PROJECT REPORT

Introduction:-

Today, we live in a time when manufactured medicines and prescriptions prevail, but do they have to be the only approach to healing? Even with all of these engineered options at our fingertips, many people find themselves turning back to the medicinal plants that started it all: Herbal remedies that have the ability to heal and boost physical and mental well-being. Mother earth is enriched and nourished with a variety of plants. These plants are useful in many ways such as drug formulation, production of herbal products, and medicines to cure many common ailments and diseases. Many parts of the plants such as leaves, bark, root, seeds, fruits, and many more are used as a vital ingredient for the production of herbal medicines.

Trying to recognize these plants just by human sight is a very tedious and time consuming process with not the most accurate results. Applications of image processing and computer vision techniques for the identification of the medicinal plants are very crucial as many of them are under extinction as per the IUCN records. Hence, the digitization of useful medicinal plants is crucial for the conservation of biodiversity.

In this project, we have built an image classification model which classifies a plant based on the images of its leaves. The algorithm used to build the neural network was CNN, but other architectures such as LeNet, ResNet, AlexNet were also quest to check their performance on the model. An analysis of each of the models is done in this report.

The types of species consider for classification consists of:

1. Amaranthus Viridis (Arive-Dantu)
2. Basella Alba (Basale)
3. Carissa Carandas (Karanda)
4. Jasminum (Jasmine)
5. Mentha (Mint)
6. Moringa Oleifera (Drumstick)
7. Punica Granatum (Pomegranate)

<u>Image Classification:</u>

Image classification refers to the task of extracting information classes from a multiband raster image. The resulting raster from image classification can be used to create thematic maps. It is the process of categorizing and labeling groups of pixels or vectors within an image based on specific rules. The categorization law can be devised using one or more spectral or textural characteristics. Two general methods of image classification are Supervised and Unsupervised image classification.

Supervised classification method is the process of visually selecting samples (training data) within the image and assigning them to pre-selected categories (i.e., roads, buildings, water body, vegetation, etc.) in order to create statistical measures to be applied to the entire image. 'maximum likelihood' and 'minimum distance' are two common methods to categorize the entire image using the training data. For example, 'maximum likelihood' classification uses the statistical characteristics of the data where the mean and standard deviation values of each spectral and textural indices of the image are computed first. Then, considering a normal distribution for the pixels in each class and using some classical statistics and probabilistic relationships, the likelihood of each pixel to belong to individual classes is computed. Finally, the pixels are labeled to a class of features that show the highest likelihood. In our project, we have made use of supervised image classification.

Unsupervised classification method is a fully automated process without the use of training data. Using a suitable algorithm, the specified characteristics of an image is detected systematically during the image processing stage. It finds spectral classes (or clusters) in a multiband image without the analyst's intervention. The Image Classification toolbar aids in unsupervised classification by providing access to the tools to create the clusters, capability to analyze the quality of the clusters, and access to classification tools. Two frequent algorithms used are called 'ISODATA' and 'K-mean'.

<u>Neural Networks</u>:-

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. Neural networks can also extract features that are fed to other algorithms for clustering and classification; so you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.

- **Working:-**

Neural networks are composed of several layers. The layers are made of nodes. A node is just a place where computation happens, loosely patterned on a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines input from the data with a set of coefficients, or weights, that either amplify or dampen that input, thereby assigning significance to inputs with regard to the task the algorithm is trying to learn; e.g. which input is most helpful is classifying data without error? These input-weight products are summed and then the sum is passed through a node's so-called activation function, to determine whether and to what extent that signal should progress further through the network to affect the ultimate outcome, say, an act of classification. If the signal passes through, the neuron has been "activated."

A node layer is a row of those neuron-like switches that turn on or off as the input is fed through the net. Each layer's output is simultaneously the subsequent layer's input, starting from an initial input layer receiving your data. Pairing the model's adjustable weights with input features is how we assign significance to those features with regard to how the neural network classifies and clusters input.

- **Types:-**

1. **Feed-Forward Neural Networks**

Feed-forward neural networks are one of the more simple types of neural networks. It conveys information in one direction through input nodes; this information continues to be processed in this single direction until it reaches the output mode. Feed-forward neural networks may have hidden layers for functionality, and this type is most often used for facial recognition technologies.

2. **Recurrent Neural Networks**

A more complex type of neural network, recurrent neural networks take the output of a processing node and transmit the information back into the network. This results in theoretical "learning" and improvement of the network. Each node stores historical processes, and these historical processes are reused in the future during processing.

This becomes especially critical for networks in which the prediction is incorrect; the system will attempt to learn why the correct outcome occurred and adjust accordingly. This type of neural network is often used in text-to-speech applications.

3. **Convolutional Neural Networks**

Convolutional neural networks, also called ConvNets or CNNs, have several layers in which data is sorted into categories. These networks have an input layer, an output layer, and a hidden multitude of convolutional layers in between. The layers create feature maps that record areas of an image that are broken down further until they generate valuable outputs. These layers can be pooled or entirely connected, and these networks are especially beneficial for image recognition applications.

4. **Deconvolutional Neural Networks**

Deconvolutional neural networks simply work in reverse of convolutional neural networks. The application of the network is to detect items that might have been recognized as important under a convolutional neural network. These items would likely have been discarded during the convolutional neural network execution process. This type of neural network is also widely used for image analysis or processing.

5. **Modular Neural Networks**

Modular neural networks contain several networks that work independently from one another. These networks do not interact with each other during an analysis process. Instead, these processes are done to allow complex, elaborate computing processes to be done more efficiently. Similar to other modular industries such as modular real estate, the goal of the network independence is to have each module responsible for a particular part of an overall bigger picture.

In our project, we have used CNN to build the model.

Convolutional Neural Network (CNN):-

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

There are three layers for CNN, namely, convolution layer, pooling layer and fully connected layers. Additionally, dropout layer and activation function are two important parameters of CNN.

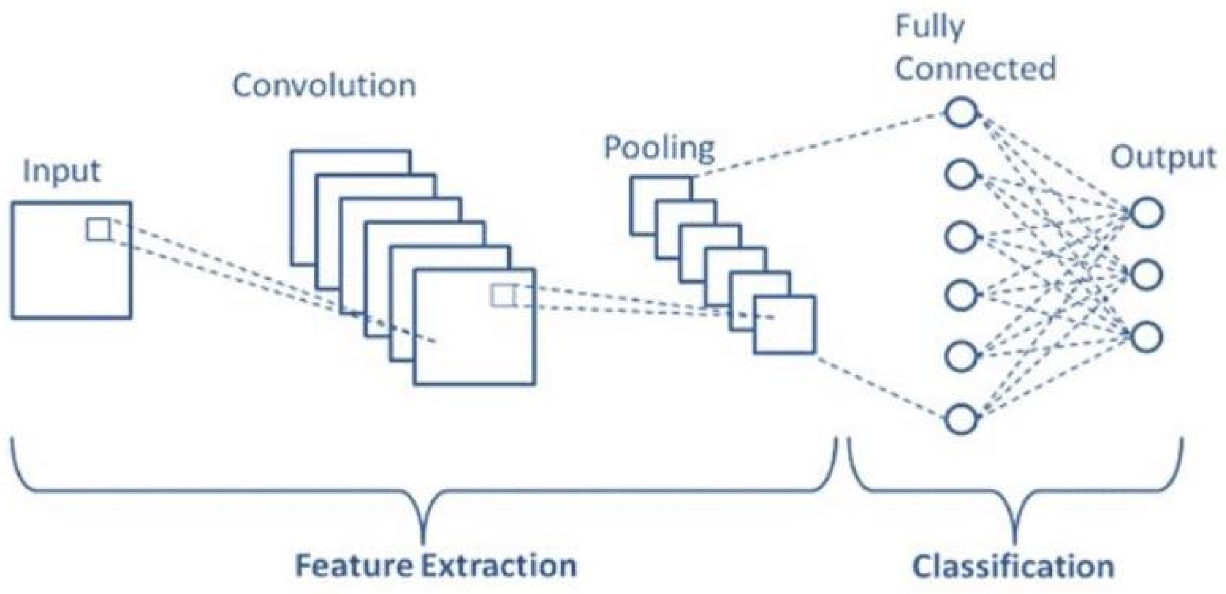

Fig. 1: Layered Details of CNN

- **Layers in CNN:**

The first layer known as the Convolution layer is used to identify different features from input images. Here, convolution is performed between input images and the predefined size filter. The filters need to slide over the input image based on that dot product should be taken for input image and filter position at that moment. The output of this product is known as a feature map. The second layer of CNN is the pooling layer. The computational costs can be reduced by decreasing the convolved feature map size. This operation can be done by decreasing the connections between layers and operates on each feature map without any dependency.There are different methods pooling operations. The average pooling is the average of the elements of the predefined image. The total sum of the elements in the predefined section is computed in sum pooling. The pooling layer is the strong connection between convolution layer and fully controlled layer. The Fully Connected (FC) layer is the combination of weights and biases of neurons. Two different layer neurons are connected through a fully connected layer. Flattened input image from the previous layer is fed to the FC layer. The vectored image modifications take place for a few layers and classification will be activated. Overfitting can occur when all features are connected through FC layers. The negative impact of over-fitting is creating the wrong impact about a model's performance on a new set of data. The solution is to drop out of the layer. 40% of the nodes are dropped out randomly on passing dropout of 0.4.

- **Activation & Loss Functions**

The most important function of CNN model is activation function i.e. a learning function. This function will decide which information in the model should fire at the end of the model. Even it can add non linearity in the model. Most popular activation functions are SoftMax, tanH, ReLU and Sigmoid Functions. The error function is conventionally known as loss function which can be used to estimate the loss of the model and weights can be used to minimize loss for next evolution.

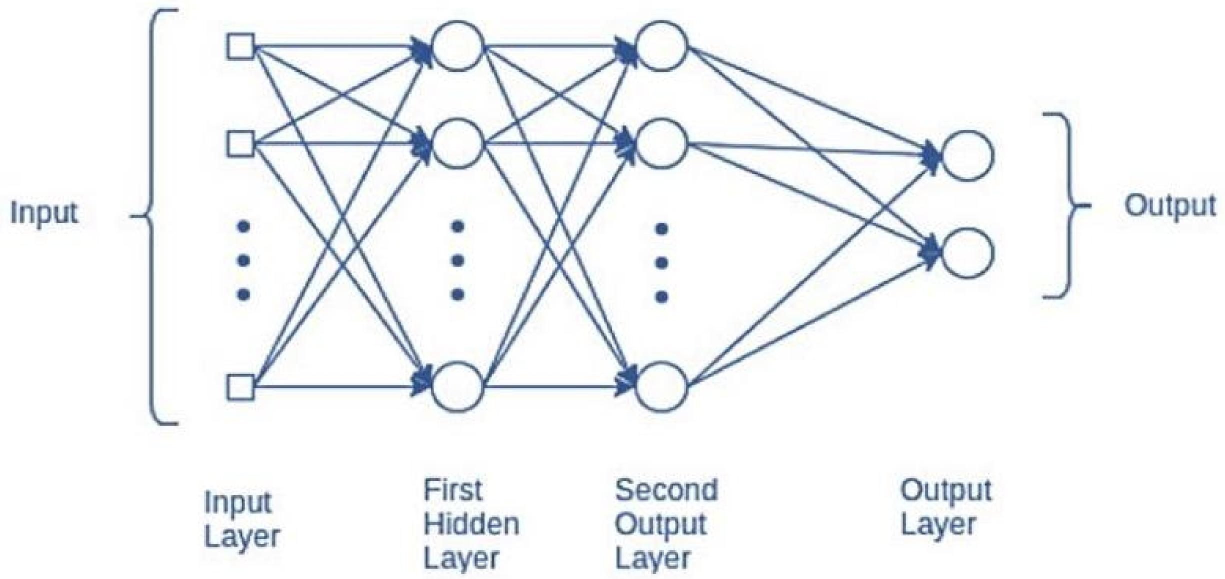

Fig. 2: Multi-Layer Perceptron architecture.

<u>Model Implementation</u>:-

The model was built using custom CNN. There are 7 layers in custom CNN. The following architecture was used after tuning the hyperparameters to achieve the best accuracy:

Layer 1:

The first layer is the convolutional layer. It consists of 32 filters. The size of each filter/kernel is (3,3). Activation function used is Relu. As we use RGB images, we have 3 layers for each image. The size of each image is (150,150). Therefore, the input size is (150,150,3).

Layer 2:

The second layer is the maxpooling layer. There are several types of pooling layers like max pooling, average pooling, etc. We have used max pooling throughout the custom architecture. The size of the pooling window is (2,2).

Layer 3:

The 3rd layer is a convolutional layer which consists of the same hyper parameters as the last convolutional layer.

Layer 4:

We add another max pooling layer with the same hyper parameters as the last max pooling layer. These layers essentially help improve the computational efficiency of the algorithm by preserving the vital features of the images.

Layer 5:

This is a flattening layer which is used to convert the 2-dimensional data to 1-dimensional data.

Layer 6:

This is a dense layer which consists of neurons with activation functions to train the data further.

Layer 7:

The 7th and the last layer is the output layer. The output Layer has 7 perceptrons for 7 classes. It gives a vector of probabilities for each class. The class with the highest probability is given as the output.

Three other architectures were used too to check their performance on the model: AlexNet, LeNet and ResNet.

1. LeNet:

- **Architecture:**

The LeNet-5 model comprises two sets of convolutional and pooling layers. Following the aforementioned 4 layers, it has another convolutional layer which acts as a flattening layer. Followed by the flattening layer, the LeNet-5 model has two back-to-back Fully Connected or Dense layers with the final layer having the Softmax activation function that acts as the final classifier layer. All the layers prior to the last classifier layer use tanh as their activation function.

First Layer:
The first layer in the LeNet-5 architecture takes 32X32 greyscale images as input. They work using 5X5 kernels and stride of 1 to generate an output of 28X28X6 shape calculating 6 features of 28X28 shape.

Second Layer:
The second layer in the LeNet-5 architecture comprises an average-pooling layer used to reduce the total parameters that are to be learned by the model. This layer works on a 2X2 kernel or filter and a stride of 2. Meaning, for every 2X2 input that it receives, it averages the values and shrinks the 2X2 data into a 1X1 data. The output computer by the average pooling layer has 6 features of 14X14 size. Hence, it generates a 14X14X6 output and feeds it forward to the next Convolutional layer/

Third Layer:
This convolutional layer works on a 5X5 kernel with stride 1 and generates 16 feature maps. Since the input has 6 features of shape 14X14, the output of these layers comes out to be 10X10X16 which is further fed to the next layer in the forward propagation.

Fourth Layer:
Just like the third layer, LeNet-4 further reduces the number of parameters to train in the fourth layer by another average pooling layer of 2X2 filter with a stride of 2. This, giving an output of 5X5X16.

Fifth Layer:
This layer is the intermediate layer between the convolutional part and the Fully-Connected part of the LeNet-5 model. It flattens the data from 5X5X16 into 1X1X120, that is, it computes 120 1X1 features to feed to the Fully Connected Dense layers as the Dense layers take "flattened" data.

Sixth Layer:
This layer is a Fully Connected or Dense layer that is connected to each of the 120 neurons in the previous layer. It outputs 84 features to the next layer.

Seventh Layer:
This is the final layer of the LeNet-5 model. It is a Fully Connected layer with a Softmax activation which gives the final probability of the output to classify the input image. Applying Argmax, we can select the final classification from the output given by the Softmax activation function.

- **Approach:**

The inspiration of our model is taken from the LeNet-5 model, but we had changed the exact model according to our input data. The original LeNet-5 model works on 32X32 images whereas the input of our dataset consists of images of shape 150X150. Furthermore, as advancements in Deep learning have been made, the use of the tanh or sigmoid activation functions has decreased by some amount due to the introduction of ReLU(Rectified Linear Unit). The drawback of using tanh or sigmoid is that as the value of our input increases, the gradient of sigmoid and tanh becomes smaller and approaches to be a constant value. Whereas ReLU can be implemented for the data point x by just max(0,x). Due to this, as the value of the input x increases, the gradient value does not become small. This can be further clarified by using the graphs of sigmoid, tanh, and ReLU.

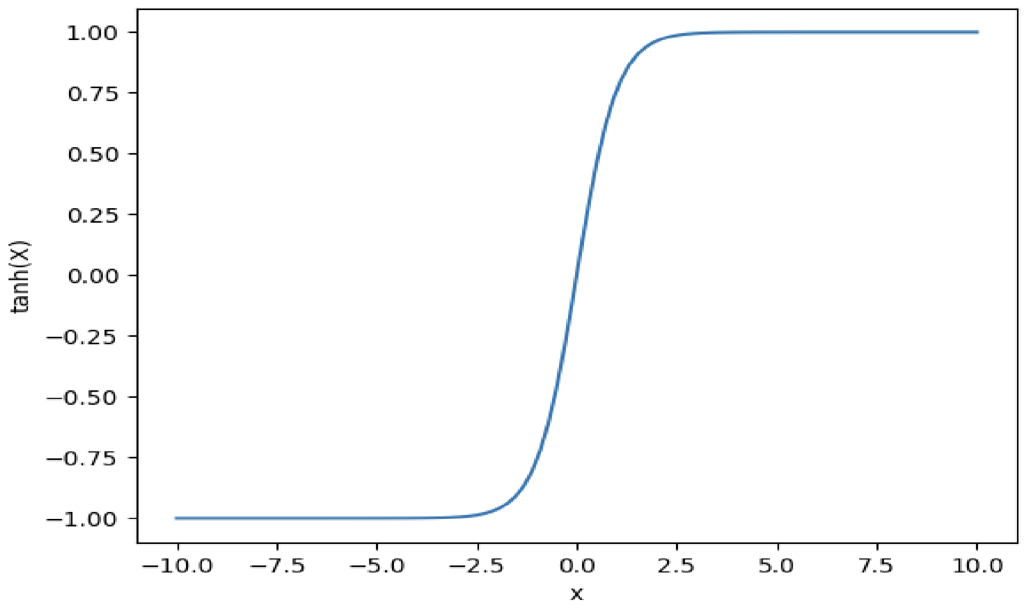

Fig 3: tanh() function graph

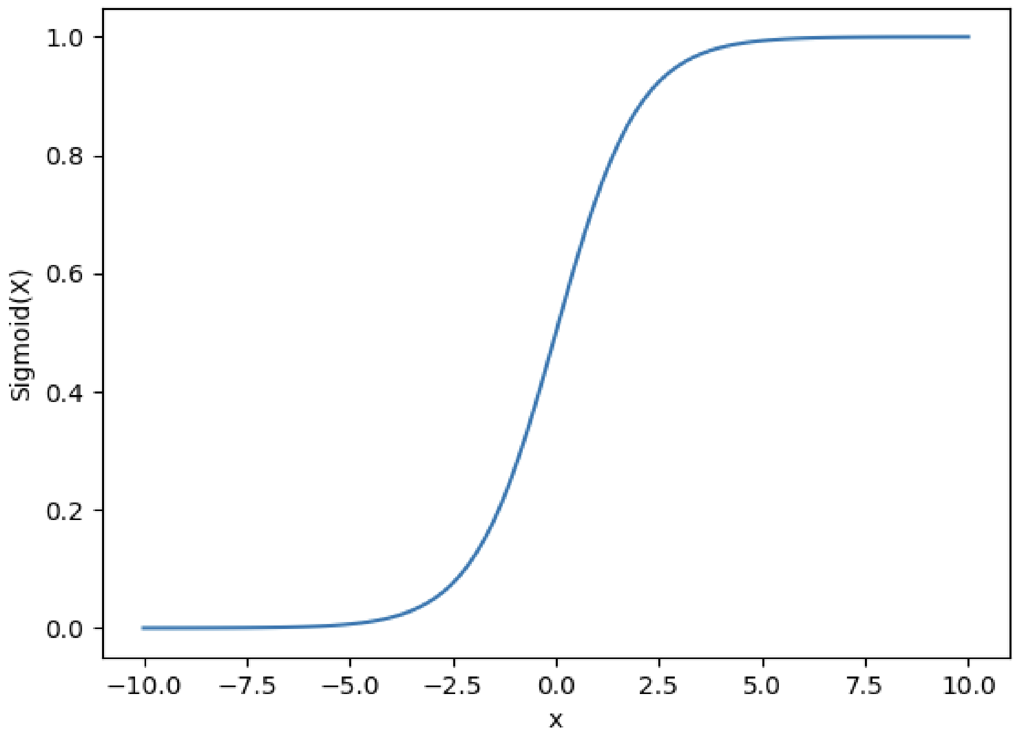

Fig 4: Sigmoid() function graph

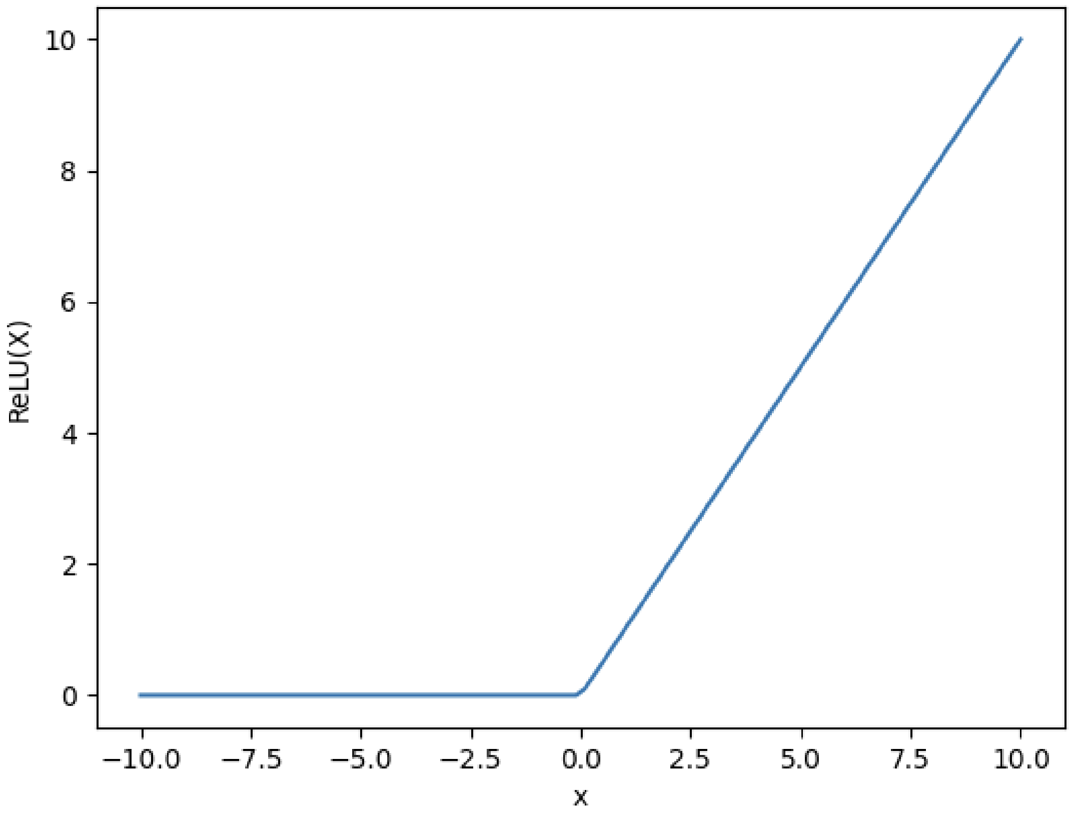

Fig 5: ReLU graph

- **<u>Our model's architecture</u>:**

First Layer:
A Convolutional Layer that takes 150X150X3 RGB images as input. Works using a 3X3 kernel to extract features, and outputs 6 features using ReLU activation.
A Normalization Layer is also here in between the first and the second layer to normalize our output of the first layer. This helps in not overfitting the model to our train data.

Second Layer:
Maxpooling layer to compress the data and reduce the number of hyperparameters to learn. Works using a 2X2 filter with stride 2. While average-pooling computes the average of the values in the filter, maxpooling shrinks the 2X2 filter in our case down to 1X1 by selecting and propagating only the maximum value of the 4 values in the 2X2 filter.

Third Layer:
A convolutional Layer that works using 5X5 kernels to extract 16 features.
A Normalization Layer is also here in between the third and the fourth layer to normalize our output of the third layer.

Fourth Layer:
Maxpooling layer to compress the data and reduce the number of hyperparameters to learn. Works using a 2X2 filter with stride 2.

Fifth layer:
A Fully Connected layer to flatten the convolutional data to be fed into the Fully Connected layers ahead.

Sixth Layer:
A Dense or Fully Connected layer of 64 neurons with ReLU activation.

Seventh Layer:
A Dense or Fully Connected layer of 32 neurons with ReLU activation.

Eighth Layer:
A Dense or Fully Connected layer of 7 neurons with Softmax activation to compute the final probabilities required for the classification of the leaf images.

- **<u>Metrics used in our model</u>:**

Optimizer: Adam optimizer.

Loss: Sparse Categorical Cross entropy (The problem of leaf classification consists of several leaves to choose from. So, our problem becomes a multi-class single-label problem and would require categorical cross entropy instead of Binary cross entropy).

Metrics measured: Accuracy of the model.

- **<u>Analysis</u>:**

Implementing the ResNet-5 model for our 150X150 images of leaf images, we got a training accuracy of 97.45% and training loss of 0.0603 after running our code for 20 epochs.

Looking at the data in the notebook, we can however say that the maximum accuracy for our model was reached after the 13th epoch. After the 13th epoch, we had a training accuracy as 99.49% and training loss 0.0304.

Hence, looking at the data after the 20th and the 13th epoch, we can say that our model starts overfitting by a bit after the 13th epoch. For optimal implementation of the LeNet-5 architecture in our case, we should stop the training right after the 13th epoch.

Plotting the accuracy and loss value graphs, we can visualize the same analysis as above.

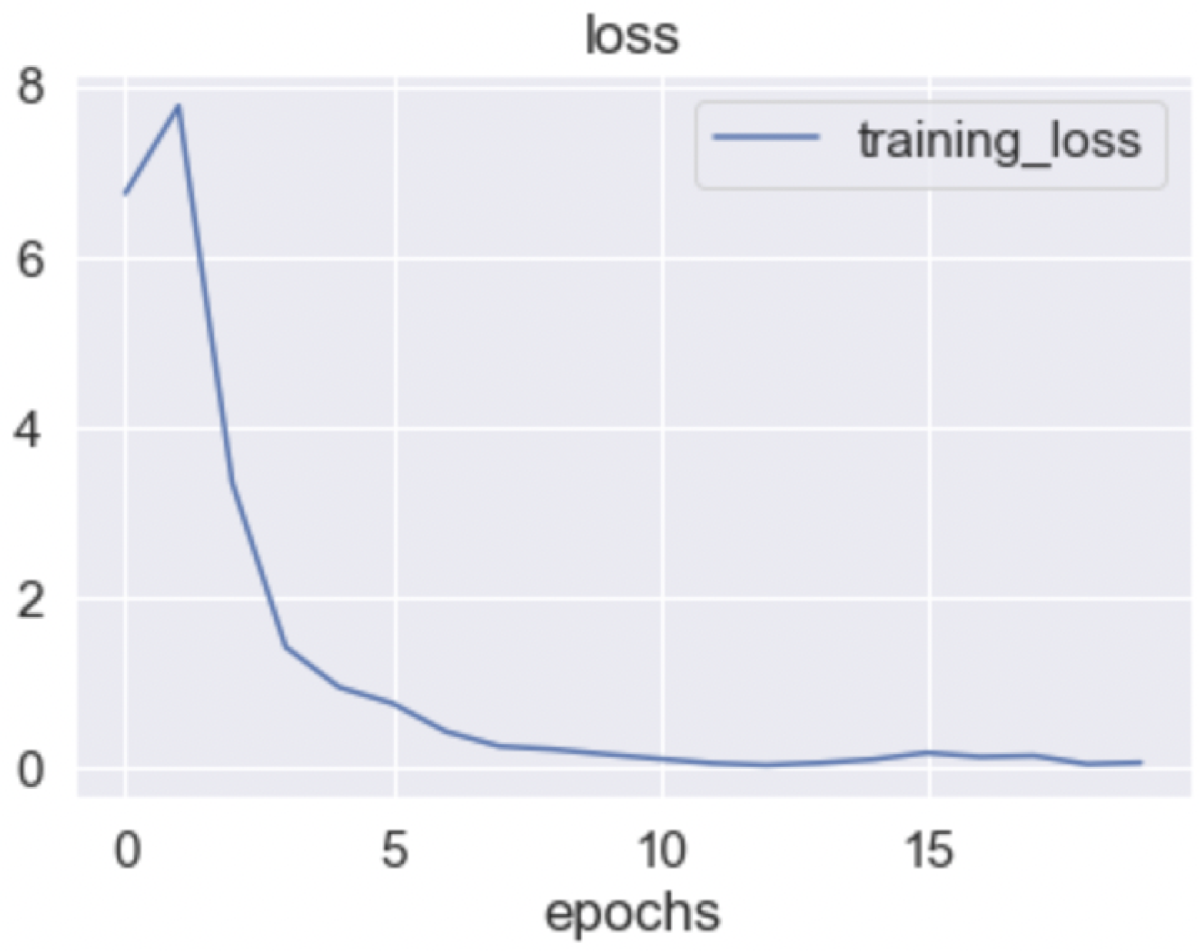

Fig 6: Loss value plot for our model during training

Looking at Fig. 6, we can see that the minimum loss we have observed is around the mid-point of 10th and the 15th epoch. Considering the output data that we have mentioned above, this should be after the 13th epoch.

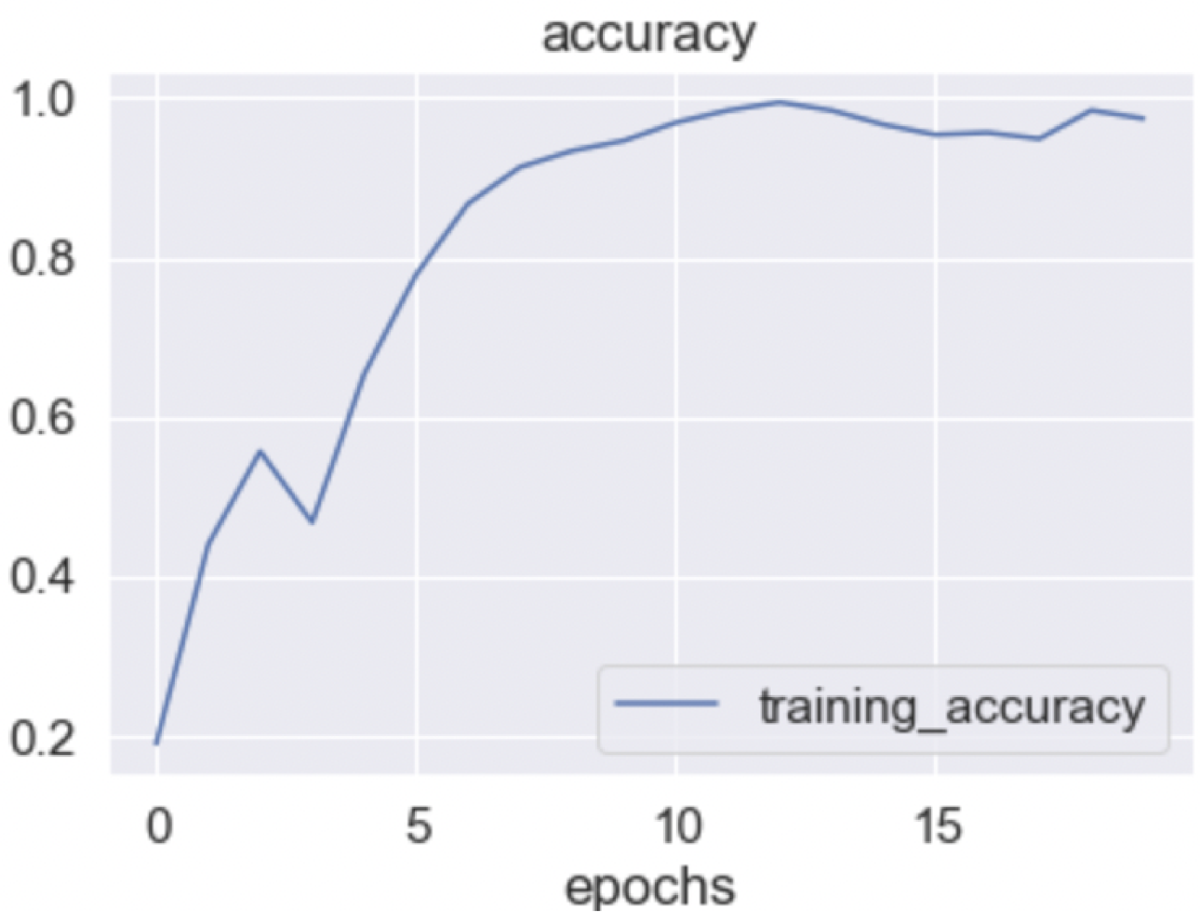

Fig 7: Accuracy plot of our model during training

Looking at the graph plotted in Fig. 7, we can confirm that we had attained maximum accuracy for the leaf classification dataset after the 13th epoch. After the 13th epoch, our model starts overfitting to the training data by a bit.

2. AlexNet:-

- **Architecture:**

AlexNet consists of 8 layers: 5 convolutional layers and 3 fully-connected layers. Some of its features include the following:

- **ReLU Nonlinearity:**
  Instead of using the tanh function, AlexNet makes use of Rectified Linear Units(ReLU) as its activation function. The advantage is seen in the training time, a CNN using ReLU was able to reach a 25% error on the CIFAR-10 dataset, which is nearly 6 times faster than a CNN using tanh.

- **Multiple GPUs:**
  Back in the day, GPUs were still rolling around with 3 GB of memory. This was especially bad because the training set had 1.2 million images. AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. This helps in cutting down on training time.

- **Overlapping Pooling:**
  CNNs traditionally "pool" outputs of neighboring groups of neurons with no overlapping. However, when overlapping was introduced, a reduction in error by about 0.5% was observed. Additionally, models using overlapping pooling are harder to overfit.

One of the major issues in using AlexNet was overfitting, since Alexnet used nearly 60 million parameters. The method used in our model to reduce overfitting is known as dropout. This technique consists of "turning off" neurons with a predetermined probability. This means that every iteration uses a different sample of the model's parameters, which forces each neuron to have more robust features that can be used with other random neurons. However, dropout also increases the training time needed for the model's convergence.

- **Hyperparameters used:**

Layer one:
The first layer is the convolutional layer. It consists of 96 filters. The size of each filter/kernel is (11,11). Activation function used is Relu. The size of each image is (150,150). Therefore, the input size is (150,150,3).

Layer two:
The second layer is the maxpooling layer. The size of the pooling window is (2,2).

Layer three:

The 3rd layer is a convolutional layer. It consists of 256 filters. The size of each filter/kernel is (5,5). Activation function used is Relu.

Layer four:

We add another max pooling layer with the same hyper parameters as the last max pooling layer. These layers essentially help improve the computational efficiency of the algorithm by preserving the vital features of the images.

Layer five:

Another convolutional layer. It consists of 384 filters. The size of each filter is (3,3). Activation function used is still Relu.

Layer six:

This is another convolutional layer with the same hyperparameters as the last convolutional layers.

Layer seven:

This is yet again a convolutional layer. It consists of 256 filters. The size of each filter is (3,3). Activation function used is still Relu.

Layer eight:

This is a maxpooling layer. The size of the pooling window is (2,2).

- **Analysis:**

In our project, we have used custom CNN, AlexNet, LeNet and Resnet. The training time required for AlexNet is 204s, which is a huge improvement in comparison to ResNet (1692s), but not as good as custom CNN (85s) and LeNet(115s). For the training dataset, accuracy for AlexNet is around 81.38%, which again is an improvement compared to ResNet (21.94%). However, we noticed that the accuracy is not as good as that of custom CNN (100%) and LeNet (97.45%). AlexNet recorded a loss of 0.5957 on the training dataset, compared to 12.5535 of ResNet, 0.0603 of LeNet and 0.0099 of custom CNN.

For the validation dataset, AlextNet achieved an accuracy of 19.39%. Once again it is an improvement only on ResNet (14.14%), but not as good as LeNet(23.47%) and custom CNN(86.73%). AlexNet recorded a loss of 10.87 on the validation dataset, in comparison with 14.1445 of ResNet, 1.6902 of LeNet and 0.3344. In all the cases, AlexNet only performs better than ResNet.

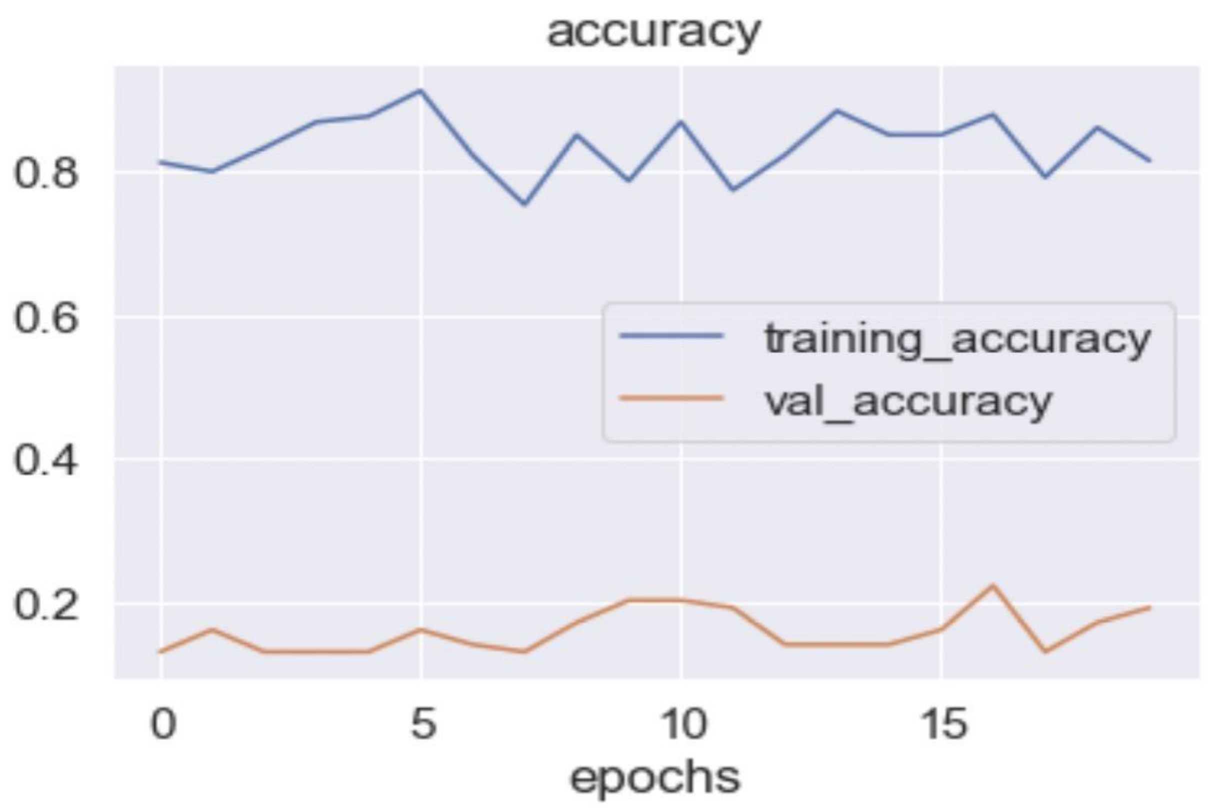

Fig. 8: Comparison of accuracy of training data vs validation data

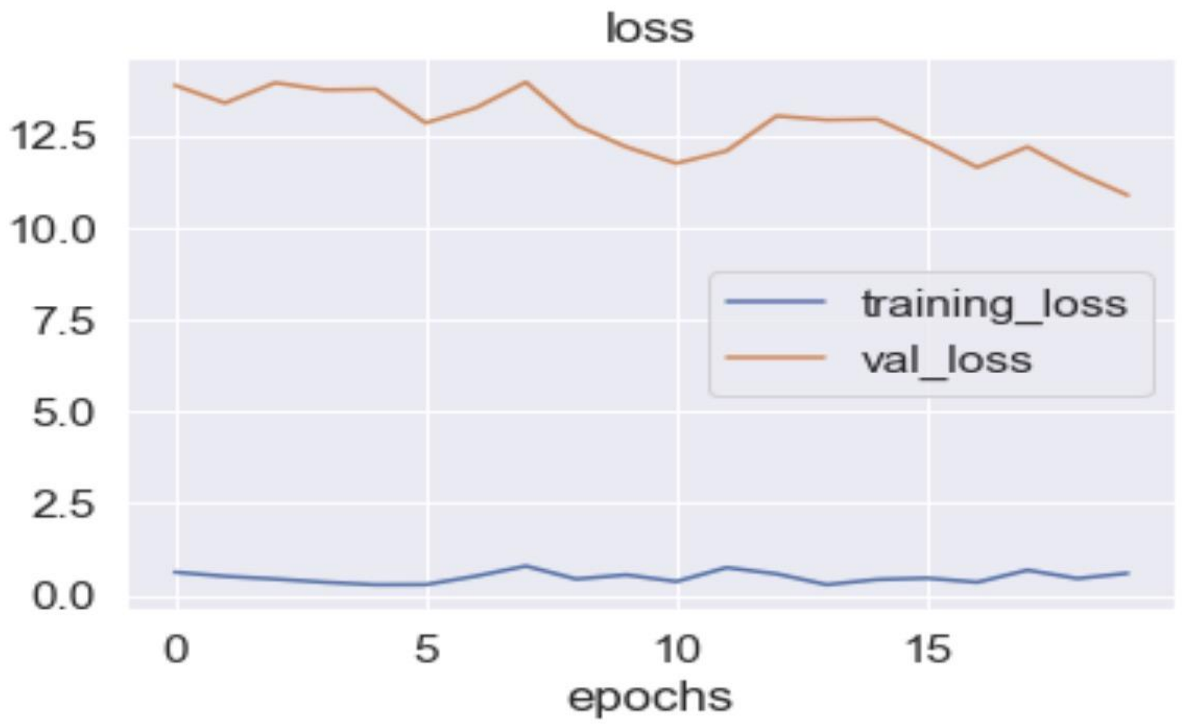

Fig 9: Comparison of loss in training data vs validation data

3. <u>ResNet</u>:-

ResNet-50 is a convolutional neural network (CNN) trained on the ImageNet dataset. It was developed by researchers at Microsoft and published in 2015. The "50" in its name refers to the fact that it has 50 layers, including convolutional layers, pooling layers, and fully connected layers. It is a deep learning model that has achieved impressive results on a variety of computer vision tasks, including image classification, object detection, and semantic segmentation.

ResNet-50 is a type of residual network, which means that it uses residual connections to help train deeper networks more efficiently. A residual connection is a shortcut that allows the model to skip one or more layers and directly pass the input through to the output of the network. This helps to reduce the vanishing gradient problem, which is a common issue that can occur when training deep neural networks.

- **Architecture:**

ResNet-50 is a deep convolutional neural network that is trained on the ImageNet dataset for image classification. It consists of 50 layers, including convolutional layers, pooling layers, and fully connected layers. The architecture of ResNet-50 can be divided into four main components:

The stem: The stem of the network consists of a few convolutional layers that are used to extract low-level features from the input image. These layers are typically followed by a max pooling layer, which reduces the spatial resolution of the feature maps and helps to reduce the computational complexity of the network.

The residual blocks: The majority of the network consists of residual blocks, which are made up of multiple convolutional layers. Each residual block contains a shortcut connection, or residual connection, that allows the input to bypass one or more of the convolutional layers and be directly added to the output of the block. This helps to alleviate the vanishing gradient problem and allows the network to learn more complex features.

The classifier: The classifier consists of a few fully connected layers that are used to make the final prediction. The output of the classifier is a vector of probabilities, where each element corresponds to the probability of the input image belonging to a particular class.

The pooling layers: ResNet-50 also includes a few global average pooling layers, which are used to reduce the spatial resolution of the feature maps and extract a compact representation of the features learned by the network.

As listed in the table below, the 50-layer ResNet design consists of the following components:

- A convolution of a 77 kernel with 64 additional kernels and a 2-sized step.
- A stride of two sizes with a maximum pooling layer.
- There are nine more layers total—three 33,64 kernel convolution levels, one 11,64 kernel layer, and one 11,256 kernel layer. Repeating these three layers three times.
- 12 more layers with iterations of 4, 1, 1, 128 and 1, 512 kernels each were added.
- Six iterations were performed on 18 additional layers, each with 1 1,256 cores, 2 3,256 cores, and 1 1,1024.
- 9 additional layers of 1, 512, 3, 512, and 1, 2048 cores iterates three times.

The overall architecture of ResNet-50 is composed of a series of blocks, each of which consists of one or more convolutional layers followed by a shortcut connection. The first layer of the network is a convolutional layer that takes the input image and applies a set of filters to it. The subsequent layers are composed of blocks, with each block consisting of several convolutional layers and a shortcut connection. The final layer of the network is a SoftMax layer, which produces a probability distribution over the classes, indicating the likelihood that the input image belongs to each class.

Overall, the 50 layers of ResNet-50 are designed to extract features from the input image and classify it into one of the 1000 classes. The residual connections help to alleviate the vanishing gradient problem and allow the network to be trained more efficiently, allowing it to achieve good results on a wide range of tasks.

- **<u>Approach</u>:-**

We have carefully selected the values of our Strides, Kernel, Filters to maximize the efficiency of the algorithm.

Our Approach is very similar to the authentic approach with ResNet-50 but we had modified the implementation characteristics of our model according to our input data in order to maximize the accuracy. Similar to the Le-Net-5 Approach we have used advanced ReLU(Rectified Linear Unit) in this approach as well. The purpose of an activation function is to introduce non-linearity into a neural network. In the case of ReLU, it outputs the input if it is positive, and outputs 0 if it is negative. This has the effect of "thresholding" the input and has been found to improve the performance of neural networks in many tasks. Advantage of the ReLU activation function is that it is computationally efficient, as it can be implemented with a simple thresholding operation. It is also differentiable, which means that it can be used in conjunction with backpropagation to train a neural network.

- **<u>Our model's architecture</u>:-**

As listed in the table below, Our 50-layer ResNet design consists of the following components:

Strides(2): The number of pixels that a convolutional filter moves, increasing the stride of a convolutional layer reduces the size of the feature maps produced by the layer and reduces the computational cost of the network have a stride of 2, which down samples the feature maps by a factor of 2 and reduces the computational cost of the network

Features(64): Feature size refers to the size of the output of a convolutional layer,

Convolution Block Implementation: Every call to Convolution Block has 3 Layers of Feature Extraction and mapping with strides and features.

Identity Block Implementation: Every call to Identify Block has 3 Layers of Feature Extraction and mapping with strides and features same as convolution

Max Pooling: Max Pooling reduces the spatial size of the feature maps by taking the maximum value within a small window and replacing the values in the window with the maximum value.

We will be using numbers to in below listing in order to demonstrate the 50 layers of ResNet-50

1: A Feature extraction of Feature= 64 Kernel: 7x7 with Stride=2

2: A Max Pool Layer of 2x2 window with Stride=2

3: A Convolution Block with Feature= 3 and with Stride=2 and Combination between Kernel of 64x64x256

6: A Identity Block : with Feature= 3 and Combination between Kernel of 64x64x256

9: A Identity Block : with Feature= 3 and Combination between Kernel of 64x64x256

12: Convolution Block with Feature= 3 and with Stride=2 and Combination between Kernel of 128x128x512

15: A Identity Block : with Feature= 3 and Combination between Kernel of 128x128x512

18: A Identity Block : with Feature= 3 and Combination between Kernel of 128x128x512

21: A Identity Block : with Feature= 3 and Combination between Kernel of 128x128x512

24: Convolution Block with Feature= 3 and with Stride=2 and Combination between Kernel of 256x256x1024

27: A Identity Block : with Feature= 3 and Combination between Kernel of 256x256x1024

30: A Identity Block : with Feature= 3 and Combination between Kernel of 256x256x1024

33: A Identity Block : with Feature= 3 and Combination between Kernel of 256x256x1024

36: A Identity Block : with Feature= 3 and Combination between Kernel of 256x256x1024

37: A Identity Block : with Feature= 3 and Combination between Kernel of 256x256x1024

38: Convolution Block with Feature= 3 and with Stride=2 and Combination between Kernel of 512x512x2048

39: A Identity Block : with Feature= 3 and Combination between Kernel of 512x512x2048

40: A Identity Block : with Feature= 3 and Combination between Kernel of 512x512x2048

41: A Max Pool Layer of 2x2 window with Stride=2

42: And finally, the classifier that makes the prediction