MSc in Data Science

Project Report

2022

Using computer vision and deep learning, what is the best model using transfer learning for the potato leaf dataset for different dataset sizes

Seyoan Santhagunam

Student Number: 150010219

Supervised by: Dr Kizito Salako

21/12/2022

Declaration:

By submitting this work, I declare that this work is entirely my own expect those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work, I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed:

Seyoan Santhagunam

Abstract:

In this dissertation report, we have applied our own Convolutional Neural Network (CNN) and other prebuilt CNN models to the potato leaf dataset for the use of predicting whether a potato will either be diseased or healthy. This was motivated by the importance of agriculture with the price of goods increasing, to hope that the vegetables we get are of the best quality. The goal of this piece of work is to determine what the best CNN model is and how businesses with a limited dataset can benefit with the application of these models.

Keywords: Convolutional Neural Network (CNN), potato disease, blight, deep learning, classification

Github link for resources:

Table of Contents

Introduction and Objectives

In the field of agriculture, fruits and vegetables face many diseases. Looking into potatoes, the diseases that are most common are either early blight or late blight. Blight is a term used to describe a number of different plant diseases that cause abrupt and severe yellowing, browning, spotting, withering, or death of leaves, flowers, fruit, stems, or the entire plant. [1]. Early blight is caused by fungus called Alternaria solani [2] and late blight, is caused by a fungus called Phytophthora infestans [3]. These diseases need to be prevented as soon as possible because if they are unnoticed, they can destroy the entire crop of potatoes.

In the majority of developing economies, blight detection and identification are done manually by trained individuals or farmers surveying the field and looking at potato leaf. This process can be very time consuming and exhausting as sometimes the process can infeasible if the plant does not have any disease or if any experts are not available in the certain region required for analysis. However, in more recent times, due to advances in technology, computer vision and image recognition allows us to identify which leaves are diseased and which are not. These methods can quickly and precisely identify plant leaf diseases without the need for human involvement. It has been noted that the agriculture field has used deep learning the most frequently [4].

Aims and objectives

According to Deep Learning models, the larger the dataset the better the accuracy, however obtaining large datasets or even producing datasets are very expensive. We aim to investigate a smaller and larger dataset in attempt to achieve high accuracies for those businesses who do not have as much free flowing money.

In order for us to be able investigate these datasets and come to a conclusion, we need to satisfy the following aims and objectives.

The aims and objectives of this thesis are to,

1) Perform image classification using 4 different types of CNN's.
2) Evaluate the performance for each neural network model.
3) Provide detailed explanation and analysis to the applied models.
4) Provide a solution for small businesses and large businesses, which CNN model would be most suitable.

Research Question

Which out of the 4 models, CNN, VGG19, GoogLeNet and ResNet, performs the best providing their accuracies and losses?

Workflow diagram

Our workflow will consist of a combination of a literature review and code. The diagram below shows the coding work flow. We will then use our results and make our evaluations from there.

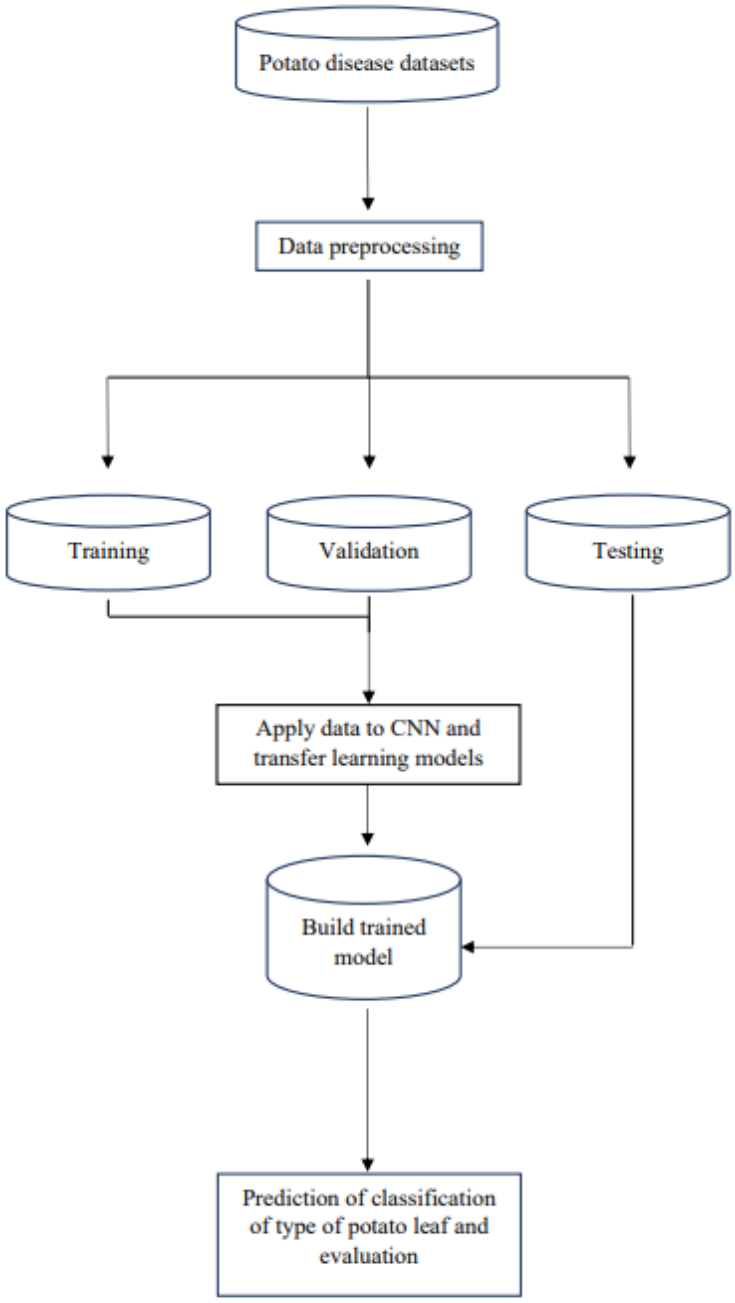

Figure 1: Work Plan

Changes made in the goals and methods

There were no major changes to the goal, or the methods complied. The same models will be used for classification; however, the only change is that, rather than partitioning the

PlantVillage dataset [5], into the whole dataset and taking a small part of it. We will be instead using the PlantVillage dataset as the smaller dataset and we will be including the Pakistan potato leaf dataset as the larger dataset. Our outcomes for this project, thus remain the same.

Report Structure

Chapter 2 will cover the state of art that is connected to this study and provide the motivation behind the use of CNN's and other models applied for the field of agriculture, including results they had produced. It will consist of what this study has changed from other studies and how this study has been developed from other studies.

Chapter 3 will investigate the methods used, explaining why the choices were made for pre-processing, the choices of models used, the networks of the chosen models.

Chapter 4, will include the results from each model, detailing a comparison of the models used.

Chapter 5 is the discussion in which we will look into the performances of the models and compare the results of our models.

Chapter 6 will then look into evaluation and reflection on the projection as a whole, seeing where we can improve for future work.

Context

In this section we look into previous studies conducted to subjects similar to this study.

Motivation

If not prevented in time, plant diseases might have a severe impact on the agriculture sector and cause mass hunger throughout human society. With the use of machine learning and deep learning models in the field of plant pathology, the early diagnosis of plant diseases will become simpler and less expensive, assisting many farmers in reducing plant loss and preventing the spread of illnesses from diseased to healthy plants.

Numerous studies have been conducted on detecting plant diseases providing a comparison study utilising various deep learning models, however they fall short of explaining the predictions that their models make. In this study, we compare how the models work for small datasets and large datasets, and we also attempt to offer an explanation for the predictions that the models make.

The dataset is divided into two classes: Late Blight and a healthy class category for the first class. “Lack of data is one of the issues with machine learning” as quoted in this paper. 596 of their own photographs and 430 from a publicly accessible collection were utilised to train a model. There are applications of data augmentation methods, transfer learning, and 5-fold cross-validation to overcome the issue of a tiny dataset. For transfer learning strategies, pre-trained models of InceptionV3, VGG16, and VGG19 were employed. Looking into their work, InceptionV3 model outperformed other pre-trained models in testing using unseen data, scoring 87%. They believe that using a much larger dataset would benefit the models accuracy [28].

CNN

With the first class being healthy and the other four being disease-infected [30], created a five-class classification model for potato tuber disease using CNN. The dataset was created with a variety of cultivars, sizes, and diseases in potato tubers. There are 400 images of diseased potato tubers with smooth skin that are different colours and shapes. The photos were captured under inconsistent lighting conditions with a camera. The photos were subsequently labelled using Matlab 2014b software and transformed to grey scale at a resolution of 224x224 pixels. Data augmentation was performed to enhance the amount of photos in the dataset prior to training.

The author's final suggestion was the VGG CNN architecture, which has eight layers, the first five of which are convolutional layers, three fully connected layers, and a Softmax end layer activation function that aids in image classification. The model had accuracy rate was 95.8%.

Methods

In this section we will look into the architecture of each model used and a step-by-step overview which entails a justification for each step.

To understand CNN's, we first need to understand feed forward neural networks (FFNN)

Feed Forward Neural Networks

A version of the neural network is the Feed forward neural network (FFNN). The FFNN is just a collection of neurons. Each layer consists of many neurons and each layer has its own name. the initial layer of the FFNN is the input layer/vector which consists of multiple features. The intermediate layers are known as the hidden layers, the amount of hidden layers depends on the use of the network. If the number of hidden layers is greater 1, it is known as a deep network. In the FFNN, the input values get multiplied by the weights and the bias is added to it, making it a linear calculation. There is also an activation function which is involved for prediction. This step is repeated in order to move forward, giving it the name feedforward neural network. The final/output layer will tell us the predictions. Mathematically speaking, the linear calculation is represented as the transpose vector of weights multiplied by the vector of inputs plus the bias.

$$\left(W^T \cdot x\right) + b$$

Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a class of deep neural networks which is most commonly applied for analysing visual images. The advances of computer vision are increasing exponentially day by day due to deep learning, which is involved in CNNs. CNNs are used in computer vision are face recognition, image classification, natural language processing (NLP) etc. It is very similar to how neural networks are built.

How do computers visualise images?

As humans, we identify an image by our previous stored knowledge of images and recognise the image we are looking at, if it looks similar to what we have seen before or not. Unfortunately, computers do not work the same way. Images are arrays or a matrix of squared pixels arranged in columns and rows made up of 3 colour channels RGB (Red, Green and Blue). Similarly, a grey scale channel, would consist of 2 colour channels (black and white).

Why should we use CNN?

Looking into the feed forward neural network (FFNN), the FFNN, generates pixels which are the total number of weights. For example, if we take a random image of pixel size 1024 and apply the fully connected layer of the feedforward neural network, then the size of the image changes to (1024 * 1024 *3) which is the width multiplied by the height multiplied by the number of colour channels. This is the sum of the total weight. This value is unmanageable and in order for us to manage the weight and the bias we apply CNNs. The aim of the CNN is to reduce the image into a form which is easier to process, without losing the features that are important for obtaining a good prediction. CNN helps for the extraction of features of the image and converts it into lower dimensions without losing its characteristics.

The most used tool for classifying plant leaf diseases is CNN. Other deep learning (DL) networks, such fully convolutional (FCN) and deconvolutional (DCN) networks, are typically employed in the fields of image segmentation [6] and medical diagnostics [7] nonetheless, they are not used to classify plant leaf diseases.

Architecture of CNN

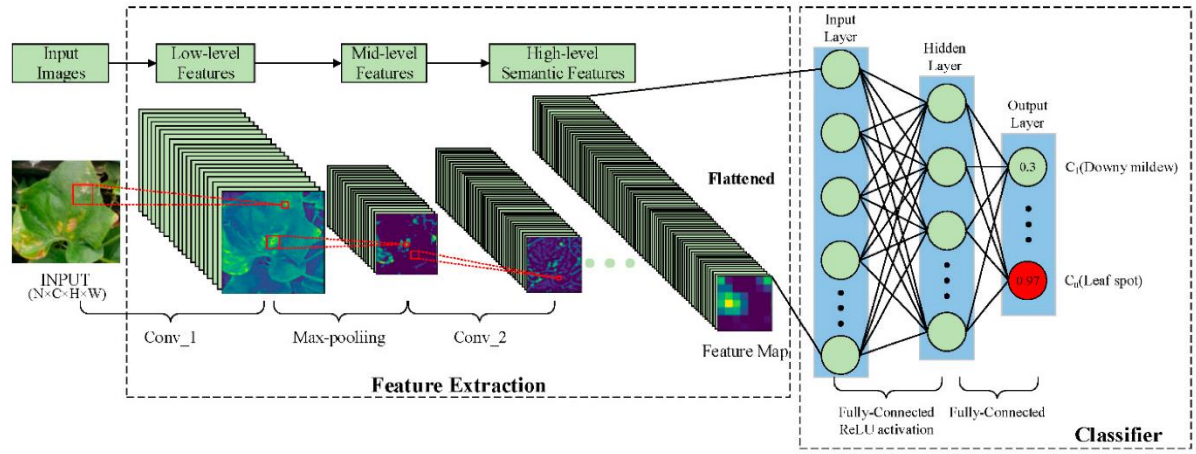

Figure 2: CNN architecture [8]

In the Convolutional Neural Networks (CNN) we have 3 main layers: convolutional layers, pooling layers and fully connected (dense) layers. We will be using CNN to create our model and for the pre-trained models.

Layers

Convolutional layers

The convolutional layer, the first layer in a CNN, is where all the subsequent layers' values are altered. This reduces the size of the image while preserving the information, giving the convolutional layer's final output in the form of a vector [9].

Pooling layers

Each channel in the feature map is covered by a two-dimensional filter during the pooling operation, and the features in the region are summarised [10]. The number of features in the feature map is decreased through pooling. The features in a section of the feature map produced by the convolutional layer are summarised using pooling [10].

There are numerous types of pooling, the pooling technique we will be using is max pooling.

Max Pooling layers

In order to create a layer where all the features have the same maximum value, max pooling selects the highest value in the area of the feature map where the filter has been applied. The features are kept when this is done, and the image size is reduced, which is highly helpful for computing.

Convolutional layer followed by max pooling layer is repeated repetitively. After being flattened, this output is subsequently used as an array of input neurons for the neural network.

The hidden layers and output layer make up the neural network after that. Softmax will be used as the activation function in our last layer because it normalises the likelihood of our classes.

Fully connecter layers and output layers

The final few levels of the CNN are the completely connected layers. There is a dense layer where it flattens into a vector after pooling. In order to use the probabilities for classification or to acquire accuracy, it next passes via the Softmax activation layer.

We'll also be adopting other pre-trained models after we finish our CNN. We shall make use of the idea of transfer learning to accomplish that.

Activation functions

Activation functions' main objective has been to guarantee non-linearity with respect to the model. This was done to modify/lower the output that was produced. It is a crucial component and affects how well the model works. Each layer is followed by the application of the activation function, and the output is then used as the input for the following layer [11], creating an iterative process. There are many different types of activation functions for CNNs.

Rectified Linear Unit (ReLU)

The most popular of activation functions is ReLU (Rectified Linear Unit). It is most commonly used for deep learning. It is a piece-wise function where if the input value is less than or equal to 0 the output is 0 and if the input value is greater than 0 then the input and output value will be the same. In other words, f(x) = x. The function is represented below, with the graph of the function.

$$\begin{cases} 0, \text{if } x \leq 0 \\ x, \text{if } x > 0 \end{cases}$$

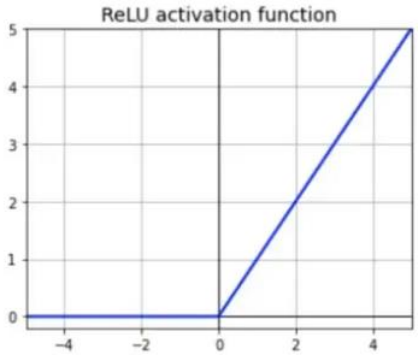

Figure 3: Diagram of ReLU function [12]

There are many other functions such as the hyperbolic function of tan(x), tan(h), or even the sigmoid function which could be applicable unfortunately these functions would both suffer from the vanishing gradient problem, as it would hinder the model's ability to train well during backpropagation, where gradient information is useful.

For the final layer of the CNN, we will use the activation of Softmax. The neural network's raw outputs are converted into a vector of probabilities—basically, a probability distribution over the input classes—by the softmax activation function [13]. It is commonly used for multi-classification tasks, such as our case. The function is represented below

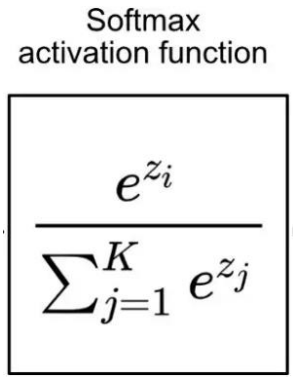

Figure 4: Function of Softmax [14].

Data augmentation

In order to expand the amount of data from the current data, data augmentation is the act of making minor changes to the data. We apply the Image Data Generator from the Keras library to reduce overfitting the model. We apply scaling where all the pixels are divided by 255 so all the pixels have a value between 0 and 1. Data augmentation methods include flipping (horizontally and vertically), rotation, cropping, and more. Since the amount of data has expanded and it can be adapted as a regularizer, we use data augmentation to prevent overfitting [15].

Transfer learning

Transfer learning is the process of applying the relevant parts of a machine learning model that has already been trained to address a distinct but connected issue [16].

The model will typically need this information as its foundation, with new components added to it to address certain problems.

Transfer learning models

VGG19

One of the major uses of VGG, which was developed by the Visual Geometry Group at Oxford, is as a classification architecture for datasets. It may also be used for transfer learning, in which an existing model is used to the solution of a separate but connected issue. To increase accuracy, it employs deep convolutional neural layers. A deep convolutional neural network (CNN) called VGG is utilised to classify images. The VGG19 model is made up of many layers.

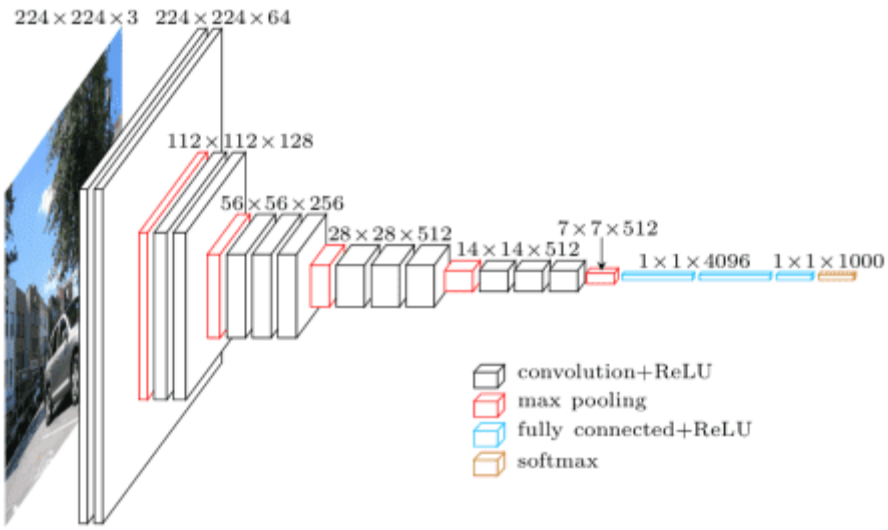

Figure 5: VGG19 architecture [18]

Simonyan and Zisserman presented the VGG network design in their 2014 publication [19].

Using only 33 convolutional layers placed on top of one another in increasing depth, this network is distinguished by its simplicity. Max pooling handles reducing volume size. Then comes a softmax classifier, which is followed by two fully connected layers with each having 4,096 nodes. In the year of 2014, a layered network of size 16, VGG16, and a layered network of size 19, VGG19, were considered to be very deep network, however that is not the case currently as models such as ResNet uses 50-200 layers with ImageNet and CIFAR-10 uses over 1000. For Simonyan and Zisserman, the task of training these networks was very difficult, so they trained networks on much smaller layers, from [19], column A and C.

This network obtained a fixed-size (224*224) RGB picture as input, indicating that the matrix was shaped (224,224,3). The mean RGB value of each pixel, calculated throughout the whole training set, was the only pre-processing that was carried out. They were able to cover the entirety of the image by using kernels that were (3*3) in size with a stride size of 1 pixel. To maintain the image's spatial resolution, spatial padding was applied. Stride 2 was used to conduct max pooling over a (2*2) pixel window. This was followed by rectified linear unit (ReLU) to introduce non-linearity to form the model classify better and to boost process time because the previous models used tan(h) or sigmoid functions this verified far better than those. This implements three fully connected layers where the first two layers are of size 4096 and the third layer is of size 1000. The final layer involves the use of softmax.

This has been represented in the diagram below.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 6: Column E: shows the network architecture for VGG19 [19]

InceptionV3 (GoogLeNet)

GoogLeNet is a 22 layer convolutional neural network. You can import a network that has already been trained using the ImageNet or Places365 datasets. The network built on ImageNet divides photos into 1000 different object categories, including several animals, mice, keyboards, and pencils [20]. Similar to networks trained on ImageNet, Places365 networks classify photos into 365 distinct kinds of places, such as lands, parks, racetracks, and halls. For a variety of image formats, these networks have learned various feature representations.

The essential role of these max pooling layers is to down sample the input as it is put through the network. To do this, they reduce the height and width of the data input. GoogLeNet is made up of 9 inception modules, including 2 max pooling layers. Another efficient way to lessen the computational strain on networks is to reduce the input size at inception. The average pooling layers restrict the input height and width to 1x1 and calculate the mean over all feature maps created since the last inception. The dropout (40%) layer decreases the number of interconnected neurons in a network at random in order to prevent overfitting. The neural network's final layer, the softmax layer, calculates the probability distribution. A vector containing the odds of each outcome is the function's output. The vectors' values sum up to 1 in total.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 6: Network architecture for InceptionV3 [21]

ResNet50

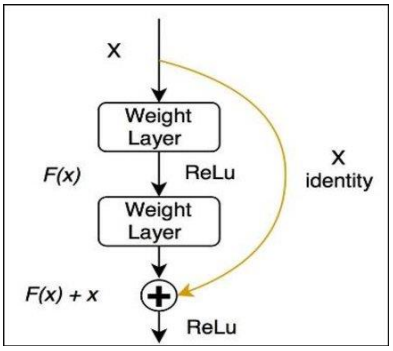

Figure 7: Functionality of ResNet

Many computer vision tasks use ResNet, sometimes referred to as Residual Networks. ResNet was a big step forward since it allowed us to train extraordinarily deep neural networks with more than 150 layers [22]. There are eight convolutional layers. The ResNet-50 model has 5 stages, each of which has a convolution and an identity block. There are three convolution layers in each of the identity and convolution blocks.

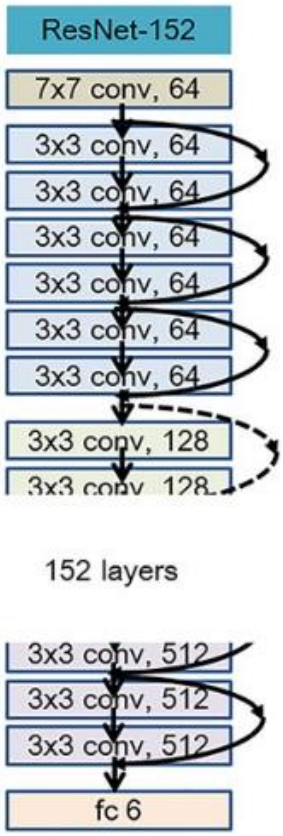

Figure 8: Network architecture for ResNet [23]

Data collection

For data collection, we need to collect images of potato leaves which are either healthy, contain early blight or late blight.

There are many ways we can collect data. We can use pre-existing data which can either be bought from an individual who is selling it or can be obtained from websites which provide data for free such as Kaggle [5].

A second option we can use to collect data is, to individually or set up a team to collect the required images from farmers and get these images classified into the desired groups being either healthy potato plants or diseased potato plants.

However, the issue with manually collecting data is that it can be very expensive. Costs include requiring a cost for travel, equipment expense such as a camera, maybe even cost of accommodation depending on how far the potato farm is. Time consumption is another factor which plays a part in collecting data manually. For small businesses, getting their own data would not be beneficial as they would have to face the setbacks mentioned above. Due to these factors, we will be using pre-existing data provided to us from Kaggle.

The smaller dataset we will be using is the PlantVillage dataset [5].

PlantVillage are a non-profiting research group who work alongside the United Nations (UN), based in Pennsylvania State University, to aid small time farmers to grow more crops [24].

The larger dataset we will be using is Pakistan potato leaf dataset [25]

The Central Punjab region of Pakistan was used to develop the Pakistan Potato Leaf Dataset. The films and pictures that make up the visuals were taken from a real-time dataset. The versions were created using a variety of capturing tools, including digital cameras, drones, and cell phone cameras. The capturing distance for digital cameras and mobile phone cameras was set at 1-2 feet, while the drone's capturing distance was set at 5-10 feet. video and image distortion caused by plant leaf movement caused by drone fanning; therefore, the plant and drone distance maximised as much as possible. The cities selected the district Okara in the Central Punjab region of Pakistan due to the higher cultivation of potato and different types of potato found in the district Okara: Coroda, Mozika and Sante [25].

Dataset

Both of the datasets used in this study are explained below.

PlantVillage dataset

This dataset consists of 2152 images of potato leaves, and there are three classes: Early blight, Healthy Leaf and Late blight. Of the 2152 images there are 1000 early blight images, 152 healthy images and 1000 late blight images. All the images are of size (256*256). These images will be resized in order to be processed into the pre-trained models.

Index	Type of leaf	Count
0	Early blight	1000
1	Healthy leaf	152
2	Late blight	1000
Total	-	2152

Table 1: Table representing the count of each variable present in the dataset

Pakistan Potato Leaf dataset

This dataset consists of 4072 images of potato leaves which is almost double the amount of data of the previous dataset. Once again, there are three classes: Early blight, Healthy Leaf and Late blight. Of the 4072 images there are 1628 early blight images, 1020 healthy images and 1424 late blight images. All the images are of size (256*256). These images will be resized in order to be processed into the pre-trained models.

Index	Type of leaf	Count
0	Early blight	1628
1	Healthy leaf	1020
2	Late blight	1424
Total	-	4072

Table 2: Table representing the count of each variable present in the dataset

Loss function

The loss function used here is the categorical cross entropy loss. Each class's predicted probability is compared to the actual class's desired output, which can be either 0 or 1, and a score or loss is computed that penalises the probability based on how far it deviates from the actual expected value. Due to the penalty's logarithmic structure, significant differences near to 1 receive a large score, while small differences close to 0 receive a small score [26]. The formula is as shown below:

$$L_{CE} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

Formula to calculate the loss function

In order to change the model weights during training, cross-entropy loss is utilised. The goal is to reduce loss; hence, the smaller the loss, the better the model. A cross-entropy loss of zero indicates a perfect model.

Model evaluation metrics

Model evaluation is essential to determining how effectively a model works and performs. In image classification, we can use the confusion matrix to help us identify how well our model is working with predicting outcomes. The confusion matrix creates predicted values and actual values after classification. The matrix looks like this:

Actual class	Predicted class	
	Positive	Negative
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

Table 3: Table for confusion matrix

True positive (TP) – tells us the number of correct predictions when the actual class was positive.

False positive (FP) – tells us the number of incorrect predictions when the actual class was positive.

True negative (TN) – tells us the number of correct predictions when the actual class was negative.

False negative (FN) – tells us the number of incorrect predictions when the actual class was positive.

These variables provide assistance in calculating the accuracy, recall, precision and F1 score.

Accuracy

The accuracy is used to identify the how efficient the model is. It takes into consideration the total amount of correct predictions. The formula is as shown below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall

The recall looks at the number of actual positive values which were predicted correctly. The formula is as shown below:

$$Recall = \frac{TP}{TP + FN}$$

Precision

The precision tells us how many of the correctly predicted values are actually positive and it provides us with how reliable the model is. The formula is as shown below:

$$Precision = \frac{TP}{FP + TP}$$

F1 Score

In a way, the F1 score is a combination of the Recall and Precision, the harmonic mean. The formula is show below:

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Results

PlantVillage

This was completed via Google Colab, Python 3.7.15 using tensorflow and keras. The optimiser present was Adam and the learning rate was kept as default. As mentioned earlier, the loss function used was categorical cross-entropy.

CNN

For the CNN model, first we mount our drive making all our file paths accessible. We then import the necessary libraries, the libraries being tensorflow, keras, matplotlib, pandas and numpy and also define our constant variables. We let our image size to be to 256, so the length and width will be 256 by 256. We assign the batch size to 32 meaning the total number of images, 2152, will be divided into image batches of 32 creating a round figure of 68 batches. We set our number of channels to be 3 which indicates the colours RGB standing for Red, Green, and Blue. We also will be using 20 epochs to run all of our models.

In this multi classification task we know that there are 3 possible outcomes: either early blight, late blight or healthy. So, when printing the class names, 0 indicates early blight, 1 indicates late blight and 2 indicates healthy. When creating the for loop we aim to see the shape of the image, which provides the shape built up of the (number of images in the batch, image size, image size, channels) and when we see the label_batch.numpy() it changes our tensor to numpy, this produces a list of classifications of each image within the batch. In trying to see the first image of the batch, our image shown is of overexposure due to the image's data type being a float.

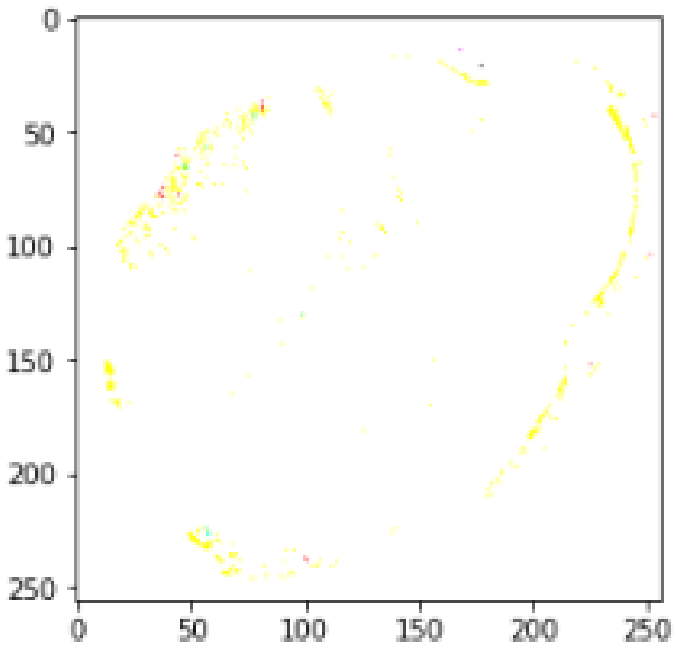

Figure 9: Overexposed image due to being a float.

To overcome this issue, we convert the image which is a float into an integer as well as including the class name above the image.

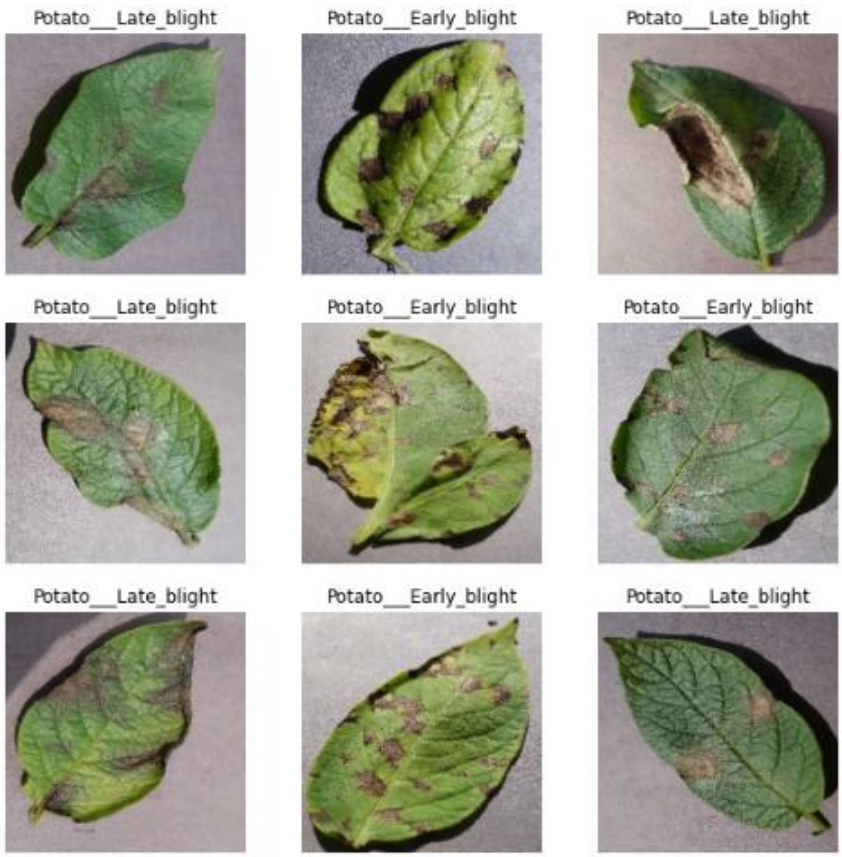

Figure 10: A visualisation of the 9 images

We then create a function in which aids in splitting the dataset, where the data is split into 80% going into the training data, 10% going to the validation data and the remaining 10% going to the testing data. We then check the length of each dataset, which sums to the total number of batches, 68.

We now rescale and resize the data, where the images are resized to the desired size and all the images are normalised to a value between 0 and 1. Data augmentation is now applied to the images are in which the images are flipped horizontally and vertically as well a random rotation of 20%.

We now begin building the model. In building the CNN model, which was explain in chapter 3, we also look at the summary where it shows the layers in the CNN. We compile the model using the Adam optimiser and use the categorical cross entropy for our loss function and we use the accuracy metrics to determine how well our model predicts images.

We begin running our model with the train data and validation data with 20 epochs, meaning that the data passes through the neural network 20 times, forwards and backwards. We apply this to the test data where we call the scores which consists of the loss and accuracy. Our test accuracy is at 99.2% and our loss is at 0.0394.

We can plot the graph of accuracy against epochs and loss against epochs for the model of CNN for both the training and validation data as shown in the diagram below.

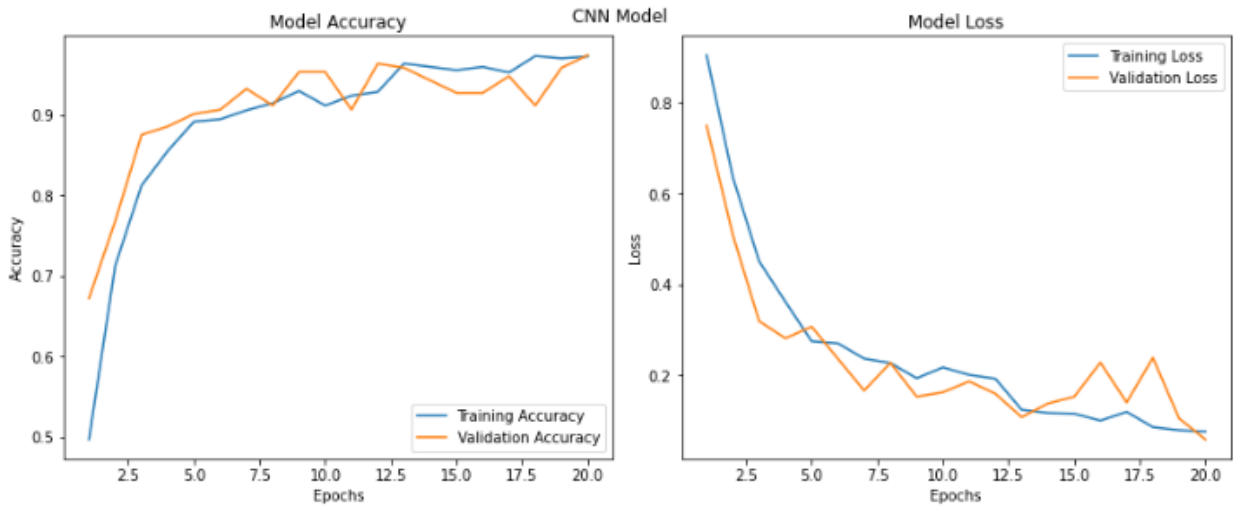

Figure 11: The model accuracy and model loss for the CNN model

We finally see how well our CNN model works at predicted which class an image belongs to. When building the function, we want the output to tell us the actual class, predicted class and the confidence of the model. This is shown below.

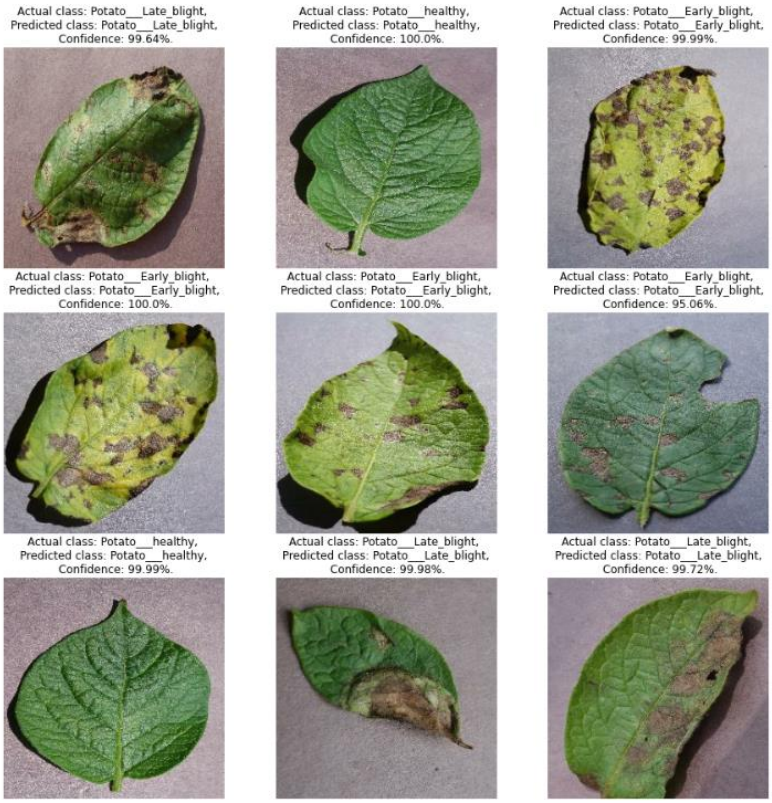

Figure 12: Prediction

Transfer Learning Models

VGG19

Similarly, to the CNN, we mount the drive and import the necessary libraries, which are the same libraries including other applications of tensorflow which helps as it loads the model for us rather than building it, step by step in CNN. Again, the constants need to be defined where the batch size, channels and epochs are the same however the image size is different. As mentioned in chapter 3, the image size is now 224, but in the CNN, it was 256. Just to make sure the data is correct; we define a function in which it tells us the possible outcomes and the count of each outcome. When we take now take the image shape it has changed to (32, 224, 224, 3) due to the image size being adjusted for the new models. We write the code to visualise 9 images from our dataset.

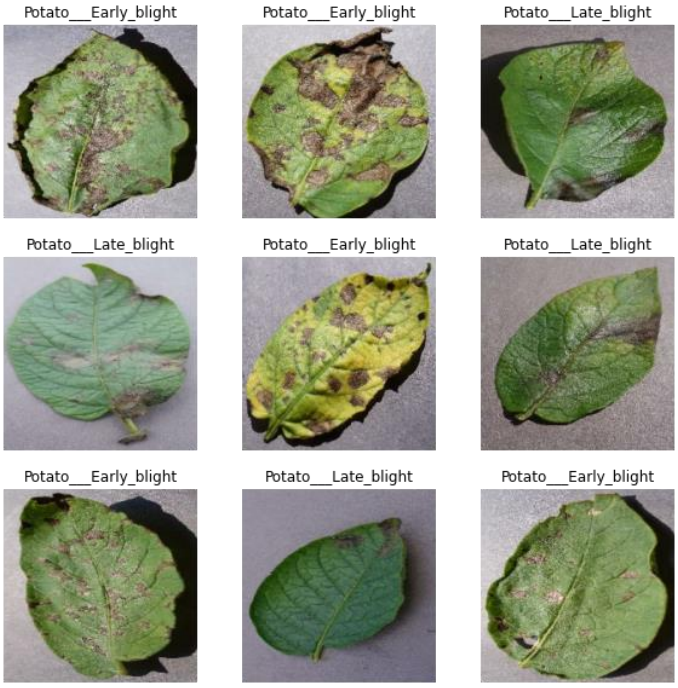

Figure 13: 9 images from the dataset

Now we split the dataset into training, validation and testing again in the same ratios of 80%, 10%, 10% correspondingly. Once again, pre-processing and augmentation. In building our model, we do not need to do much as this is a pre-built model, not something we are creating ourselves, so the necessary layers have been included and the same activation functions have been included. We look at the summary and compile the model with the Adam optimiser and categorical cross entropy for the loss where the accuracy is also measured. The model is run with the same variables as the CNN model and after 20 epochs the model ends. This produces a test accuracy of 97.7% and a loss of 0.08556. We can plot the graph of accuracy against epochs and loss against epochs for the model of VGG19 for both the training and validation data.

Figure 14: The model accuracy and model loss for the VGG19 model

We'll look at how well the model is at predicting images, giving us an output in the same form which includes, providing us with the actual class, predicted class and the confidence of the model.

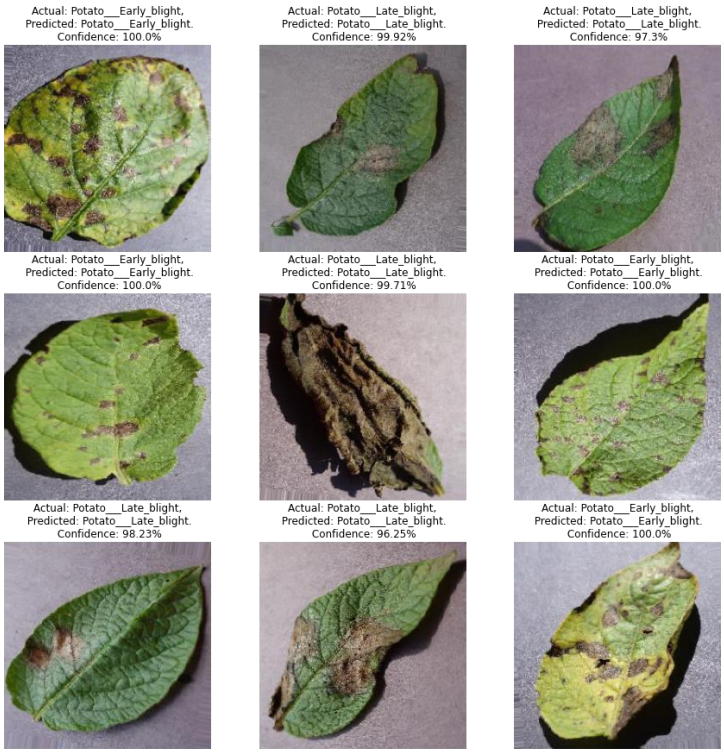

Figure 15: Prediction of VGG19

As explained for VGG19, the code works exactly the same for both InceptionV3 and ResNet50 however the only change is that the imported model will change to the model we are applying the dataset to.

InceptionV3 (GoogLeNet)

This produces a test accuracy of 94.0% and a loss of 0.17685. We can plot the graph of accuracy against epochs and loss against epochs for the model of InceptionV3 for both the training and validation data.

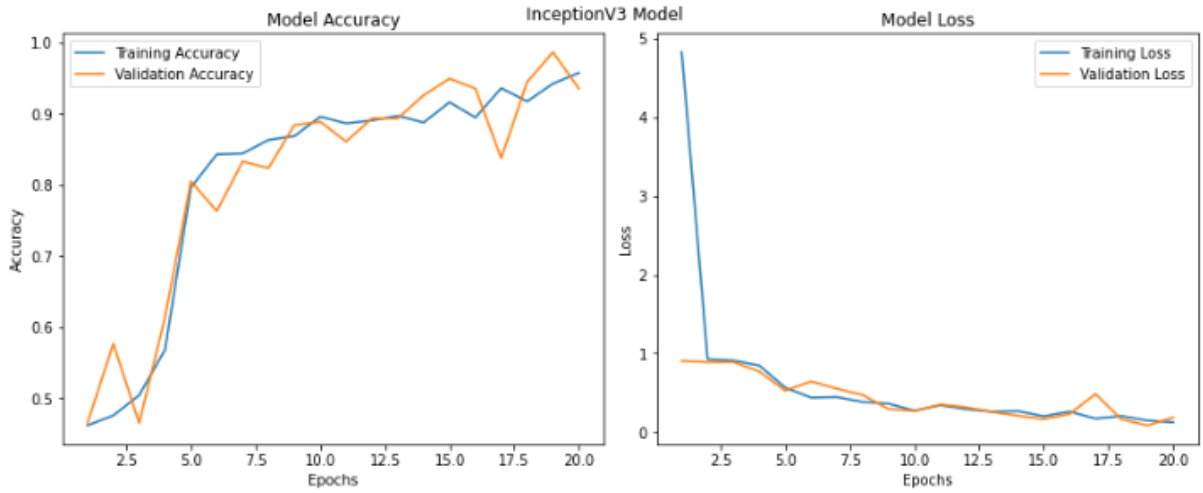

Figure 16: The model accuracy and model loss for the InceptionV3 model

We'll look at how well the model is at predicting images, giving us an output in the same form which includes, providing us with the actual class, predicted class and the confidence of the model.

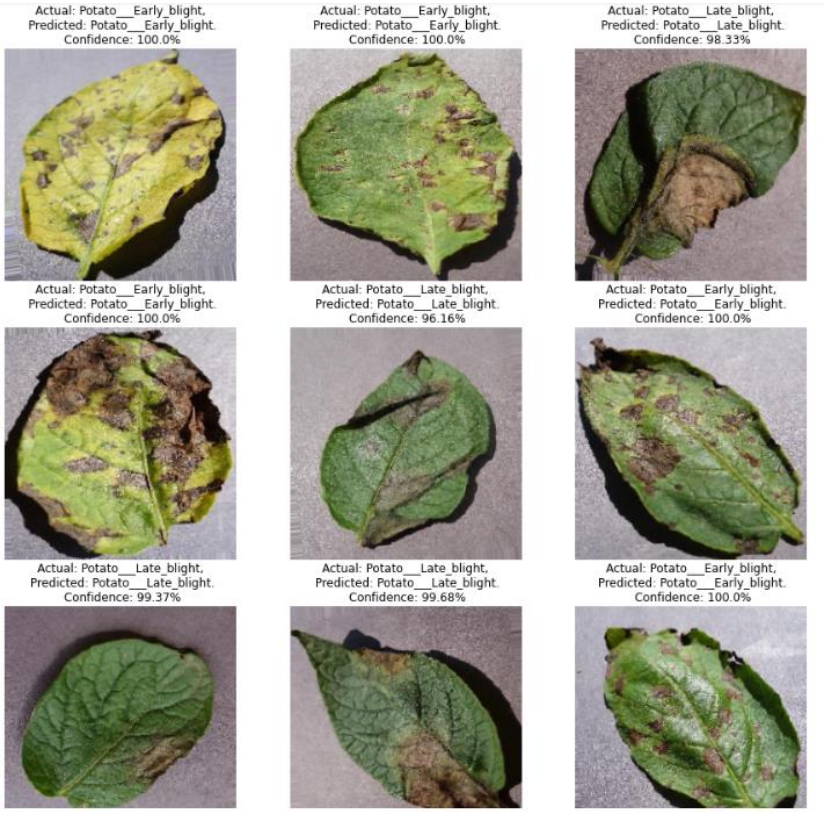

Figure 17: Prediction on Inception V3

ResNet50

This produces a test accuracy of 91.7% and a loss of 0.18201. We can plot the graph of accuracy against epochs and loss against epochs for the model of ResNet50 for both the training and validation data.

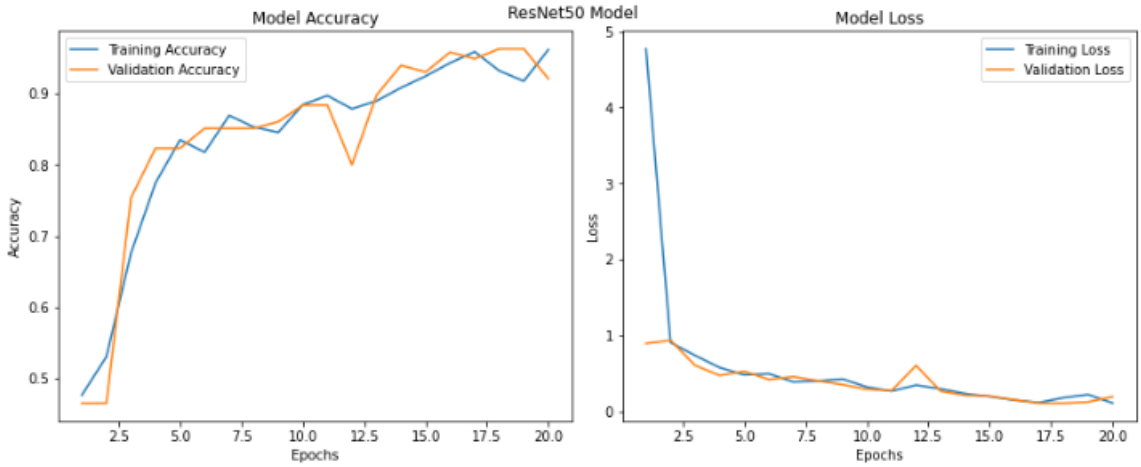

Figure 18: The model accuracy and model loss for the ResNet50 model

We'll look at how well the model is at predicting images, giving us an output in the same form which includes, providing us with the actual class, predicted class and the confidence of the model.

Figure 19: Prediction on ResNet

Comparing the transfer learning models

Test accuracy

Here we plot a bar plot to see the comparisons of the test accuracies between the transfer learning models. As you can see VGG19 stands with the highest test accuracy, with an accuracy of 97.685% and ResNet50 has the lowest test accuracy, with an accuracy of 91.667%. InceptionV3 has an accuracy of 93.981 which is in between both. All three of these models produce a very good accuracy.

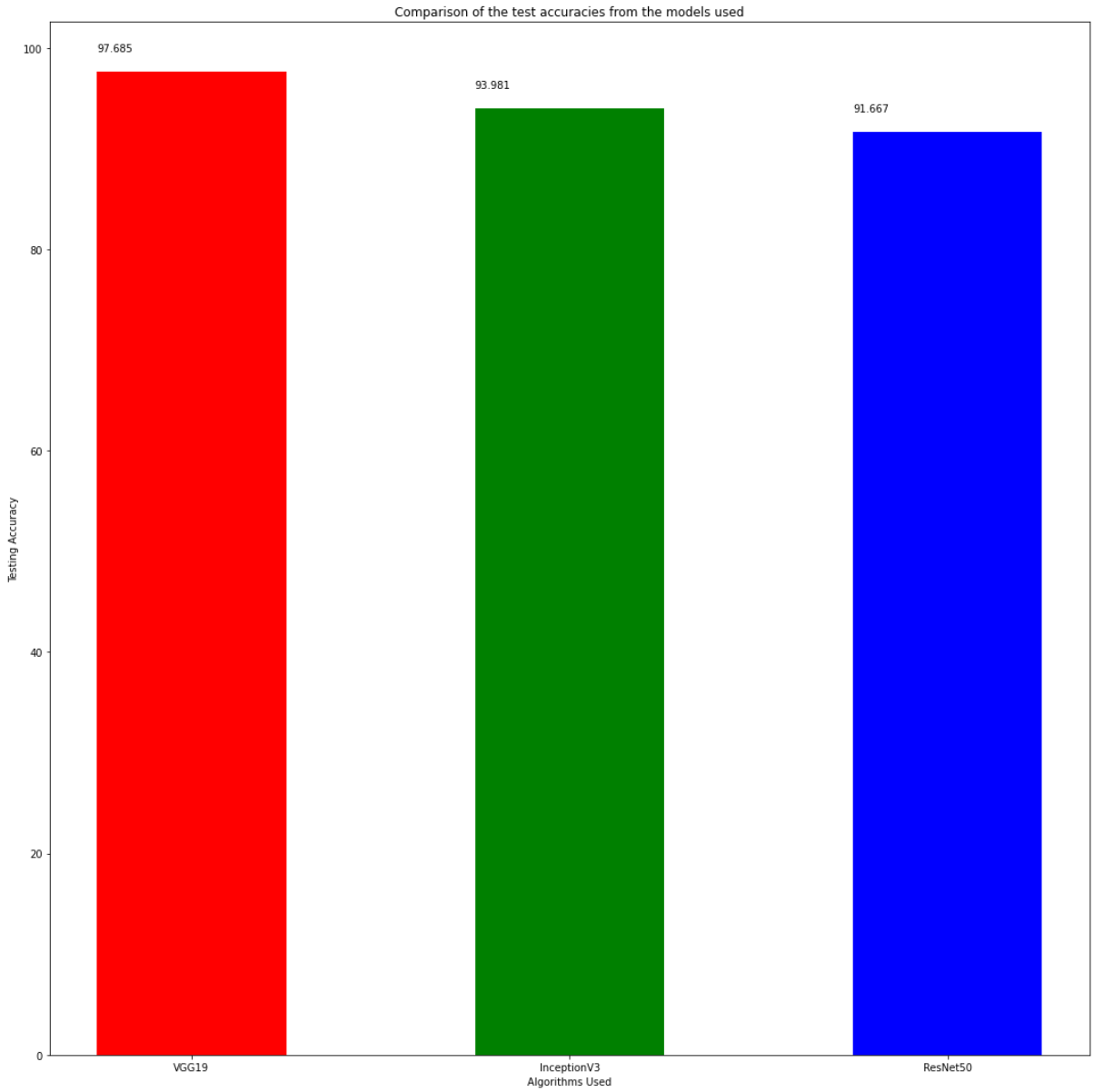

Figure 20: Comparisons of accuracies

Test loss

Another bar plot is created to see the comparisons of the test losses between the transfer learning models. In reverse order, VGG19 has the lowest test loss with a loss of 0.086, followed by InceptionV3 which had a loss of 0.177 closely followed by ResNet50 which had a loss of 0.182.

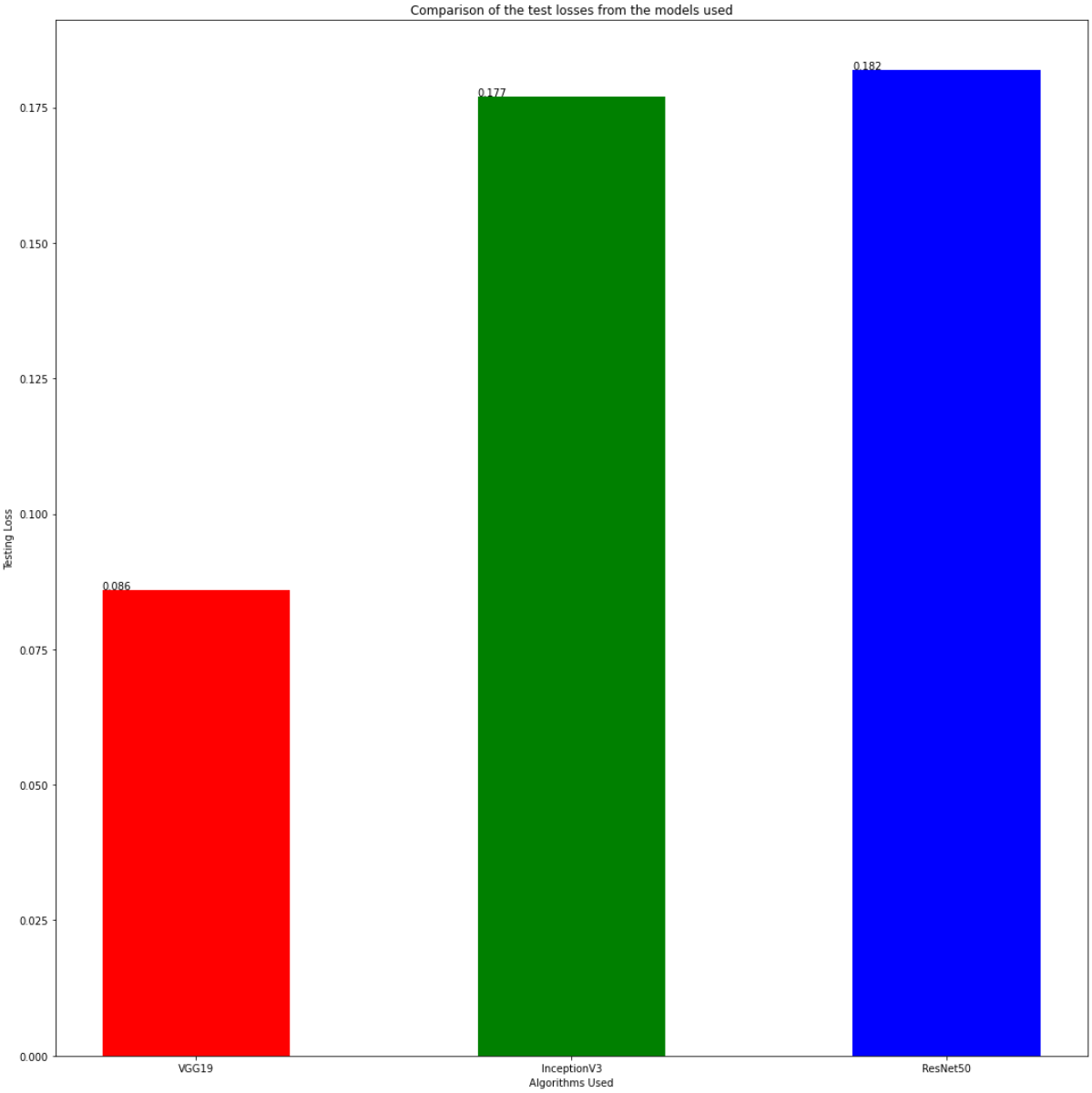

Figure 21: Comparison of test losses

Pakistan Potato Leaf Dataset

This was completed via Google Colab, Python 3.7.15 using tensorflow and keras. The optimiser present was Adam and the learning rate was kept as default. As mentioned earlier, the loss function used was categorical cross-entropy.

CNN

The code that was run for the CNN for PlantVillage is the exact same here as the model has already been built. However, when building the model, we decided to change the number of epochs for this dataset. This is because that increase the number of epochs, the more number of times the weights are changed. This means the model moves from being underfit to optimum to possibly overfit [27]. Rather using 20 epochs, we increased our epoch size to 50.

We visualise 9 images from the dataset and apply our CNN model.

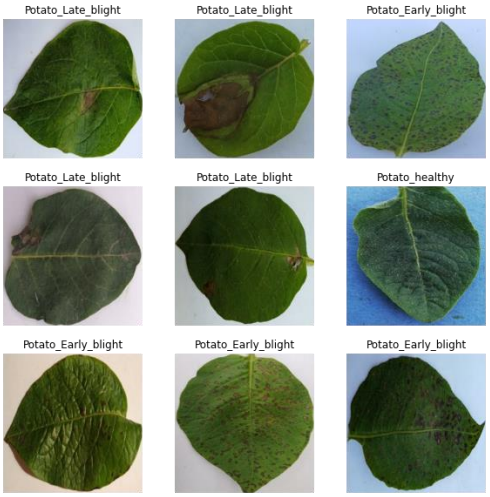

Figure 22: A visualisation of the 9 images

The split of training, validation and testing remains the same of 80%, 10% and 10% correspondingly.

We begin running our model with the train data and validation data with 50 epochs, meaning that the data passes through the neural network 50 times, forwards and backwards. We apply this to the test data where we call the scores which consists of the loss and accuracy. Our test accuracy is at 99.1% and our loss is at 0.04708.

We can plot the graph of accuracy against epochs and loss against epochs for the model of CNN for both the training and validation data as shown in the diagram below.

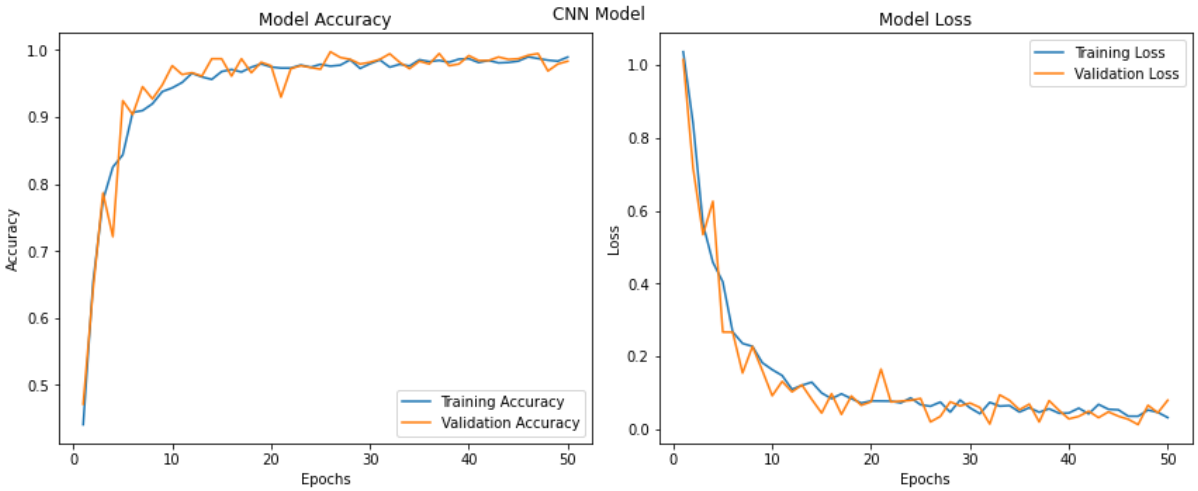

Figure 23: The model accuracy and model loss for the CNN model

We finally see how well our CNN model works at predicted which class an image belongs to. When building the function, we want the output to tell us the actual class, predicted class and the confidence of the model. This is shown below.

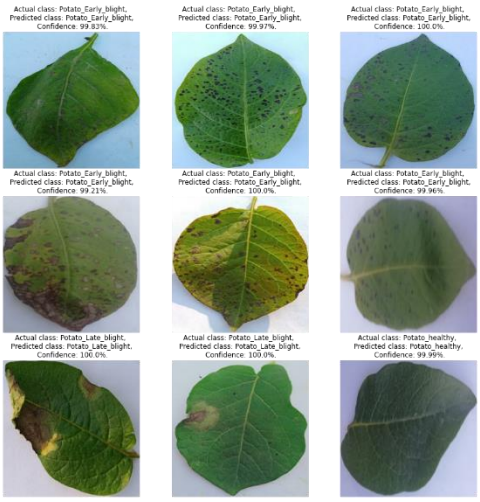

Figure 24: Prediction on CNN

Transfer Learning Models

Again, there are no changes to the code for the transfer learning for the PlantVillage, apart from the epochs being change to 50 rather than 20.

We display 9 images from the dataset

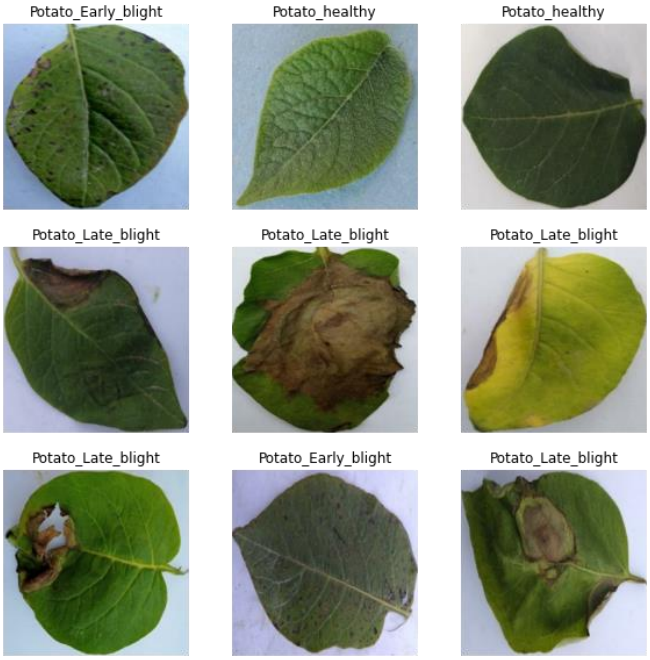

Figure 25: 9 images from the dataset

Once again split the data in the previously mention ratios.

VGG19

The test accuracy produced is 40.1% and the test loss is 1.08026.

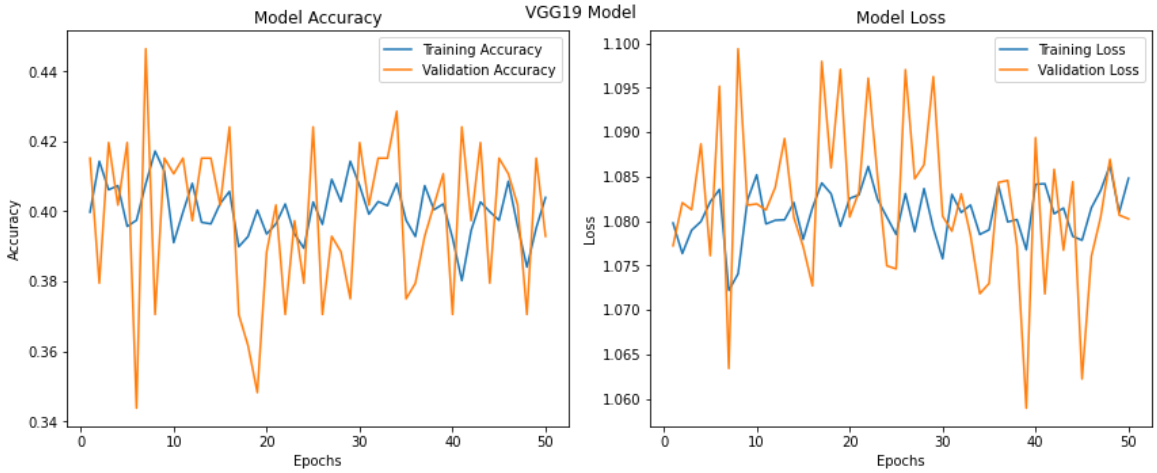

Figure 26: The model accuracy and model loss for the VGG19 model

We'll look at how well the model is at predicting images, giving us an output in the same form which includes, providing us with the actual class, predicted class and the confidence of the model.

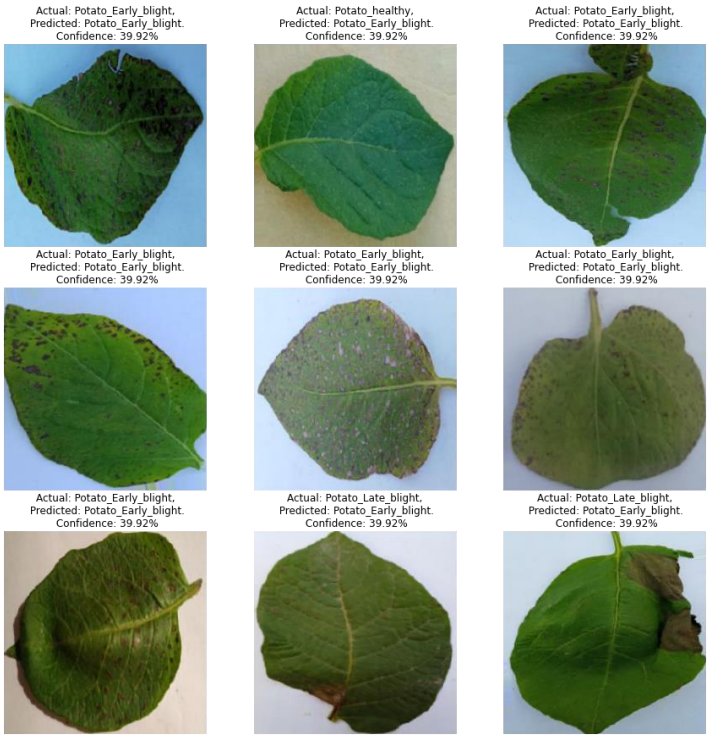

Figure 27: Prediction on VGG19

InceptionV3

The test accuracy produced here is 40.1% and the test loss is 1.0803.

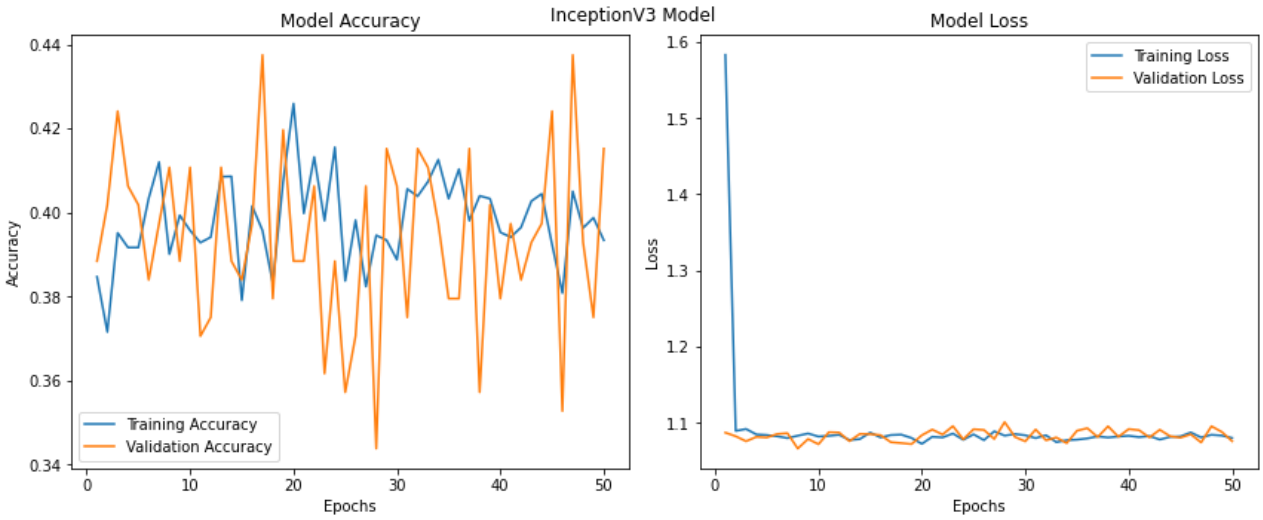

Figure 28: The model accuracy and model loss for the InceptionV3 model

The predictions are as shown below

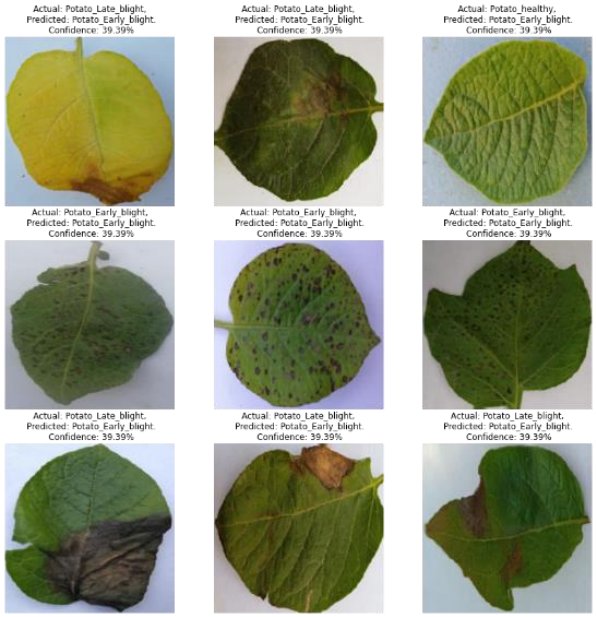

Figure 29: Prediction on InceptionV3

ResNet

The test accuracy was 40.1% and the test loss was 1.0803.

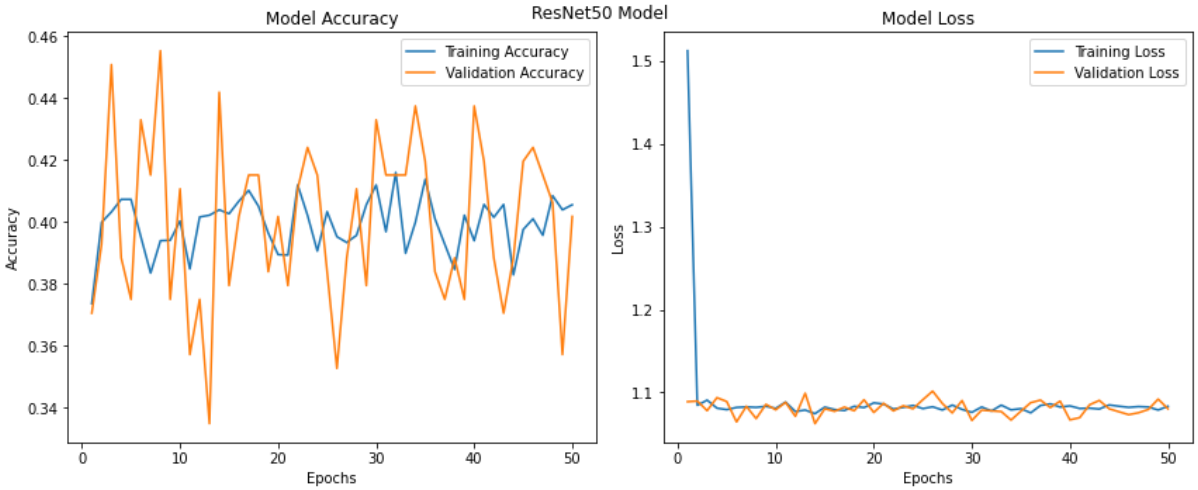

Figure 30: The model accuracy and model loss for the ResNet model

The predictions are given below.

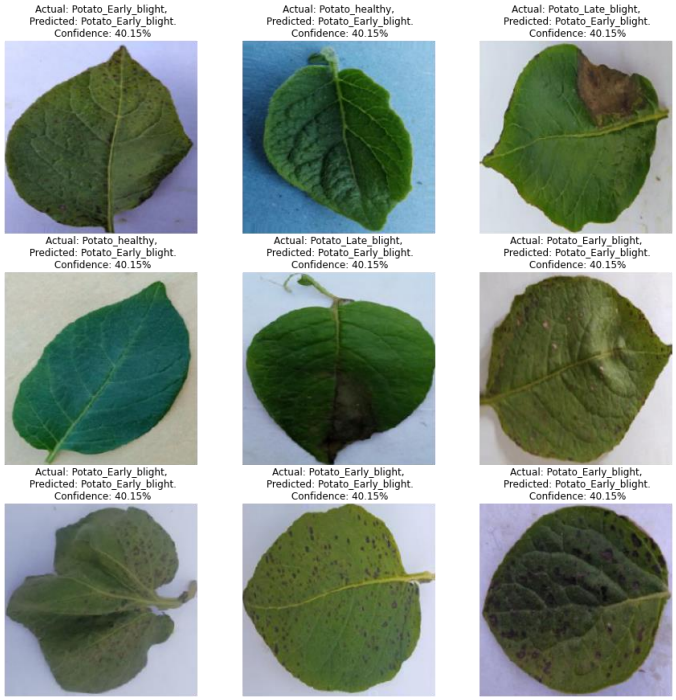

Figure 31: Prediction on ResNet

Comparing the transfer learning models

Test accuracy

Looking at the bar plot produced, correct to 3 decimal places, the test accuracy is all the same, at an accuracy of 40.098%.

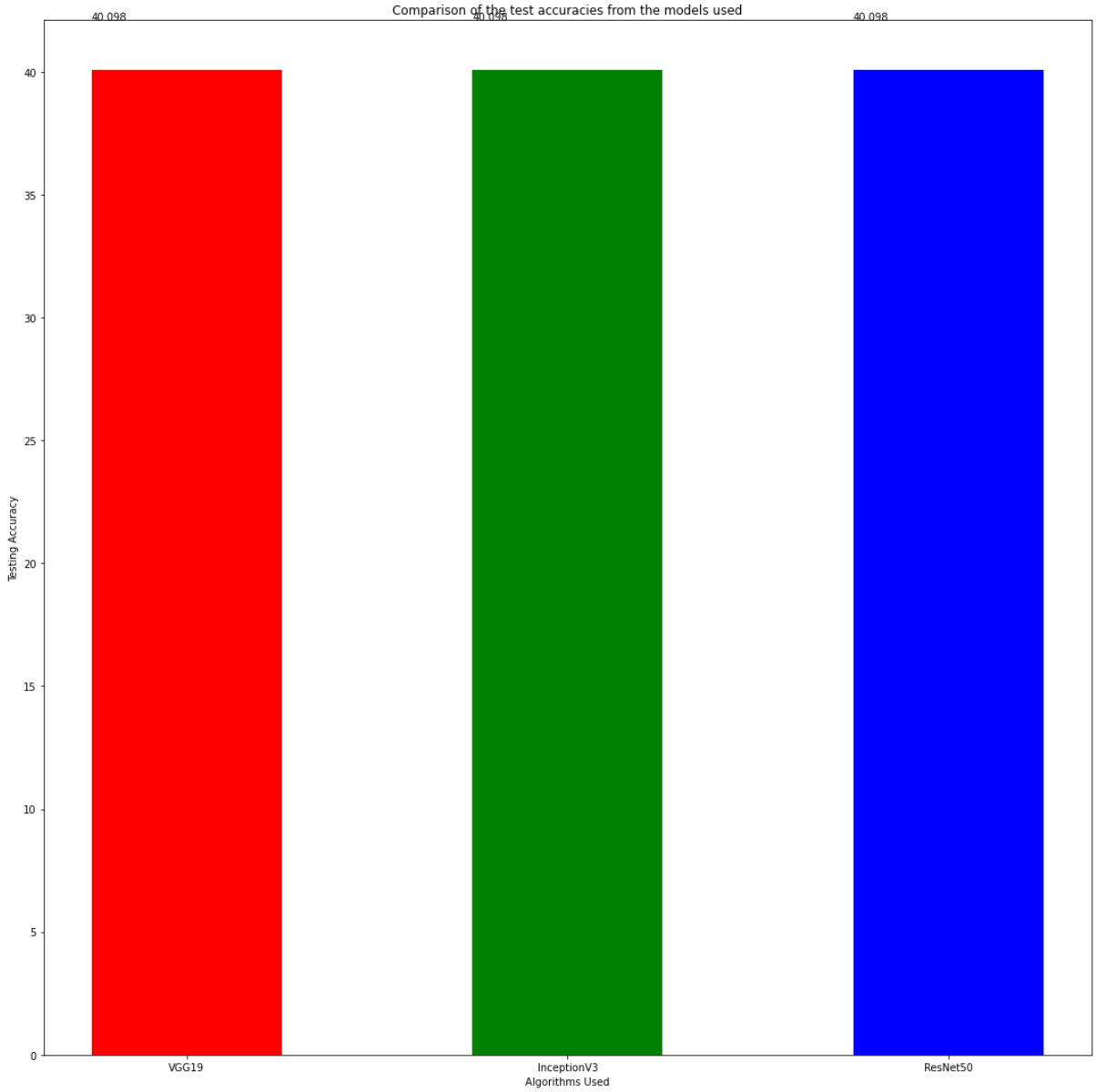

Figure 32: Comparison of test accuracies on transfer learning

Test loss

As well as the test accuracy being the same, the test loss is also the same at a loss of 1.08 correct to 2 decimal places.

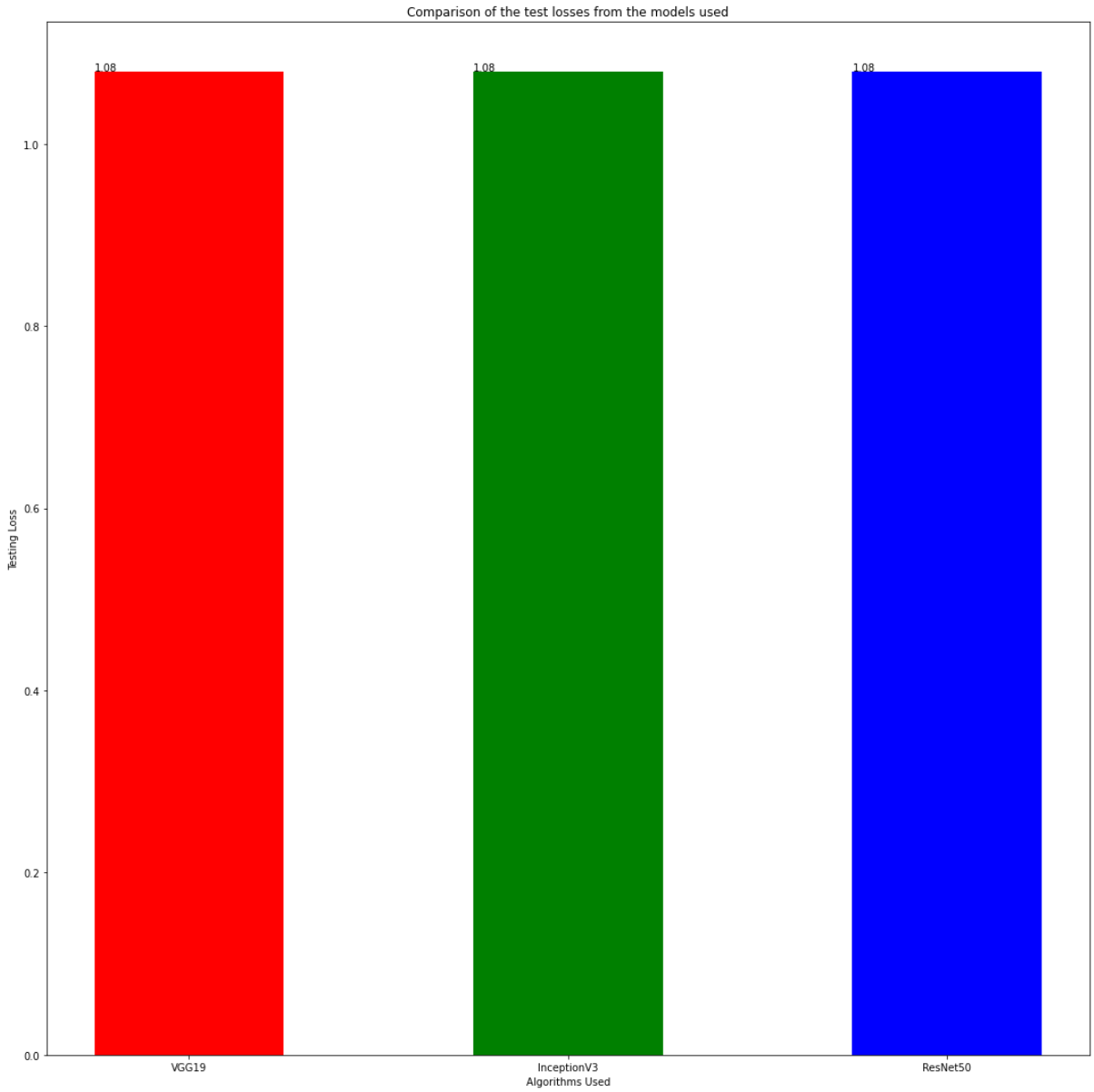

Figure 33: Comparison of losses on transfer learning models

Accuracy and loss tables for both datasets

PlantVillage dataset

Model	Accuracy (%)	Loss
CNN	99.2	0.03946
VGG19	97.7	0.08556
GoogLeNet	94.0	0.17685
ResNet	91.7	0.18201

Table 4: Accuracy and losses of PlantVillage

Pakistan Potato Leaf dataset

Model	Accuracy (%)	Loss

CNN	99.1	0.04708
VGG19	40.1	1.08026
GoogLeNet	40.1	1.0803
ResNet	40.1	1.0803

Table 5: Accuracy and losses of Pakistan Potato Leaf Dataset

Discussion

The table above summarises the results of the accuracies and losses from the models applied to both datasets. As you can see in both cases the CNN model performed the best with the highest accuracy. The model that performed the best for PlantVillage was CNN. The model that performed the worst in PlantVillage was ResNet. The model that performed the best for Pakistan Potato Leaf Dataset was CNN. The model that performed the worst in PlantVillage was VGG19, GoogLeNet and ResNet.

This is quite astonishing as we expect those 3 models to perform the best as they already pre-built where it would have taken a very long time to come up with that network. However, with that being said, all 4 models perform very well, all producing an accuracy above 91%. Since the dataset was smaller in comparison to the Pakistan potato leaf dataset, using augmentation would sway the accuracy to a much higher value. This is because using 1 image with as many rotations and translations is still identified as 1 image but using many images and augmenting them makes the model learn much more.

The accuracy of the pre-trained models being low could be due to computational error too. A way we could overcome this low error is to adjust the learning rate so the model can learn the output more accurately.

Personal Reflections

This process has helped me to educate myself with how the applications of CNNs work and how other transfer learning models can be used to apply the same dataset for prediction. It has helped in understanding how much importance the agriculture industry has on every day life and how well image classification can help produce maximum yield. This project has helped me to understand the vital components needs for image classification that will affect the accuracy of a model.

Future work

With time being a limited factor, we could have investigated other models, apart from the four models we have already looked at for the use of comparison purposes seeing which would provide the optimum accuracy for prediction. As the run time of these models was also not short, applying more epochs could have increased the accuracy for prediction, for the Pakistan potato leaf dataset in particular.

We could have also compared how the models work with both augmented and non-augmented data, to see if the models would be better or not at prediction. Using cross-

validation could be another factor to take into consideration for the future as it would aid in reducing the bias, but again this would very time consuming.

Using dropout layers will also helped in preventing overfitting and it helps in randomly decreasing the number of interconnected neurons within a network. The usage of batch normalisation would have bettered our model as it facilitates learning by increasing the speed of training and incorporating higher learning rates.

For this sort of research work, always use Google Colab. This is because of the pre-exisiting GPU which helps the model learn much quicker rather than work on a PC's CPU. For example, the ResNet model for the Pakistan potato leaf dataset took a maximum of 15 minutes to run on Google Colab, however on JuPyter Notebook, it took approximately 80 minutes. Another alternative is to invest into a computer with much higher specifications, making the runtime much more efficient.

We could even make this into a mobile application where farmers would just scan an image of a potato leaf and on the application, it would provide the result of if the potato would diseased or not. This would be very time efficient as there would be not need to collect images and run the images through each model. As well as this, the scanned images could be stored in a database in which whenever these images need to be used for any other purpose, they can be accessed very easily.

Conclusion

In conclusion, I believe that for businesses of small structure, using the CNN model would be the best due to the outcome of the accuracy, and for larger businesses, using my output CNN would be the best. On the other hand, from research and the papers I had read I was able to understand that pre trained models can produce the highest accuracies. Depending on how free flowing the money is, CNN would be the most reliable and cost-effective model.

References

[1] - Britannica, T. Editors of Encyclopaedia (2017, June 22). blight. Encyclopedia Britannica. https://www.britannica.com/science/blight

[2] - En.wikipedia.org. 2022. *Alternaria solani - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Alternaria_solani>

[3] - En.wikipedia.org. 2022. *Phytophthora infestans - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Phytophthora_infestans>

[4] - U. P. Singh, S. S. Chouhan, S. Jain and S. Jain, "Multilayer Convolution Neural Network for the Classification of Mango Leaves Infected by Anthracnose Disease," in IEEE Access, vol. 7, pp. 43721-43729, 2019, doi: 10.1109/ACCESS.2019.2907383.

[5] - 2022. [online] Available at: <https://www.kaggle.com/datasets/arjuntejaswi/plant-village - 6>

[6] - Guan, S.; Kamona, N.; Loew, M. Segmentation of Thermal Breast Images Using Convolutional and Deconvolutional Neural Networks. In Proceedings of the 2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, USA, 9–11 October 2018. [**Google Scholar**]

[7] - Gehlot, S.; Gupta, A.; Gupta, R. SDCT-AuxNet: DCT Augmented Stain Deconvolutional CNN with Auxiliary Classifier for Cancer Diagnosis. *Med. Image Anal.* **2020**, *61*, 101661. [**Google Scholar**]

[8] - Lu, J; Tan L; Jiang H; [*Lu, J.; Tan, L.; Jiang, H. Review on Convolutional Neural Network (CNN) Applied to Plant Leaf Disease Classification. *Agriculture* **2021**, *11*, 707. https://doi.org/10.3390/agriculture11080707]

[9] - [Databricks. 2022. *What is a Convolutional Layer? - Databricks*. [online] Available at: https://databricks.com/glossary/convolutional-layer#:~:text=A%20convolution%20converts%20all%20the,convolutional%20layer%20is%20a%20vector.

[10] - GeeksforGeeks. 2022. *CNN | Introduction to Pooling Layer - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>]

[11] - Agarap, A.F. (2018) 'Deep Learning using Rectified Linear Units (ReLU)'

[12] - Chen, B. (2021) "7 popular activation functions you should know in Deep Learning and how to use them with Keras and TensorFlow 2," *Towards Data Science*, 3 January. Available at: https://towardsdatascience.com/7-popular-activation-functions-you-should-know-in-deep-learning-and-how-to-use-them-with-keras-and-27b4d838dfe6.

[13] - Bala, P.C. (no date) *Pinecone*. Available at: https://www.pinecone.io/learn/softmax-activation/#:~:text=The%20softmax%20activation%20function%20transforms,distribution%20over%20the%20input%20classes

[14] - (2020) *Softmax Activation Function Explained*, 18 June. Available at: https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60

[15] - Soni, P., 2022. *Data augmentation: Techniques, Benefits and Applications | Analytics Steps*. [online] Analyticssteps.com. Available at: <https://www.analyticssteps.com/blogs/data-augmentation-techniques-benefits-and-applications>

[16] Seldon. 2022. *Transfer Learning for Machine Learning*. [online] Available at: <https://www.seldon.io/transfer-learning#:~:text=Transfer%20learning%20means%20taking%20the,to%20solve%20a%20specific%20task.>

[17] - "ImageNetVGGNet, ResNet, Inception, and Xception with Keras" (2017) *pyimagesearch*, 20 March. Available at: https://pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/.

[18] - Frossard, D. (2016) *cs toronto edu*, 17 June. Available at: https://www.cs.toronto.edu/~frossard/post/vgg16/.

[19] - [1409.1556] Very Deep Convolutional Networks for Large-Scale Image Recognition (arxiv.org)]

[20] - 2022. [online] Available at: https://uk.mathworks.com/help/deeplearning/ref/googlenet.html

[21] - Kurama, V. (2019) *PaperspaceBlog*. Available at: https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/#:~:text=The%20GoogleNet%20Architecture%20is%2022,the%20global%20average%20pooling%20layer.

[22] - Medium. 2022. *Understanding and Coding a ResNet in Keras*. [online] Available at: <https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33#:~:text=The%20ResNet%2D50%20model%20consists,over%2023%20million%20trainable%20parameters.>

[23] - [He K, Zhang X, Ren S, Sun J, editors. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition; 2016]

[24] - 2022. [online] Available at: https://www.linkedin.com/company/plantvillage/

[25] - Rashid, J.; Khan, I.; Ali, G.; Almotiri, S.H.; AlGhamdi, M.A.; Masood, K. Multi-Level Deep Learning Model for Potato Leaf Disease Recognition. *Electronics* **2021**, *10*, 2064. https://doi.org/10.3390/electronics10172064]

[26] - Koech, K.E. (October 2, 2020) "Cross-Entropy Loss Function," *Towards Data Science*. Available at: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

[27] - Sharma, S. (2017) "Epoch vs Batch Size vs Iterations," *Towards Data Science*, 23 September. Available at: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

[28] - : N. T. Sinshaw, B. G. Assefa and S. K. Mohapatra, "Transfer Learning and Data Augmentation Based CNN Model for Potato Late Blight Disease Detection," 2021

International Conference on Information and Communication Technology for Development for Africa (ICT4DA), 2021, pp. 30-35, doi: 10.1109/ICT4DA53266.2021.9672243.

[30] - D. Oppenheim, G. Shani, O. Erlich, and L. Tsror, “Using deep learning for image-based potato tuber disease detection,” The American Phytopathological Society, vol. 109, no. 6, p. 1083–1087, dec 2019