**DETECTION OF CROP DISEASE**

**USING CONVOLUTION**

**NEURAL NETWORK (CNN)**

A PROJECT(I) REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF TECHNOLOGY

IN

DIVISION OF INFORMATION TECHNOLOGY

Submitted by:

**ANSHU YADAV**

**(2016UIT2599)**

**TANISHA RASTOGI**

**(2016UIT2609)**

Under the supervision of

**Dr. PRITI BANSAL**



DIVISION OF INFORMATION TECHNOLOGY

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF DELHI

DECEMBER, 2019

**CERTIFICATE**



Department of Information technology

University of Delhi

Delhi-110007, India

This is to certify that the Project(I)- Report titled “**Detection of Crop Disease Using Convolution Neural Network (CNN)**” which is being submitted by **Anshu Yadav** (2016UIT2599) and **Tanisha Rastogi** (2016UIT2609) to the **Department of Information Technology, Netaji Subhas Institute of Technology, Delhi (University of Delhi)** in partial fulfillmentof the requirement for the award of the degree of Bachelor of Technology, is a record of thework carried out by the students under my supervision and guidance. The content in the reportis original and not copied from source without proper citation. This work has not previouslyformed the basis for the award of any Degree.

Place: Delhi (DR. PRITI BANSAL)

Date: SUPERVISOR

**ACKNOWLEDGEMENT**

Our project work was very time consuming and proceeded with small steps towards the goal completion. Sometimes these steps lead us close to the solution and sometimes a bit away. Also, the project work was full of research and experimental work. So, first of all we are thankful to God for helping us to have patience during the project.

We would like to express our sincere gratitude and indebtedness to our mentor, Dr. Priti Bansal for her invaluable guidance and enormous help and encouragement, which helped us to complete our project successfully. Her way of working was a constant motivation throughout the project term. This project would not have been possible without her constant supervision.

ANSHU YADAV (2016UIT2599)

TANISHA RASTOGI (2016UIT2609)

# **DECLARATION**

We hereby declare that the work presented in the report entitled: **Detection of crop disease using Convolution Neural Network (CNN)** submitted by us in partial fulfilment of the requirements for the degree of Bachelor of Technology in Information Technology at Netaji Subhas Institute Technology, Dwarka, Delhi, is an authentic record of our work carried out under guidance of Dr. Priti Bansal. Due acknowledgments have been given in the report to all the materials used. This work has not been submitted anywhere else for the reward of any other degree.

## Anshu Yadav (2016UIT2599)

Tanisha Rastogi (2016UIT2609)

**ABSTRACT**

A substantial loss in plant’s yield is due to the plant diseases. In turn of loss due to yield of plants we also face a huge economic losses. However, accurate recognition and timely diagnosis of these disease may prove to be a life savior. Pre diagnosis of plant disease becomes very important in the era of constantly changing climate and globalization. It is important to ensure food security as well as to save plants from spread of invasive pests and pathogens. Traditionally the detection of diseases was carried out through manual methodologies. Different parts of a plant serve as a medium to study over them and to carry out diagnosis of different types of diseases affecting different plant parts respectively.

With the advancement in science and technology, we have developed a lot methods to train machines to perform tasks that were carried by human in past. Some of the well known field of study in the era of training machines to perform like human and even better than them are Artificial Intelligence, Deep Learning, Neural Networks, Computer Vision and many more.

In this paper we have used Convolution Neural Network to train and validate our machine on plant village dataset. Convolution Neural Network is a branch of Deep Neural Networks which processes in two stages that are feature extraction and training or classification. It is much faster and efficient compared to other methods of image classification due to its property of feature extraction via machine and then training via Artificial Neural Network.

Plant Village dataset contains images of tomato, potato and pepper bell leaves. Out of this the project work is based on tomato plant. The project has been developed with the motive of classifying tomato leaves into different type of diseases (10 classes as per dataset). In order to make the system efficient, two models have been developed that are binary model and multiclass model. Binary model distinguishes between tomato and non-tomato leaf. On the other hand multiclass model try to identify the type of disease if the tomato leaf is not healthy and labels it healthy if it falls under the category of healthy leaf. Also, another model has been tried to assign both the labels to an image at the same time. This model is called multilabel classifier.

**INDEX**

CERTIFICATE i

ACKNOWLEDGEMENT ii

DECLARATION iii

ANSTRACT iv

INDEX v

LIST OF FIGURES vi

LIST OF TABLES vii

**CHAPTER 1**

* 1. Introduction
  2. Motivation of the work
  3. Methodology of the work
  4. Application of the work

**CHAPTER 2**

2.1 Data Set Breakdown

2.2 Work Flow

2.3 Experimental Results

REFERENCES

PLAGIARISM REPORT

**LIST OF FIGURES**

1. Graph representation of accuracy values of reviewed paper
2. Samples plant village Image Dataset
3. Basic workflow of methodology used
4. Resize operation over dataset
5. Image pre-processing technique applied to the dataset
6. Result of Binary Classifier (Model 1)
7. Result of Multi-class Classifier (Model 2)
8. Result of Multi-label Classifier (Model 3)
9. Model Architecture (Model 1 & Model 2)

**LIST OF TABLES**

1. Tabular list of names of reviewed paper, their methodology and future works.
2. List of reviewed papers with accuracy value and used methods.

**CHAPTER 1 INTRODUCTION**

* 1. **BACKGROUND**

Plants are among the important elements of human life. Existing diseases disrupt the growth of plants and cause economic, social and ecological losses. Most of them, produce some form of manifestation in the visible portions of plants. Correct recognition of diseases when they first appear, is a crucial step for effective disease management. In most cases, human experts identify diseases visually, who may be efficient in the recognition and quantification of diseases, but, they are engaged with some difficulties that may harm their efforts. In this context, diagnosing diseases in an exact and timely way is of the great importance. There are various types of diseases that harm the quality of tomatoes. Some of them have visible symptoms on the plant leaves such as Bacterial Spot. The disease starts by touching the leaves and yields of the tomato plants and continues, resulting in complete defoliation and sun scalded fruits. As the disease extends leaves and arts appear as they were slightly burned, foliage turns yellowish and dies, with severe defoliation exposing fruits and stems. Other types of tomato diseases that affect leaves, include Early Blight, Late Blight, Septoria leaf spot and etc.

In this project, an application of Convolutional Neural Networks (CNNs) with a customized architecture, in the leaf disease recognition is experimented. Specifically a CNN architecture, is proposed to classify leaves of tomato plants, infected with various diseases, including Bacterial Spot, Early Blight, Late blight, Septoria leaf spot, Spider mites (Two-spotted spider mite), Tomato mosaic virus, Leaf Mold, Target Spot and Tomato Yellow Leaf Curl disease. The used dataset is the Plant Village image dataset [1], including 16012 images of tomato leaves and 4627 images of other crop leaves. These images are resized to 128×128 pixels which is a fraction of the average size of all images. The size is chosen to minimize the amount of squash and stretch in images. First a classification model based on CNN is prepared to distinguish tomato leaves from other crop leaves. Second CNN architecture is used to distinguish the type of tomato crop disease along with healthy ones. This sum up to a multiclass CNN model, where total number of classes are 10. The accuracy on the validation set is tracked to choose the best state of the trained architecture and the parameters of the model.

* 1. **MOTIVATION OF THE WORK**
* Field of study i.e. Detection of crop disease. It is one of the concerned field of study as around 60 percent of world’s population depends on agriculture for their survival and around 70 percent of India’s population is involved in the profession of farming. Clearly, pre-diagnosis of crop disease helps in increasing crop productivity and thus increase in economic growth of an individual and the Nation.
* Related Work in form of a research paper and scope of different methodology for the same.
* Accuracy of Convolution Neural Network over any other methodology.
  1. **METHODOLOGY OF THE WORK**

There exist a number of methods which can be used to automate the detection of crop disease. Some of the methods are Image Processing, Convolution neural network, Image Segmentation along with a classification algorithm and so on. A comparison between different methodologies has been drawn below to compare their scope and accuracy.

***Table 1.1:*** *Tabular list of names of reviewed paper, their methodology and future works [2-5]*

|  |  |  |
| --- | --- | --- |
| **Paper** | **Methodology** | **Future Work** |
| 1.Detecting Jute Plant Disease  Using Image Processing and  Machine Learning [2] | Color co-occurrence methods,  Multi SVM classifier. | Nil |
| 2.Detection and measurement  of paddy leaf disease symptoms  using image pro-cessing. [3] | ANN, FUZZY classification, SVM,  K-means algorithm, color  co-occurrence meth-od. | It evaluates the techniques in image  processing, detecting diagnosing of  crop leaf disease. |
| 3.Detection of leaf disease and  classifi-cation using digital image  processing. [4] | GLCM, SVM, K-means | Classifying different plant disease and  improve the classification accuracy. |
| 4.Deep Residual Learning for  Tomato Plant Leaf Disease  Identification [5] | CNN- Deep Residual Learning  Method from scratch, Pre-trained CNN (Vgg16, Vgg19) | Devising methods to automate the  assessing of CNN settings instead of  carrying it out empirically. |

***Table 1.2 :*** *List of reviewed papers with accuracy values and used methods.*

|  |  |  |
| --- | --- | --- |
| **Paper Number** | **Methods** | **Accuracy Value** |
| Paper 1 | Color co-occurrence meth-od, Multi SVM classifier | 86% |
| Paper 2 | ANN, FUZZY classifica-tion, SVM, K-means algo-rithm,  Color co-occurrence method. | 94.70% |
| Paper 3 | GLCM, SVM, K-means | 90% |
| Paper 4 | CNN- Deep Residual Learning Method from scratch,  Pre-trained CNN (Vgg16, Vgg19) | 97.53% |

***Figure 1.1:*** *Graph Representation of accuracy values of reviewed paper*

From the comparison drawn based on the different papers, it can be concluded that Convolution Neural Network is the most accurate technique applicable to the problem of detection of crop disease. Based on the conclusion, the method was chosen for the work. Finally the paper referred for further working is the paper named Deep Residual Learning for Tomato Plant Leaf Disease Identification.

* 1. **APPLICTAION OF THE WORK**

Training a model based on convolution neural network has helped a lot in pre-diagnosis of crop disease. The considered field for study is itself an application of Deep Neural Networks in Farming Sector. The work carried under the project is mainly concerned with the detection of disease in tomato crop. However, the same methodology can be applied to any other crop disease detection.

**CHAPTER 2 PROJECT WORK**

**2.1. DATASET BREAKDOWN**

Referred Dataset is Plant Village Dataset. It consist of total 20639 images of different crop leaves. This image dataset covers 3 different crop leaf images that are Tomato, Pepper and Potato. Along with this there is a subdivision between different types of diseased and healthy leaves under each category. The full break down of dataset is shown in below,

* **Pepper\_\_bell\_\_\_Bacterial\_spot ---997**
* **Pepper\_\_bell\_\_\_healthy            -- 1478**
* **Potato\_\_\_Early\_blight            --- 1000**
* **Potato\_\_\_healthy                    --- 152**
* **Potato\_\_\_Late\_blight                  --- 1000**
* **Tomato\_\_Target\_Spot                  --- 1404  - - - - - - - - - - -- -|**
* **Tomato\_\_Tomato\_mosaic\_virus  --- 373                                 |**
* **Tomato\_\_Tomato\_YellowLeaf\_\_Curl\_Virus   --- 3209                |**
* **Tomato\_Bacterial\_spot                    --- 2127                            |**
* **Tomato\_Early\_blight                        --- 1000                          | ->16012**
* **Tomato\_healthy                                    --- 1591                       |**
* **Tomato\_Late\_blight                                --- 1909                      |**
* **Tomato\_Leaf\_Mold                                  --- 952                       |**
* **Tomato\_Septoria\_leaf\_spot                    --- 1771                     |**
* **Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite   --- 1676  - - - - - - - - - - |**

**-------------------------------------------------------**

**20639**

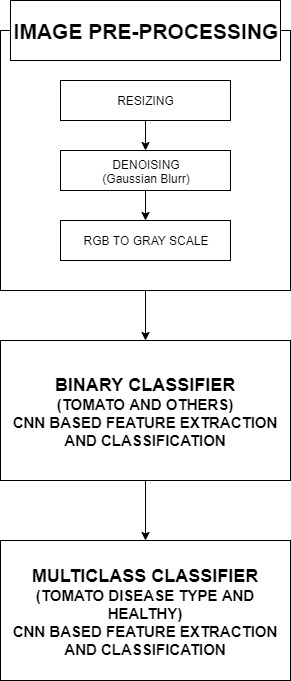
**------------------------------------------------------**

** **

Healthy Plant Leaf Images Diseased Plant Leaf Images

***Figure 2.1 Samples of Plant Village Image Dataset***

**2.2. WORK FLOW**



***Figure 2.2. Basic Workflow of Methodology Used***

Inspired from the work mentioned in the paper, we tried to implement the same concept using a different methodology. Under our project, we decided to utilize the Plant Village Dataset fully unlike work mentioned in the paper where they dropped 20% of the dataset images that belonged to different plant categories i.e. pepper and potato.

We first prepared a model to distinguish tomato from other crops (Binary Classification). After that another model is prepared to identify the type of tomato disease including the healthy class (Multi Class Classification).

**2.2.1. IMAGE PRE-PROCESSING**

While carrying out the process, images are first pre-processed in order to reduce the data size. It helps in bringing the generalization to the data set and reduces the time consumed in training and validation.

Steps followed in pre-processing of images are listed below:

**Resizing:** Resizing images to the average of sum of dimensions of all images. (128\*128)



**Before Resizing**

****

**After Resizing**

***Figure 2.3. Resize Operation over dataset***

**Blur:** Blurring images to generalize dataset and to reduce image data size.

**RGB to Gray**: Converting images from RGB scale to Gray Scale.

**Dimensions**: 256\*256 **Dimensions**: 128\*128

**Size**: 18.0 KB **Size**: 3.72KB

**Scale**: RGB Color Scale **Scale**: Gray Color Scale

***Figure 2.4 Image Pre-Processing Techniques applied to the dataset***

**2.2.2. MODEL 1. BINARY CLASSIFIER**

**Model 1**. This model aims to distinguish tomato from others. The model architecture is shown below in Figure 8. Training Dataset: 15996 (77.5%), Test Dataset: 4642 (22.5%)

**INPUT IMAGES**



**Tomato Leaf Others Others**

***Figure 2.5 Result of Binary Classifier (Model 1)***

Final accuracy achieved in the model 1 are,

Model 1. Accuracy: 0.9604 Loss: 0.1002 Epochs: 20

**2.2.3. MODEL 2. MULTI-CLASS CLASSIFIER**

**Model 2.** This model aims at identifying the type of disease present in a tomato leaf if it is infected and labels it healthy if it is not. The plant village dataset covers 10 classes under multi-class classification. Among these 10 classes, 9 classes belongs to the different types of diseases present in tomato plant. The remaining one belongs to the healthy category. The architecture is shown below in Figure 8. Training dataset (90%) and Test Set(10%).

Two models have been developed for multi-class classifier. One with dropout and another without dropout. Accuracies observed in both the models were approximately the same. It implies we can trade on accuracy for bringing more generalization to our model.

**INPUT IMAGE**

**Target Spot Mosaic Virus Healthy**

***Figure 2.6. Result of Multi-Class Classifier (Model 2)***

Final accuracy achieved in the model 2 are,

Model 2 (without dropout). Accuracy: 0.8080 Loss: 0.5494 Epochs: 30

Model 2 (with dropout, 0.2). Accuracy: 0.7971 Loss: 0.5676 Epochs: 30

**2.2.4. MODEL 3. MULTI-LABEL CLASSIFIER**

**Model 3.** This model aims at bringing the functionality of Model1 and Model2 into one by assigning multi labels at one time.

**INPUT IMAGE**

**Target Spot , Tomato Mosaic Virus , Tomato Healthy, Tomato**

***Figure 2.7. Result of Multi-Label Classifier (Model 3)***

Final accuracy achieved in the model 3 are,

Model 3. Accuracy: 0.9222 Loss: 0.2074 Epochs: 20

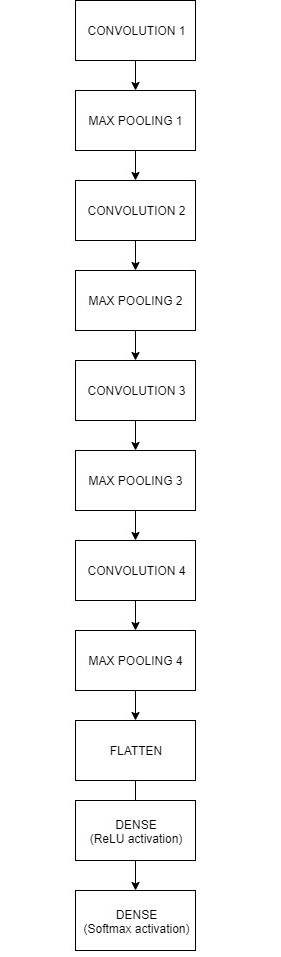
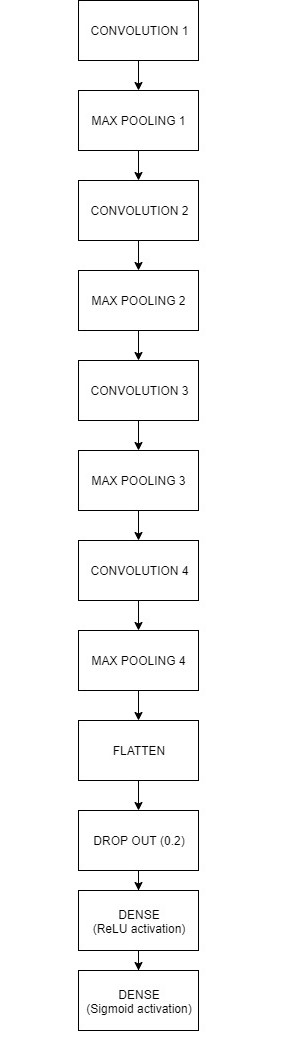
**2.2.5. MODEL ARCHITECTURE**

Model architecture of both the models are shown in Figure 8. The main difference between the architecture of both the models is of activation function used in last layer of classifier and of some other parametric values.

Binary classifier uses sigmoid activation function in the last layer of classifier which is responsible for calculating individual probability for each test point.

Multi-Class classifier uses softmax activation function in the last layer of classifier which is responsible for giving one class as output class. This is because it assigns probability to a test point compared to the whole probabilistic sum of each class for that particular test point.

Multi-Label classifier uses same architecture as that of multi-class classifier. The only difference is of sigmoid activation function in the last layer of classifier. For a particular test point, if the probability of belonging to a particular class is more than or equal to 0.5, then it assign that class as a label to the considered test point.



***Figure 2.8. Model Architecture (Model 1 and Model 2)***

**2.3. EXPERIMENTAL RESULTS**

**MODEL 1.**

Epoch 1/20

499/499 [==============================] - 466s 933ms/step - loss: 0.4745 - acc: 0.7911

Epoch 2/20

499/499 [==============================] - 256s 513ms/step - loss: 0.3336 - acc: 0.8576

Epoch 3/20

499/499 [==============================] - 231s 464ms/step - loss: 0.2656 - acc: 0.8881

Epoch 4/20

499/499 [==============================] - 233s 466ms/step - loss: 0.2288 - acc: 0.9073

Epoch 5/20

499/499 [==============================] - 230s 462ms/step - loss: 0.2028 - acc: 0.9179

Epoch 6/20

499/499 [==============================] - 237s 474ms/step - loss: 0.1894 - acc: 0.9222

Epoch 7/20

499/499 [==============================] - 231s 463ms/step - loss: 0.1714 - acc: 0.9290

Epoch 8/20

499/499 [==============================] - 231s 462ms/step - loss: 0.1611 - acc: 0.9343

Epoch 9/20

499/499 [==============================] - 229s 459ms/step - loss: 0.1469 - acc: 0.9413

Epoch 10/20

499/499 [==============================] - 230s 461ms/step - loss: 0.1411 - acc: 0.9457

Epoch 11/20

499/499 [==============================] - 229s 460ms/step - loss: 0.1343 - acc: 0.9478

Epoch 12/20

499/499 [==============================] - 229s 460ms/step - loss: 0.1298 - acc: 0.9493

Epoch 13/20

499/499 [==============================] - 234s 469ms/step - loss: 0.1210 - acc: 0.9520

Epoch 14/20

499/499 [==============================] - 254s 509ms/step - loss: 0.1167 - acc: 0.9539

Epoch 15/20

499/499 [==============================] - 243s 488ms/step - loss: 0.1132 - acc: 0.9583

Epoch 16/20

499/499 [==============================] - 266s 533ms/step - loss: 0.1152 - acc: 0.9538

Epoch 17/20

499/499 [==============================] - 286s 573ms/step - loss: 0.1096 - acc: 0.9570

Epoch 18/20

499/499 [==============================] - 276s 554ms/step - loss: 0.0995 - acc: 0.9624

Epoch 19/20

499/499 [==============================] - 293s 588ms/step - loss: 0.1012 - acc: 0.9604

Epoch 20/20

499/499 [==============================] - 249s 500ms/step - loss: 0.1002 - acc: 0.9604

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_13 (Conv2D) (None, 62, 62, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_13 (MaxPooling (None, 62, 31, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_14 (Conv2D) (None, 60, 29, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_14 (MaxPooling (None, 60, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_15 (Conv2D) (None, 58, 12, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_15 (MaxPooling (None, 58, 6, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_16 (Conv2D) (None, 56, 4, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_16 (MaxPooling (None, 56, 2, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 1792) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_4 (Dropout) (None, 1792) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

feature\_dense (Dense) (None, 128) 229504

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 1) 129

=================================================================

Total params: 244,449

Trainable params: 244,449

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MODEL 2 (without dropout)**

Epoch 1/30

464/464 [==============================] - 248s 536ms/step - loss: 1.8152 - acc: 0.3622

Epoch 2/30

464/464 [==============================] - 276s 595ms/step - loss: 1.4178 - acc: 0.5146

Epoch 3/30

464/464 [==============================] - 273s 588ms/step - loss: 1.2140 - acc: 0.5831

Epoch 4/30

464/464 [==============================] - 265s 571ms/step - loss: 1.0889 - acc: 0.6203

Epoch 5/30

464/464 [==============================] - 241s 520ms/step - loss: 0.9907 - acc: 0.6572

Epoch 6/30

464/464 [==============================] - 255s 550ms/step - loss: 0.9260 - acc: 0.6765

Epoch 7/30

464/464 [==============================] - 251s 542ms/step - loss: 0.8844 - acc: 0.6914

Epoch 8/30

464/464 [==============================] - 251s 541ms/step - loss: 0.8526 - acc: 0.7012

Epoch 9/30

464/464 [==============================] - 251s 541ms/step - loss: 0.8233 - acc: 0.7106

Epoch 10/30

464/464 [==============================] - 243s 524ms/step - loss: 0.7830 - acc: 0.7236

Epoch 11/30

464/464 [==============================] - 228s 491ms/step - loss: 0.7567 - acc: 0.7352

Epoch 12/30

464/464 [==============================] - 211s 456ms/step - loss: 0.7365 - acc: 0.7429

Epoch 13/30

464/464 [==============================] - 212s 457ms/step - loss: 0.7327 - acc: 0.7427

Epoch 14/30

464/464 [==============================] - 210s 454ms/step - loss: 0.7205 - acc: 0.7464

Epoch 15/30

464/464 [==============================] - 198s 428ms/step - loss: 0.6960 - acc: 0.7610

Epoch 16/30

464/464 [==============================] - 201s 434ms/step - loss: 0.6735 - acc: 0.7616

Epoch 17/30

464/464 [==============================] - 205s 443ms/step - loss: 0.6613 - acc: 0.7675

Epoch 18/30

464/464 [==============================] - 199s 429ms/step - loss: 0.6517 - acc: 0.7672

Epoch 19/30

464/464 [==============================] - 199s 428ms/step - loss: 0.6408 - acc: 0.7727

Epoch 20/30

464/464 [==============================] - 200s 431ms/step - loss: 0.6236 - acc: 0.7787

Epoch 21/30

464/464 [==============================] - 200s 430ms/step - loss: 0.6197 - acc: 0.7800

Epoch 22/30

464/464 [==============================] - 199s 429ms/step - loss: 0.6075 - acc: 0.7862

Epoch 23/30

464/464 [==============================] - 199s 429ms/step - loss: 0.5917 - acc: 0.7899

Epoch 24/30

464/464 [==============================] - 241s 520ms/step - loss: 0.5832 - acc: 0.7928

Epoch 25/30

464/464 [==============================] - 236s 509ms/step - loss: 0.5859 - acc: 0.7958

Epoch 26/30

464/464 [==============================] - 201s 433ms/step - loss: 0.5723 - acc: 0.8000

Epoch 27/30

464/464 [==============================] - 198s 428ms/step - loss: 0.5653 - acc: 0.8012

Epoch 28/30

464/464 [==============================] - 201s 433ms/step - loss: 0.5609 - acc: 0.8044

Epoch 29/30

464/464 [==============================] - 199s 429ms/step - loss: 0.5519 - acc: 0.8036

Epoch 30/30

464/464 [==============================] - 199s 428ms/step - loss: 0.5494 - acc: 0.8080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_5 (Conv2D) (None, 62, 62, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_5 (MaxPooling2 (None, 62, 31, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_6 (Conv2D) (None, 60, 29, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_6 (MaxPooling2 (None, 60, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_7 (Conv2D) (None, 58, 12, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_7 (MaxPooling2 (None, 58, 6, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_8 (Conv2D) (None, 56, 4, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_8 (MaxPooling2 (None, 56, 2, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 1792) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 128) 229504

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 10) 1290

=================================================================

Total params: 245,610

Trainable params: 245,610

Non-trainable params: 0

**MODEL 2 (with dropout 0.2)**

Epoch 1/30

464/464 [==============================] - 206s 443ms/step - loss: 1.8731 - acc: 0.3322

Epoch 2/30

464/464 [==============================] - 199s 430ms/step - loss: 1.4570 - acc: 0.4994

Epoch 3/30

464/464 [==============================] - 198s 427ms/step - loss: 1.2484 - acc: 0.5675

Epoch 4/30

464/464 [==============================] - 199s 430ms/step - loss: 1.0903 - acc: 0.6202

Epoch 5/30

464/464 [==============================] - 200s 430ms/step - loss: 0.9893 - acc: 0.6551

Epoch 6/30

464/464 [==============================] - 199s 428ms/step - loss: 0.9380 - acc: 0.6680

Epoch 7/30

464/464 [==============================] - 199s 429ms/step - loss: 0.8870 - acc: 0.6867

Epoch 8/30

464/464 [==============================] - 199s 429ms/step - loss: 0.8363 - acc: 0.7050

Epoch 9/30

464/464 [==============================] - 199s 429ms/step - loss: 0.8093 - acc: 0.7131

Epoch 10/30

464/464 [==============================] - 199s 428ms/step - loss: 0.7860 - acc: 0.7212

Epoch 11/30

464/464 [==============================] - 199s 428ms/step - loss: 0.7486 - acc: 0.7374

Epoch 12/30

464/464 [==============================] - 199s 428ms/step - loss: 0.7364 - acc: 0.7411

Epoch 13/30

464/464 [==============================] - 199s 429ms/step - loss: 0.7230 - acc: 0.7410

Epoch 14/30

464/464 [==============================] - 199s 429ms/step - loss: 0.7161 - acc: 0.7480

Epoch 15/30

464/464 [==============================] - 199s 429ms/step - loss: 0.6845 - acc: 0.7589

Epoch 16/30

464/464 [==============================] - 199s 430ms/step - loss: 0.6740 - acc: 0.7618

Epoch 17/30

464/464 [==============================] - 199s 429ms/step - loss: 0.6722 - acc: 0.7614

Epoch 18/30

464/464 [==============================] - 198s 427ms/step - loss: 0.6536 - acc: 0.7717

Epoch 19/30

464/464 [==============================] - 223s 480ms/step - loss: 0.6355 - acc: 0.7739

Epoch 20/30

464/464 [==============================] - 222s 479ms/step - loss: 0.6275 - acc: 0.7797

Epoch 21/30

464/464 [==============================] - 203s 438ms/step - loss: 0.6256 - acc: 0.7779

Epoch 22/30

464/464 [==============================] - 198s 426ms/step - loss: 0.6048 - acc: 0.7855

Epoch 23/30

464/464 [==============================] - 199s 429ms/step - loss: 0.6113 - acc: 0.7866

Epoch 24/30

464/464 [==============================] - 199s 429ms/step - loss: 0.6030 - acc: 0.7895

Epoch 25/30

464/464 [==============================] - 202s 434ms/step - loss: 0.5988 - acc: 0.7855

Epoch 26/30

464/464 [==============================] - 200s 431ms/step - loss: 0.5718 - acc: 0.7968

Epoch 27/30

464/464 [==============================] - 200s 431ms/step - loss: 0.5789 - acc: 0.7965

Epoch 28/30

464/464 [==============================] - 199s 429ms/step - loss: 0.5741 - acc: 0.7973

Epoch 29/30

464/464 [==============================] - 200s 431ms/step - loss: 0.5846 - acc: 0.7931

Epoch 30/30

464/464 [==============================] - 203s 437ms/step - loss: 0.5676 - acc: 0.7971

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 62, 62, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 62, 31, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 60, 29, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 60, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 58, 12, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_3 (MaxPooling2 (None, 58, 6, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_4 (Conv2D) (None, 56, 4, 32) 4640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_4 (MaxPooling2 (None, 56, 2, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 56, 2, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 1792) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 128) 229504

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 10) 1290

=================================================================

Total params: 245,610

Trainable params: 245,610

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MODEL 3**

Epoch 1/20

464/464 [==============================] - 530s 1s/step - loss: 0.3420 - acc: 0.8630

Epoch 2/20

464/464 [==============================] - 526s 1s/step - loss: 0.2440 - acc: 0.9159

Epoch 3/20

464/464 [==============================] - 399s 860ms/step - loss: 0.2415 - acc: 0.9159

Epoch 4/20

464/464 [==============================] - 384s 827ms/step - loss: 0.2428 - acc: 0.9156

Epoch 5/20

464/464 [==============================] - 383s 825ms/step - loss: 0.2398 - acc: 0.9150

Epoch 6/20

464/464 [==============================] - 397s 856ms/step - loss: 0.2377 - acc: 0.9151

Epoch 7/20

464/464 [==============================] - 387s 834ms/step - loss: 0.2341 - acc: 0.9149

Epoch 8/20

464/464 [==============================] - 394s 849ms/step - loss: 0.2289 - acc: 0.9160

Epoch 9/20

464/464 [==============================] - 386s 832ms/step - loss: 0.2252 - acc: 0.9183

Epoch 10/20

464/464 [==============================] - 398s 858ms/step - loss: 0.2206 - acc: 0.9195

Epoch 11/20

464/464 [==============================] - 390s 840ms/step - loss: 0.2180 - acc: 0.9201

Epoch 12/20

464/464 [==============================] - 387s 834ms/step - loss: 0.2150 - acc: 0.9210

Epoch 13/20

464/464 [==============================] - 386s 832ms/step - loss: 0.2141 - acc: 0.9210

Epoch 14/20

464/464 [==============================] - 384s 828ms/step - loss: 0.2128 - acc: 0.9217

Epoch 15/20

464/464 [==============================] - 390s 841ms/step - loss: 0.2113 - acc: 0.9216

Epoch 16/20

464/464 [==============================] - 385s 830ms/step - loss: 0.2097 - acc: 0.9216

Epoch 17/20

464/464 [==============================] - 385s 829ms/step - loss: 0.2087 - acc: 0.9219

Epoch 18/20

464/464 [==============================] - 385s 830ms/step - loss: 0.2083 - acc: 0.9226

Epoch 19/20

464/464 [==============================] - 385s 830ms/step - loss: 0.2069 - acc: 0.9224

Epoch 20/20

464/464 [==============================] - 385s 829ms/step - loss: 0.2074 - acc: 0.9222

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 62, 62, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 62, 31, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 60, 29, 64) 9280

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 60, 14, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 58, 12, 128) 36992

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_3 (MaxPooling2 (None, 58, 6, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_4 (Conv2D) (None, 56, 4, 256) 147712

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_4 (MaxPooling2 (None, 56, 2, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 14336) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 8) 114696

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 12) 108

=================================================================

Total params: 309,684

Trainable params: 309,684

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**REFERENCES**

1. Tairu Oluwafemi Emmanuel. [n.d]. Plant Village Dataset, Dataset of diseased plant leaf images and corresponding labels. Retrieved from <https://www.kaggle.com/emmarex/plantdisease>
2. Zarreen Naowal Reza , Faiza Nuzhat , Nuzhat Ashfat Mahsa , Md. Haider Ali . Detecting Jute Plant Disease Using Image Processing and MachineLearning.Retrievedfromhttps://[www.researchgate.net/publication/314176336\_Detecting\_Jute\_Plant\_Disease\_Using\_Image\_Processing\_and\_Machine\_Learning](http://www.researchgate.net/publication/314176336_Detecting_Jute_Plant_Disease_Using_Image_Processing_and_Machine_Learning)
3. Neha Mangla, Priyanka B Raj, Soumya J Hegde [2019,Feb]. Paddy leaf disease detection using Image Processing and Machine Learning. Retrieved from <https://www.researchgate.net/publication/333292667_Paddy_Leaf_Disease_Detection_Using_Image_Processing_and_Machine_Learning>
4. Meena Prakash R. , G.P. Saraswathy , G. Ramalakshmi , K.H. Mangaleswari [2017, March] . Detection of Leaf Diseases and Classifications using Digital Image Processing. Retrieved from <https://www.researchgate.net/publication/323000855_Detection_of_leaf_diseases_and_classification_using_digital_image_processing>
5. K. Vinoth Kumar and T. Jayasankar. [n.d]. An Identification of crop disease using image segmentation. Retrieved from <http://ijpsr.com/bft-article/an-identification-of-crop-disease-using-image-segmentation/?view=fulltext>.