FYP Documentation

# Dataset: Apple Disease Dataset

## About Dataset

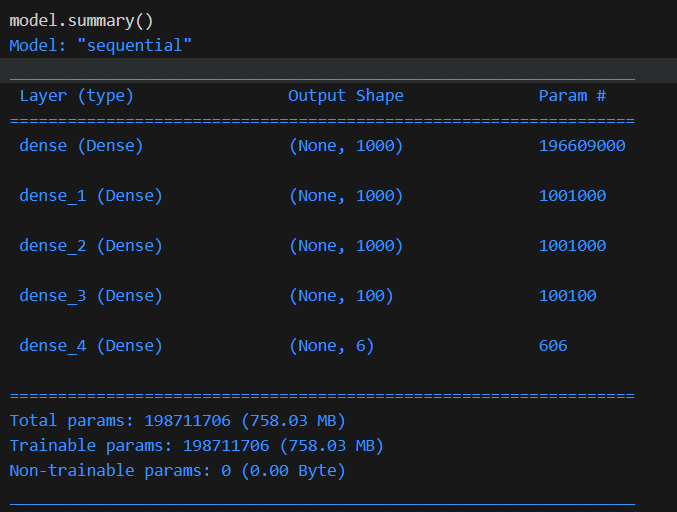
|  |  |
| --- | --- |
| Dataset Source | Challenger.ai |
| We got it from | Sir Hashim |
| Dataset used by | Yong Zhong and Ming Zhao (2020) |

## Approach: Using Feedforward ANN for Classification

### Preprocessing

All images resized to 256 x 256. Train, test split (85% )was as provided by sir.

### Model Summary



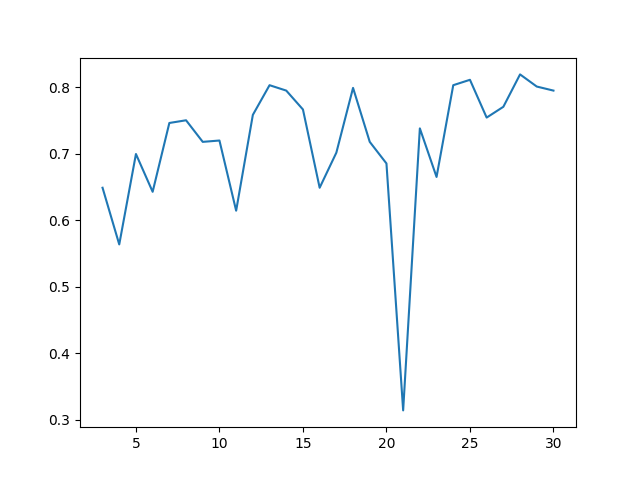
## Results

After 10 training epochs, the model gave accuracy of 0.6795 on test data.

First, we resized data images into 256 x 256 pixels Then we trained the data for 10 epochs. After this training, the model performed on test data with an accuracy of 0.6795. With an image resize of 128 x 128 and 10 epochs, accuracy of the model was 0.6815.

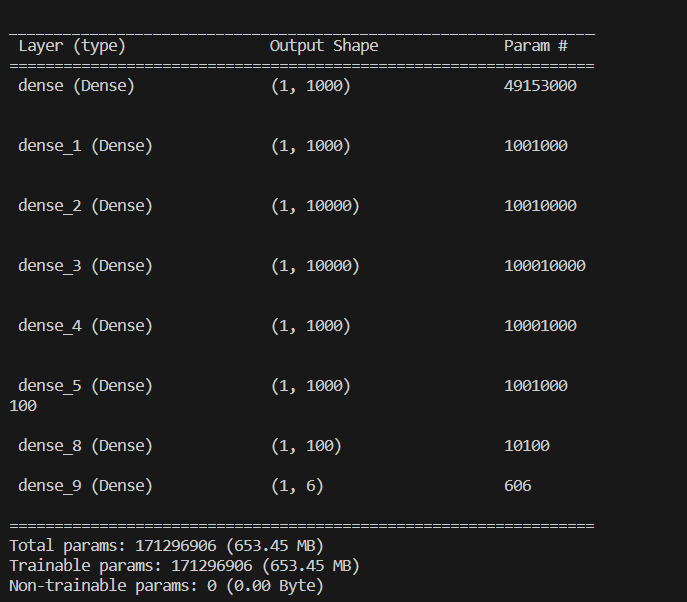
We resized the images into 128x128 pixels, and trained the ANN model to classify the images, using values of epochs ranging from 3 to 30 and 40,50, and 100. With epoch 100, the model tended to overfit. The accuracies of the ANN on test data for each epoch are given below.

|  |  |
| --- | --- |
| Epoch | Accuracy |
| 3 | 0.649087221 |
| 4 | 0.563894523 |
| 5 | 0.69979716 |
| 6 | 0.643002028 |
| 7 | 0.746450304 |
| 8 | 0.750507099 |
| 9 | 0.718052738 |
| 10 | 0.720081136 |
| 11 | 0.614604462 |
| 12 | 0.75862069 |
| 13 | 0.803245436 |
| 14 | 0.795131846 |
| 15 | 0.76673428 |
| 16 | 0.649087221 |
| 17 | 0.701825558 |
| 18 | 0.799188641 |
| 19 | 0.718052738 |
| 20 | 0.685598377 |
| 21 | 0.314401623 |
| 22 | 0.738336714 |
| 23 | 0.665314402 |
| 24 | 0.803245436 |
| 25 | 0.811359026 |
| 26 | 0.754563895 |
| 27 | 0.770791075 |
| 28 | 0.819472617 |
| 29 | 0.801217039 |
| 30 | 0.795131846 |
| 40 | 0.8356998 |
| 50 | 0.83772819 |
| 100 | 0.59229209 |



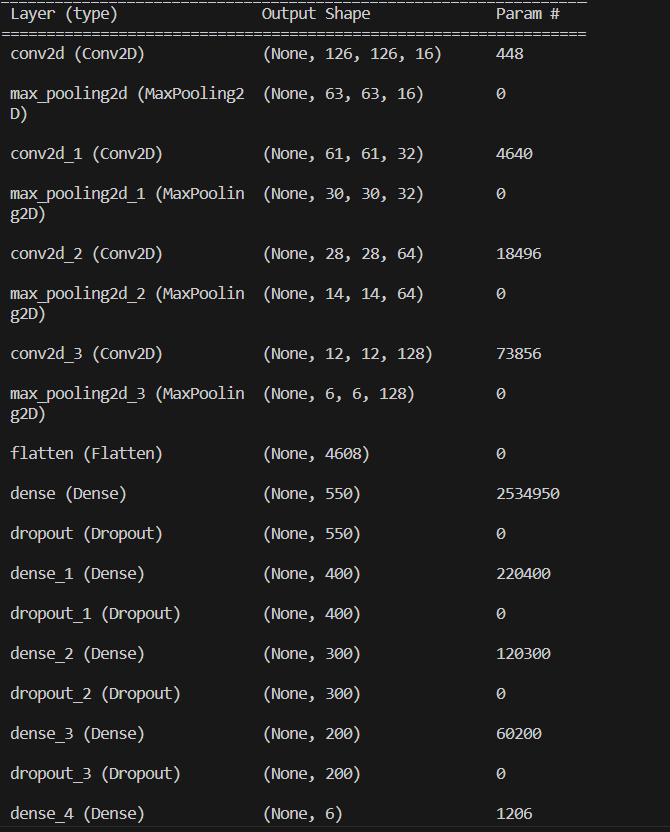
Considering the possibility that ANN was not giving its full potential accuracy because of a lack of depth, we added several layers to the architecture and retrained the model. After 50 epochs of training, the model gave accuracy of 0.8316, which was similar to the one obtained by the previous model. However, the training time increased substantially.

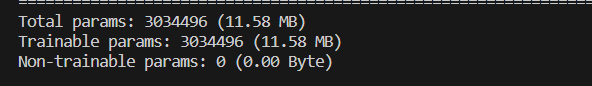
The architecture of the ANN after addition of new layers is given below



## Approach: Using Convolutional Neural Networks

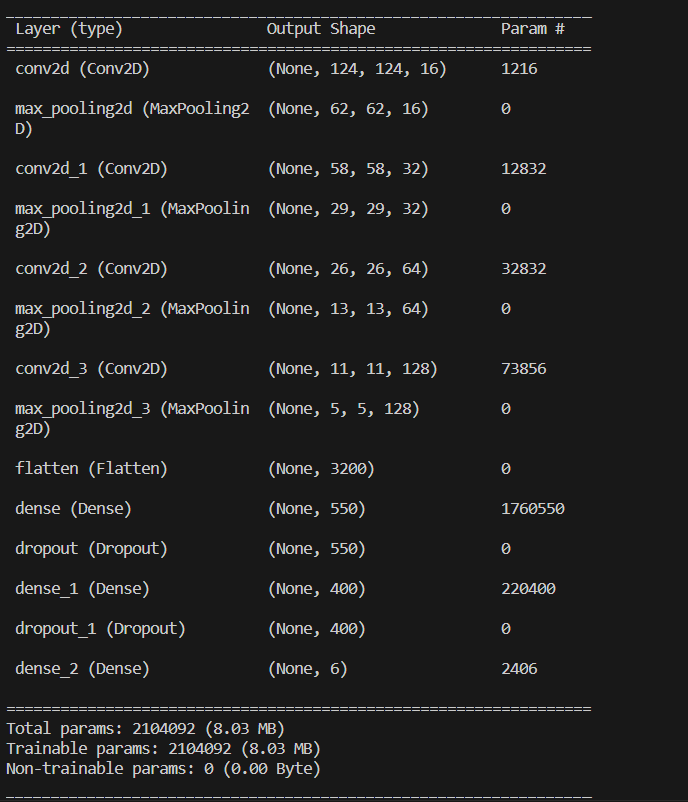
We started by picking an arbitrary CNN architecture from a website. The architecture is given below





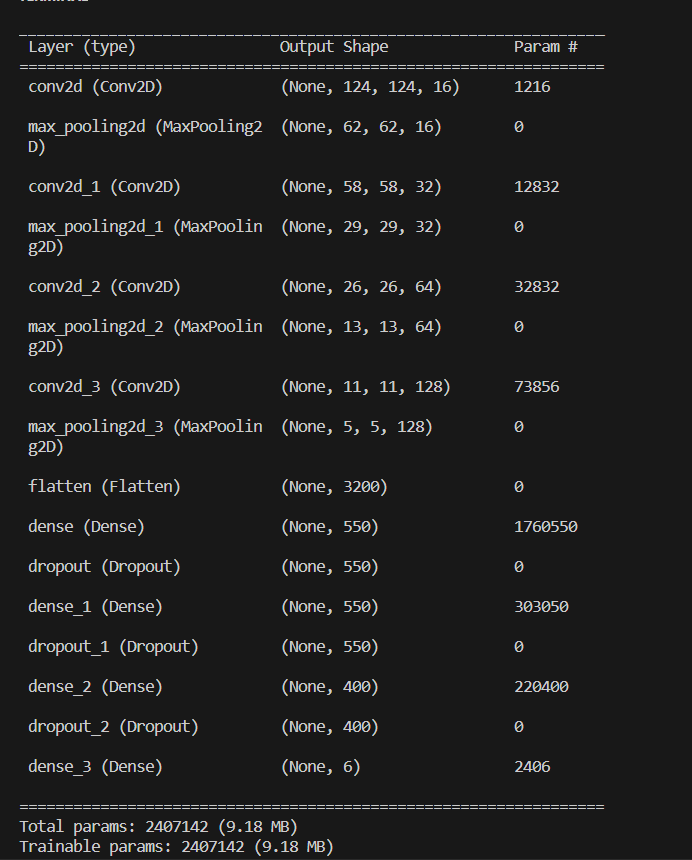
After being trained for 20 and 30 learning epochs, the model yielded respective accuracies of 0.8499 and 0.84 on test data.

Then we changed the architecture by removing some dense layers, and increasing kernel sizes.



After a training of 25 epochs, the accuracy was 0.827. Suspecting overfitting, we tried retraining the model for 15 epochs only, but still we got accuracy of 0.839.

Suspecting that there was something related to feature overrepresentation going on, we added a dropout layer

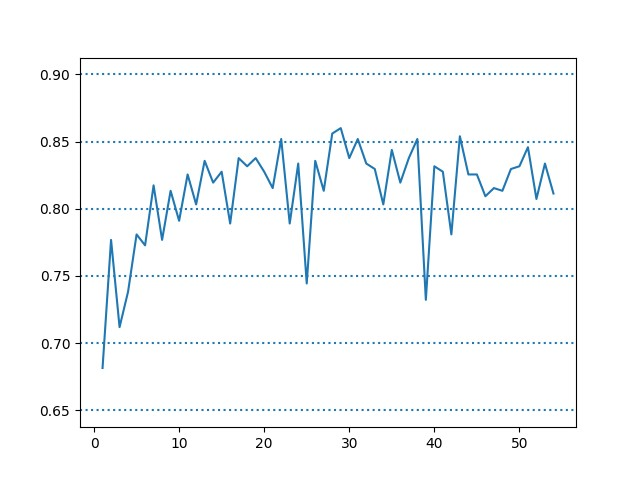


This time accuracy increased to 0.8621 after 25 epochs.

The training data consisting of epochs and accuracies for this model is given below.

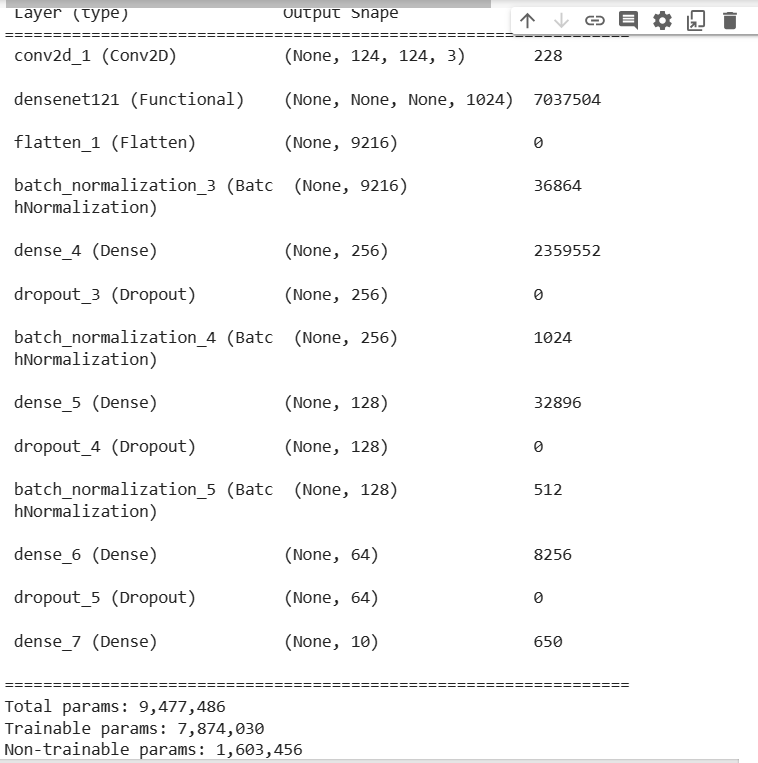
|  |  |
| --- | --- |
| Epoch | Accuracy |
| 1 | 0.681542 |
| 2 | 0.776876 |
| 3 | 0.711968 |
| 4 | 0.738337 |
| 5 | 0.780933 |
| 6 | 0.772819 |
| 7 | 0.817444 |
| 8 | 0.776876 |
| 9 | 0.813387 |
| 10 | 0.791075 |
| 11 | 0.825558 |
| 12 | 0.803245 |
| 13 | 0.8357 |
| 14 | 0.819473 |
| 15 | 0.827586 |
| 16 | 0.789047 |
| 17 | 0.837728 |
| 18 | 0.831643 |
| 19 | 0.837728 |
| 20 | 0.827586 |
| 21 | 0.815416 |
| 22 | 0.851927 |
| 23 | 0.789047 |
| 24 | 0.833671 |
| 25 | 0.744422 |
| 26 | 0.8357 |
| 27 | 0.813387 |
| 28 | 0.855984 |
| 29 | 0.860041 |
| 30 | 0.837728 |
| 31 | 0.851927 |
| 32 | 0.833671 |
| 33 | 0.829615 |
| 34 | 0.803245 |
| 35 | 0.843813 |
| 36 | 0.819473 |
| 37 | 0.837728 |
| 38 | 0.851927 |
| 39 | 0.732252 |
| 40 | 0.831643 |
| 41 | 0.827586 |
| 42 | 0.780933 |
| 43 | 0.853955 |
| 44 | 0.825558 |
| 45 | 0.825558 |
| 46 | 0.809331 |
| 47 | 0.815416 |
| 48 | 0.813387 |
| 49 | 0.829615 |
| 50 | 0.831643 |
| 51 | 0.845842 |
| 52 | 0.807302 |
| 53 | 0.833671 |
| 54 | 0.811359 |

The accuracy vs epoch plot:



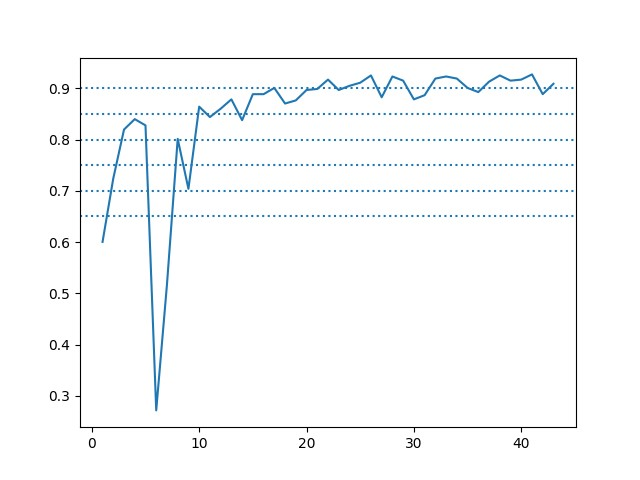
## Model: Densenet121

We imported the Densenet121 pretrained classification model from keras, and trained it for the apple leaf disease dataset. We then augmented four convolutional layers, followed by 4 dense and 3 dropout layers. We then applied the softmax activation function. The summary of the architecture of the network is described below



The training summary data of epochs and accuracies is recorded in below table, and plotted after the table.

|  |  |
| --- | --- |
| Epoch | Accuracy |
| 1 | 0.60040568 |
| 2 | 0.724137931 |
| 3 | 0.819472617 |
| 4 | 0.839756592 |
| 5 | 0.827586207 |
| 6 | 0.271805274 |
| 7 | 0.517241379 |
| 8 | 0.801217039 |
| 9 | 0.703853955 |
| 10 | 0.864097363 |
| 11 | 0.843813387 |
| 12 | 0.860040568 |
| 13 | 0.878296146 |
| 14 | 0.837728195 |
| 15 | 0.888438134 |
| 16 | 0.888438134 |
| 17 | 0.900608519 |
| 18 | 0.870182556 |
| 19 | 0.876267748 |
| 20 | 0.896551724 |
| 21 | 0.898580122 |
| 22 | 0.9168357 |
| 23 | 0.896551724 |
| 24 | 0.904665314 |
| 25 | 0.910750507 |
| 26 | 0.92494929 |
| 27 | 0.882352941 |
| 28 | 0.922920892 |
| 29 | 0.914807302 |
| 30 | 0.878296146 |
| 31 | 0.886409736 |
| 32 | 0.918864097 |
| 33 | 0.922920892 |
| 34 | 0.918864097 |
| 35 | 0.900608519 |
| 36 | 0.892494929 |
| 37 | 0.912778905 |
| 38 | 0.92494929 |
| 39 | 0.914807302 |
| 40 | 0.9168357 |
| 41 | 0.926977688 |
| 42 | 0.888438134 |
| 43 | 0.90872211 |



Now we will try the performance of densenet121 using a linear function in activation layer, and a simplex algorithm to find thresholds corresponding to the target classes.

# Augmentation of the image data

We realized that we were not obtaining sufficiently good results because we did not augment the image data. Now we will augment the images and then train densenet121 on the enlarged training datasrt.