A Mini Project Report on

Identification of Tomato Plant Disease Using Leaf Images

undergone at

Department of Computer Science and Engineering

under the guidance of

Asst. Prof. jeny rajan

Submitted by

G Sai Yeshwanth

192CS006

II Sem M.Tech (CS)

in partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science & Engineering

National Institute of Technology Karnataka, Surathkal.

May 2020

ABSTRACT

The tomato crop has high commercial value in the Indian market and produced in large quantities, and there are several kinds of diseases that these plants undergo which effects the growth of the plant. To ensure minimal loss to the crop its better we identify the diseases and treat it within time. So here a method is proposed to identify or classify the disease of plant using neural networks, more precisely convolution neural network, which is proving itself to be state of art model in recent times, images of the diseased plant are taken as input and we classify them. different models such as Le Net, Alex Net and inceptionv3 are used to train the images, all of them gave very high accuracy and inceptionV3 which is final proposed model achieved accuracy of 100% which shows it is best way to decide the type of diseases.

INDEX

1. INTRODUCTION

India is a country with a majority of the population relying heavily on the agricultural sector. Tomato is the most common vegetable used across India. The three most important antioxidants namely vitamin E, vitamin C and beta-carotene are present in tomatoes. They are also rich in potassium, a very important mineral for good health. Tomato crop cultivation area in India spans around 3,50,000 hectares approximately and the production quantities roughly sum up to 53,00,000 tons, making India the third largest tomato producer in the world. The sensitivity of crops coupled with climatic conditions have made diseases common in the tomato crop during all the stages of its growth. Disease affected plants constitute 10-30% of the total crop loss. Identification of such diseases in the plant is very important in preventing any heavy losses in yield as well as the quantity of the agricultural product. Monitoring the plant diseases manually is a difficult task due to its complex nature and is a time consuming process. Therefore, there is a need to reduce the manual effort put into this task, while making accurate predictions and ensuring that the farmers' lives are hassle free.

Visually observable patterns are difficult to decipher at a single glance, leading to many farmers making inaccurate assumptions regarding the disease. As a result, prevention mechanisms taken by the farmers may be ineffective and sometimes harmful. Farmers usually come together and implement common disease prevention mechanisms, as they lack expert advice on how to deal with their crop infestation. There have been circumstances where due to inadequate knowledge or misinterpretation regarding the intensity of the disease, over-dosage or under-dosage of the pesticide has resulted in crop damage. This is the underlying motivation for the proposed methodology that aims to accurately detect and classify diseases in the tomato crop.

The methodology suggested here pertains to the most common diseases found in the tomato plant like, Bacterial leaf spot and Septorial leaf spot, Yellow Leaf Curl among many others. Any leaf image given as input can be classified into one of the disease classes or can be deemed healthy. The database used for evaluation is a subset of Plant Village [1], a repository that contains 54,306 images of 14 crops infested with 26 diseases. The subset includes around 18160 images of tomato leaf diseases.

Broadly, the proposed methodology consists of three major steps: data acquisition, pre-processing and classification. The images used for the implementation of the proposed methodology were acquired from a publicly available dataset called Plant Village, as mentioned earlier. In the next step, the images were re-sized to a standard size before feeding it into the classification model. The final step is the classification of the input images with the use of a different deep learning convolutional neural network (CNN) standard model called the LeNet, AlexNet and InceptionV3 which consists of the convolutional, activation, pooling and fully connected layers

2. DATA SET

The dataset used here is taken from plant Village Dataset form Kaggle[1], it contains 18160 images of tomato leafs which are diseased and categorized into 10 different types of diseases

• Tomato_Bacterial_spot	2127 images
• Tomato_Early_blight	1000 images
• Tomato_Late_blight	1909 images
• Tomato_Leaf_Mold	952 images
• Tomato_Septoria_leaf_spot	1771 images
• omato_Spider_mites_Two_spotted_spider_mite	1676 images
• Tomato__Target_Spot	1404 images
• Tomato__Tomato_Yellow_curl_virus	3209 images
• Tomato__Tomato_mosaic_virus	373 images
• Tomato_healthy	1951 images

These images are resized to 256*256*3 pixels before giving them to model for training

3. PROPOSED METHODOLOGY

The proposed approach includes the three important stages namely: Data Acquisition, Data pre-processing and Classification. The project done on google celebratory. Flow diagram is shown in Fig. 1 and current section includes the brief discussions of the same.

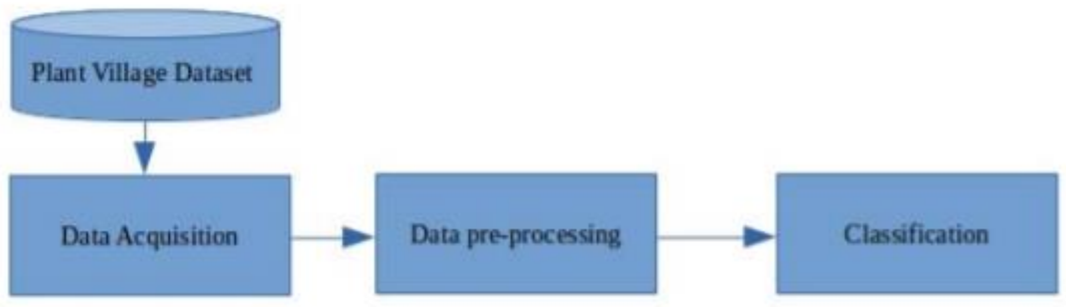


Figure 1: Proposed methodology

3.1 Data Acquisition

The tomato leaf disease images have been taken from the Plant Village repository [1]. Images for the diseases were downloaded and uploaded into drive. The acquired dataset consists of around 18160 images belonging to 10 different classes. The dataset includes images of all major kinds of leaf diseases that could affect the tomato crop. Each of the downloaded images belongs to the RGB color space by default and were stored in the uncompressed JPG format.

3.2 Data Pre-processing

The acquired dataset consisted of images with minimal noise and hence noise removal was not a necessary preprocessing step. The images in the dataset were resized to 256 × 256*3 resolution in order to speed up the training process and make the model training computationally feasible. The process of standardizing either the input or target variables tends to speed up the training process. This is done through improvement of the numerical condition of the optimization problem. It is also made sure that the several default values involved in initialization and termination are appropriate. For our purpose, we normalize the images to get all the pixel values in the same range by using the mean and the standard deviation. In machine learning terms, it is called as the Z-score.

3.3 Classification

Convolutional neural networks (CNN) can be used for the creation of a computational model that works on the unstructured image inputs and converts them to corresponding classification output labels. They belong to the category of multi-layer neural networks which can be trained to learn the required features for classification purposes. They require less pre-processing in comparison to traditional approaches and perform automatic feature extraction which gives better performance. For the purpose of tomato leaf disease detection, we have experimented with several standard deep learning architectures like LeNet[2],AlexNet [3], GoogleNet(Inception) [4] and the best results could be seen with the use InceptionV3 architecture [4].

The architectures used for the classification of the tomato leaf diseases consists of convolutional, activation and pooling layers. Convolutional and pooling layers are used for feature extraction whereas the fully connected layers are used for classification. Activation layers are used for introducing non-linearity into the network. Convolutional layer applies convolution operation for extraction of features. With the increase in depth, the complexity of the extracted features increases. The max pooling layer is used for reduction in size of the feature maps, speeding up the training process, and making the model less variant to minor changes in input. The kernel size for max pooling is 2×2. ReLU activation layer is used in each of the blocks for the introduction of non-linearity. Also, Dropout regularization technique has been used to avoid overfitting the train set. Dropout regularization randomly drops neurons in the network during each iteration of training in order to reduce the variance of the model and simplify the network which aids in prevention of overfitting. Finally, the classification block consists of two sets fully connected neural network layers. The second dense layer is followed by a softmax activation function to compute the probability scores for the ten classes.

For all these models training set and validation set combined 66% and testing set 33% and all are trained for 20 epochs and one of Adam or sgd as optimizer

3.2.1 LeNet Architecture

Le-Net [2] is a simple CNN model that consists of convolutional, activation, pooling and fully connected layers. LeNet uses sigmoid function as its activation function and has 5*5*x convolution layers, it uses Average pooling of size 2*2*it is 5 layers and whole architecture is summarized in below figure 2.

LeNet-5 comprises 5 layers (excluding input layer), all of which contains trainable parameters.

• The input is a 32 x 32-pixel image. The values of the input pixels are normalized so that the background level

(white) corresponds to a value of -0.1 and the foreground (black) corresponds to 1.175.

• Convolutional layers are labeled Cx, sub sampling layers are labeled Sx and fully connected layers are labeled Fx, where x is the layer index.

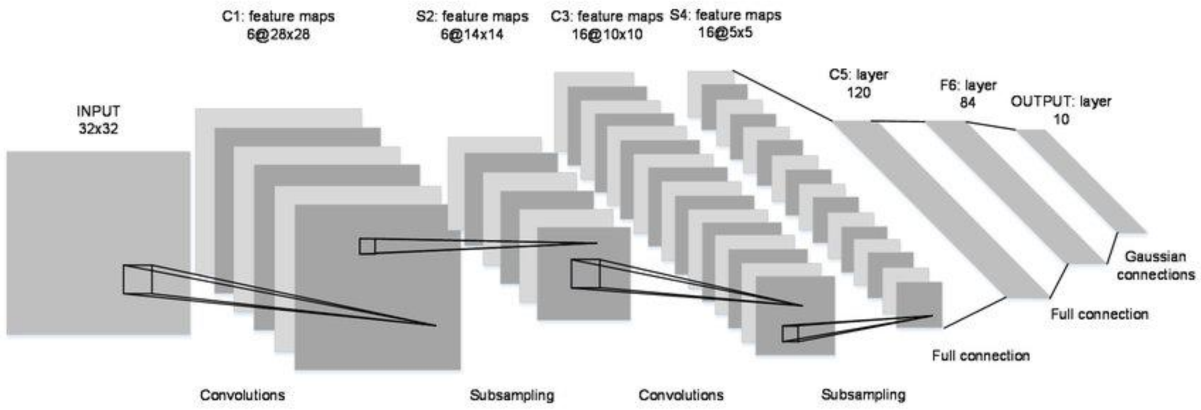


Figure 2: LeNet Architecture

Layer C1

• Convolutional layer with 6 feature maps (6 filters)

• Convolutional filter size: 5 x 5

• Feature map size (image size after convolution): 28 x 28

• Total trainable parameters: 156 (5 x 5 x 6 = 150w + 6b = 156)

Layer S2

• Subsampling layer (pooling layer)

• Each unit in each feature map is connected to a 2 x 2 neighborhood in the

corresponding feature map in C1.

• The four inputs to a unit in S2 are added, then multiplied by a trainable

coefficient, and added to a trainable bias.

• The result is passed through a tanh function.

• The 2 x 2 receptive fields are non-overlapping, hence the output will be half the

size of the input.

• Layer S2 has 12 trainable parameters (6w+6b)

Layer C3

• Convolutional layer with 16 feature maps (16 filters)

• Convolutional filter size: 5 x 5 x Z

• The first six C3 feature maps take inputs from every contiguous subsets of three feature maps in S2 (filter size: 5 x 5 x 3)

• The next six take input from every contiguous subset of four (filter size: 5 x 5 x 4)

• The next three take input from some discontinuous subsets of four (filter size: 5 x 5 x 4)

• The last one takes input from all S2 feature maps (filter size: 5 x 5 x 6)

• Total trainable parameters: 1516 (450w + 600w + 300w + 150w + 16b)

Layer S4

• Subsampling layer (pooling layer) with 16 feature maps of size 5 x 5

• Each unit in each feature map is connected to a 2 x 2 neighborhood in the corresponding feature map in C3, in a similar way as C1 and S2.

• Layer S4 has 32 trainable parameters (16w+16b)

Layer C5

• Convolutional layer with 120 feature maps (120 filters)

• Convolutional filter size: 5 x 5 x 16

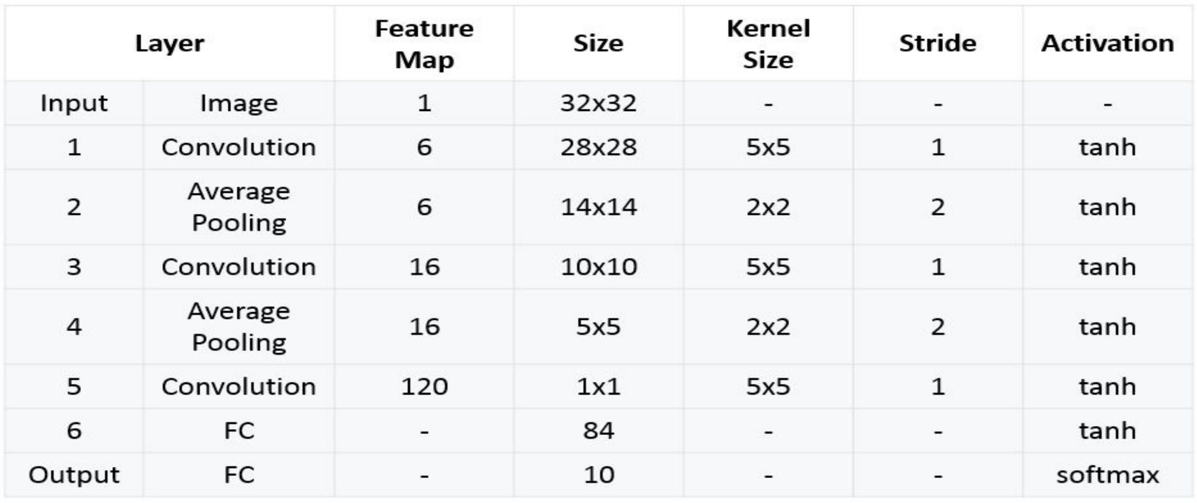
• Total trainable parameters: 48120 (48000w+120b)

Layer F6

• Contains 84 neurons and is fully connected to C5.

• Layer F6 has 10164 trainable parameters (10080 w + 84 b)

And the details of convolution layers and pooling layers are summarized in below figure 3



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Figure 3 Lenet Details

This model gave an accuracy of 97.7 % for 2 classes

3.2.2 AlexNet Model

AlexNet[3] is more advanced model compared to LeNet

AlexNet won the 2012 ImageNet LSVRC-2012 competition by a large margin (15.3% VS 26.2% (second place) error

rates).

• First successful CNN application for such a big dataset.

• AlexNet is considered one of the most influential papers published in computer vision.

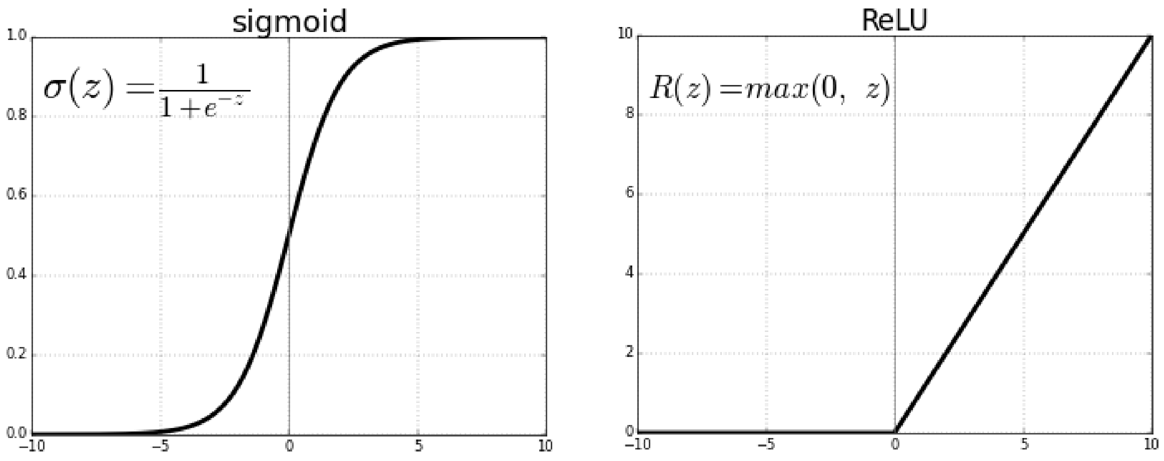


Figure 4: sigmoid vs relu activation function

Advantages of Alexnet over LeNet

1. AlexNet uses Relu Activation function which is 6 times faster compared to sigmoid used in LeNet.
2. AlextNet uses Max pooling and overlapped polling which has more advantages than average pooling.
3. AlexNet uses concepts like dropout to avoid overfitting.

Here is the Architecture of Alexnet

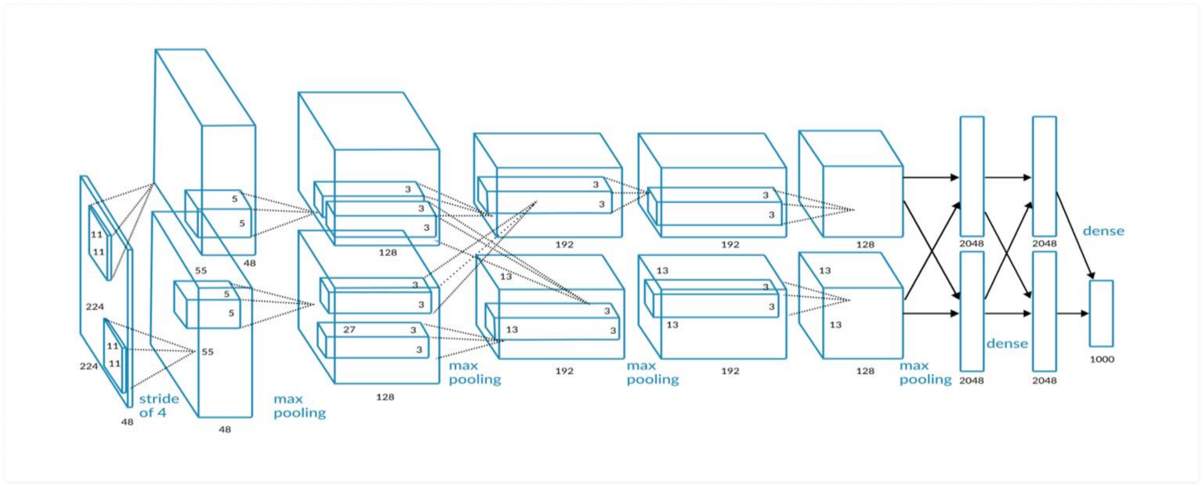
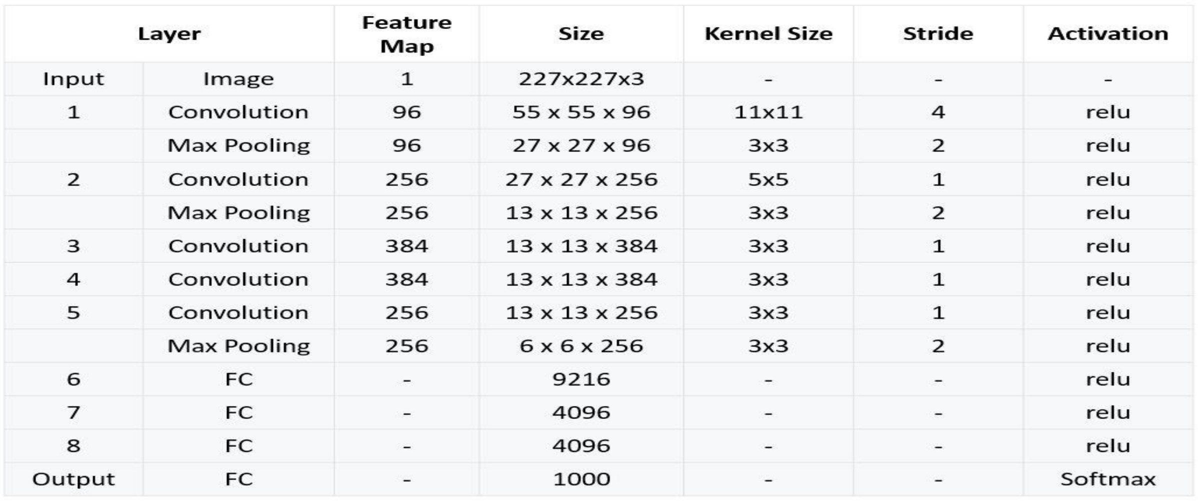


Figure 5: Alexnet Architecture



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Figure 6: Alexnet Details

AlextNet achieved accuracy of 97.9% for 2 classes

3.2.3 InceptionV3

Inception v3 [4] is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via Softmax.

A high-level diagram of the model is shown below:

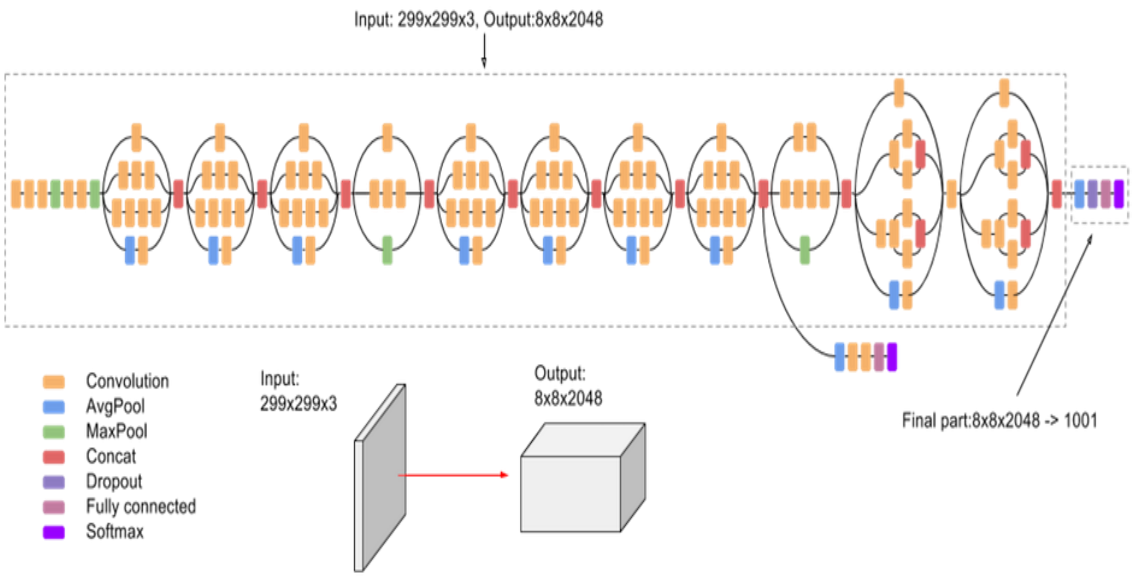


Figure 7: InceptionV3 architecture

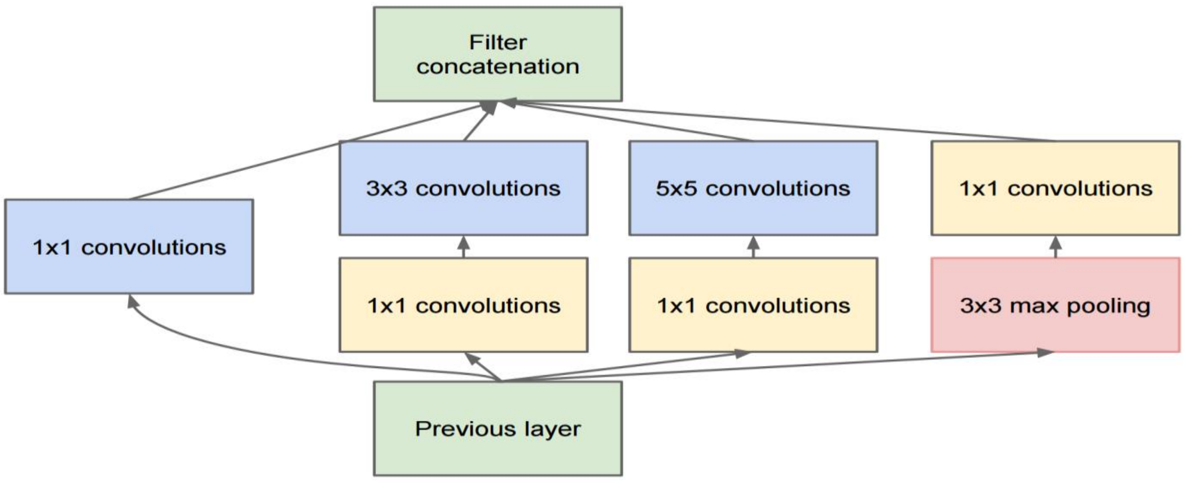
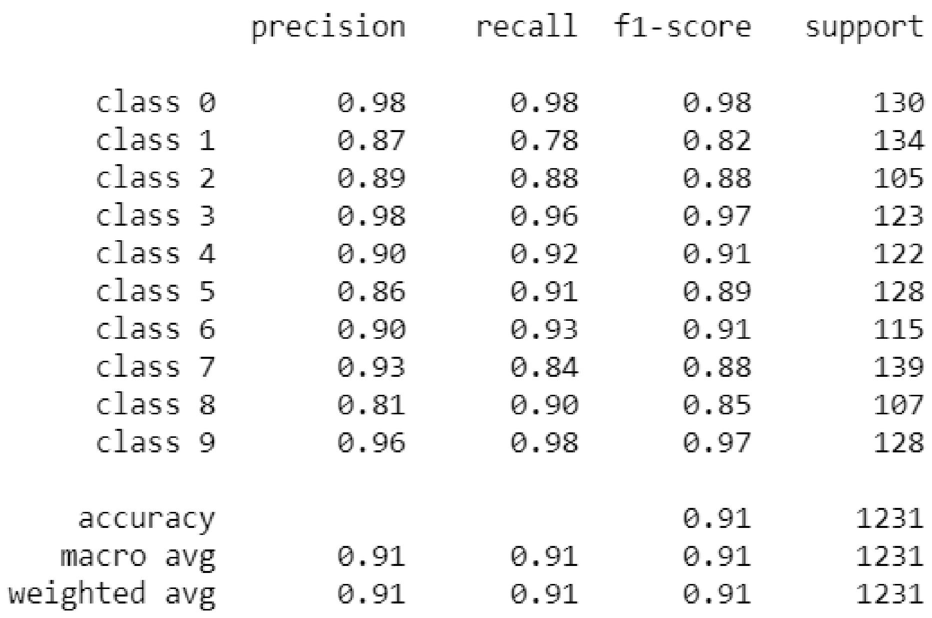


Figure 8: building block of inception v3

This inceptionV3 model is taken from the Keras and used for our problem based on Transfer learning. Transfer learning allows you to retrain the final layer of an existing model, resulting in a significant decrease in not only training time, but also the size of the dataset required. One of the most famous models that can be used for transfer learning is Inception V3. As mentioned above, this model was originally trained on over a million images from 1,000 classes on some very powerful machines. Being able to retrain the final layer means that you can maintain the knowledge that the model had learned during its original training and apply it to your smaller dataset, resulting in highly accurate classifications without the need for extensive training and computational power. Inception gave an accuracy of 100% accuracy for 2 classes and 91% for 10 classes. All models can be checked at GitHub link [5] that is provided in references

4. RESULTS

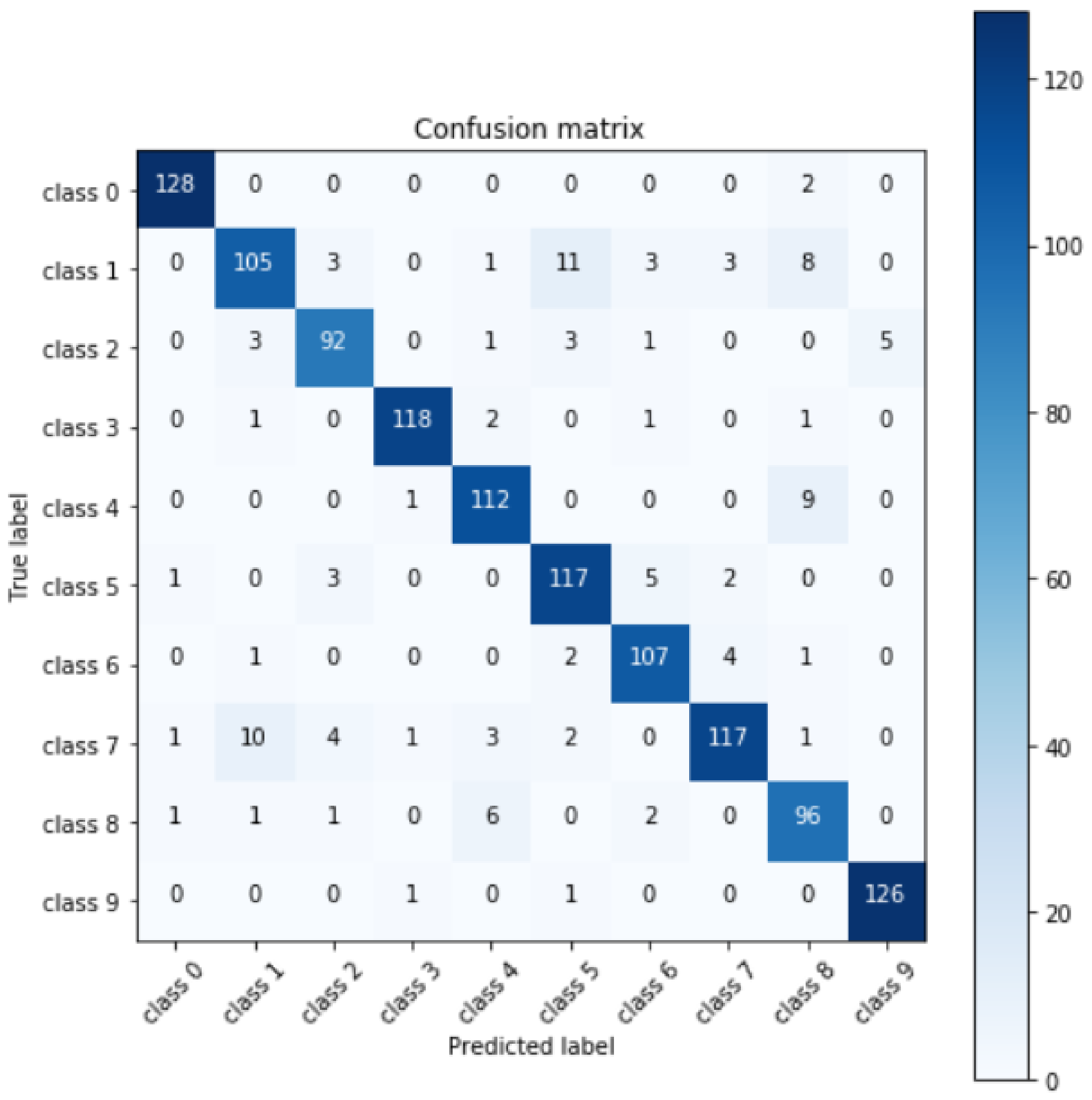


```
              precision    recall  f1-score   support

     class 0       0.98      0.98      0.98       130
     class 1       0.87      0.78      0.82       134
     class 2       0.89      0.88      0.88       105
     class 3       0.98      0.96      0.97       123
     class 4       0.90      0.92      0.91       122
     class 5       0.86      0.91      0.89       128
     class 6       0.90      0.93      0.91       115
     class 7       0.93      0.84      0.88       139
     class 8       0.81      0.90      0.85       107
     class 9       0.96      0.98      0.97       128

    accuracy                           0.91      1231
   macro avg       0.91      0.91      0.91      1231
weighted avg       0.91      0.91      0.91      1231
```

Figure 9: accuracy of inceptionV3



	class 0	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	class 9
class 0	128	0	0	0	0	0	0	0	2	0
class 1	0	105	3	0	1	11	3	3	8	0
class 2	0	3	92	0	1	3	1	0	0	5
class 3	0	1	0	118	2	0	1	0	1	0
class 4	0	0	0	1	112	0	0	0	9	0
class 5	1	0	3	0	0	117	5	2	0	0
class 6	0	1	0	0	0	2	107	4	1	0
class 7	1	10	4	1	3	2	0	117	1	0
class 8	1	1	1	0	6	0	2	0	96	0
class 9	0	0	0	1	0	1	0	0	0	126

Figure 10: Confusion matrix after classification

5. CONCLUSION

Agricultural sector is still one of the most important sector over which the majority of the Indian population relies on. Detection of diseases in these crops is hence critical to the growth of the economy. Tomato is one of the staple crops which is produced in large quantities. Hence, this work aims at detection and identification of 10 different diseases in the tomato crop. The proposed methodology uses a convolutional neural network model to classify tomato leaf diseases obtained from the Plant Village dataset. The architecture used is a state of art model called InceptionV3 convolutional neural network to classify the tomato leaf diseases into 10 different classes. Thus, the above mentioned model can be made use of as a decision tool to help and support farmers in identifying the diseases that can be found in the tomato plant. With an accuracy of 91% the methodology proposed can make an accurate detection of the leaf diseases with little computational effort.

6. REFERENCES

[1] https://www.kaggle.com/emmarex/plantdisease?

[2] http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

[3]https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[4]https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/44903.pdf

[5] https://github.com/gundasai/deeplearning