MID SEMESTER I PROJECT REPORT

Potato Leaf Disease Classification Using Deep Learning

4EE32: Project Stage I

By

Project Group No: 7

Galav Bhatt (18EE027)

Under the guidance of

Prof. Dipan A Parikh

Electrical Engineering Department

Birla Vishvakarma Mahavidyalaya (Engineering College)

(An Autonomous Institute)

Vallabh Vidyanagar: 388120

(Month & Year)

Birla Vishvakarma Mahavidyalaya (Engineering College)

An Autonomous Institution

Vallabh Vidyanagar 388120

TABLE OF CONTENTS

Contents

Chapter 1: Project Orientation

The potato (Solanum tuberosum) is an herbaceous annual that grows up to 100 cm (40 inches) tall and produces a tuber - also called potato - so rich in starch that it ranks as the world's fourth most important food crop, after maize, wheat and rice. As the potato plant grows, its compound leaves manufacture starch that is transferred to the ends of its underground stems. The stems thicken to form a few

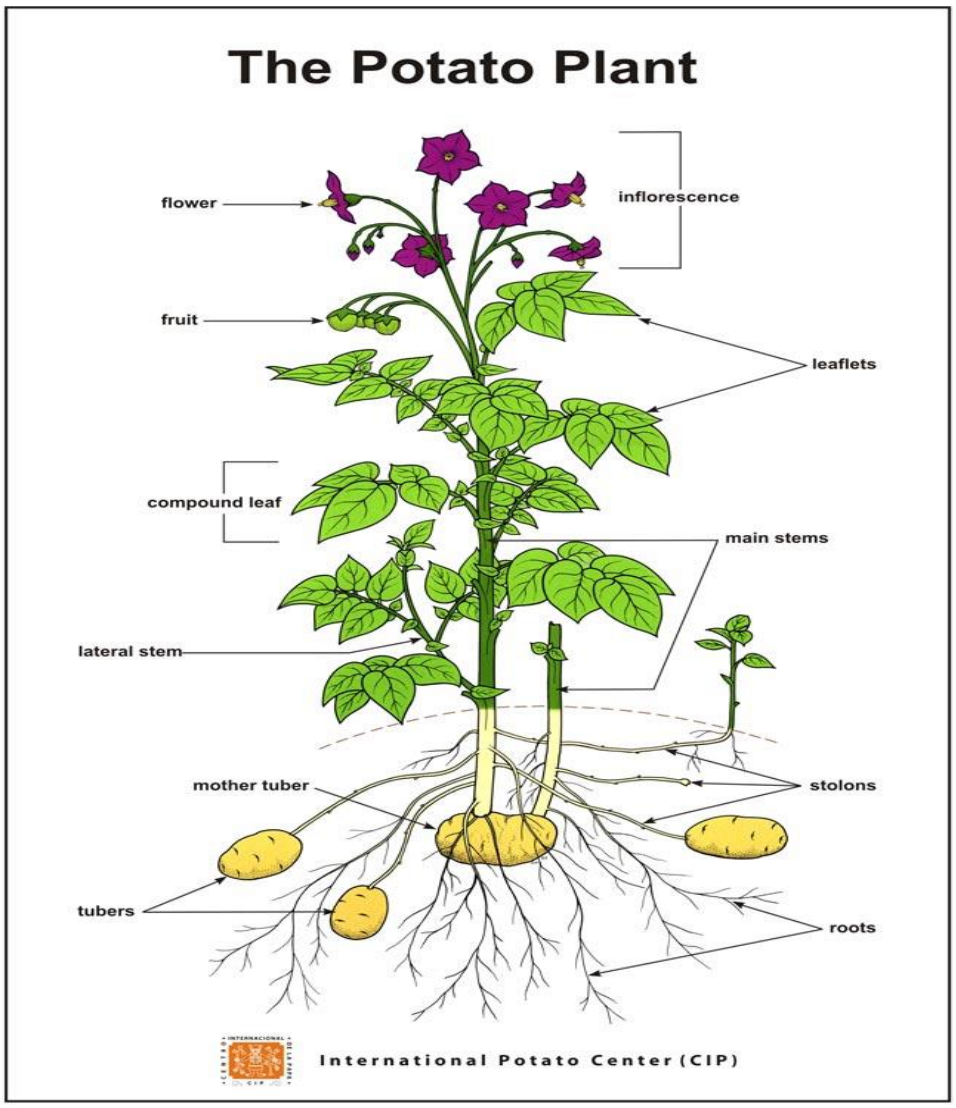


Figure 1: Potato Plant

or as many as 20 tubers close to the soil surface. The number of tubers that actually reach maturity depends on available moisture and soil nutrients. Tubers may vary in shape and size, and normally weigh up to 300 g each. The cut potato piece or "seed" piece provides the new sprout or seedling with nourishment from its supply

of stored starch. After you've planted a seed piece, it usually takes one to two weeks for the main stem and first leaves to appear above ground. The root system develops quickly and begins to absorb nutrients as the food supply in the seed piece is used up.

Nutritive Value Potato in India as per Central Potato Research Institute (CPRI), Shimla is given below:

Constituents	Weight (grams)
Water	74.70
Carbohydrates	22.60
Protein	1.60
Fat	0.40
Fiber	0.10
Minerals	0.60
Minerals and Vitamins Constituent in Potato	
Calcium	7.7
Copper	0.15
Iron	0.75
Magnesium	24.2
Phosphorus	40.3
Potassium	568.0
Sodium	6.5
Vitamin C.	14.0 – 25.0
Thiamin 10.	0.18
Riboflavin	0.01-0.07
Niacin 12.	0.4 –3.1
Total Folate	5.0-35.0
Pyridoxine	0.13-0.25

The top, leafy part of the plant puts on a lot of growth in the first four to five weeks after planting. Then the main stem of the plant stops growing and produces a flower bud. When that happens, the plant will have as many leaves as it will ever have. With proper sunshine, the leaves eventually produce more food than the plant needs, and the excess energy is channeled downward to be stored in the "tubers" -- thick, short, underground stems -- which we simply call potatoes.

Chapter 2: Project Introduction

Potato is the most versatile crop which contributes to about 28.9% of total agricultural crop production in India. Potato is the fourth largest agricultural food crop in the world after maize, wheat, and rice. India is the 2nd largest country in the production of potatoes which produces 48.5 million tons every year. However, the crops like potato also face the issue of disease caused due to microorganism and its adverse effects.

Diseases have adverse effects on plant and agricultural lands. The main causes of these diseases are microorganisms, genetic disorders, and infectious agents like bacteria, fungi, and viruses. Fungi and bacteria are mainly responsible for potato leaf diseases. Late blight and early blight are fungal diseases.

Early blight of potato is caused by **the fungal pathogen Alternaria solani**. The disease affects leaves, stems and tubers and can reduce yield, tuber size, storability of tubers, quality of fresh-market and processing tubers and marketability of the crop. Here, Circular and pale brown spots appear on the leaves the spot subsequently get covered with grayish fungal growth. Further, the tissues turn necrotic and concentric ring of raised and depressed necrotic zones are formed, giving the typical target board appearance. Photosynthesis gets completely distorted due to it.

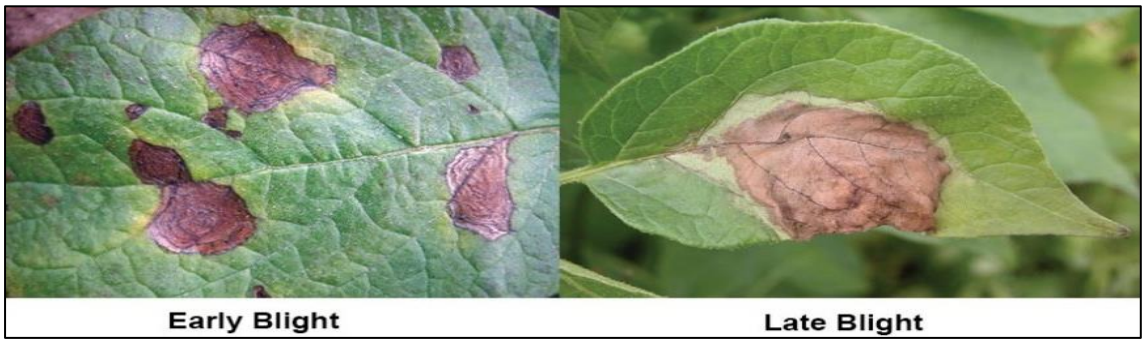


Figure 2 Early Blight vs Late Blight

Late blight caused by **the fungus Phytophthora infestans** is the most important disease of potato that can result into crop failures in a short period if appropriate control measures are not adopted. Usually, it seems at the stage of flowering in the potato plant brown spots appear on the leaves and stem. The spots grow fast

if there is favorable condition like low temperature and high humidity. The most prominent initial signs of late blight are brown spots or disease spots on plant stems. The Key difference between both the types of disease are given below:

Parameter	Early Blight of Potato	Late Blight of Potato
Definition & Infectious Agent	early blight of potato is a disease caused mainly by the fungus Alternaria solani	late blight of potato is a disease caused by the oomycete Phytophthora infectants.
Occurrence	Appears First	Appears Second
Preferred Climate	Warm Temperature and High Humidity	Cool and Moist Weather
Symptoms	Small Dark Spots on the lower and older leaves of the plant	Dark and irregular spots with a drenching appearance

The manual classification of such disease requires domain knowledge and some sort of expertise in the field as it depends on certain technical parameters. Due to such disease the average annual loss is estimated to 15% of total production in the country and also the reduced tuber size decreases the market value of this crop. Hence, in order to automate the process of classification we will use advance computer methodology know as deep learning.

The Deep Learning technique like Convolutional Neural Network can be applied for the image classification task of such disease affected plant, which will help farmers and agriculturist to take precautionary steps in order to control the disease.

The model can also be used by agriculture academicians, farmers and students to have the correct identification of plant disease. After, Identification control steps can be taken like follow a complete and regular foliar fungicide spray program, Practice good killing techniques to lessen tuber infections, sallow tubers to mature before digging, dig when vines are dry, not wet, and avoid excessive wounding of potatoes during harvesting and handling. Plow under all plant debris and volunteer potatoes after harvest.

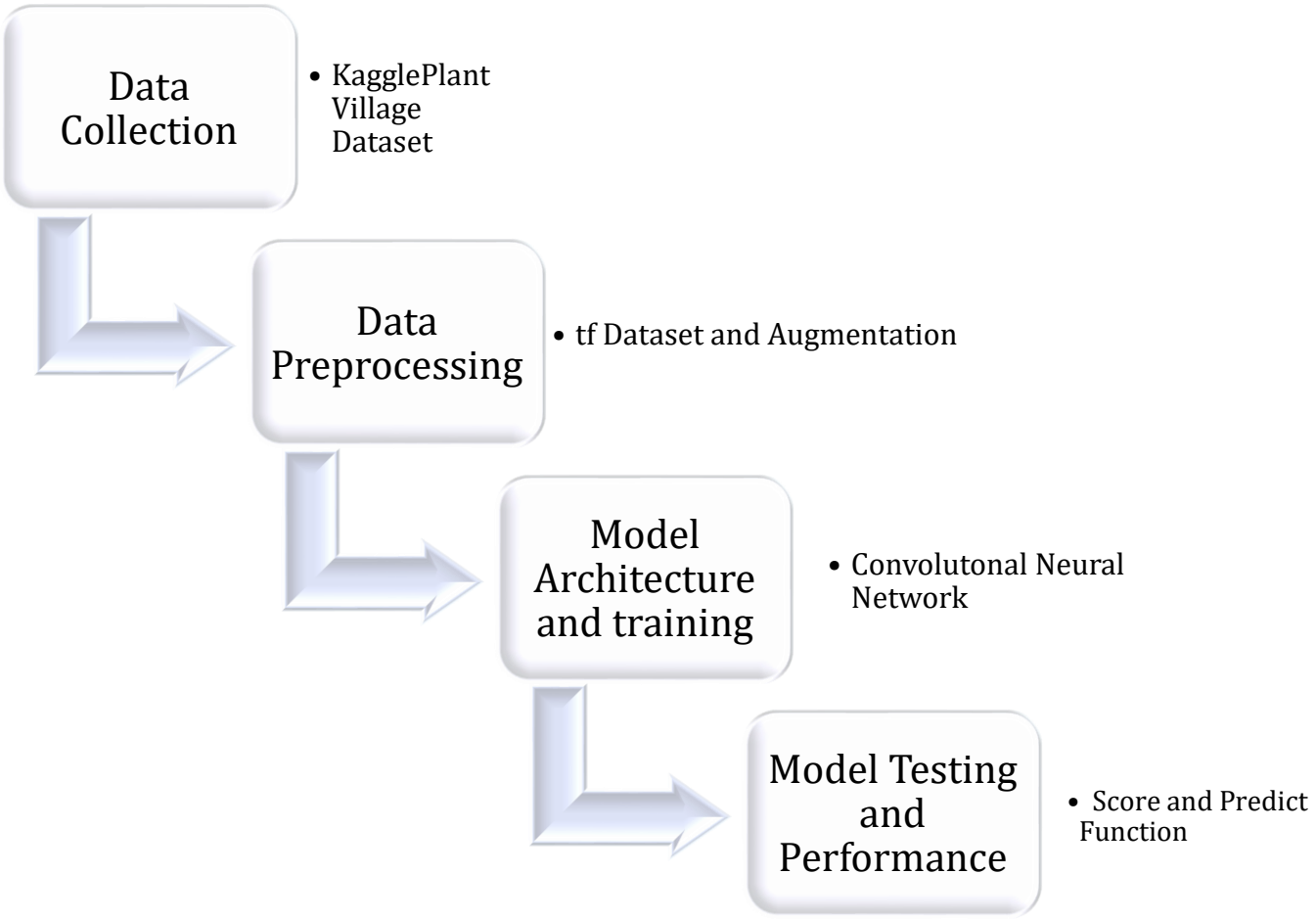
Chapter 3: Project Definition, Identification and Objective

In the age of computing and with the advancement in agricultural technology and the use of artificial intelligence in diagnosing plant diseases, it becomes important to make pertinent research to sustainable agricultural development. Various diseases like early blight and late blight immensely influence the quality and quantity of the potatoes and manual interpretation of these leaf diseases is quite time-taking and cumbersome. As it requires tremendously a good level of expertise, efficient and automated detection of these diseases in the budding phase can assist in ameliorating the potato crop production. Previously, various models have been proposed to detect several plant diseases. In this project a model is presented that uses Convolutional Neural Network (CNN) and fine-tuning (transfer learning) to extract the relevant features from the dataset. Then, with the help of multiple classifiers

The identification of project is held by taking into consideration the impact of plant diseases in the agricultural crop production. Farmers spend billions of dollars are on disease management, often without adequate technical support, resulting in poor disease control, pollution and harmful results. In addition, plant disease can devastate natural ecosystems, compounding environmental problems caused by habitat loss and poor land management and weather conditions. This disease occurred during 2013-14 crop season and a roving survey was carried out from July 2013 to February 2014 in the states of Uttar Pradesh, West Bengal, Punjab, Karnataka covering about 65% potato area. The technical support may help the farmers to improve and take the precautionary measures beforehand the entire crop gets distorted.

The Main Objective of the project is to help the farmers in the identification of Early Blight and Late Blight disease in the potato crop production. The early detection of such disease can lead to save a lot of waste, prevent the economic loss and a suitable treatment can be given to the crop. Certain precautionary measures can be taken after the accurate classification of the respective disease. Overall crop production rate will have a significant positive impact due to early classification of such disease.

Chapter 4: Project Plan



The Entire Project includes four steps which involves Data Collection, Data Cleaning & Preprocessing, Model Building and performance evaluation & testing at the end. Data Collection is a crucial step because the quality of our model will depend upon the data quality. Data cleaning involves the removal of unnecessary and noisy data which will be negligible in our case. Preprocessing indicates the distribution and shuffling of data in accordance for the model training. The Model building step involves the model selection and its parameter tuning in order to get the best results out of it. After, the training of model the final step involves its performance evaluation and testing on our test dataset.

Chapter 5: Literature Review

Potato Leaf Diseases Detection Using Deep Learning – (IEEE Xplore Publication)

Citation: - D. Tiwari, M. Ashish, N. Gangwar, A. Sharma, S. Patel and S. Bhardwaj, "Potato Leaf Diseases Detection Using Deep Learning," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 461-466, doi: 10.1109/ICICCS48265.2020.9121067.

The Publication on Potato Leaf Diseases Detection Using Deep Learning depicts about the importance of classification of plant diseases and its impact on the agricultural crop production of our country. Initially the significance of potato crop is shown like Potato is the most versatile crop which contributes to about 28.9% of total agricultural crop production in India. Potato is the fourth largest agricultural food crop in the world after maize, wheat, and rice. Further, the related work of different scientists and researchers were shown along with their final results like P Badar et. have used an approach of segmentation using K Means Clustering on various features of Potato leaf image samples such as color, texture, area, etc. and applied Back Propagation Neural Network algorithm for identifying and classifying the disease in the leaf image in which they obtained a classification accuracy of 92%.

The dataset collected by the authors to execute their own model were from Kaggle Plant Village Dataset. The machine over which this research has been accomplished is having an NVIDIA GeForce N16VGMR1 graphic card. VGG19 is a CNN based approach proposed by K. Simonian and A. Zisserman. The dataset used to train this model as ImageNet, which contains more than 15 million labeled high-resolution images belonging to 22000 categories. After evaluating results from all the mentioned approaches, we found that CNN along with logistic regression gave the state -of-the art solution. After the feature extraction from the convolutional neural network the author used multiple models to classify the images in classification part. Multiple classifiers namely KNN, SVM, Neural Network and logistic regression are used for classification. Among which Logistic Regression gives the state-of-the-art solution with a classification accuracy of 97.7%.

Receiver operating characteristic (ROC) curve were plotted by taking into consideration different models. The author finally concluded that the concept of

transfer learning and have developed an automated system to diagnose and classify diseases in the potato leaves like early blight, late blight and healthy with a **novel solution achieving classification accuracy of 97.8% over the test dataset**. The future scope of the project is that disease detection needs a lot of expertise so it would be very beneficial if we could implement this system on the smartphone in which farmers can click a picture of the leaf and send it to the server. The server will automatically identify and classify the type of disease and send results along with prescribed medicines back to the smartphone.

TensorFlow

TensorFlow is a free and open-source software library for machine learning developed by Google Brain Team with the help of Python, C++ and CUDA. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming.

In the TensorFlow website, I have gone through the explanation of Convolution Neural Network where they demonstrated the CNN with the CIFAR 10 Dataset (The CIFAR10 dataset contains 60,000 color images in 10 classes, with 6,000 images in each class). Further, I went through the Image classification section where they implemented the flower classification following the complete Machine Learning cycle and implemented the CNN for the better result.

Transfer Learning section involves the training of model using pre trained model known as MobilenetV2 such training is called as Transfer Learning. The Data Augmentation section includes implementation of data augmentation techniques in order to increase the volume of the dataset and diverse number of images on

the basis of contrast, position, opacity etc. further, I have gone through the procedure to save the model and load it for the deployment purpose.

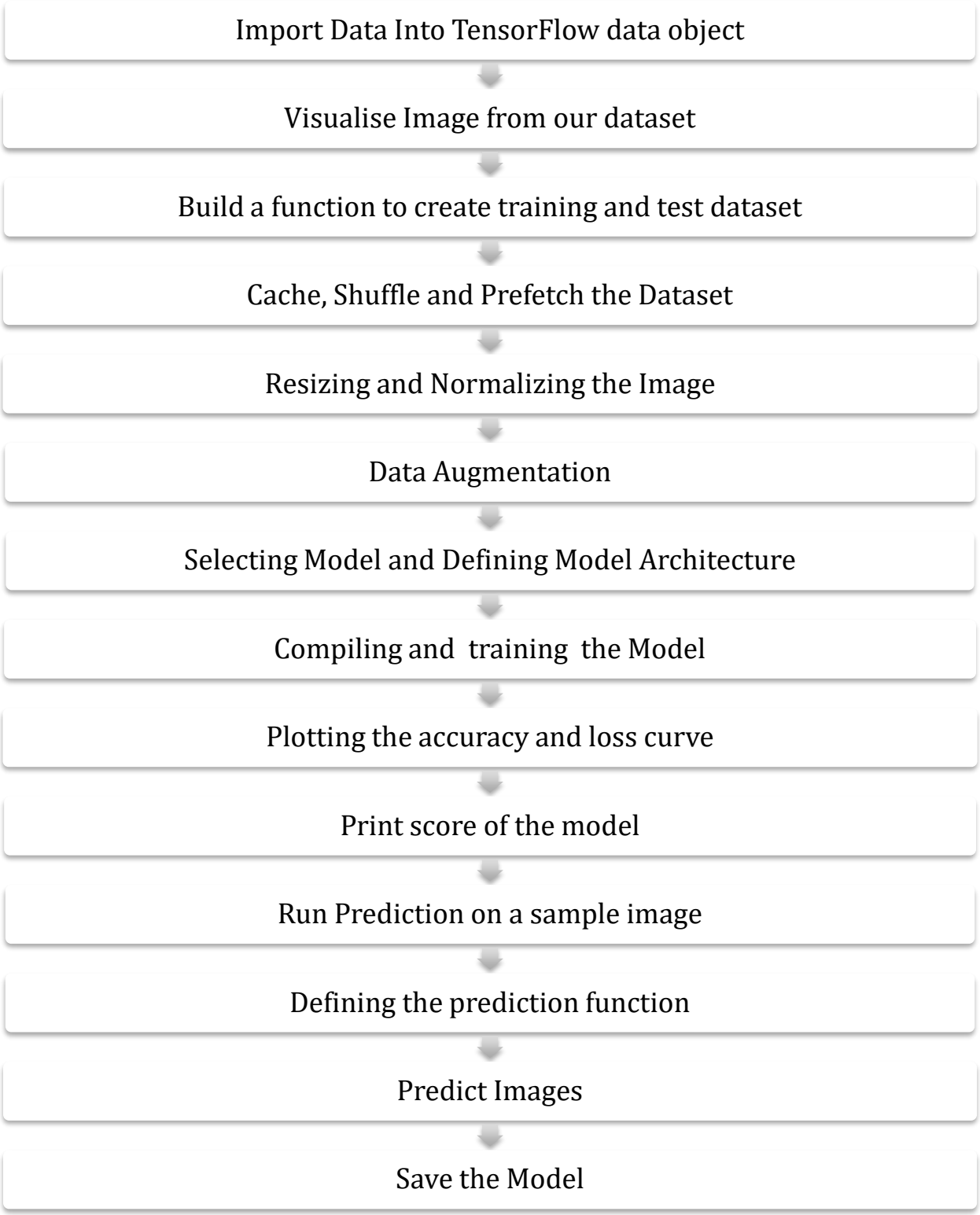
Skill Basics

I have gone through skill basics website in order to understand the core concepts of deep learning. The course was taught by Mr. Dhaval Patel who is a software engineer at Bloomberg USA. His YouTube channel came under Top 15 YouTube Channel for Deep Learning enthusiasts. Initially I have gone through basic concepts of deep learning and its application in real world. The course starts from basic concepts like Neural Network, Neuron and Installation of TensorFlow. Projects like Handwritten Digit Classification, Customer Churn Prediction, CIFAR10 dataset classification etc. Base Concepts like Precision, Recall, F1 Score, activation function and gradient descent method was taught theoretically and practically.

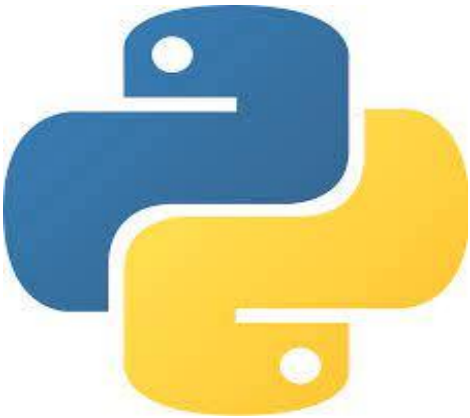
Artificial Neural Network (ANN) was explained in a simple way along with the exercises and projects like customer churn prediction. Further, the topic was computer vision was taught and its significance was explained in the Layman Language. Practical Implementation of each and every concept was given through coding. CNN was taught from basic to advance along with project like Celebrity Image Classification. The project involves Model Building as well as deployment.

Image classification, segmentation and detection were explained clearly like Image Classification helps us to classify what is contained in an image. Image Localization will specify the location of single object in an image whereas Object Detection specifies the location of multiple objects in the image. Finally, Image Segmentation will create a pixel wise mask of each object in the images. Yolo V4 algorithm was discussed and further RNN was also explained by giving the real-life examples and implementation in TensorFlow. Overall, I gained a significant theoretical and practical knowledge associated with deep learning project cycle and core concepts.

Project Flow Diagram and Explanation



Project tools and technology



Import data into TensorFlow dataset object

The tf.data API enables you to build complex input pipelines from simple, reusable pieces. For example, the pipeline for an image model might aggregate data from files in a distributed file system, apply random perturbations to each image, and merge randomly selected images into a batch for training. The pipeline for a text model might involve extracting symbols from raw text data, converting them to embedding identifiers with a lookup table, and batching together sequences of different lengths. The tf.data API makes it possible to handle large amounts of data, read from different data formats, and perform complex transformations. Here we will provide our directory to TensorFlow dataset object.

Visualise Image from our dataset

Visualizing the image becomes necessary in order to verify the image data which we have collected and here we'll be able to explore the dataset in a much better way. Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. In order to view and explore the image we will use Matplotlib Library.

Build a Function to create training and test dataset

In order to train our model, we require training dataset through which the model can learn the image features and extract the boundary and differentiating features among the images. The TensorFlow Library have train test split methods which can be used to do and also, we can define our own function which can do the train test split for the given dataset. After, the train test split we will train the model with training dataset and evaluate the model on test dataset

Cache, Shuffle and Prefetch the Dataset

Shuffling the training data is an important part of data preparation before it moves to the model for training. By shuffling the data, we ensure that **each data point creates an "independent" change on the model**, without being biased by the same points before them. The buffer size in Dataset. shuffle () can affect the randomness of your dataset, and hence the order in which elements are produced.

A **cache's** primary purpose is to **increase data retrieval performance** by reducing the need to access the underlying slower storage layer. The tf data Dataset cache transformation can cache a dataset, either in memory or on local storage. This will save some operations (like file opening and data reading) from being executed during each epoch. The next epochs will reuse the data cached by the cache transformation.

Prefetching is the **loading of a resource before it is required** to decrease the time waiting for that resource. It overlaps the preprocessing and model execution of a training step. While the model is executing training step s, the input pipeline is reading the data for step s+1. Doing so reduces the step time to the maximum (as opposed to the sum) of the training and the time it takes to extract the data.

Resizing and Normalizing the Image

Before we feed our images to network, we should be resizing it to the desired size. Moreover, to improve model performance, we should normalize the image pixel value (keeping them in range 0 and 1 by dividing by 256). This should happen while training as well as inference. Hence, we can add that as a layer in our Sequential Model.

The Resizing of image is also an important operation but it becomes necessary for us to build a model which gives the same performance with the images of varied dimensions. Therefore, here we will resize the image using TensorFlow keras library.

Data Augmentation

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model. Data Augmentation is needed when we have less data, this boosts the accuracy of our model by augmenting the data. It helps us to increase the size of the dataset and introduce variability in the dataset, without actually collecting new data. We will perform the data augmentation with the help of TensorFlow library it has an in-built function which can help us with the data augmentation process.Selecting Model and Defining Model Architecture

After going through research papers and articles I found that Convolutional Neural Network performs best when it comes to image classification tasks. Therefore, I have used CNN as my preferred model for the given problem.

Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which 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. The term “Convolution” is associated with the filtration of features and creating a feature map with respect to a particular feature. The feature map indicates the particular feature location in the image. Let’s say we want to detect the loopy patter from the image then we can apply the loopy pattern filter to the image and in the output we will get feature map which have the location of that feature.

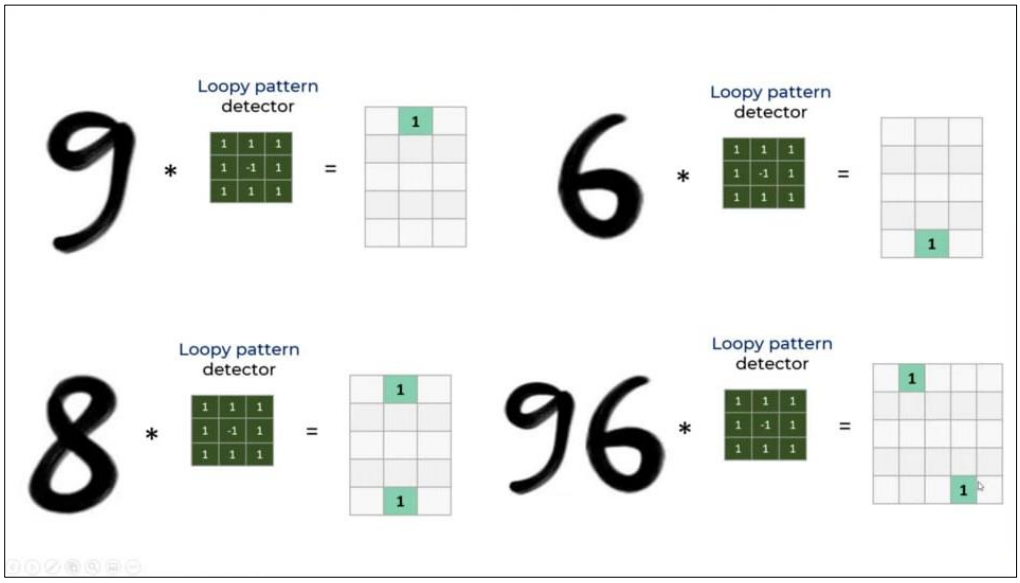


Figure 3 loopy pattern convolution

similarly, we apply different type of filter as per the given image and in the output, we will get the feature map with respect to all the features separately. Once, we get feature map then we apply the ReLU (rectified linear activation unit) activation function in order to inculcate non linearity, reduce the load of computation and simplify the feature map. After, applying the ReLU activation we will apply pooling layer it is used to reduce the size of a feature map. There are different types of pooling layers like Maximum Pool, Minimum Pool Average Pool, Adaptive Pool. The benefits of pooling layer are it reduces dimension and computation, reduce overfitting as there are less parameters and the model will be tolerant towards variations and distortions. The pooling layer works on the basis of strides in the feature map usually one stride is applied to the feature map.

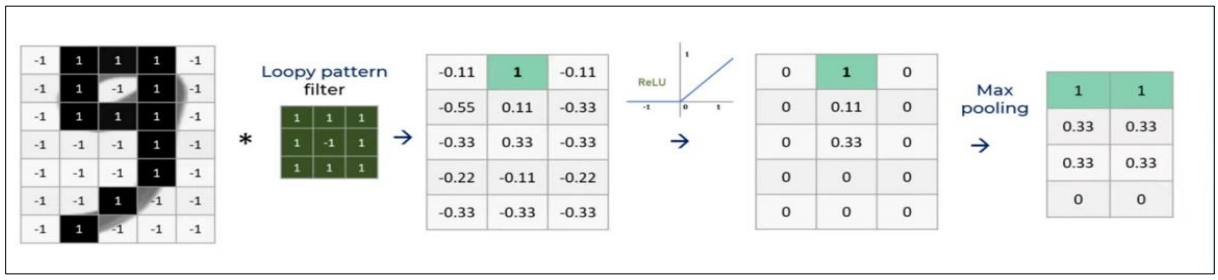


Figure 4 ReLU and Pooling layer operation on Feature Map

Once we get the output from pooling layer it acts as input for the next layer of our entire convolutional neural network and further in the similar way the feature

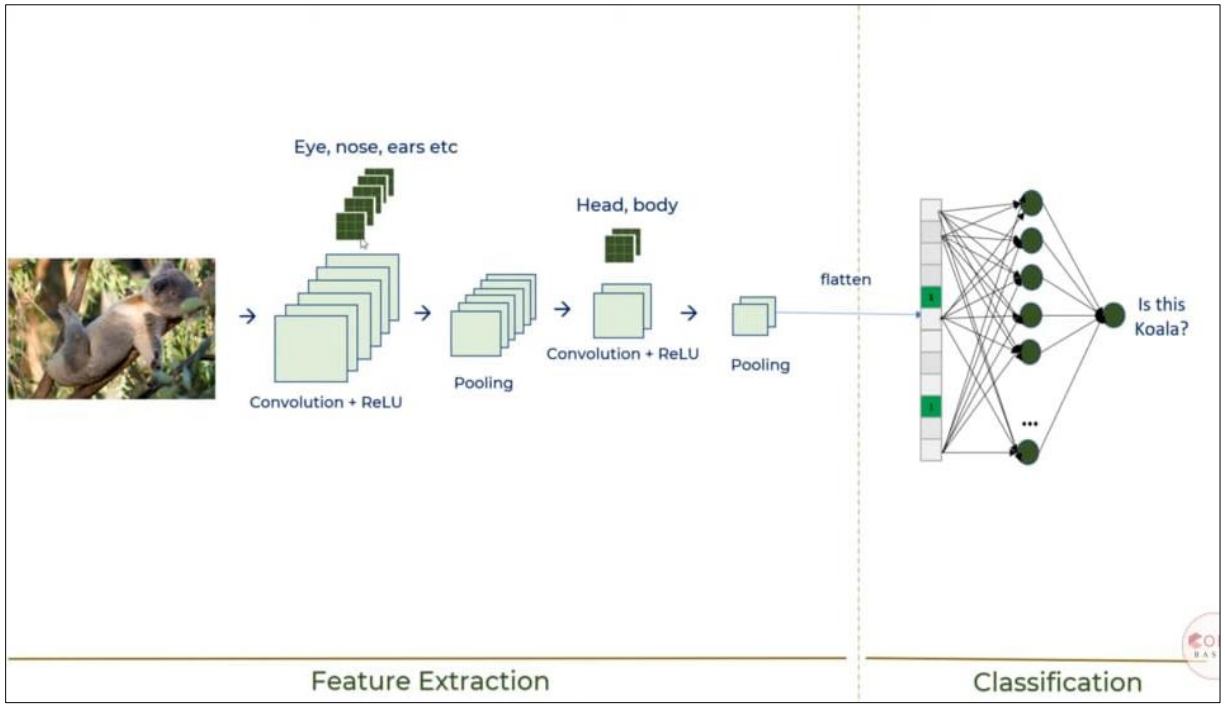


Figure 5 Convolution Neural Network

extraction takes place and finally when we go through all the convolutional layer we will get the final feature map which are similar to matrices as an output. Those matrices will be flattened and applied to simple dense artificial neural network which will give output as a classifier and in accordance with our desired classification operation. The weights and biases of the neural network will be adjusted until the loss is minimum (gradient descent operation) and finally the SoftMax activation will be applied which simplify and optimize the neural network output and we will get the final classification as a label index.

Compiling and training the model

Compile **defines the loss function, the optimizer and the metrics**. That's all. It has nothing to do with the weights and you can compile a model as many times as you want without causing any problem to pretrained weights. The model compile operation is performed once the model architecture is defined and then using TensorFlow library we will execute model compilation before the training and evaluation.

Model training is the phase in the data science development lifecycle where practitioners try to **fit the best combination of weights and bias** to a machine learning algorithm to minimize a loss function over the prediction range. The purpose of model training is to build the best mathematical representation of the relationship between data features and a target label (in supervised learning or among the features themselves (unsupervised learning). There are certain parameters like epochs, filter layers, data size and local system GPU which will impact the duration of model training.

In our case we are using Convolutional Neural Network (CNN) which includes the number of convolution layers, ReLU activation transformation equal to the number of CNN Layers, Pooling layers to optimize the feature maps and a Artificial Neural Network (ANN) to finally give the output as a label index.

Plotting the accuracy and loss curve

Once the model architecture and compile operation are performed, we will move towards plotting the accuracy and loss curve of the model. Here in our dataset, we have three splits train, validation and test split. When we train the model during

that time after an epoch it gets validated with the validation dataset. Hence, we will have the four parameters like Model Accuracy, Model Loss, Validation Accuracy and Validation Loss. We will plot their graphs and visualize that as the epochs increases the loss comes down and accuracy moves up.

The plotting operation is performed using Matplotlib Library of python.

Print score of the model

Scoring is widely used in machine learning to **mean the process of generating new values**, given a model and some new input. The generic term "score" is used, rather than "prediction," because the scoring process can generate so many different types of values: - A predicted class or outcome, for classification models.

The score of the model will be achieved with the help of model. evaluate () function of Deep Learning. The scores show the accuracy of model when the testing dataset is passed to it.

If you are working on a classification problem, the best score is **100% accuracy**. If you are working on a regression problem, the best score is 0.0 error. These scores are an impossible to achieve upper/lower bound. All predictive modeling problems have prediction error.

Run Prediction on a sample image

Once we completed the entire process of training and compiling the model, we will test the model on a sample image and verify its accuracy and confidence Interval.

Defining the prediction function

While running the predict function on a single separate image is one thing but when we want to run a batch or a series of image on a prediction function its altogether a different thing. Hence, we are required to create a function which can predict the class of an image along with its confidence interval. The function is created simply with the help of python.

Predict Images

Once the function is ready where are prepared to predict the images and help farmers, agriculture academicians and students to classify the potato Leaf Disaese.

Save the Model

After the completion of entire model life cycle and verifying the performance of the model we are supposed to save the model in our disk. The model is saved with the help of TensorFlow Library in built function.

Code of the project

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
from IPython.display import HTML
In [3]:
img_size = 256
Batch_size = 32
epochs = 50
rgb_channel = 3
```

Stage 1: Data Collection

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
   "Potato-Leaf-Data",
   shuffle = True,
   image_size = (img_size,img_size),
   batch_size = Batch_size
)
```

Output: Found 2152 files belonging to 3 classes.

```
In [5]:

class_names = dataset.class_names

class_names
```

Output: ['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']

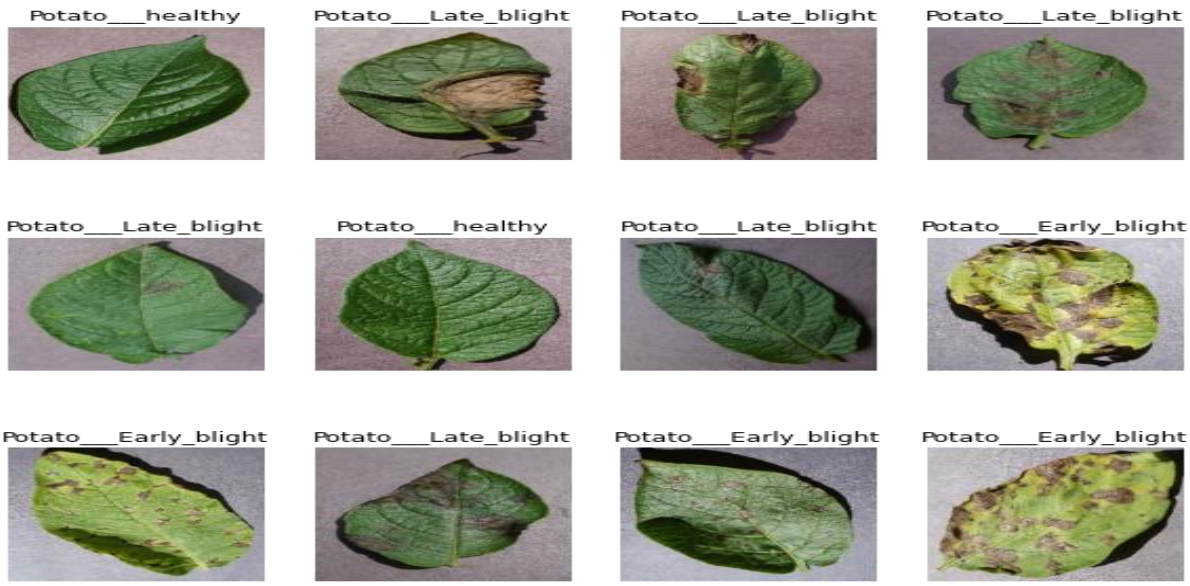
```
print(f"Total Batches in our dataset:{len(dataset)}\nThe total images are:{len(dataset)*Batch_size}")
```

Output:
Total Batches in our dataset:68
The total images are:2176

```
for image_batch,label_batch in dataset.take(1):
    print(image_batch.shape)
    print(label_batch.numpy())
```

Output:
(32, 256, 256, 3)
[1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 0 1 1 2 0 0 0 1 0 2 2 0 2 1 0 0 0]

```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1) #3 ROW 4 COLUMN and i+1th plot
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



Stage 2 : Data Preprocessing

```
defget_dataset_partitions_tf(ds,train_split                                    =
0.8,test_split=0.1,val_split=0.1,shuffle = True,shuffle_size=10000):

    if shuffle:
        ds = ds.shuffle(shuffle_size,seed=10)

    ds_size = len(ds)
    train_size=int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds,val_ds,test_ds


train_ds,val_ds,test_ds = get_dataset_partitions_tf(dataset)

print(len(train_ds))
print(len(val_ds))
print(len(test_ds))
```

Output:

54

6

8

```
train_ds= train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

resize_and_rescale = tf.keras.Sequential([
```

```
  layers.experimental.preprocessing.Resizing(img_size, img_size),
  layers.experimental.preprocessing.Rescaling(1./255),
])


data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),
])
```

Stage 3: Model Building

```
input_shape = (Batch_size, img_size, img_size, rgb_channel)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32,          kernel_size          =          (3,3),          activation='relu',
input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])
```

```
model.build(input_shape=input_shape)


model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential (Sequential)      (32, 256, 256, 3)         0
_________________________________________________________________
sequential_1 (Sequential)    (32, 256, 256, 3)         0
_________________________________________________________________
.
.
.
.
.
_________________________________________________________________
dense (Dense)                (32, 64)                  16448
_________________________________________________________________
dense_1 (Dense)              (32, 3)                   195
=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(
   optimizer='adam',
   loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
   metrics=['accuracy']
)

history = model.fit(
   train_ds,
   batch_size=Batch_size,
   validation_data=val_ds,
   verbose=1,
   epochs=50,
)
```

```
Epoch 1/50
```

```
54/54 [==============================] - 137s 2s/step - loss: 0.9294 - accura
cy: 0.4842 - val_loss: 0.8975 - val_accuracy: 0.4792
Epoch 2/50
54/54 [==============================] - 105s 2s/step - loss: 0.8391 - accura
cy: 0.5329 - val_loss: 0.6988 - val_accuracy: 0.6458
Epoch 3/50
54/54 [==============================] - 89s 2s/step - loss: 0.6570 - accurac
y: 0.6931 - val_loss: 0.4798 - val_accuracy: 0.7917
Epoch 4/50
54/54 [==============================] - 91s 2s/step - loss: 0.5348 - accurac
y: 0.7617 - val_loss: 0.6303 - val_accuracy: 0.7552
Epoch 5/50
54/54 [==============================] - 86s 2s/step - loss: 0.3844 - accurac
y: 0.8498 - val_loss: 0.3008 - val_accuracy: 0.9167
Epoch 6/50
54/54 [==============================] - 92s 2s/step - loss: 0.2442 - accurac
y: 0.9038 - val_loss: 0.4352 - val_accuracy: 0.8385
Epoch 7/50
54/54 [==============================] - 91s 2s/step - loss: 0.1785 - accurac
y: 0.9331 - val_loss: 0.9654 - val_accuracy: 0.7240
Epoch 8/50
54/54 [==============================] - 101s 2s/step - loss: 0.1627 - accura
cy: 0.9337 - val_loss: 0.2542 - val_accuracy: 0.8958
Epoch 9/50
54/54 [==============================] - 94s 2s/step - loss: 0.1306 - accurac
y: 0.9501 - val_loss: 0.1034 - val_accuracy: 0.9635
Epoch 10/50
54/54 [==============================] - 100s 2s/step - loss: 0.1082 - accura
cy: 0.9583 - val_loss: 0.7597 - val_accuracy: 0.7760
Epoch 46/50
54/54 [==============================] - 70s 1s/step - loss: 0.0160 - accurac
y: 0.9935 - val_loss: 0.0097 - val_accuracy: 1.0000
Epoch 47/50
54/54 [==============================] - 69s 1s/step - loss: 0.0279 - accurac
y: 0.9888 - val_loss: 0.0051 - val_accuracy: 1.0000
Epoch 48/50
54/54 [==============================] - 69s 1s/step - loss: 0.0222 - accurac
y: 0.9924 - val_loss: 0.0047 - val_accuracy: 1.0000
Epoch 49/50
54/54 [==============================] - 69s 1s/step - loss: 0.0299 - accurac
y: 0.9883 - val_loss: 0.0165 - val_accuracy: 0.9948
Epoch 50/50
54/54 [==============================] - 72s 1s/step - loss: 0.0246 - accurac
y: 0.9918 - val_loss: 0.0207 - val_accuracy: 0.9896
```

Stage:4 Model Testing and Evaluation

```
scores = model.evaluate(test_ds)
scores
```

8/8 [==============================] - 2s 286ms/step - loss: 0.1249 - accuracy: 0.9688

Output:

```
[0.12487272918224335, 0.96875]
```

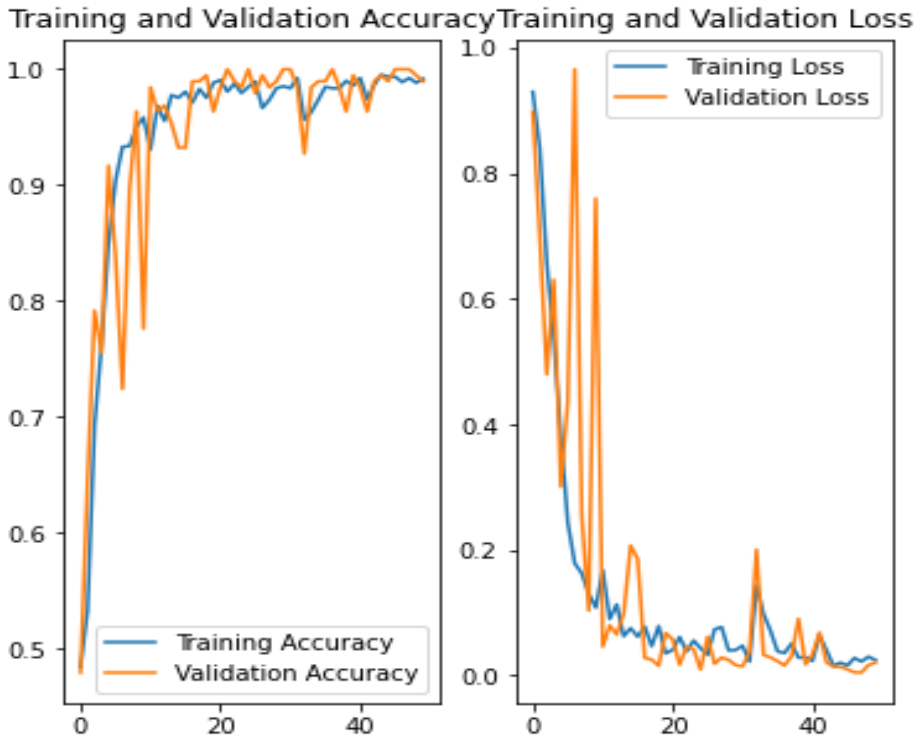
```
history.params
```

output: {'verbose': 1, 'epochs': 50, 'steps': 54}

```
history.history.keys()
```

output: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(6,6))
plt.subplot(1, 2, 1)
plt.plot(range(epochs), acc, label='Training Accuracy')
plt.plot(range(epochs), val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(range(epochs), loss, label='Training Loss')
plt.plot(range(epochs), val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

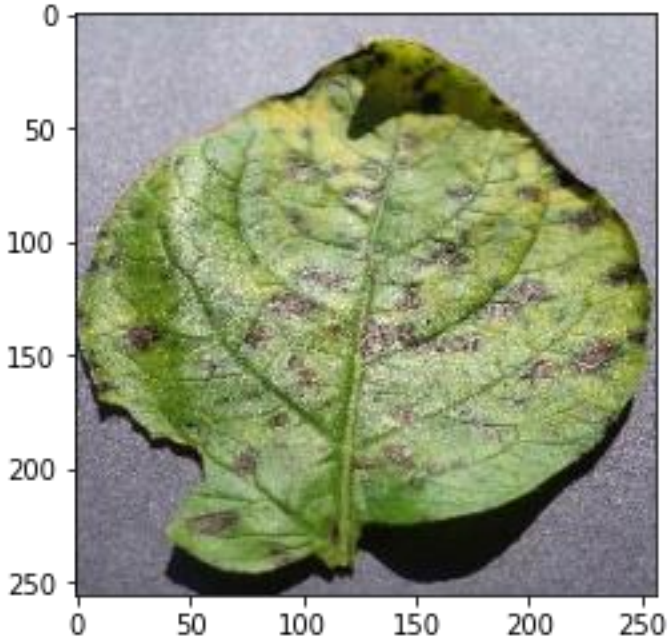



```
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
```

first image to predict

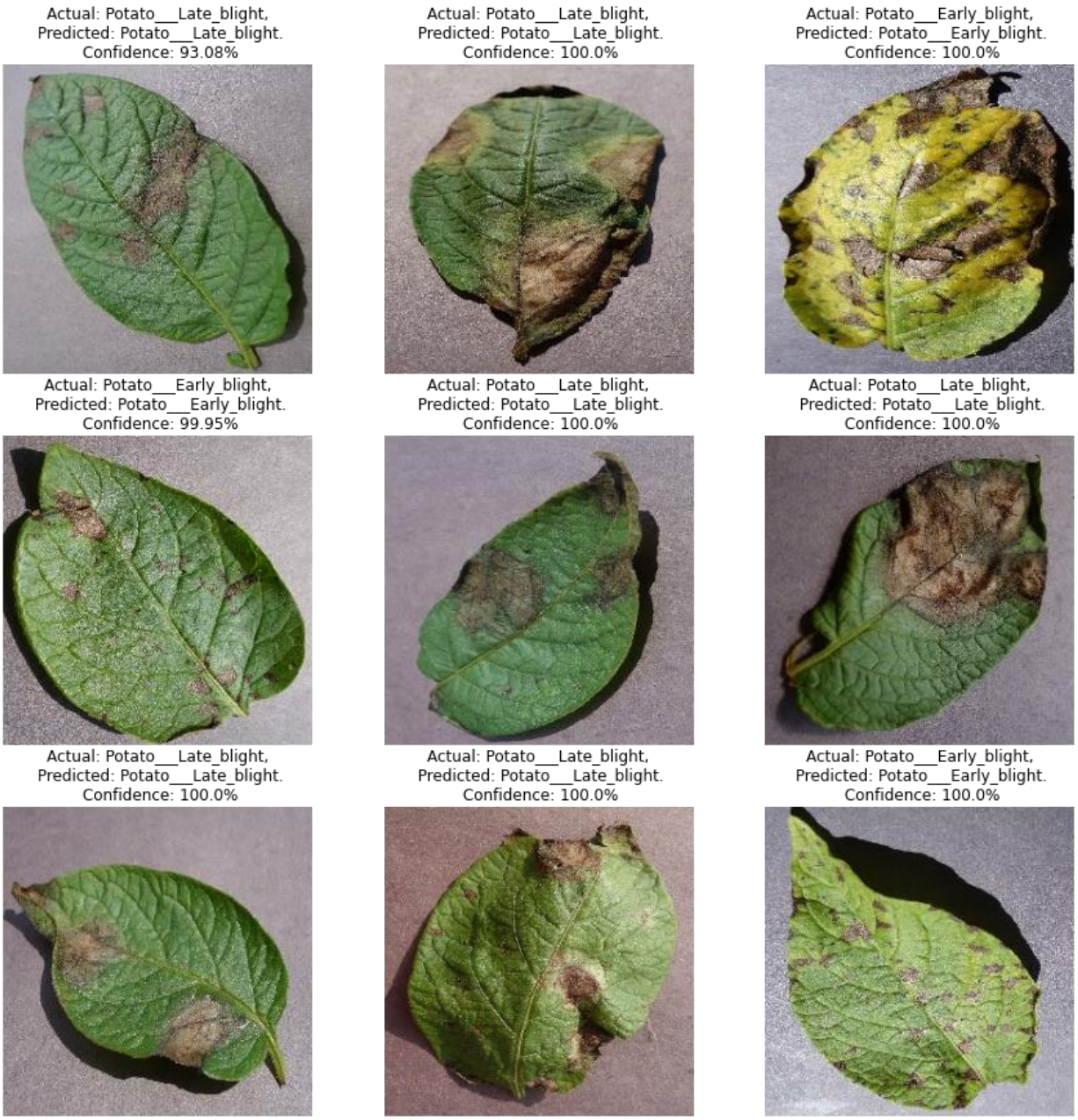
actual label: Potato___Early_blight

predicted label: Potato___Early_blight




```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0) # Create a batch

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence
In [34]:
plt.figure(figsize=(15, 15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]
```

```
    plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")

    plt.axis("off")
```




```
model.save('./models', save_format='tf')
INFO:tensorflow:Assets written to: ./models\assets
```

Conclusion and future scope

As we can see from the stage 4 of the model life cycle, we can clearly say that,

The model achieves an accuracy score of **96.87% on the test dataset** and **99.18% on the training dataset**. This model will help farmers in the early classification of Early Blight and Late Blight disease on Potato Plant.

Farmers can take precautionary measures and control like follow a complete and regular foliar fungicide spray program, Practice good killing techniques to lessen tuber infections, harvesting etc.

The future scope for the project is to build a system where farmers can upload the picture or take the picture of a plant and they get the result with respect to disease associated with plant. Sustainable agriculture crop production and economic stability will be enhanced using such technology.

Future Work

Forest Cover Type prediction.

Problem Statement: -

Forest land is highly required for developing ecosystem management. Any changes that occur in ecosystem should be carefully noticed to avoid further loss. This model helps in noticing the changes occurred due to heavy floods or other calamities which affected the forest land. The goal is to predict seven different cover types in four wilderness area.

Further, i have to build a platform where user can interact with my model.
The database allotted for the project is Cassandra.

End to end deployment should be executed after the model formation in GCP, AWS or Azure.
Dataset: - https://www.kaggle.com/c/forest-cover-type-prediction/data

iNeuron

iNeuron Intelligence Pvt Ltd

#51/27 Swamy towers-1st Floor, 1st Main, RJ Garden, ORR, Marathahalli, Bangalore - 560037

DATE: 1st September 2021

Internship Offer Letter

Dear Galav Bhatt,

Following your application, we are pleased to inform you that you have been considered for an internship with iNeuron for **Forest Cover Type Prediction** project. As a result, you will be contributing to our project from 1st September 2021.

As a part of your internship, you will be proactively contributing to your selected project, besides product development & PoCs. In addition, you will be required to complete performance & learning goals for your current project with us.

We hope that your association with the company will be successful and rewarding.

Regards,
Sudhanshu Kumar
CEO & Chief AI Engineer at iNeuron.ai

I accept the offer with the company on the terms and conditions set out in this letter.

Galav Bhatt **DATE: 1st September 2021**

References

1) D. Tiwari, M. Ashish, N. Gangwar, A. Sharma, S. Patel and S. Bhardwaj, "Potato Leaf Diseases Detection Using Deep Learning," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 461-466, doi: 10.1109/ICICCS48265.2020.9121067.
2) https://www.tensorflow.org/
3) https://www.skillbasics.com/
4) http://www.fao.org/potato-2008/en/potato/utilization.html
5) https://garden.org/learn/articles/view/560/
6) https://agmarknet.gov.in/Others/profile-potato.pdf
7) https://doraagri.com/difference-of-early-late-blight/