A Project Report on

# Detecting plants leaf disease using deep learning

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in Electronics and Communication Engineering.


**UNDER THE GUIDANCE :**

Mr.M.A.MUTHIAH
Dept of Electroics and Communication Engineering.

**SUBMITTED BY :**

Volipi UdayKumar - 38130240
Thalluru Uday Kiran - 38130221

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

(DEEMED TO BE UNIVERSITY )

Accredited with Grade "A" by NAAc I 12B S tatusby UGC I Approved by **AICTE**

**JEPPIAR NAGAR,RAJIV GANDHI SALAI,CHENNAI _ 600119.**

**I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.**

I convey my thanks to Dr. **N.M. NANDITHA, M.E., Ph.D.**, Dean, School of **ELECTRONICS AND COMMUNICATION ENGINEERING** and Dr. **T. RAVI, M.E., Ph.D.**, Head of the Department, Dept. of **ELECTRONICS AND COMMUNICATION ENGINEERING** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr.M.A.MUTHIAH** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **ELECTRONICS AND COMMUNICATION ENGINEERING** who were helpful in many ways for the completion of the project.

• This is to certify that the " Project Report " by Batch No : ECE - 79

1) VOLIPI UDAY KUMAR (Regd. No : 38130240)
2) THALLURU UDAY KIRAN(Regd. No : 38130221)

Is work done by his and submitted during 2018 – 2022 academic year, in partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF ENGINEERING in ELECTRONICS AND COMMUNICATIONS AND ENGINEERING .**

| Internal Guide | Head of the Department |
|---|---|
| Mr.**M.A.MUTHIAH** | Dr. **T. RAVI, M.E., Ph.D.** |

## Abstract

One of the important and tedious task in agricultural practices is detection of disease on crops. It requires huge time as well as skilled labor. This paper proposes a smart and efficient technique for detection of crop disease which uses computer vision and machine learning techniques.

The proposed system is able to detect 5 different diseases of Appel plants with 95% accuracy.

***Key Words:*** Digital image processing, CNN, Machine learning, Plant disease detection..

## INTRODUCTION

In India about 70% of the populace relies on agriculture. Identification of the plant diseases is important in order to prevent the losses within the yield. It's terribly troublesome to observe the plant diseases manually. It needs tremendous quantity of labor, expertize within the plant diseases, and conjointly need the excessive time interval. Hence, image processing and machine learning models can be employed for the detection of plant diseases. In this project, we have described the technique for the detection of plant diseases with the help of their leaves pictures. Image processing is a branch of signal processing which can extract the image properties or useful information from the image. Machine learning is a sub part of artificial intelligence which works automatically or give instructions to do a particular task. The main aim of machine learning is to understand the training data and fit that training data into models that should be useful to the people. So it can assist in good decisions making and predicting the correct output using the large amount of training data. The color of leaves, amount of damage to leaves, area of the leaf, texture parameters are used for classification. In this project we have analyzed different image parameters or features to identifying different plant leaves diseases to achieve the best accuracy. Previously plant disease detection is done by visual inspection of the leaves or some chemical processes by experts. For doing so, a large team of experts as well as continuous observation of plant is needed, which costs high when we do with large farms. In such conditions, the recommended system proves to be helpful in monitoring large fields of crops. Automatic detection of the diseases by simply seeing the symptoms on

the plant leaves makes it easier as well as cheaper. The proposed solution for plant disease detection is computationally less expensive and requires less time for prediction than other deep learning based approaches since it uses statistical machine learning and image processing algorithm.

## LITERATURE SURVEY

In 2015, S. Khirade et Al. tackled the problem of plant disease detection using digital image processing techniques and back propagation neural network (BPNN) [1]. Authors have elaborated different techniques for the detection of plant disease using the images of leaves. They have implemented Otsu's thresholding followed by boundary detection and spot detection algorithm to segment the infected part in leaf. After that they have extracted the features such as color, texture, morphology, edges etc. for classification of plant disease. BPNN is used for classification i.e. to detect the plant disease. Shiroop Madiwalar and Medha Wyawahare analyzed different image processing approaches for plant disease detection in their research [2]. Authors analyzed the color and texture features for the detection of plant disease. They have experimented their algorithms on the dataset of 110 RGB images. The features extracted for classification were mean and standard deviation of RGB and YCbCr channels, grey level cooccurrence matrix (GLCM) features, the mean and standard deviation of the image convolved with Gabor filter. Support vector machine classifier was used for classification. Authors concluded that GCLM features are effective to detect normal leaves. Whereas color features and Gabor filter features are considered as best for detecting anthracnose affected leaves and leaf spot respectively. They have achieved highest accuracy of 73.34% using all the extracted features. Peyman Moghadam et Al. demonstrated the application of hyperspectral imaging in plant disease detection task [3]. visible and near-infrared (VNIR) and short-wave infrared (SWIR) spectrums were used in this research. Authors have used k-means clustering algorithm in spectral domain for the segmentation of leaf. They have proposed a novel grid removal algorithm to remove the grid from hyperspectral images. Authors have achieved the accuracy of 73% with vegetation indices in VNIR spectral range and 93% accuracy with full spectrum. Though the proposed

method achieved higher accuracy, it requires the hyperspectral camera with 324 spectral bands so the solution becomes too costly. Sharath D. M. et Al. developed the Bacterial Blight detection system for Pomegranate plant by using features such as color, mean, homogeneity, SD, variance, correlation, entropy, edges etc. Authors have implemented grab cut segmentation for segmenting the region of interest in the image [4]. Canny edge detector was used to extract the edges from the images. Authors have successfully developed a system which can predict the infection level in the fruit. Garima Shrestha et Al. deployed the convolutional neural network to detect the plant disease [5]. Authors have successfully classified 12 plant diseases with 88.80% accuracy. The dataset of 3000 high resolution RGB images were used for experimentation. The network has 3 blocks of convolution and pooling layers. This makes the network computationally expensive. Also the F1 score of the model is 0.12 which is very low because of higher number of false negative predictions.

## **Existing System** :-

The system detects disease in plant leaf image and show the percentage of the predicted disease with acutal disease and show the value of accuracy it helps to know only the efficiency of the system.But not the actual disease in leaf of the plant.The accuracy of the model is not efficient and it takes more time to load data and run the model.In Normal Vgg19 and Darknet it showing less accuracy . Loss keep on increasing with epochs.Accuracy decreasing keep on decrasing on increasing epochs which is not a good model.

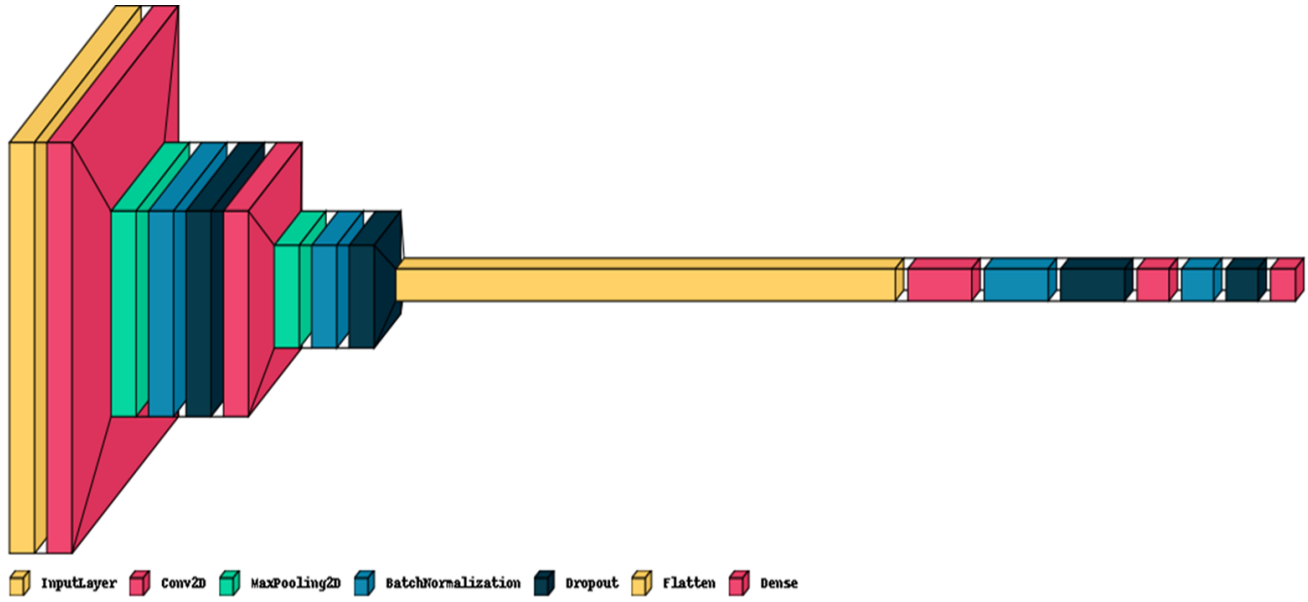
## **Proposed System** :-

- The challenging part of our approach is not only deal with the accuracy of model and detect the actual disease and tries to give the type of disease leaf having so that we can have particular for it.Our model give the 92% accuracy.We have Customize the Vgg19 architecture.Because in normal vgg19 we are getting 32% as accuracy.This model as good training accuracy and Validation accuracy.Training loss and validation losss goes decreasing on increasing epochs.

## **Description:**

We create one cnn model and train the model using training dataset and predict the test images disease dataset .For more understanding we spilt the train dataset itself into train,val,test dataset.we train the model with train and val dataset and predict the test dataset and see the accuracy of the model and fscore of the model.This can helps to see both actual and predicted disease on particular test image.

**Block Diagram** :




## Applications and Advantages:

1) "Leaf Disease Detection" can be used in the various types of fields for detecting whether the leaf is healthy r not healthy.

2) This project can also be used in agriculture to detect type of disease in plants .

**Formula:**

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

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

**F1 = 2 * (precision * recall) / (precision + recall)**

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

| | | **Predicted** | |
|---|---|---|---|
| | | **Negative** | **Positive** |
| **Actual** | **Negative** | True Negative | False Positive |
| | **Positive** | False Negative | True Positive |

**Software used:**

- Kaggle, Jupyter Notebook

## 3)Methodology :

### 3.a)Dataset:

**Healthy:**

**Rust:**

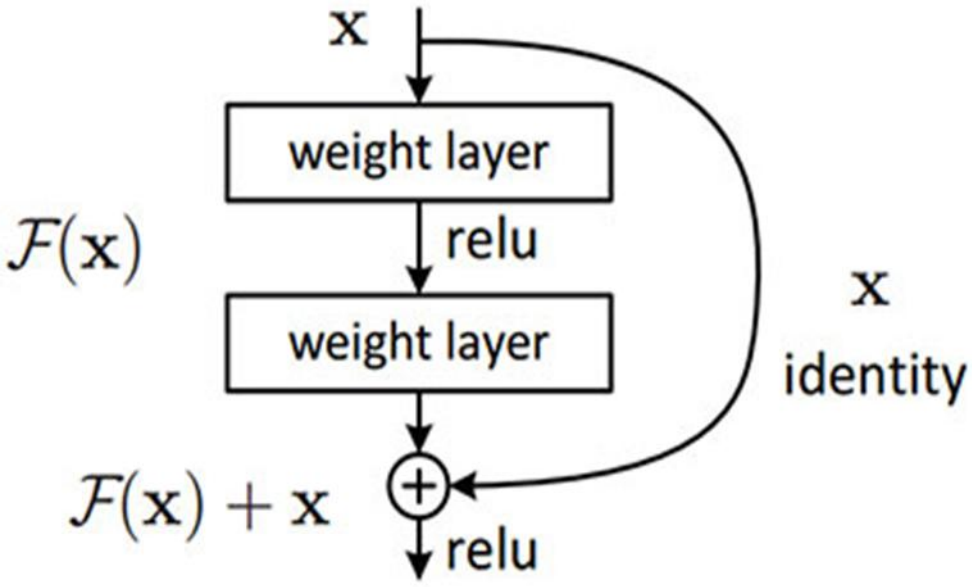
**Scab:**

**Multiple Disease:**

**Steps involved:**

- Collection and cleaning of image dataset. Basic exploratory analysis of the collected dataset is performed.
- A neural network architecture suitable to the task at hand is selected and coded up using the Keras framework.
- The NN is trained and the metrics relevant to the current task is measured (accuracy, F1 score etc.)

**RESNET have more accuracy due to Residual block:**



# Introduction of Machine Learning

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

**Machine learning** (**ML**) is the study of computer algorithms that can improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.[2] Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where

it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

**Introduction to  Machine Learning**

- Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

**Types of Learning:**

**1)Supervised Learning**

**2)Unsupervised Learning**

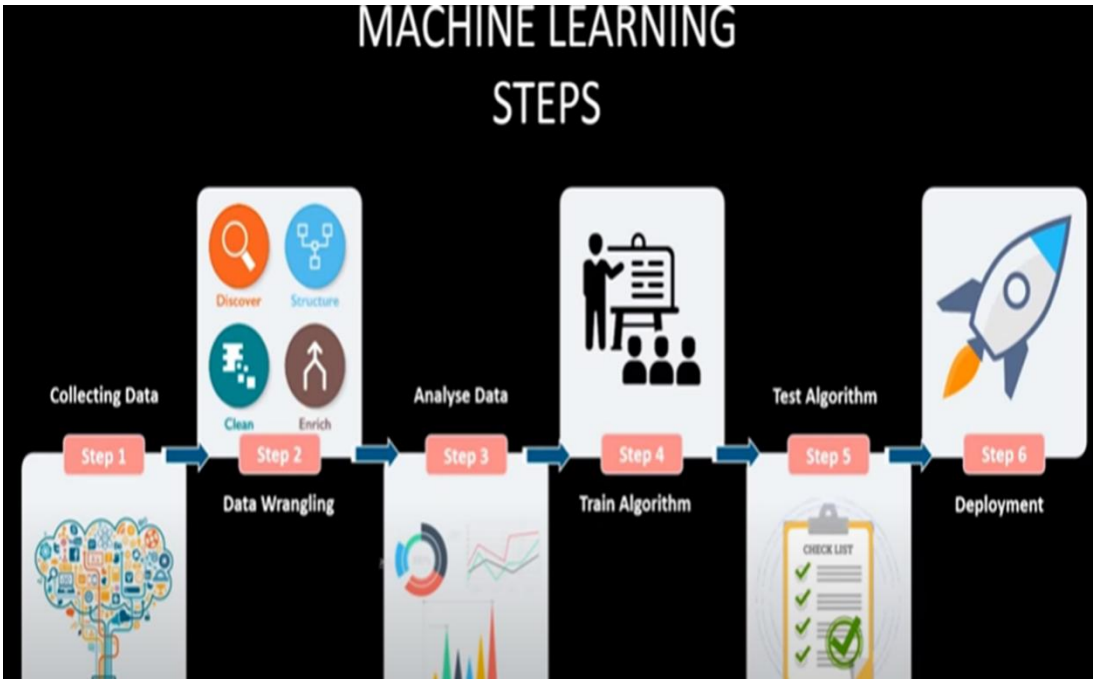
**3) Semi-Supervised Learning**

**4)Reinforcement Learning**

- **Supervised learning:** In this type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.
- **Unsupervised learning:** This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through data sets looking for any meaningful connection. The

data that algorithms train on as well as the predictions or recommendations they output are predetermined

- **Semi-supervised learning:** This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.
- **Reinforcement learning:** Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.




# Need for Machine Learning

- Human beings, at this moment, are the most intelligent and advanced type on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".
- Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it

data-driven decisions taken by machines, particularly to automate the process.

- These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

## Why & When to Make Machines Learn?

- There can be several circumstances where we need machines to take data-driven decisions with efficiency and at a huge scale. The followings are some of such circumstances where making machines learn would be more effective
- Lack of human expertise
- The very first scenario in which we want a machine to learn and take data-driven decisions, can be the domain where there is a lack of human expertise. The examples can be navigations in unknown territories or spatial planets.
- Dynamic scenarios
- There are some scenarios which are dynamic in nature i.e. they keep changing over time. In case of these scenarios and behaviors, we want a machine to learn and take data-driven decisions. Some of the examples can be network connectivity and availability of infrastructure in an organization.
- Difficulty in translating expertise into computational tasks

- There can be various domains in which humans have their expertise,; however, they are unable to translate this expertise into computational tasks. In such circumstances we want machine learning. The examples can be the domains of speech recognition, cognitive tasks etc.
- “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”
- The above definition is basically focusing on three parameters, also the main components of any learning algorithm, namely Task(T), Performance(P) and experience (E). In this context, we can simplify this definition as −
- ML is a field of AI consisting of learning algorithms that −
- Improve their performance (P)
- At executing some task (T)
- Over time with experience

Task(T)

- From the perspective of problem, we may define the task T as the real-world problem to be solved. The problem can be anything like finding best house price in a specific location or to find best marketing strategy etc. On the other hand, if we talk about machine learning, the definition of task is different because it is difficult to solve ML based tasks by conventional programming approach.
- A task T is said to be a ML based task when it is based on the process and the system must follow for operating on data

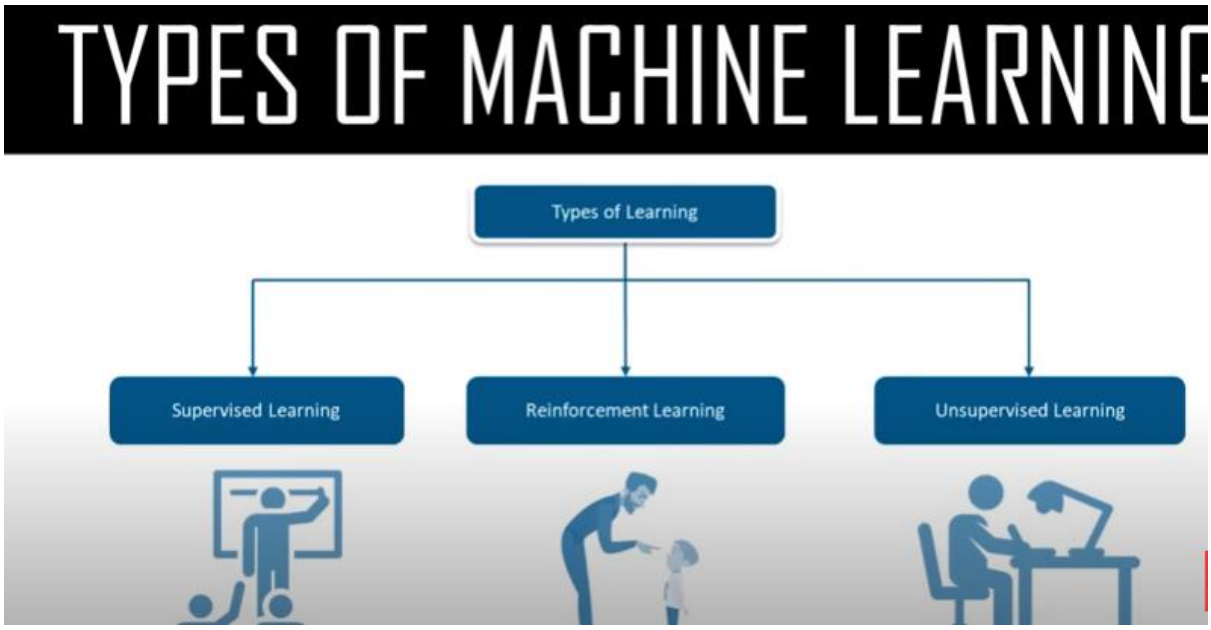
points. The examples of ML based tasks are Classification, Regression, Structured annotation, Clustering, Transcription etc.

- Experience (E)
- As name suggests, it is the knowledge gained from data points provided to the algorithm or model. Once provided with the dataset, the model will run iteratively and will learn some inherent pattern. The learning thus acquired is called experience(E). Making an analogy with human learning, we can think of this situation as in which a human being is learning or gaining some experience from various attributes like situation, relationships etc. Supervised, unsupervised and reinforcement learning are some ways to learn or gain experience. The experience gained by out ML model or algorithm will be used to solve the task T.
- Performance (P)
- An ML algorithm is supposed to perform task and gain experience with the passage of time. The measure which tells whether ML algorithm is performing as per expectation or not is its performance (P). P is basically a quantitative metric that tells how a model is performing the task, T, using its experience, E. There are many metrics that help to understand the ML performance, such as accuracy score, F1 score, confusion matrix, precision, recall, sensitivity etc.

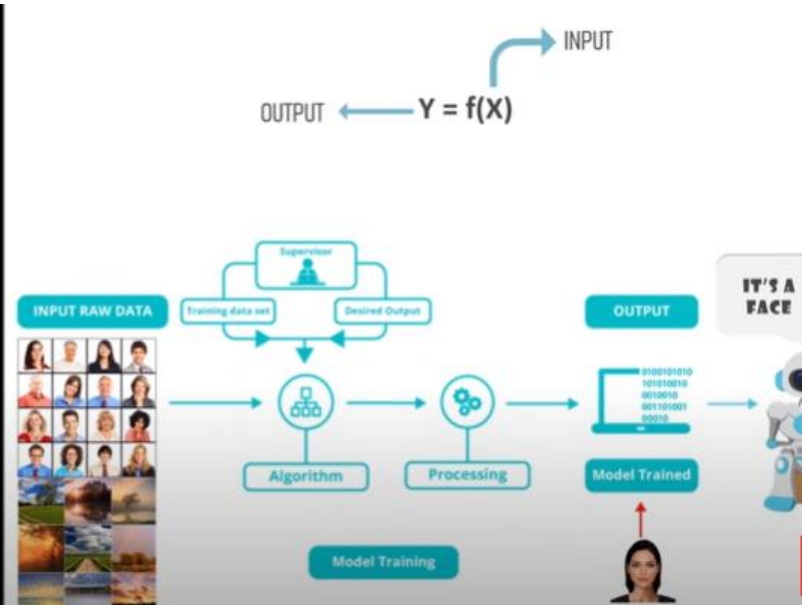
# Challenges in Machines Learning

- **Quality of data** − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
- **Time-Consuming task** − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
- **Lack of specialist persons** − As ML technology is still in its infancy stage, availability of expert resources is a tough job.
- **No clear objective for formulating business problems** − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
- **Issue of overfitting & underfitting** − If the model is overfitting or underfitting, it cannot be represented well for the problem.
- **Curse of dimensionality** − Another challenge ML model faces is too many features of data points. This can be a real hindrance.
- **Difficulty in deployment** − Complexity of the ML model makes it quite difficult to be deployed in real life.
- Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML
- Emotion analysis
- Sentiment analysis
- Error detection and prevention

- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping.

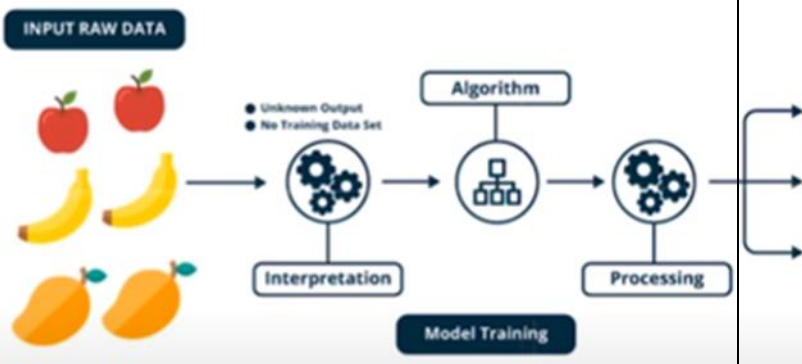


# SUPERVISED LEARNING

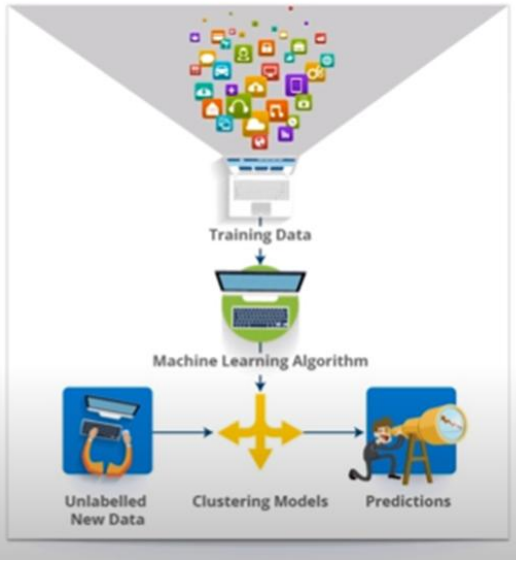



# SUPERVISED LEARNING EXAMPLES

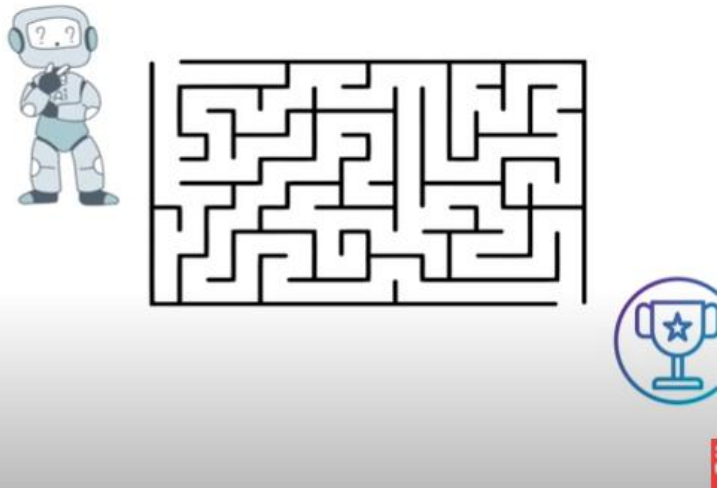

# UNSUPERVISED LEARNING




# UNSUPERVISED LEARNING EXAMPLES

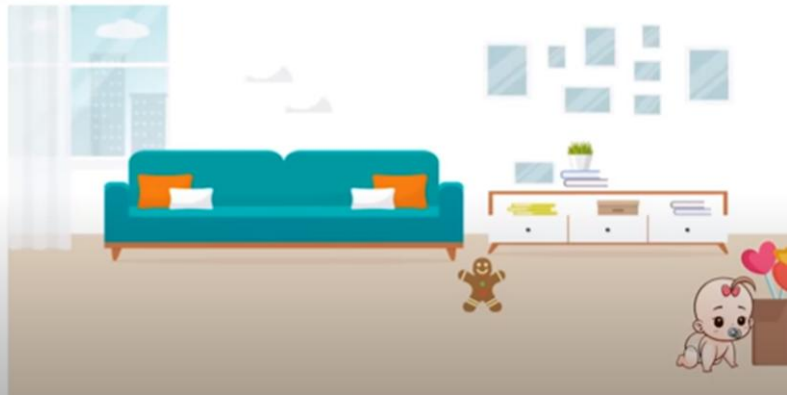


# REINFORCEMENT LEARNING



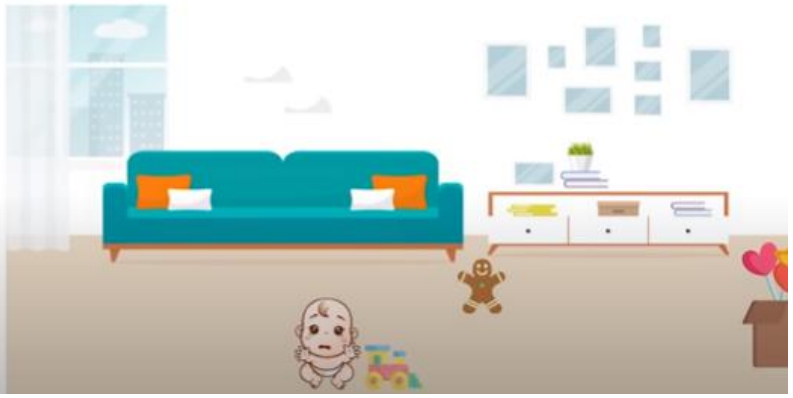
# REINFORCEMENT LEARNING EXAMPLE

Scenario - 1

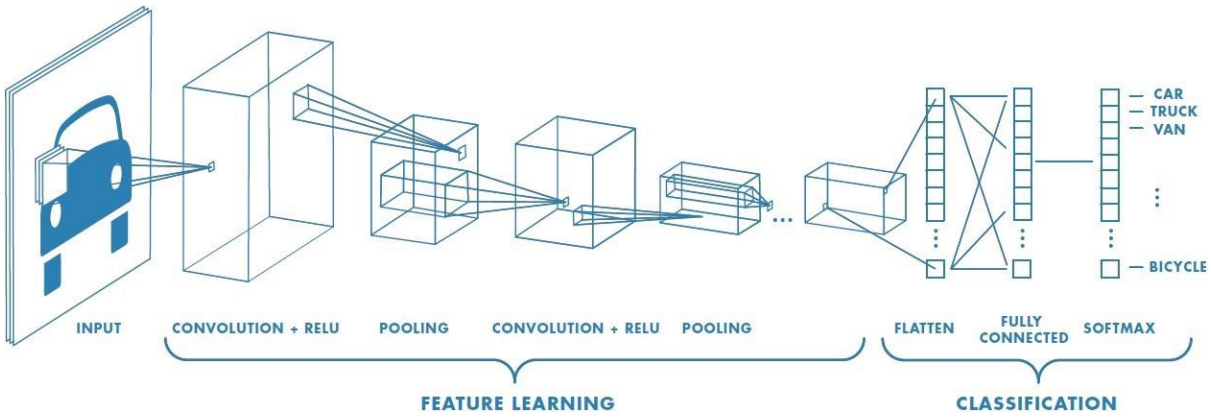


# REINFORCEMENT LEARNING EXAMPLE

Scenario - 2

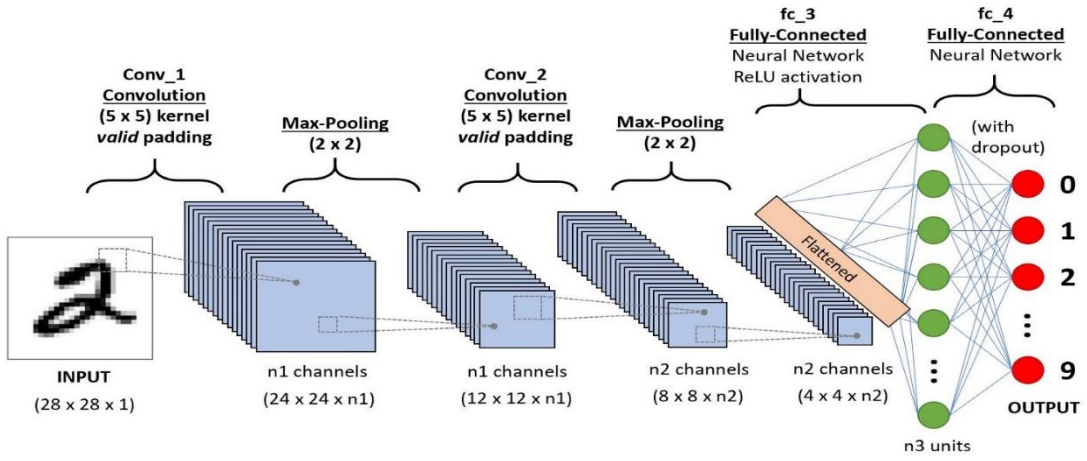


# Convolutional Neural Networks



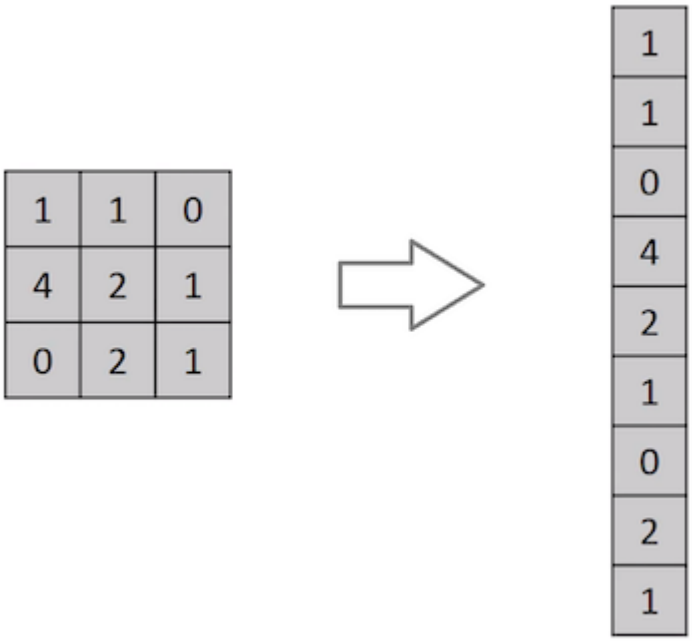

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision.

The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm — a **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. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.



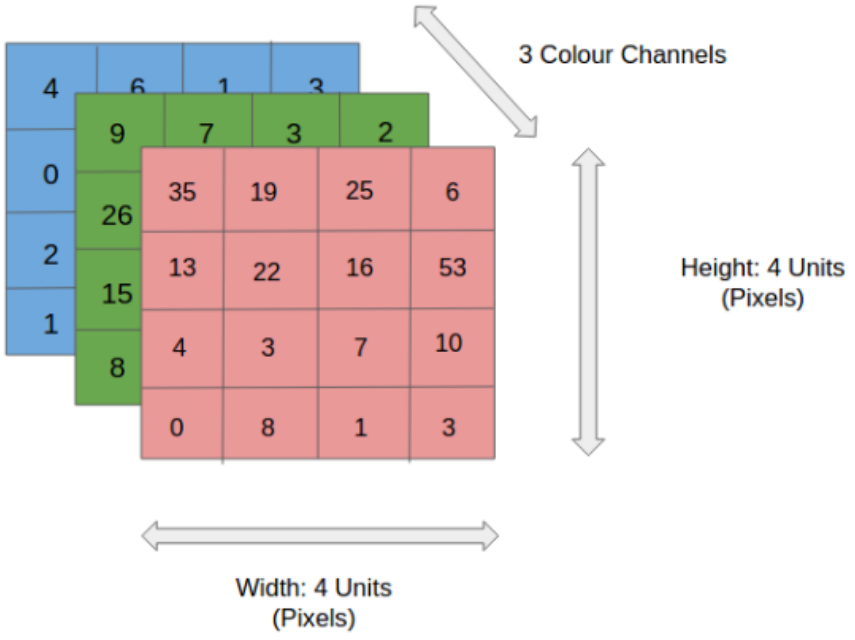

Flattening of a 3x3 image matrix into a 9x1 vector

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

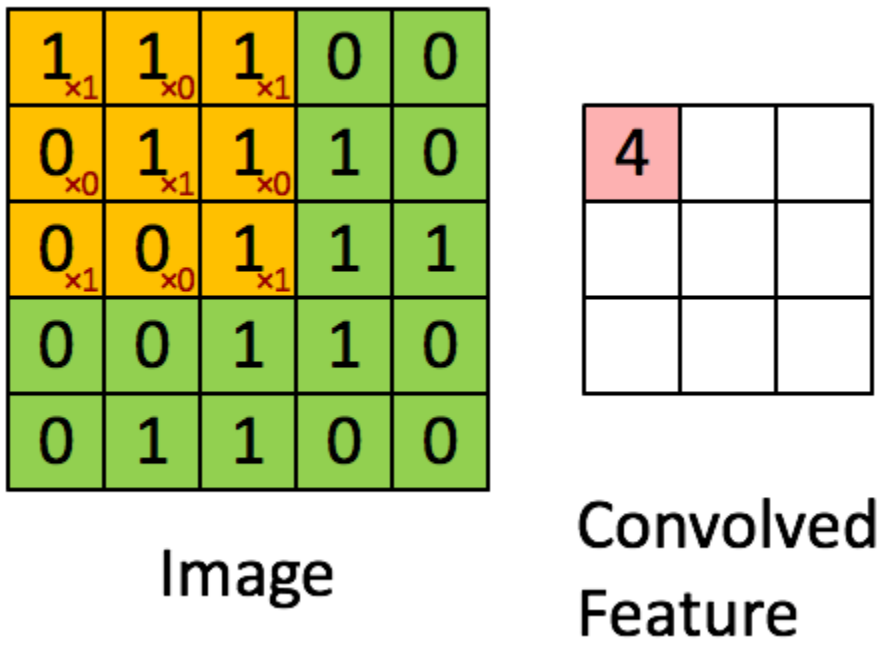
## Input Image



In the figure, we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

## Convolution Layer — The Kernel

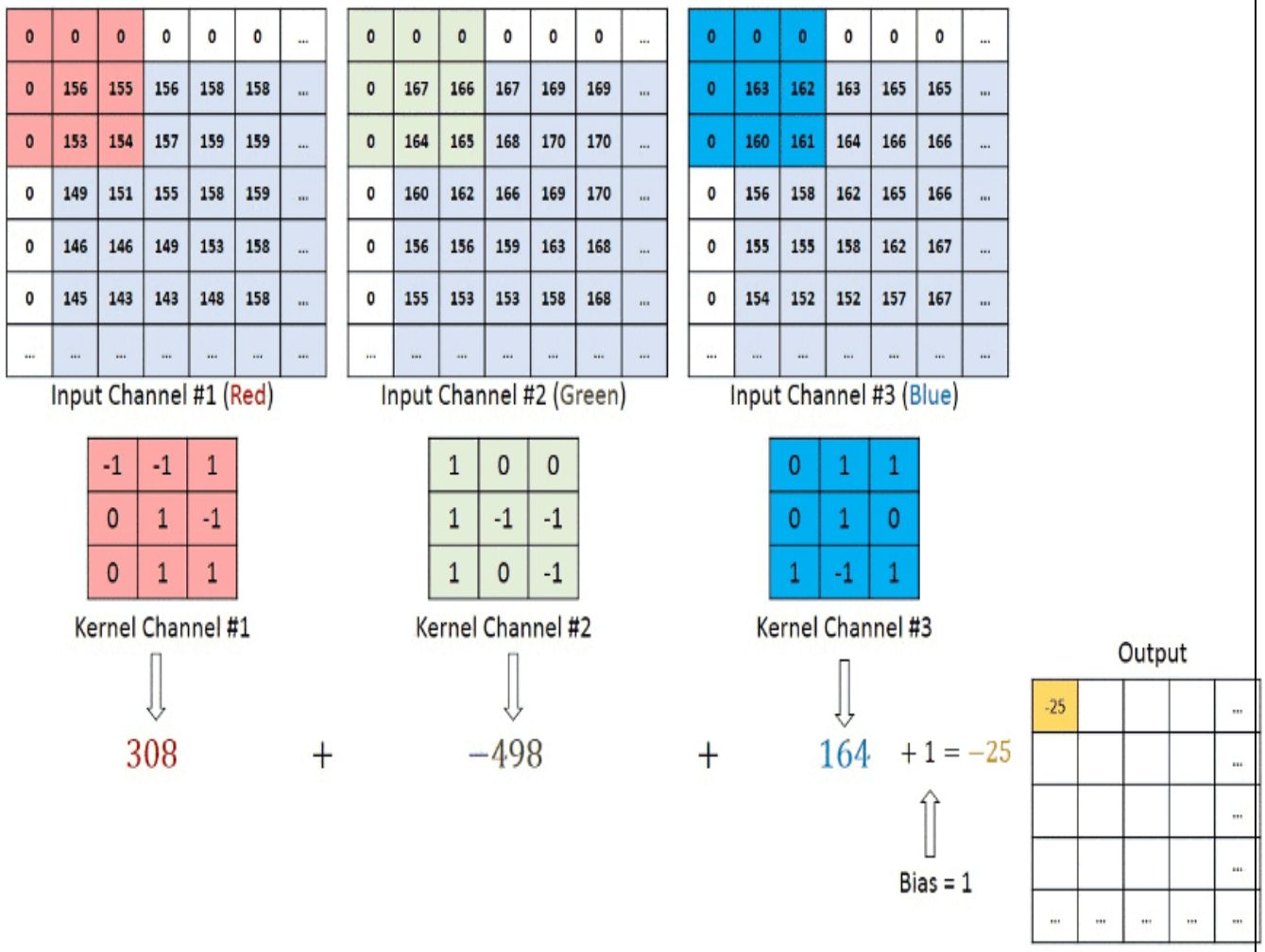



Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

In the above demonstration, the green section resembles our **5x5x1 input image, I**. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter, K**, represented in the color yellow. We have selected **K as a 3x3x1 matrix.**

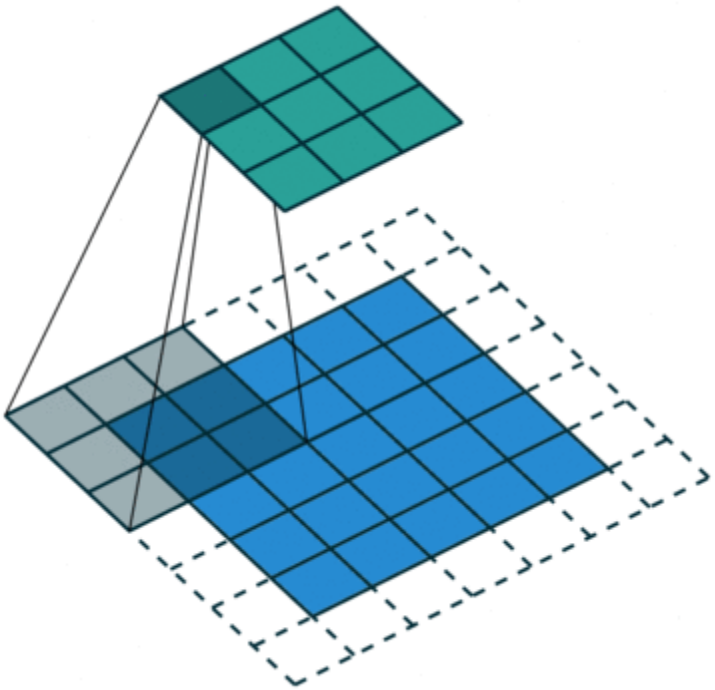
The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.




Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

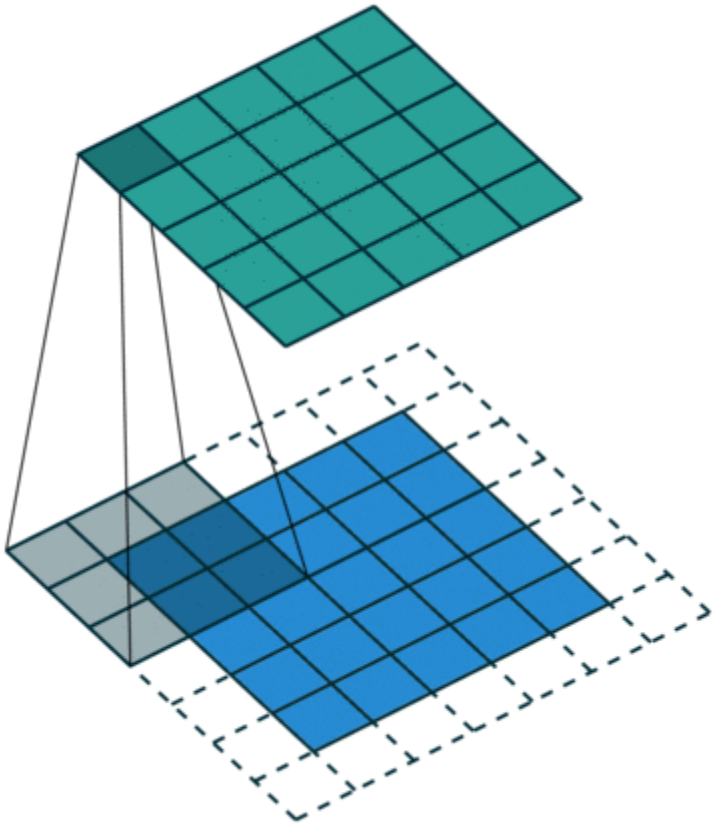
n the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output

Convolution Operation with Stride Length = 2



The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying **Valid Padding** in case of the former, or **Same Padding** in the case of the latter.

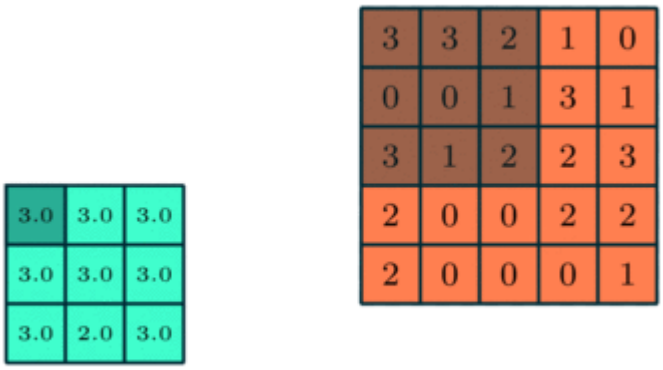


**SAME padding:** 5x5x1 image is padded with 0s to create a 6x6x1 image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name — **Same Padding**.

On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself — **Valid Padding**.

## Pooling Layer

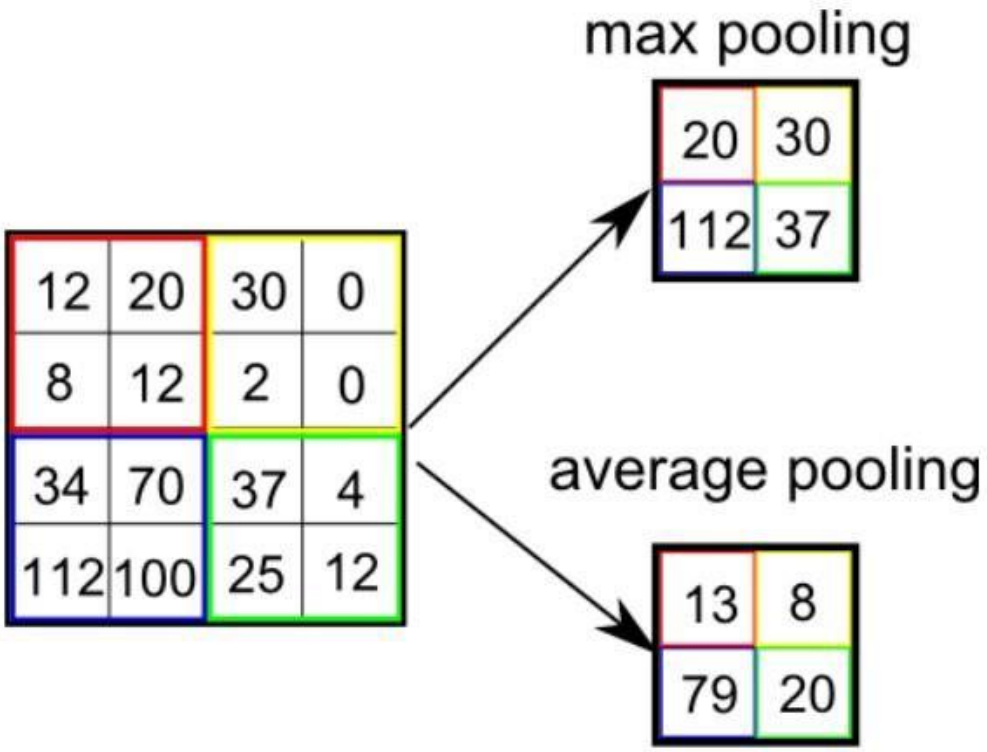



3x3 pooling over 5x5 convolved feature

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** through dimensionality reduction. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. **Max Pooling** returns the **maximum value** from the portion of the image covered by the Kernel. On the other hand, **Average Pooling** returns the **average of all the values** from the portion of the image covered by the Kernel.

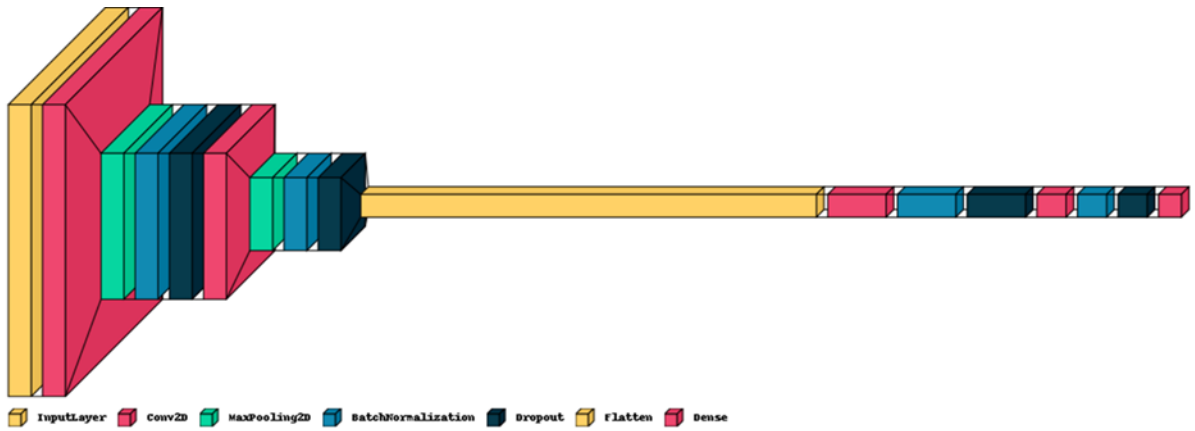
Max Pooling also performs as a **Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.



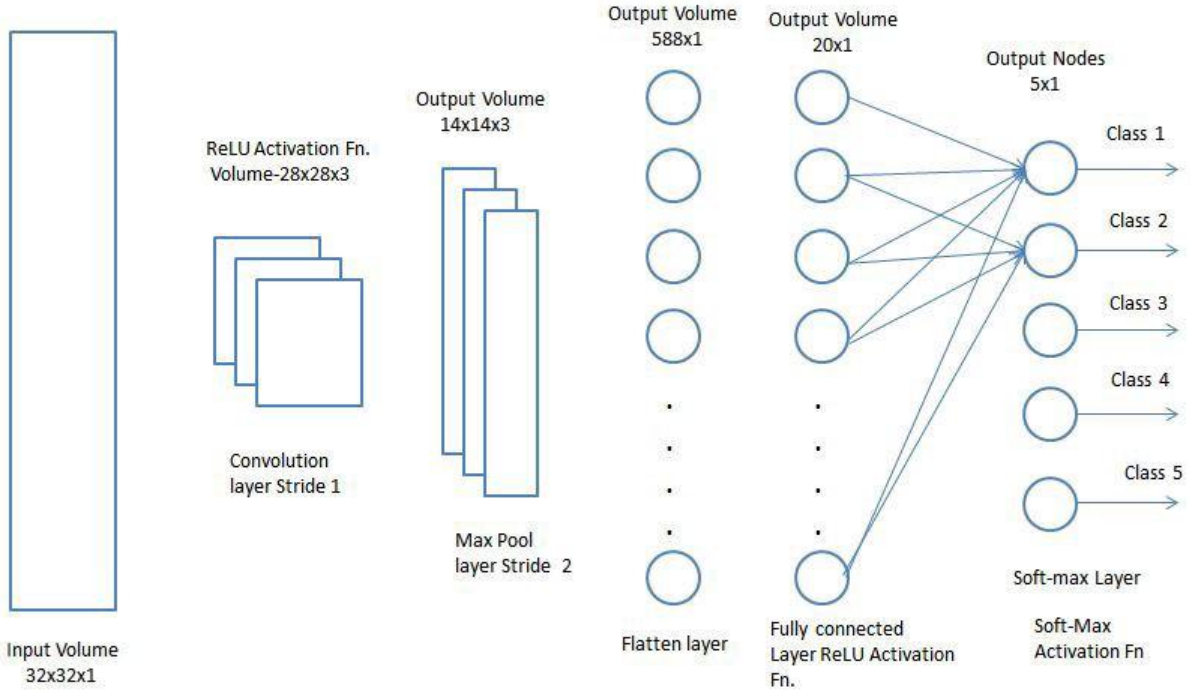

Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.



## Classification — Fully Connected Layer (FC Layer)




Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

**Result:**

We train the model using training images and predicted the test image dataset. For more understanding we spilt the train dataset itself into train,val,test dataset.we train the model with train and val dataset and predict the test dataset and see the accuracy of the model and fscore of the model.This can helps to see both actual and predicted disease on particular test image.we created the 3 models and vgg-19 custom model has better accuracy with 95%.

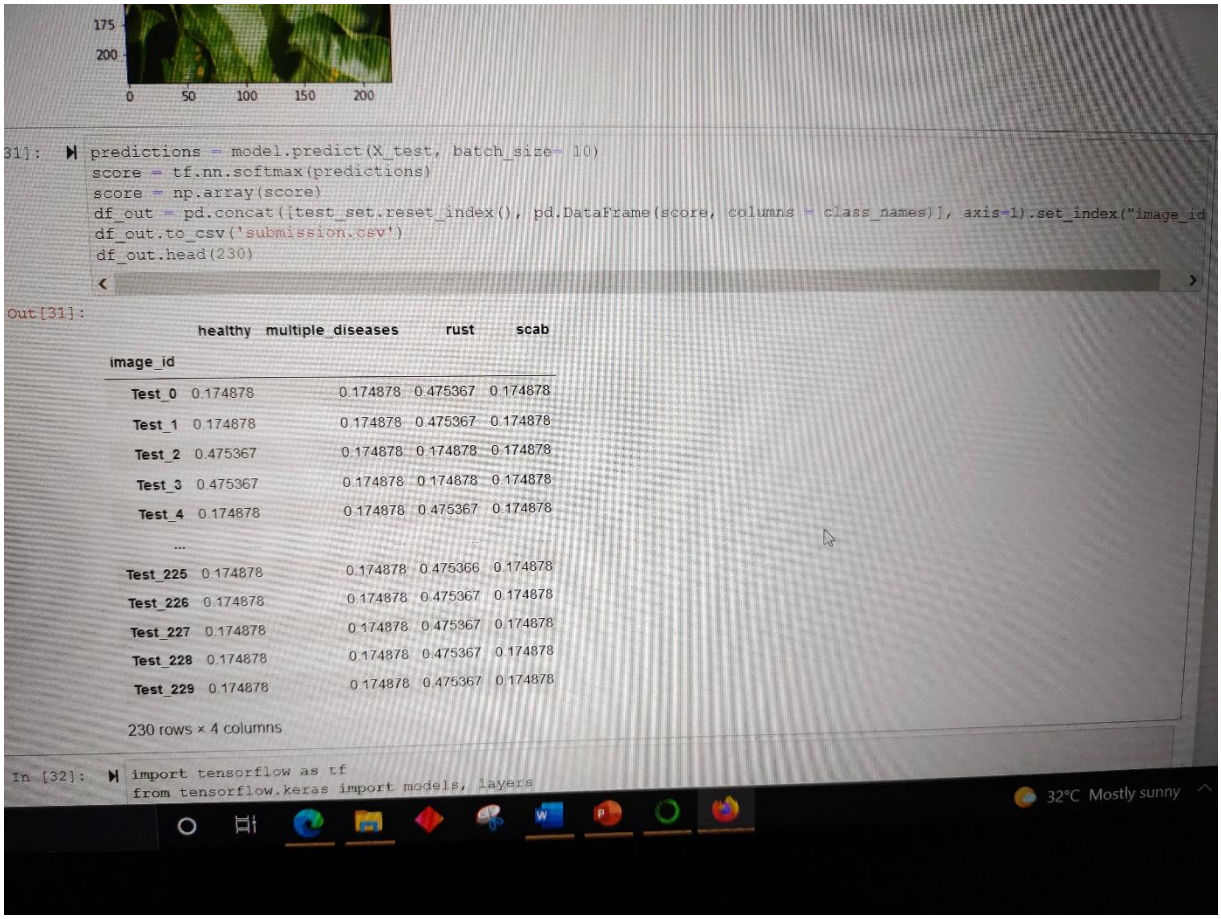
**Conclusion:**

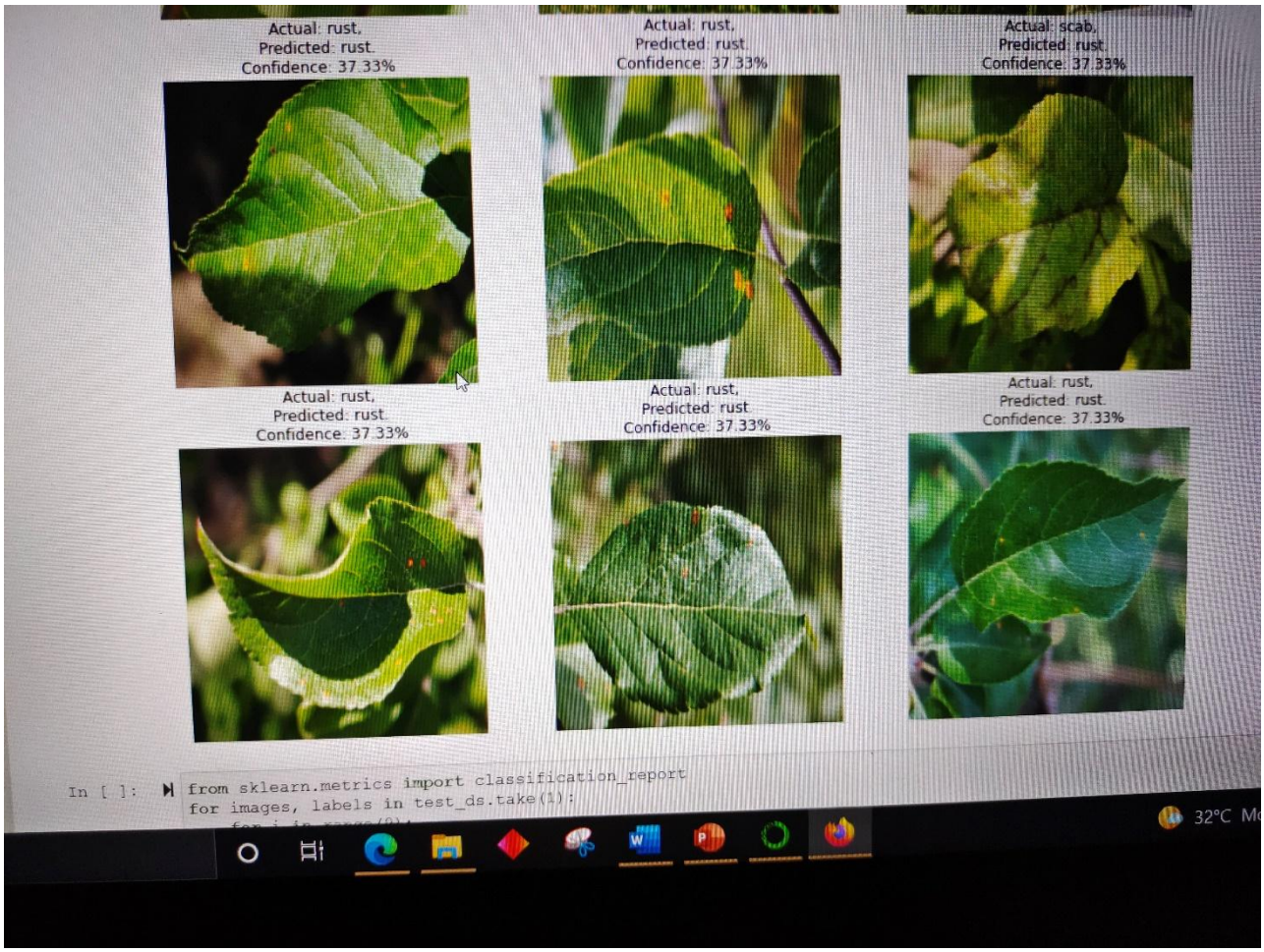
- As now a days due to climate and chemicals diseases in leaf became common so we need to aware of that particular disease before treating.so,that we can treat it accorndingly.Our model detect the disease in leaf with good accuracy.we predict the type of disease and percentange of chances of disease.

| Algorithm | Accuracy |
|---|---|
| Resnet | 94.60 |
| Vgg19 | 32.77 |
| Darknet | 33.20 |

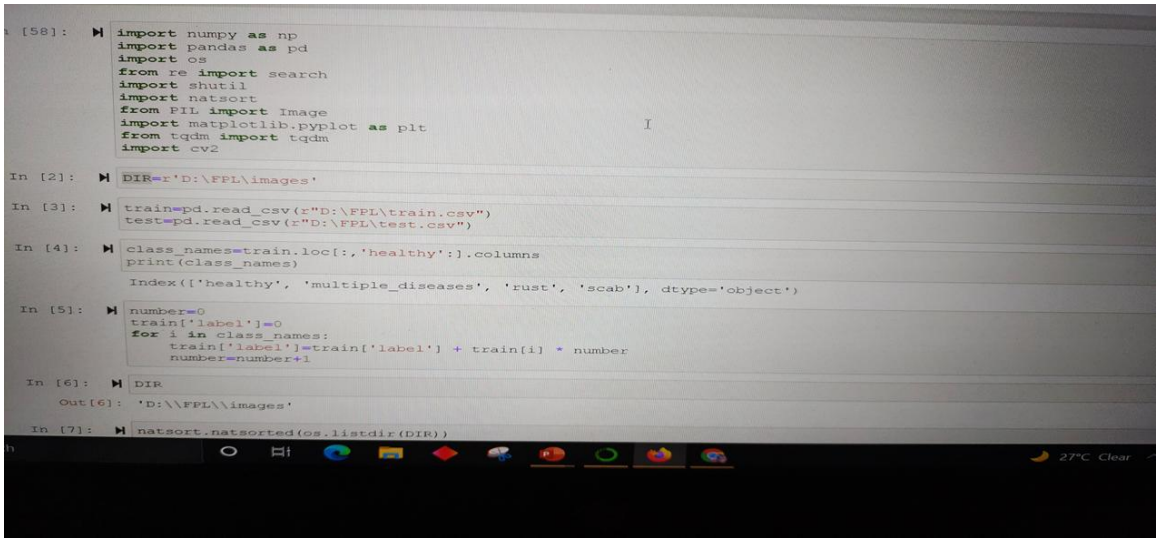
For 1st 5 test images

| | healthy | multiple_diseases | rust | scab |
|---|---|---|---|---|
| **image_id** | | | | |
| **Test_0** | 0.174878 | 0.174878 | 0.475367 | 0.174878 |
| **Test_1** | 0.174878 | 0.174878 | 0.475367 | 0.174878 |
| **Test_2** | 0.174878 | 0.174878 | 0.475367 | 0.174878 |
| **Test_3** | 0.475311 | 0.174887 | 0.174887 | 0.174916 |
| **Test_4** | 0.174878 | 0.174878 | 0.475367 | 0.174878 |



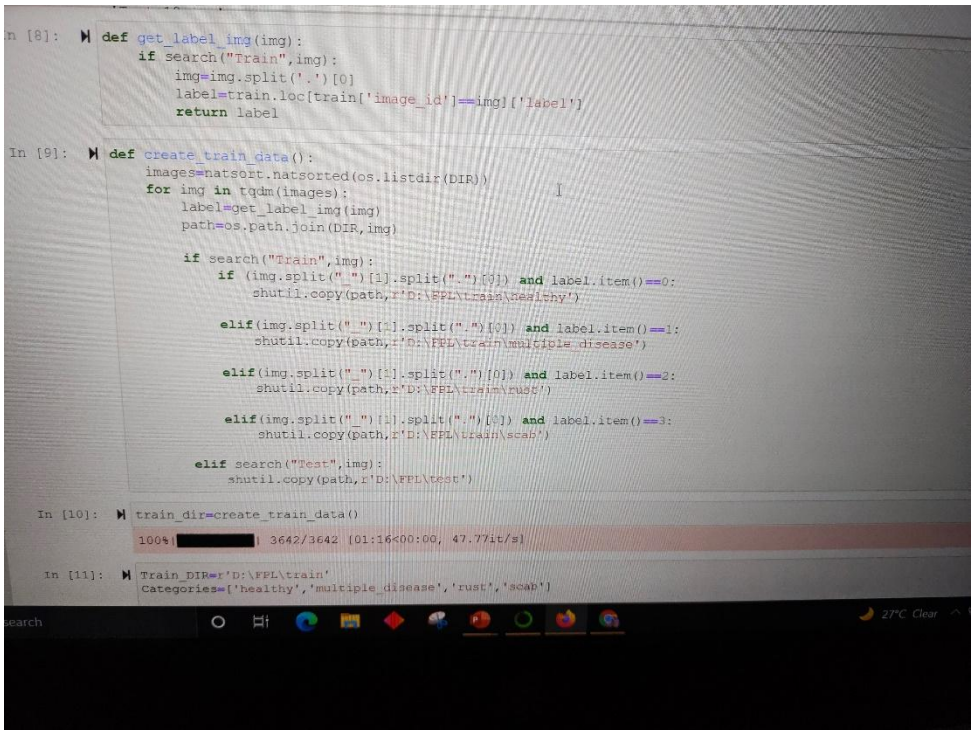

Actual: rust,
Predicted: rust.
Confidence: 37.33%
Actual: rust,
Predicted: rust.
Confidence: 37.33%
Actual: scab,
Predicted: rust.
Confidence: 37.33%
Actual: rust,
Predicted: rust.
Confidence: 37.33%
Actual: rust,
Predicted: rust.
Confidence: 37.33%
Actual: rust,
Predicted: rust.
Confidence: 37.33%
In [ ]:
from sklearn.metrics import classification_report
for images, labels in test_ds.take(1):
32°C

# Data collection:



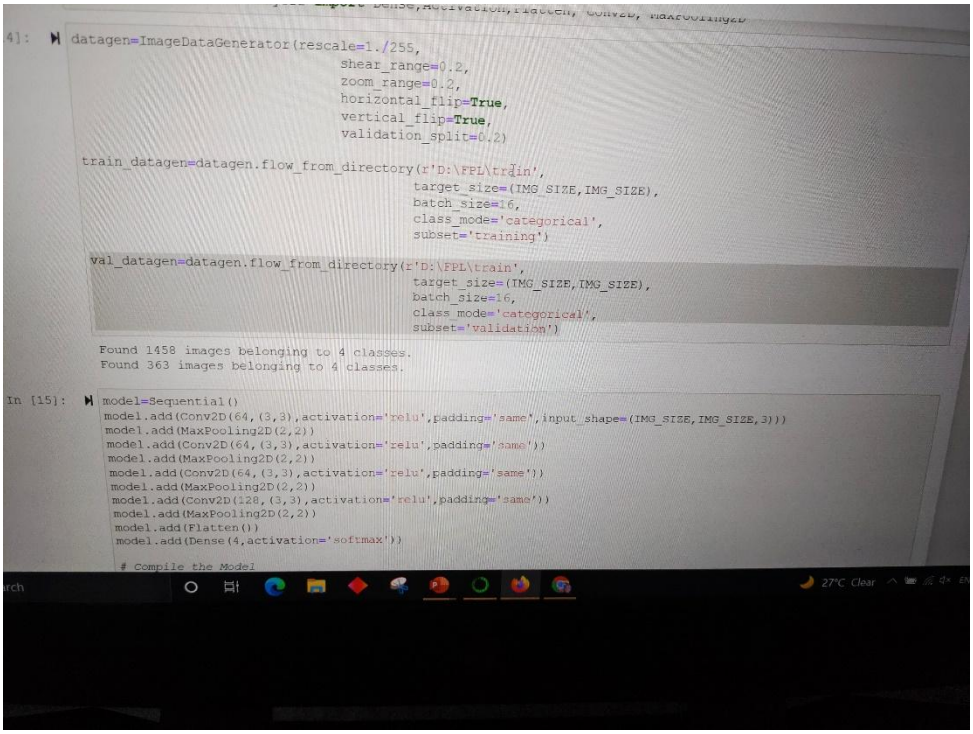
```
In [58]: import numpy as np
         import pandas as pd
         import os
         from re import search
         import shutil
         import natsort
         from PIL import Image
         import matplotlib.pyplot as plt
         from tqdm import tqdm
         import cv2

In [2]:  DIR=r'D:\FPL\images'

In [3]:  train=pd.read_csv(r"D:\FPL\train.csv")
         test=pd.read_csv(r"D:\FPL\test.csv")

In [4]:  class_names=train.loc[:,'healthy':].columns
         print(class_names)

         Index(['healthy', 'multiple_diseases', 'rust', 'scab'], dtype='object')

In [5]:  number=0
         train['label']=0
         for i in class_names:
             train['label']=train['label'] + train[i] * number
             number=number+1

In [6]:  DIR
Out[6]:  'D:\\FPL\\images'

In [7]:  natsort.natsorted(os.listdir(DIR))
```

# Data cleaning:



```
In [8]:  def get_label_img(img):
             if search("Train",img):
                 img=img.split('.')[0]
                 label=train.loc[train['image_id']==img]['label']
                 return label

In [9]:  def create_train_data():
             images=natsort.natsorted(os.listdir(DIR))
             for img in tqdm(images):
                 label=get_label_img(img)
                 path=os.path.join(DIR,img)

                 if search("Train",img):
                     if (img.split("_")[1].split(".")[0]) and label.item()==0:
                         shutil.copy(path,r'D:\FPL\train\healthy')

                     elif(img.split("_")[1].split(".")[0]) and label.item()==1:
                         shutil.copy(path,r'D:\FPL\train\multiple_disease')

                     elif(img.split("_")[1].split(".")[0]) and label.item()==2:
                         shutil.copy(path,r'D:\FPL\train\rust')

                     elif(img.split("_")[1].split(".")[0]) and label.item()==3:
                         shutil.copy(path,r'D:\FPL\train\scab')

                 elif search("Test",img):
                     shutil.copy(path,r'D:\FPL\test')

In [10]: train_dir=create_train_data()
         100%|[redacted]| 3642/3642 [01:16<00:00, 47.77it/s]

In [11]: Train_DIR=r'D:\FPL\train'
         Categories=['healthy','multiple_disease','rust','scab']
```

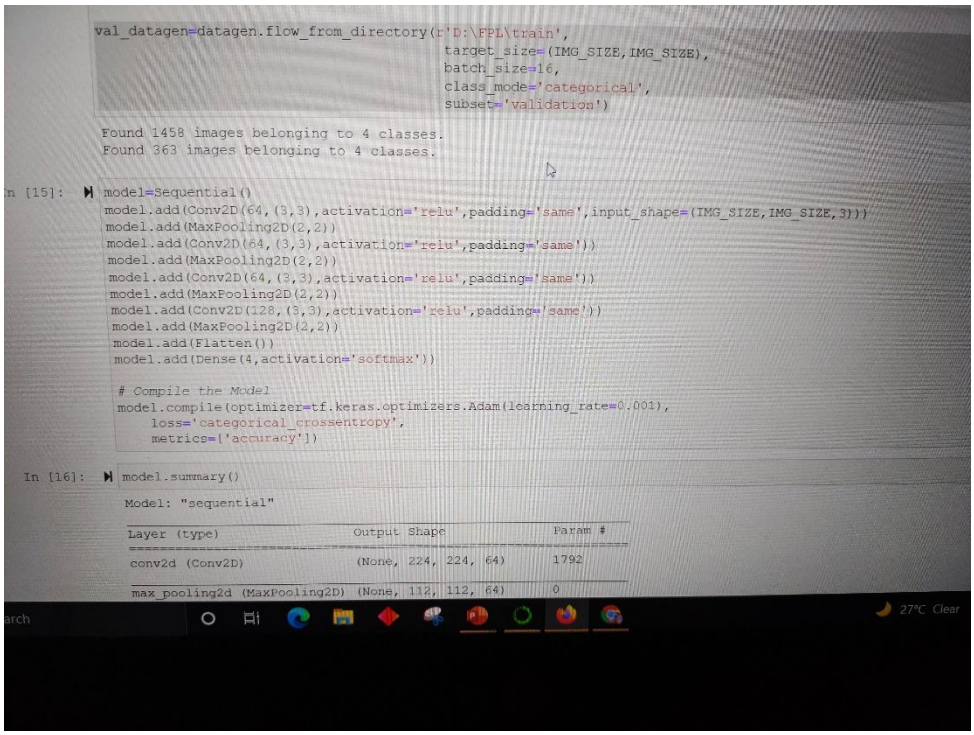
# Creating Model:



```
datagen=ImageDataGenerator(rescale=1./255,
                           shear_range=0.2,
                           zoom_range=0.2,
                           horizontal_flip=True,
                           vertical_flip=True,
                           validation_split=0.2)

train_datagen=datagen.flow_from_directory(r'D:\FPL\train',
                                          target_size=(IMG_SIZE,IMG_SIZE),
                                          batch_size=16,
                                          class_mode='categorical',
                                          subset='training')

val_datagen=datagen.flow_from_directory(r'D:\FPL\train',
                                        target_size=(IMG_SIZE,IMG_SIZE),
                                        batch_size=16,
                                        class_mode='categorical',
                                        subset='validation')
```

```
Found 1458 images belonging to 4 classes.
Found 363 images belonging to 4 classes.
```

```
model=Sequential()
model.add(Conv2D(64,(3,3),activation='relu',padding='same',input_shape=(IMG_SIZE,IMG_SIZE,3)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(4,activation='softmax'))

# Compile the Model
```



```
val_datagen=datagen.flow_from_directory(r'D:\FPL\train',
                                        target_size=(IMG_SIZE,IMG_SIZE),
                                        batch_size=16,
                                        class_mode='categorical',
                                        subset='validation')
```

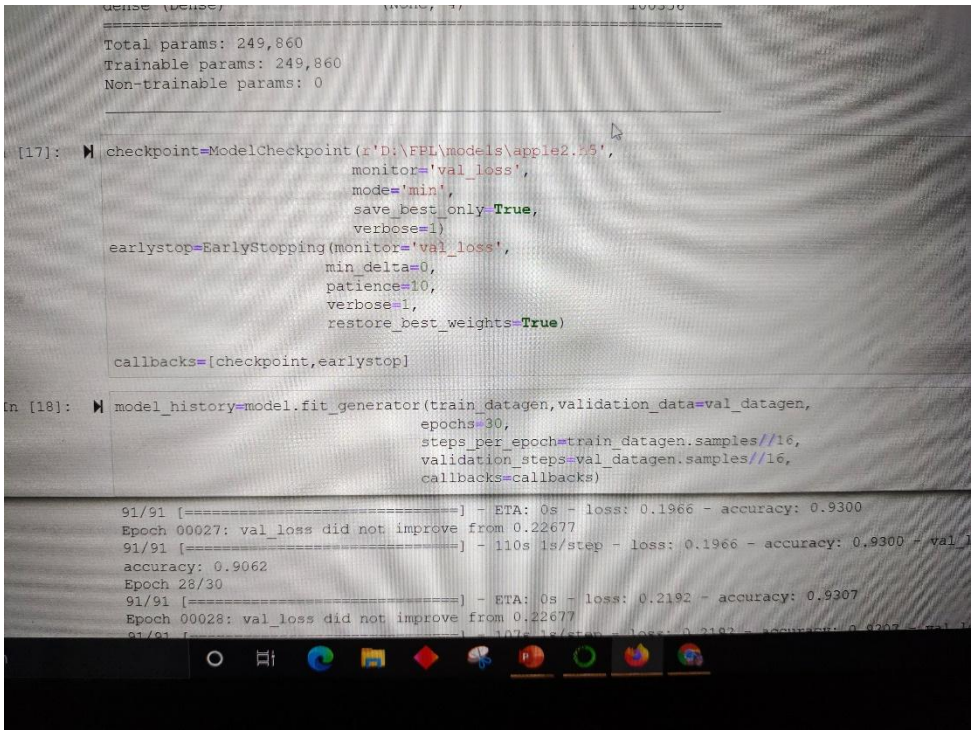
```
Found 1458 images belonging to 4 classes.
Found 363 images belonging to 4 classes.
```

```
model=Sequential()
model.add(Conv2D(64,(3,3),activation='relu',padding='same',input_shape=(IMG_SIZE,IMG_SIZE,3)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(4,activation='softmax'))

# Compile the Model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 64)      1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 64)      0
```

## Training the model:




## Performance of the model:

## Train Accuracy

## 92%

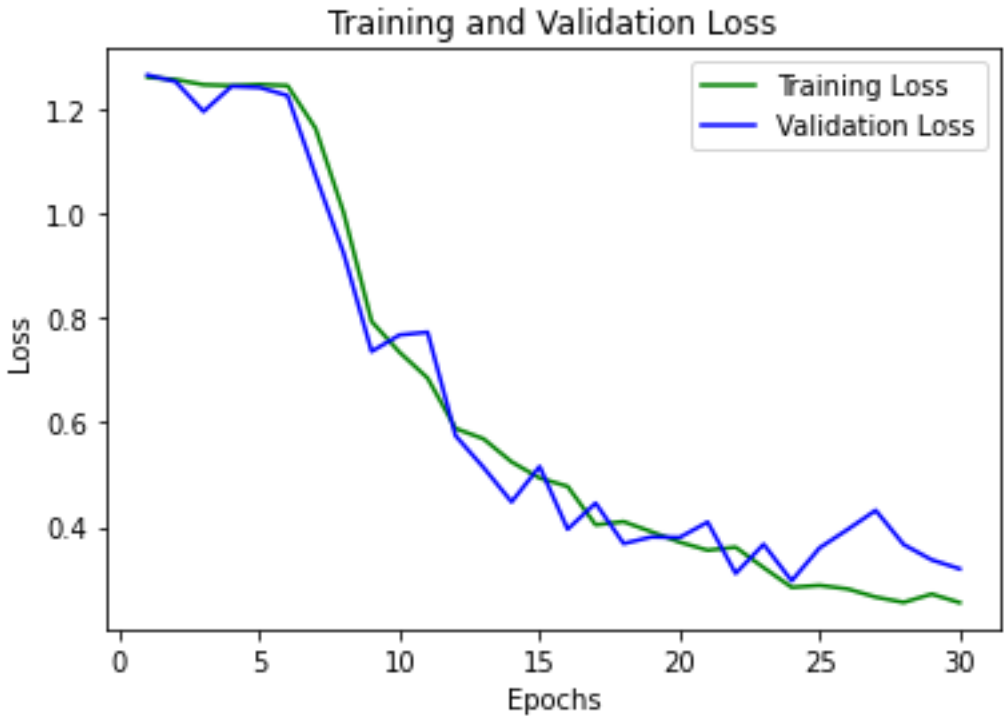
- - - - - - - - -

## Validation Accuracy

## 94%

**Graphs:**


Training Accuracy
Validation Accuracy


Training and Validation Loss
Training Loss
Validation Loss
Loss
1.2
1.0
0.8
0.6
0.4
0
5
10
15
20
25
30
Epochs

**Refernces:**


- **1) N.E.Abdullah,A.A. Rahim,Classification of Rubber Tree Leaf Diseases Using Multilayer Perceptron Neural Network, 5th Student Conference on Research and Development, DOI: 10.1109 ,2007, pp. 1 6.**
- **2) Adrian Ford, Alan Roberts, Colour Space Conversions, August11, 1998.**
- **3) H.Al-hiary, bani-ahmad, Fast and Accurate detection and classifica- tion of plant diseases, International Journal of Computer Applica- tions (0975-8887), March 2011, Volume 17-No.1.**
- **4)Ali Gholipour, Estroff JudyA, Barnewolt CarolE, Connolly SusanA, Warfield SimonK. Fetal brain volumetry through MRI volumetric reconstruction and segmentation. Int. J. Comput. Assist. Radiol. Surg. 2011;6(3):329–39.**
- **5)Anton-Rodriguez J. M, Peter Julyan, Ibrahim Djoukhadar, David Russell, D. Gareth Evans, Alan Jackson, and Julian C. Matthews, (2019) "Comparison of a Standard Resolution PET-CT Scanner With an HRRT Brain Scanner for Imaging Small Tumors Within the Head".**