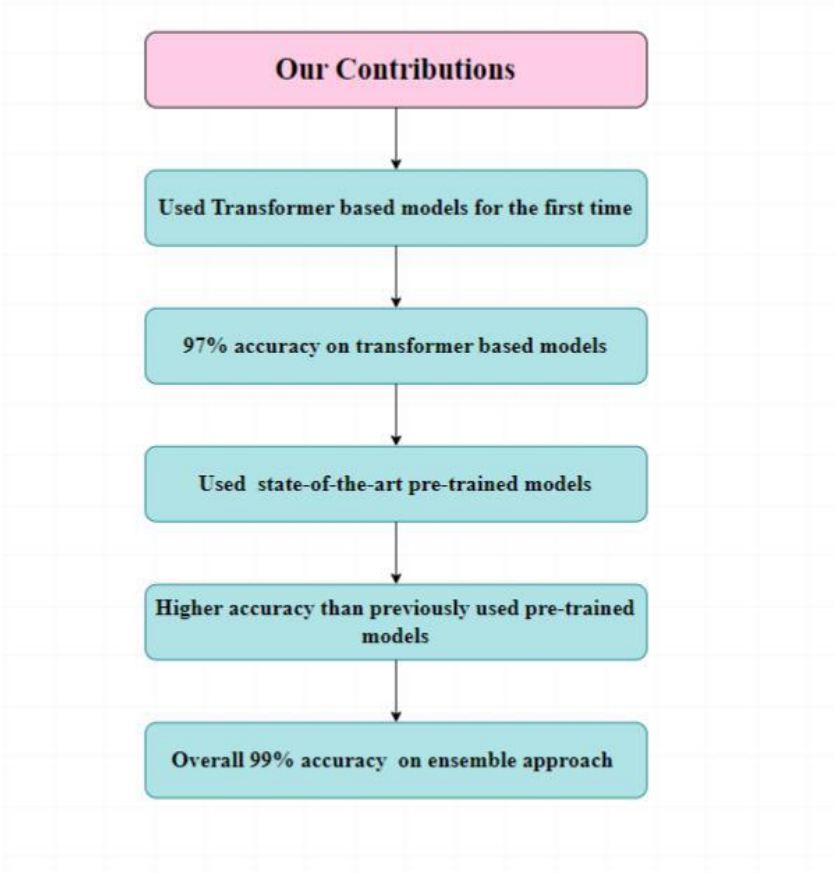


Figure 55: Research Contribution

Data Availability

The image data used to support the findings of this study are from previously reported studies and datasets, which have been cited and included within the article.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

ORCID Id's for Emergency Communication:
Ibne Hassan (0009-0008-9438-6587), Raida Mobashshira Tahsin (0009-0000-6907-1163), Sadia Shara Toma (0009-0008-0484-870X), Tausif Tazwar Quadria Utchhash (0009-0006-5150-5158), Dr. Muhammad Iqbal Hossain (0000-0002-0915-9291).

References:

[1] McCollum AM and Damon IK (Clin Infect Dis 2013; 58:260-7). (2014). *Clinical Infectious Diseases*, *58*(12), 1792. https://doi.org/10.1093/cid/ciu196

[2] Alakunle, E., Moens, U., Nchinda, G., & Okeke, M. I. (2020). Monkeypox Virus in Nigeria: Infection Biology, Epidemiology, and Evolution. *Viruses*, *12*(11), 1257. https://doi.org/10.3390/v12111257

[3] *Human monkeypox, 1970-79*. (1980). PubMed. https://pubmed.ncbi.nlm.nih.gov/6249508/

[4] Nolen, L. D., Osadebe, L., Katomba, J., Likofata, J., Mukadi, D., Monroe, B., Doty, J. B., Hughes, C. A., Kabamba, J., Malekani, J., Bomponda, P. L., Lokota, J. I., Balilo, M. P., Likafi, T., Lushima, R. S., Ilunga, B. K., N'Kawa, F., Pukuta, E., Karhemere, S., . . . Reynolds, M. G. (2016). Extended Human-to-Human Transmission during a Monkeypox Outbreak in the Democratic Republic of the Congo. *Emerging Infectious Diseases*, *22*(6), 1014–1021. https://doi.org/10.3201/eid2206.150579

[5] Kumar, N., Acharya, A., Gendelman, H. E., & Byrareddy, S. N. (2022). The 2022 outbreak and the pathobiology of the monkeypox virus. *Journal of Autoimmunity*, *131*, 102855. https://doi.org/10.1016/j.jaut.2022.102855

[6] Ali, S. N. (2022, July 6). *Monkeypox Skin Lesion Detection Using Deep Learning Models: A Feasibility Study*. arXiv.org. https://arxiv.org/abs/2207.03342

[7] Nguyen, P., Ajisegiri, W. S., Costantino, V., Chughtai, A. A., & MacIntyre, C. R. (2021). Reemergence of Human Monkeypox and Declining Population Immunity in the Context of Urbanization, Nigeria, 2017–2020. *Emerging Infectious Diseases*, *27*(4). https://doi.org/10.3201/eid2704.203569

[8] El-Kenawy, E. M., Albalawi, F., Ward, S. A., Ghoneim, S. S. M., Eid, M. M., Abdelhamid, A. A., Bailek, N., & Ibrahim, A. (2022). Feature Selection and Classification of Transformer Faults Based on Novel Meta-Heuristic Algorithm. *Mathematics*, *10*(17), 3144. https://doi.org/10.3390/math10173144

[9] Salamai, A. A., El-Kenawy, E. M., & Abdelhameed, I. (2021). Dynamic Voting Classifier for Risk Identification in Supply Chain 4.0. *Computers, Materials & Continua*, *69*(3), 3749–3766. https://doi.org/10.32604/cmc.2021.018179

[10] *Mpox (monkeypox) outbreak 2022 - Global* (2022, July 23). https://www.who.int/emergencies/situations/monkeypox-oubreak-2022

[11] *Addressing the physicians' shortage in developing countries by accelerating and reforming the medical education: Is it possible?* (2017, October 1). PubMed. https://pubmed.ncbi.nlm.nih.gov/28979916/

[12] Ali, S. N. (2022b, July 6). *Monkeypox Skin Lesion Detection Using Deep Learning Models: A Feasibility Study*. arXiv.org. https://arxiv.org/abs/2207.03342v1

[13] Akın, K. D. , Gurkan, C. , Budak, A. & Karataş, H. (2022). Classification of Monkeypox Skin Lesion using the Explainable Artificial Intelligence Assisted Convolutional Neural Networks . Avrupa Bilim ve Teknoloji Dergisi , ICIAS 2022 , 106-110 . DOI: 10.31590/ejosat.1171816

[14] Abdelhamid, A. A., El-Kenawy, E. M., Khodadadi, N., Mirjalili, S., Khafga, D., Alharbi, A., Ibrahim, A., Eid, M. M., & Saber, M. (2022). Classification of Monkeypox Images Based on Transfer Learning and the Al-Biruni Earth Radius Optimization Algorithm. *Mathematics*, *10*(19), 3614. https://doi.org/10.3390/math10193614

[15] CWI (Centre for Mathematics and Computer Science)P. O. Box 94079 NL-1090 GB Amsterdam Netherlands. (1995). *Python reference manual*.

[16] *Multi-classification Network for Detecting Skin Diseases using Deep Learning and XAI*. (2022, November 20). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9990755

[17] Sitaula, C., & Hossain, M. M. (2020). Attention-based VGG-16 model for COVID-19 chest X-ray image classification. *Applied Intelligence*, *51*(5), 2850–2863. https://doi.org/10.1007/s10489-020-02055-x

[18] Keras-Team. (n.d.). *GitHub - keras-team/keras: Deep Learning for humans*. GitHub. https://github.com/keras-team/keras

[19] Ahsan, M. M. (2022, June 4). *Image Data collection and implementation of deep learning-based model in detecting Monkeypox disease using modified VGG16*. arXiv.org. https://arxiv.org/abs/2206.01862

[20] Akin, K. D., Gurkan, C., Budak, A., & Karataş, H. (2022). fication of Monkeypox Skin Lesion using the Explainable Artificial Intelligence Assisted Convolutional Neural Networks. *European Journal of Science and Technology Special Issue*. https://doi.org/10.31590/ejosat.1171816

[21] Sitaula, C., & Shahi, T. B. (2022). Monkeypox Virus Detection Using Pre-trained Deep Learning-based Approaches. *Journal of Medical Systems*, *46*(11). https://doi.org/10.1007/s10916-022-01868-2

[22] Ahsan, M. M. (2022a, June 3). *Monkeypox Image Data collection*. arXiv.org. https://arxiv.org/abs/2206.01774