International Journal of Formal Sciences: Current and Future Research Trends (IJFSCFRT)
ISSN (Print) , ISSN (Online)
© International Scientific Research and Researchers *Association*
https://ijfscfrtjournal.isrra.org/index.php/Formal_Sciences_Journal/index

A Deep Learning Approach to Classifying MonkeyPox from Lesion Images

Alex W. Zhu[a]*, Aayush Goel [b]

[a]*Cupertino High School*

[b] *Cupertino High School*

[a]*Email: alexzhu1058@gmail.com*

[b]*Email: aayush.goel1@gmail.com*

Abstract

Following the outbreak of the COVID-19 virus, the Monkeypox virus poses a new threat of a potential global pandemic. Although the severity of the Monkeypox virus is not yet at the level of COVID-19, important measures are needed to prevent another major outbreak. Deep learning methods have proven efficient at classifying monkeypox lesions, which can help with preventing the spread of the virus as it can be a substitute for in-person testing. Our research implements the “Monkeypox Skin Lesion Dataset”, a collection of monkeypox, measles, and chickenpox images collected from websites, news portals, and publicly accessible case reports. The original images were augmented to increase the dataset size. Our deep learning approaches include training fine-tuned convolutional neural networks as well as a deep hybrid learning approach in which we extract features using a convolutional neural network. We then proceed by passing the features into a variety of machine learning classifiers including XGBoost, K-Nearest Neighbors, and Logistic Regression, yielding respective accuracies of 95.81%, 97.07%, and 97.28%.

Keywords: Convolutional Neural Network; Deep Hybrid Learning; K-Nearest Neighbors; Logistic Regression; Machine Learning; Monkeypox; Transfer Learning; XGBoost.

1. Introduction

The recent outbreak of the Monkeypox virus in 2022 in many nations has proposed another global health challenge as the world is still recovering from the COVID-19 pandemic. Monkeypox is a zoonotic disease that belongs to the Orthopoxvirus genus in the Poxviridae family which includes smallpox and cowpox[1]. The monkeypox virus was initially diagnosed in the Democratic Republic of Congo in 1970, and since has spread to other nations in Africa as well as outside Africa[2]. The virus is usually transmitted through monkeys and rodents. However, recently its contamination through human contact has also become prevalent[3]. It is spread through salivary or respiratory droplets as well as contact with lesion exudates[4, 5]. Some of the symptoms of the monkeypox virus are similar but less severe to those of smallpox. When infected with smallpox, patients develop a fever as well as rashes in the following days[6]. Swelling of the lymph nodes, known as Lymphadenopathy, occurs around the time when rashes start to appear[7]. Long term symptoms of Monkeypox include red-bumps on the skin[8] - one of the most common symptoms the images in the dataset implemented contains.

Within the past few decades, the growth of monkeypox has been exponential. Although its growth is not as sudden and explosive as the COVID-19 virus, which reached millions of cases within a few months, it is nonetheless significant. For instance, on May 31, 2022, the total number of cases in the U.S. was 31[9]. And as of August 19, 2022, the total number of cases has reached 14, 594 in the US[10]. Therefore, it is fair to say that this rapid growth is of concern to health officials as it has proven it can reach high numbers very quickly unless treated with proper caution.

As of now, the CDC(Centers of Disease Control and Prevention) states that there is no specific treatment for the monkeypox virus. However, antiviral drugs and vaccines developed against smallpox can be used to treat monkeypox[11]. Some of these antivirals include ST-246(TPOXX), brincidofovir(Tembexa), and cidofovir(Vistide)[12]. Additionally, some vaccines that are currently being used are JYNNEOS(Imvamune or Imvamax), NIAD, and the ACAM2000 smallpox vaccine[13].

In order to test for monkeypox, people have to meet a certain epidemiologic criteria which is the initial stage of the diagnosis. This criteria includes contact with people who have had a similar lesion, intimate contact with others experiencing monkeypox symptoms, or traveled to countries where monkeypox is endemic. If any of these criteria are met, the individual moves on to the next stage of diagnosis which is testing for monkeypox. To test, they can either swab samples from a lesion and send it to a laboratory or take an Orthopoxvirus PCR test[14].

Our research proposes another diagnosis of monkeypox, which includes using machine learning(ML) and deep learning(DL) image classification methods to classify images skin lesions as being infected with monkeypox or not. In the past decade, image classification methods have developed rapidly and have been implemented in the field of bio-medical imagery. In particular, Convolutional Neural Networks(CNNs) have the ability to identify and learn features or patterns otherwise not always recognizable by humans. Moreso, CNNs have demonstrated the ability to achieve highly accurate classifications of skin lesions correlated with cancer or other skin conditions, which will be a potential tool to assist dermatologists in diagnosis. Some advantages of the CNN include high accuracy with proper training, short execution time, usefulness in diagnosis, and its relatively cheap cost to train. Some of the drawbacks include the need for large and diverse training datasets, obscure decision making process(known as the "black box" in deep learning), and limited accuracy on under-represented conditions[15]. However, with the right data, the advantages outweigh the drawbacks. For instance, authors of [16] use InceptionV3 trained on ImageNet weights and transfer learning on a medical image dataset containing 108,312 optical coherence tomography (OCT) images with an average accuracy of 96.6%, sensitivity of 97.8%, and specificity of 97.4%. The authors in [17] used CapsNet to classify four types of breast tissue biopsies from breast cancer images with an accuracy of 87%. The authors of [18] used a combination of CNN and LSTM(long-short-term memory) for feature extraction and Softmax along with SVM layers for decision making using the extracted features for an accuracy of 91% on biomedical bread cancer images from the BreakHis dataset. We found two papers which used CNN architectures for monkeypox image classification. The authors of [19] used the VGG16 model for an accuracy of 97% with an AUC of 97.2 on their own created monkeypox dataset. The authors of [20] used the Monkeypox Skin Lesion Dataset for classification using the pretrained models VGG16, ResNet50, and InceptionV3 with respective accuracies of 81.48%, 82.96%, and 79.26%.

Before we began our study, we realized that not much research has been done on utilizing deep learning for monkeypox classification, and more so we could improve on the research already conducted[19, 20] to achieve higher accuracies. The remaining paper will discuss the following: Section 2 describes the methodology, followed by Section 3 with results, and Section 4 with discussion and conclusions.

2. Materials and Data

To develop our models, we implemented the Monkeypox Skin Lesion Dataset from Kaggle[20], a collection of monkeypox, measles, and chickenpox images collected from websites, news portals, and publicly accessible case reports. This dataset includes three folders, the first being the one with original images, the second with augmented images, and the third, Fold1, being the three-fold cross validation dataset. The original image dataset contains 228 images, where 102 of those images belong to the monkeypox class, and the remaining 126 belong to the "Others" class which include images of other non-monkeypox diseases such as measles and chicken. The third dataset, Fold1, uses three-fold cross validation to reduce bias in training, such as overfitting. Three-fold cross validation is where the dataset is split into three folds: two of those folds being used for training, and one of those folds being used for testing[21]. This way, the same data points are used for both training and testing which makes sure that no one particular datapoint was overfit. The last dataset, the one which we used, was the Augmented Image dataset. The images were split into 1428 monkeypox images, and 1764 "Other" images. Some of the augmentations include rotation, translation, reflection, shear, hue, saturation, contrast and brightness jitter, noise, scaling etc. which have been implemented using MATLAB R2020a. Figure 1, shown below, is an example of an original image. Figures 2, shown below as well, is a collection of augmented versions of the original image.

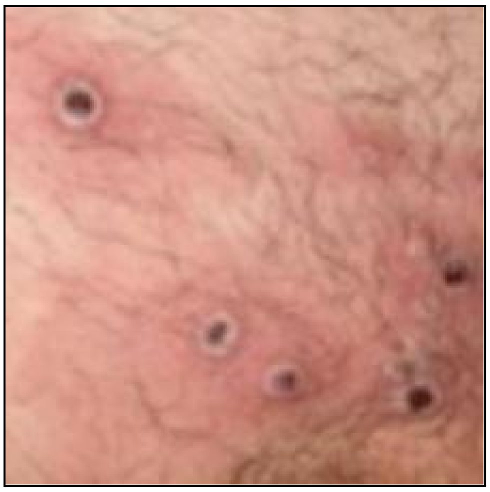

Figure 1: Original Image

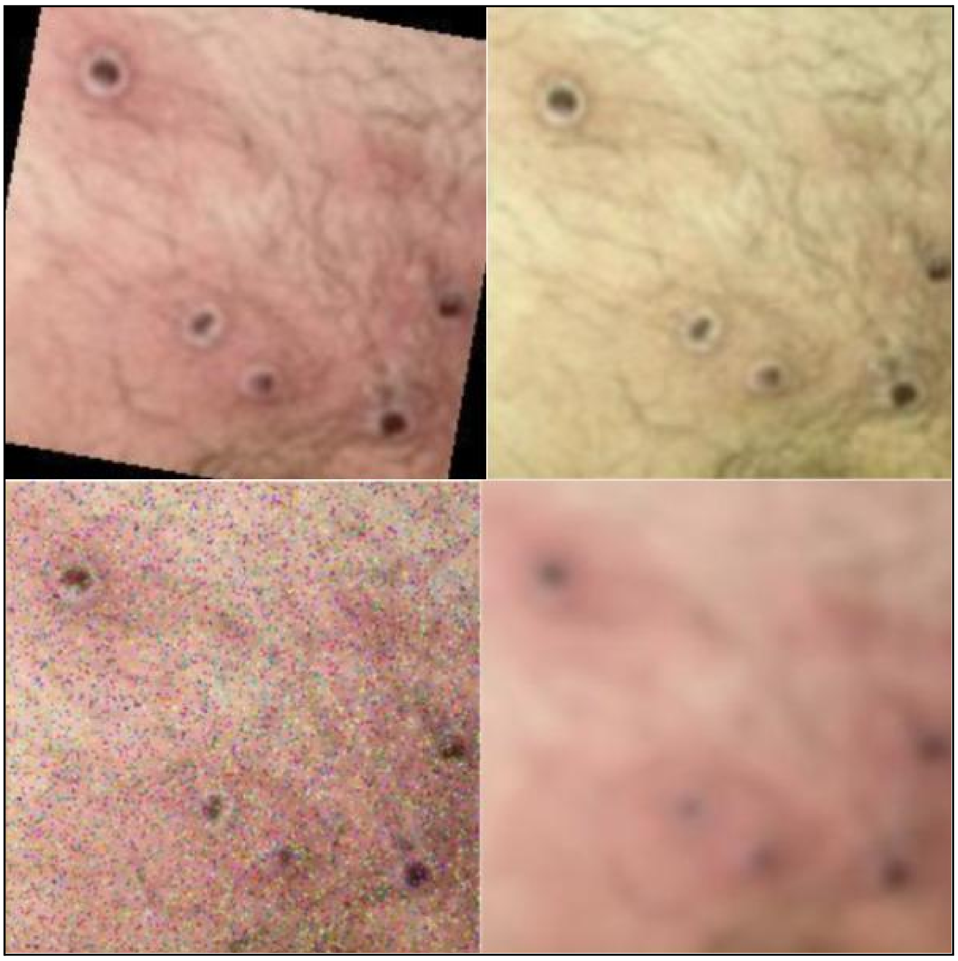

Figure 2: Collection of augmented versions of original image

Given that the image sizes for the original and three-fold set were relatively small to train large deep learning models such as CNNs, we utilized data augmentation to increase the image size. Although the dataset size is still relatively small(3192 images in total), we found that it is possible to develop models for diagnosis as demonstrated by researchers who created these models on the onset of COVID-19. For instance, in [22], the authors used three datasets of chest x-ray images, where each dataset was roughly the size of three thousand images to train CNN models including ResNet50, ResNet101, ResNet152, InceptionV3 and Inception-ResNetV2, which are similar to the models our research utilizes. We therefore assumed that it is possible to train large CNN models with limited data, as well as achieving high accuracy with the data.

2.1 CNN Architecture

A CNN, Convolutional Neural Network, is a deep learning model with a feed-forward architecture that is able to identify abstract features similar to how biological detectors do through mathematical calculations known as convolutions. Within the context of neural networks, and for the sake of simplicity, a convolution is in specific a linear operation is a multiplication between a set of weights, also known as a filter or kernel. This multiplication between the input and kernel is used to extract features from the image[23]. However, besides just performing convolutions, a CNN has much more to it. A CNN often consists of several components: an input layer, convolution layers, pooling layers, fully-connected layers, and an output layer.

The input layer takes the input of an image, and splits the image into generally a three dimensional matrix where the dimensions are the width of the image in pixels, the height in pixels, and three values for RGB. Then the convolution layers are where most of the feature extraction happens. Convolution layers use kernels to iterate over the input of the previous layer, and perform the mathematical operation above to return a new matrix, which is known as a feature map to be used by the next layer. The pooling layers are used to further reduce the size of the feature map of the previous layer, by applying a window on the feature map and returning only one value from that window into the new matrix. After the features are extracted, the fully-connected layers take in a single vector which is the summary of all the features to perform a classification, and then go to the output layer which gives a probability of the image being a certain class[24]. Figure 3, listed below, is an example of a CNN architecture.

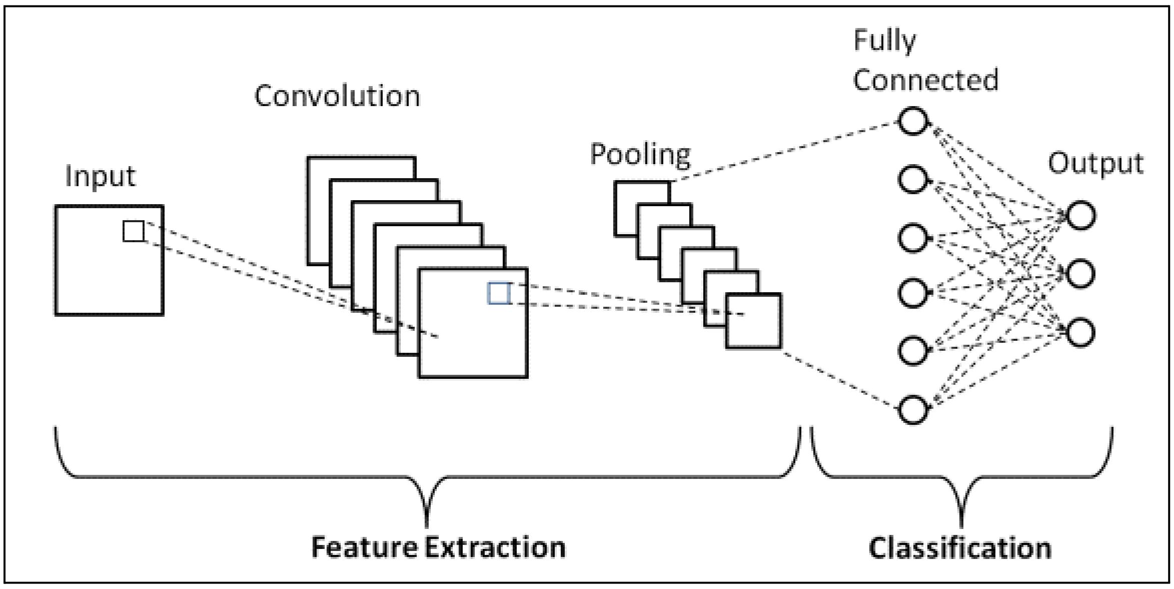

Figure 3: CNN Architecture Example

2.2 Resnet50

ResNet50, also known as a Deep Residual Network, is a type of CNN that adds more layers to activation and batch normalization where the layers are reformulated to learn residual functions. Originally, it was assumed that in order to increase accuracy, one should just add more layers as well as intermediate normalization layers which at the time was used to reduce overfitting by a bit. However, what ended up happening was after a certain number of added layers, the accuracy would decrease. This decrease in accuracy is not due to overfitting, but gradients being less effective. The deep residual network architecture addressed this problem by changing the mapping of the layers from being direct to residual. A residual mapping works like the following: say that in a traditional CNN, the input, x, goes through some weight layers, with some activation function a, and identity mapping F which is simply equal to a[25]. What residual mapping does is that it changes the identity mapping so that the original input is not changed too much by the time it reaches the deeper layers. The following identity mapping can be represented by equation(1).

$$F(x) = a(x)+x \qquad (1)$$

The advantage of this identity mapping is that it functions as a skip connection, where they add outputs from the previous layer to the output of the current layers. This ensures that any of the higher layers of the model will perform equal to or better than the previous layers which then allows ResNets to improve efficiency as well as accuracy[26]. Figure 4 below shows an example in-play of how the identity, or residual, mapping works where along with the ReLU activation, the identity, or the input x is used as the new activation compared to ReLU

activation without the identity mapping.

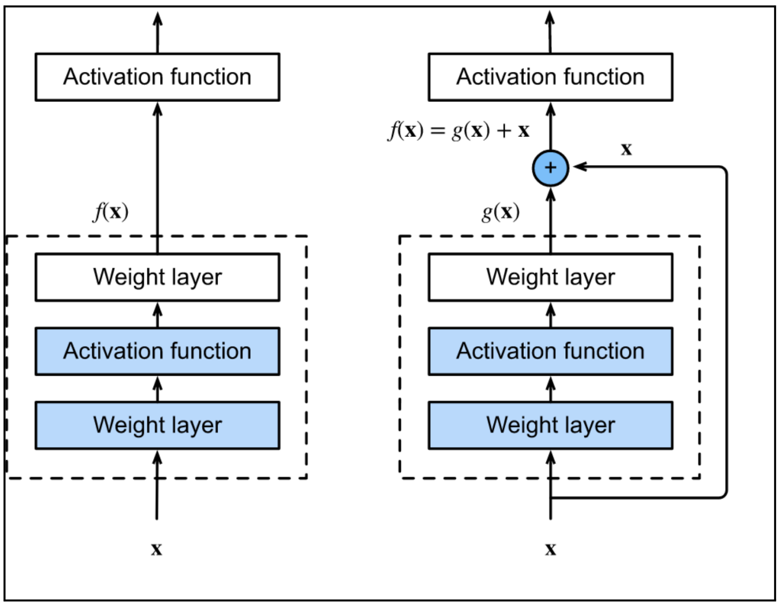

Figure 4: Resnet Architecture example

The specific model of residual networks, ResNets, we used was ResNet50. ResNet50 is unique compared to the other models where if there is a current layer L_i, its output will be used as input for layer L_{i+4}. Figure 5 shows a sample of the architecture for ResNet50[27]:

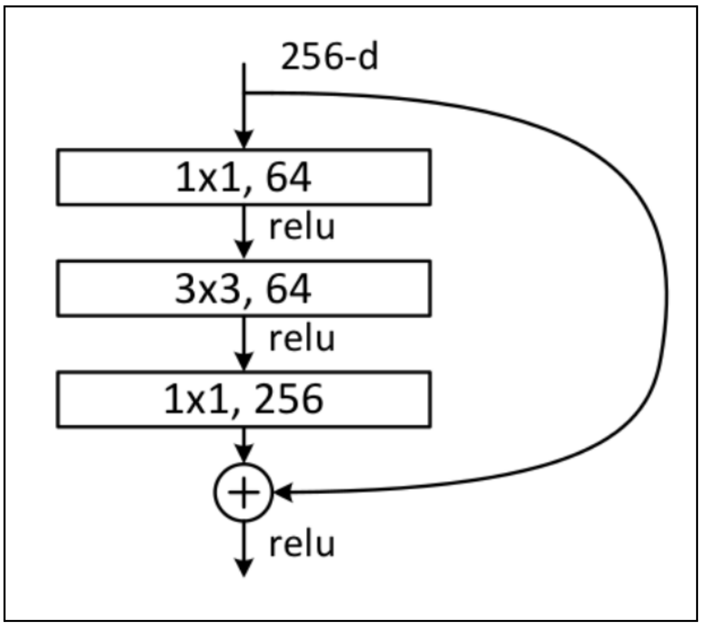

Figure 5: ResNet50 Architecture example

2.3 EfficientNetB3

EfficientNetB3 is a type of CNN that uses a type of scaling, compound scaling to scale models to obtain better accuracy more efficiently[29]. Before compound scaling, it was common to scale models by only one dimension, depth, width, or image size of model and scaling multiple of these dimensions was tedious and achieved sub-par results[30, 31, 32]. A new scaling method, compound scaling, was then introduced which uniformly scales width, depth, and image size each with a fixed coefficient.

Each of the different scaling methods, depth, width, and resolution all have different purposes for their scaling. For instance, depth is scaled to capture richer and more complex features, as well as to generalize on different tasks. Scaling network width and resolution is most commonly used in scaling small size models, which is then used to capture fine-grained features. Although scaling these features is advantageous towards achieving higher accuracy, these showed that accuracy gains significantly diminish overtime. In figure 5, where accuracy is plotted against a number of FLOPS(floating point operations per second) with scaling a particular network feature, we can see that the graph is that of a logistic shape where the accuracy initially rapidly increases, but then flattens out. Therefore, compound scaling was introduced, which uses a compound coefficient, with constant ratio say φ, to uniformly scale depth, width, and resolution. For instance, say you have 2^N more computational resources. The following list of mathematical relations demonstrates this.

- depth: a, $a \geq 1$
- width: $b \geq 1$
- resolution: $c \geq 1$
- $a^N {}_* b^N * c^N \approx 2^N$

These coefficients are determined by a small grid-search. Grid search is a hyperparameter optimization technique. Grid searching uses a combination of hyper-parameters and their values and calculates the performance for each combination and picks the best value. This helps evaluate how efficient is the processing-time and resources needed based on the number of hyperparameters involved. The hyperparameters which EfficientNet grid searches for include depth, width, and image resolution. For EfficientNetB0, the first version, it was found that the optimal coefficients for depth, width, and image resolution were 1.2, 1.1, and 1.15. What makes EfficientNetB3 different is that it has coefficients 1.2, 1.4, and 1.3[33].

2.4 XGBoost

XGBoost is a scalable end-to-end tree boosting system[1]. The system is built to increase machine learning model performance and computational speed by utilizing specific loss functions and several regularization techniques. Before piecing together XGBoost, it is important to understand both Decision trees and Gradient Boosting.

The Decision Tree algorithm's main goal is to develop a model that can be used to predict classes of given target values[1]. Figure 6 is an example of a Decision Tree structure.

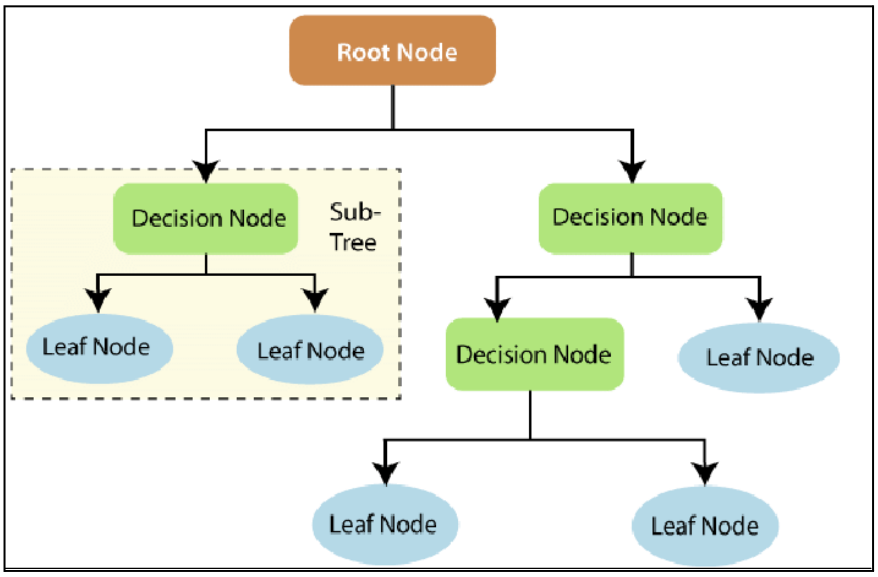

Figure 6: Decision Tree structure

In a Decision Tree structure, each decision node represents a test on an attribute. The following branches, as shown in Figure 6, are the test outcomes. Each leaf node contains a class label. Decision Trees learn from a process known as Information gain. Information gain is the mutual information between the decision node and the branch output[1]. Information gain utilizes Entropy - or a measure of a dataset's impurity[1][1]. Entropy of the classification set X with c classes is shown in equation(2).

$$E(X) = -\sum_{i=1}^{c} p_i \log_2 p_i \qquad (2)$$

p_i is the proportion the class i makes within the presented dataset. Information gain simply becomes the *E(parent) - E(children).* The *parent* and *children* inputs represent the parent node and child node within the Decision Tree structure hierarchy. Higher information gain represents more entropy being removed. A high information gain represents a better split. In terms of constructing a Decision Tree, parent nodes should be the attribute with the highest information gain from a specific set. Child nodes are then added for every parent node until a tree is built. These trees emulate the human thought process in making decisions and can be used to solve classification problems.

In short, Gradient Boosting is an ensemble technique. The ensemble technique involves creating multiple weak models and combining them. The Gradient Boosting learning procedure involves constructing and fitting new base-learners that are maximally correlated to the negative gradient of the provided loss function[1].

XGBoost is an application of optimized Gradient Boosted Decision Trees. XGBoost algorithms use an objective function consisting of both Training Loss and Regularization[1].

$$obj(\theta) = TL(\theta) + R(\theta) \qquad (3)$$

In equation(3), θ represents the parameters of a dataset[1]. Training loss is represented by *TL* and regularization is represented by *R.* In order to determine the parameters used to construct a tree, a function f_t is used to calculate the structure of the tree and their leaf node scores[1]. A new objective function at step t is born. The following equation represents this new objective function.

$$obj^{(t)} = \sum_{i=1}^{c} [m_i f_t(p_i) + \frac{1}{2} c_i f_t^2 (p_i)] + R(f_t) \qquad (4)$$

In equation(4), m_i and c_i represent inputs[1]. Equation(4), the objective function, yields the optimization for new trees being iteratively added to the model itself[1]. The final tree structure itself is formed by calculating the regularization, objective function, and leaf node scores at each level within the hierarchy[1]. At each leaf node split, the information gain is calculated. A branch is pruned if the information gain calculated is less than the regularization value calculated. The final Gradient Boosted Decision Tree can be used to classify data.

2.5 K-Nearest Neighbors

K-Nearest Neighbors is a powerful non-parametric classifier[1]. The K-Nearest Neighbors, commonly known as KNN algorithm, is relatively simple compared to many other machine learning classifiers. Basically, the classification is based on closest training examples[1]. The class most frequent amongst its k nearest neighbors is signed to the input[1]. There are many well known metrics used to calculate the distances between values in the KNN algorithm. Notable metrics include: Mahalanobis Distance, City Block Distance, Jaccard Distance, and the Hamming Distance. We will be using the Euclidean metric to calculate distances. The mathematical representation of the Euclidean metric is provided in equation(5) below.

$$d(x_1 - x'_1)^2 = (x_1 - x'_1)^2 + \ldots + (x_n - x'_n)^2 \quad (5)$$

The Euclidean metric will be used to calculate the distances between the input value, represented in equation(5) as x, and the other data values represented as points. All the neighbors are ranked by increasing distance based on the Euclidean metric calculations. A majority vote is held in order to classify the input value.

2.6 Logistic Regression

The Logistic Regression algorithm classifies an input variable through the relationship between existing independent variables. Say we have two classes. The goal, in this case, is to use the Logistic Regression algorithm to classify the classes. Logistic regression calculates the probability of an input being either one of the classes. In order to transform these calculated probabilities to a continuous scale that is unbounded, a sigmoid function is used. The sigmoid function is given in equation(6).

$$1/(1 + e^{-(x)}) \quad (6)$$

This sigmoid function is used to add non-linearity in the Logistic Regression model. Essentially deciding what class to output. The final Logistic Regression model is given by:

$$y = 1/(1 + e^{-(w_0 + w_1x_1 + w_nx_n)}) \quad (7)$$

Finally, because probability values are typically between 0 and 1, a cross-entropy loss function is used to calculate the loss value.

3. Methodology and Model Evaluation

In this study, we implement a Deep Hybrid Learning(DHL) approach. The reason for a DHL approach is to utilize the benefits of both deep learning and classical machine learning approaches and make up for their shortcomings. Deep learning removes pretty much any need to perform any feature engineering. However, the deep learning model's fully connected layers often overfits training data[1] - especially when trained using small datasets. To solve all these problems, we decided to implement a DHL approach. In a DHL approach, deep learning algorithms are used to extract features out of images[1]. The features are then passed into either a different deep learning architecture or classical machine learning algorithms for classification[1]. Our approach workflow is presented below in Figure 7.

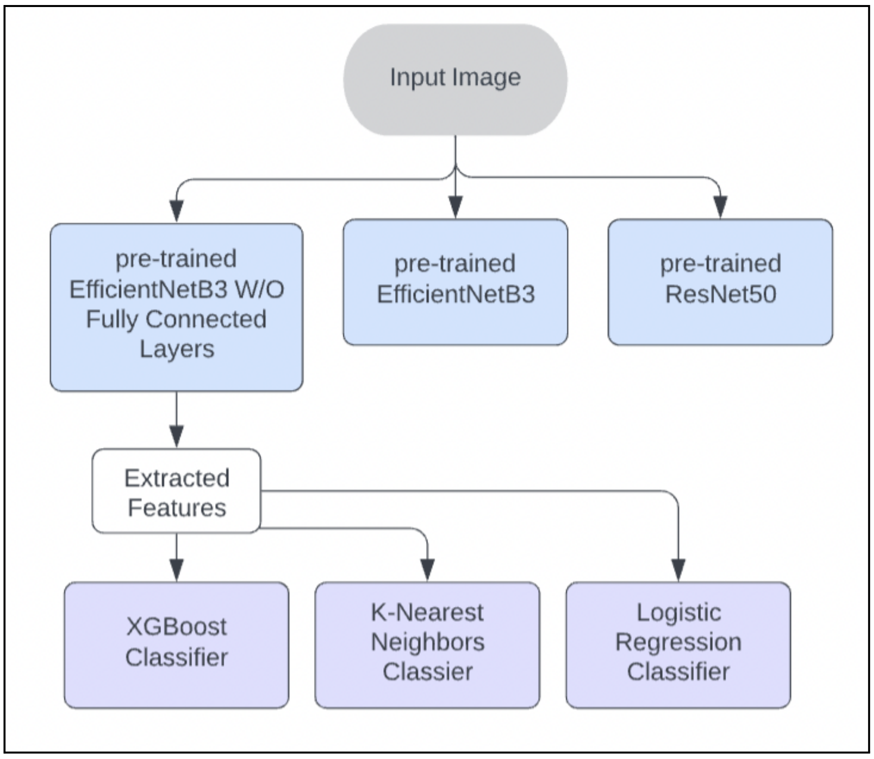
Input Image
pre-trained EfficientNetB3 W/O Fully Connected Layers
pre-trained EfficientNetB3
pre-trained ResNet50
Extracted Features
XGBoost Classifier
K-Nearest Neighbors Classier
Logistic Regression Classifier

Figure 7: Proposed DHL approach workflow

As seen in Figure 7, we train three deep learning models. The first deep learning model is a pre-trained Resnet50 model. The second is a pre-trained EfficientNetB3. The third and final deep learning model we trained was a pre-trained EfficientNetB3 that did not include fully connected layers; this was the model we used as our feature extractor in our deep hybrid learning approach. In each deep learning model we trained, we froze the weights obtained by the model during pre-training. In order to further increase the performance of all these models, we fine tuned them by adding Flatten, Dense, and Dropout layers. The EfficientNetB3 model we used for our deep hybrid learning approach finished with a Dense layer containing 512 units as output. The two other deep learning models contained the fully connected layers and finished with Dense layers containing 2 units as output; the two units corresponded to either of the dataset classes: Monkeypox or Other. The features extracted in the EfficientNetB3 without fully connected layers were then passed into classical machine learning classifiers. The features were passed into an XGBoost model, a K-Nearest Neighbor model, and finally a Logistic Regression model. The fine-tuned pre-trained EfficientNetB3 model yielded a test accuracy of 97.07%. The fine-tuned pre-trained ResNet50 model yielded a test accuracy of 94.35%. The DHL approach featuring an EfficientNetB3 model as an extractor and a XGBoost model as a classifier yielded a test accuracy of 95.81%. The following confusion matrix displays the EfficientNetB3 extractor to XGBoost classifier model performance.

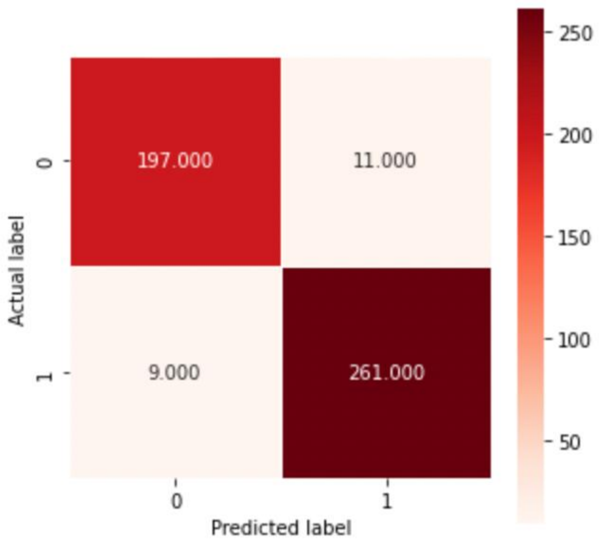

Figure 9: EfficientNetB3 extractor to XGBoost classifier confusion matrix

The DHL approach featuring an EfficientNetB3 model as an extractor and a K-Nearest-Neighbors model as a classifier yielded a test accuracy of 97.07%. The following confusion matrix displays the EfficientNetB3 extractor to K-Nearest-Neighbors classifier model performance.

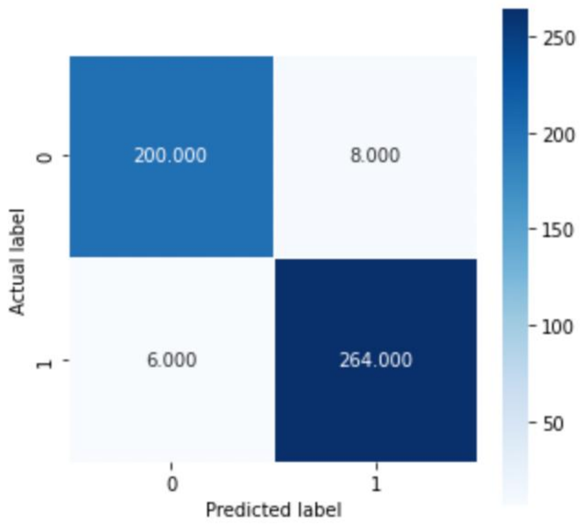

Figure 8: EfficientNetB3 extractor to K-Nearest-Neighbors classifier confusion matrix

The DHL approach featuring an EfficientNetB3 model as an extractor and a Logistic Regression model as a classifier yielded a test accuracy of 97.28%. The following confusion matrix displays the EfficientNetB3 extractor to Logistic Regression classifier model performance.

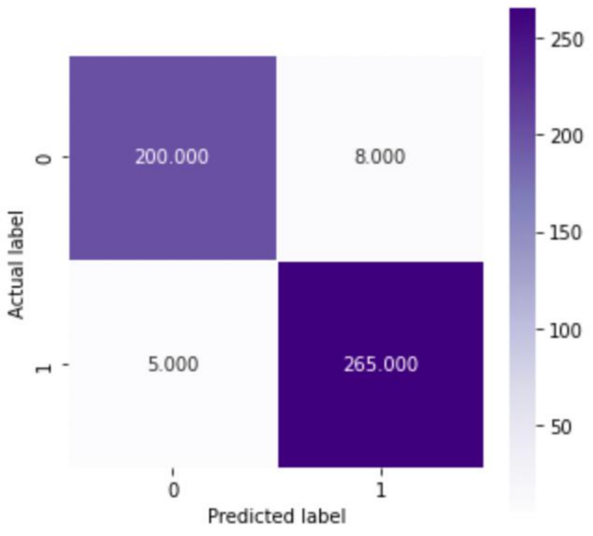

Figure 10: EfficientNetB3 extractor to Linear Regression classifier confusion matrix

In total, 2 deep learning models with fully connected layers, a deep learning model without fully connected layers, and 3 separate Deep Hybrid Learning models were developed in this study.

4. Conclusion

Monkeypox, a zoonotic disease, has shown to be of great danger through its recent 2022 outbreak where the number of cases grew exponentially. In this day and age where hospital availability is becoming more and more scarce, there is a dire need to classify diseases such as Monkeypox without the use of hospital equipment. In this study, we use deep learning and machine learning to classify medical images containing lesions as containing Monkeypox or not. Our study finds that our proposed deep learning methods can be implemented to develop high performing models.

References