

**CSCE4604: Practical deep machine Learning**

**Assignment 3- Report**

|  |  |
| --- | --- |
| **Submitted by:** | **Mohammed Abuelwafa** |
| **Student ID:** | **900172603** |

|  |  |
| --- | --- |
| **Date:** | 2th December 2020 |

**Fall 2020**

# Data Preprocessing:

* The following was done while preprocessing the data for the training :

1. Subtract the mean image from the data set   
   X -= np.mean (X , axis = 0 )
2. Divide by 255.0 in order to normalize the pixel values to be between 0 and 1  
   X /= 255.0
3. Before calculating the loss, the largest score was subtracted from the scores array in order to avoid numerical explosion when calculating the NLL loss.
4. One hot encoding was used for labelling the labels of the testing data, validation data, training data since it was found more useful in calculating the accuracies and loss (like most of the current frameworks)

# Choosing the network Architecure:

**Implementation notes:**

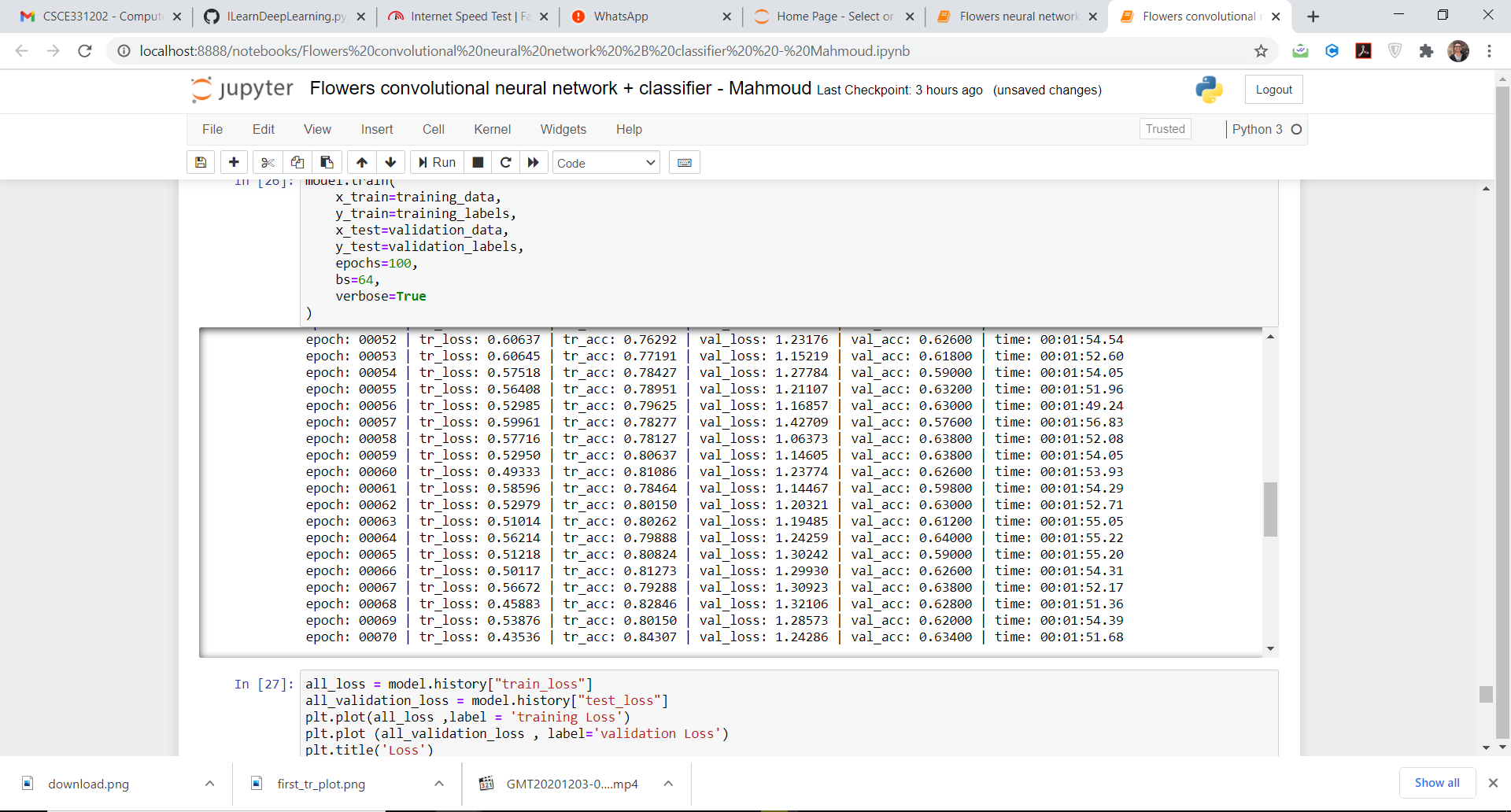
* Multiple learning rates were tried ( 0.001 , 0.002 , 0.005 , 0.001 , 0.008 ). However, 0.005 gave the best results.
* No regularization was used. However, in order to avoid overfitting dropout was implemented with a keep ratio = 85%
* SGD was used and the weights were updated after each epoch with minibatches was used for the best training. However, Adam was implemented as well, but did not give proper results (maybe had an implementation error from my side).
* The ReLU activation function was used in addition to keeping a close eye on the learning rate in addition to softmax function in the last classification layer.

Best training had a value of : learning rate = 5e-3

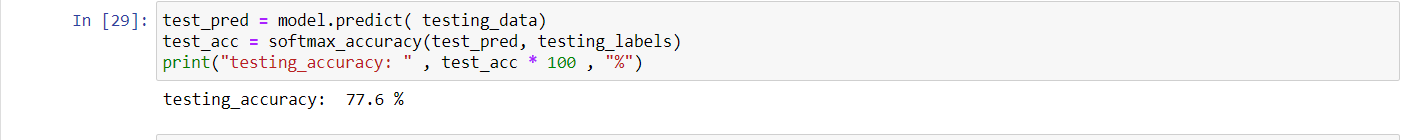
**The used architecture:**

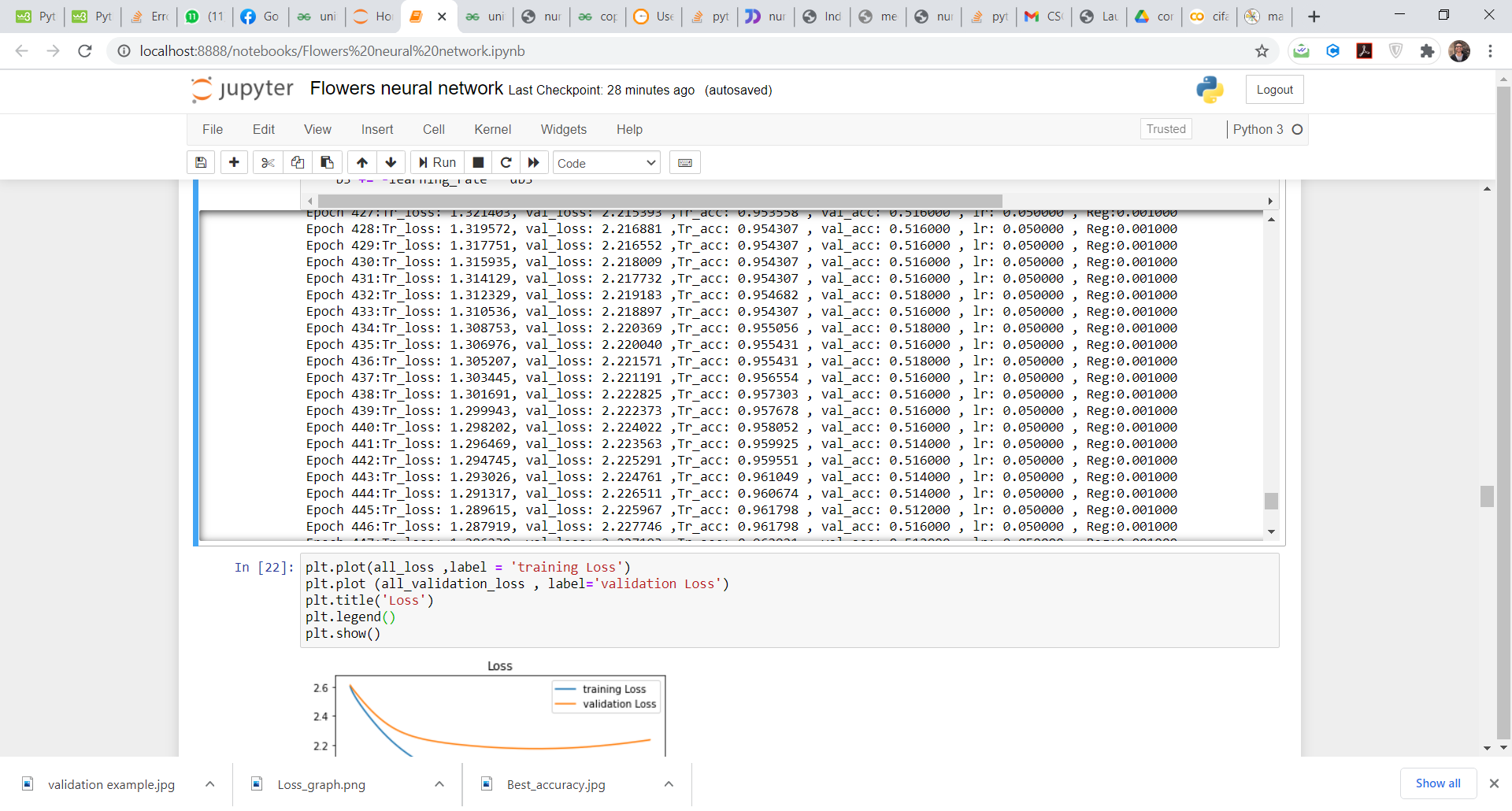
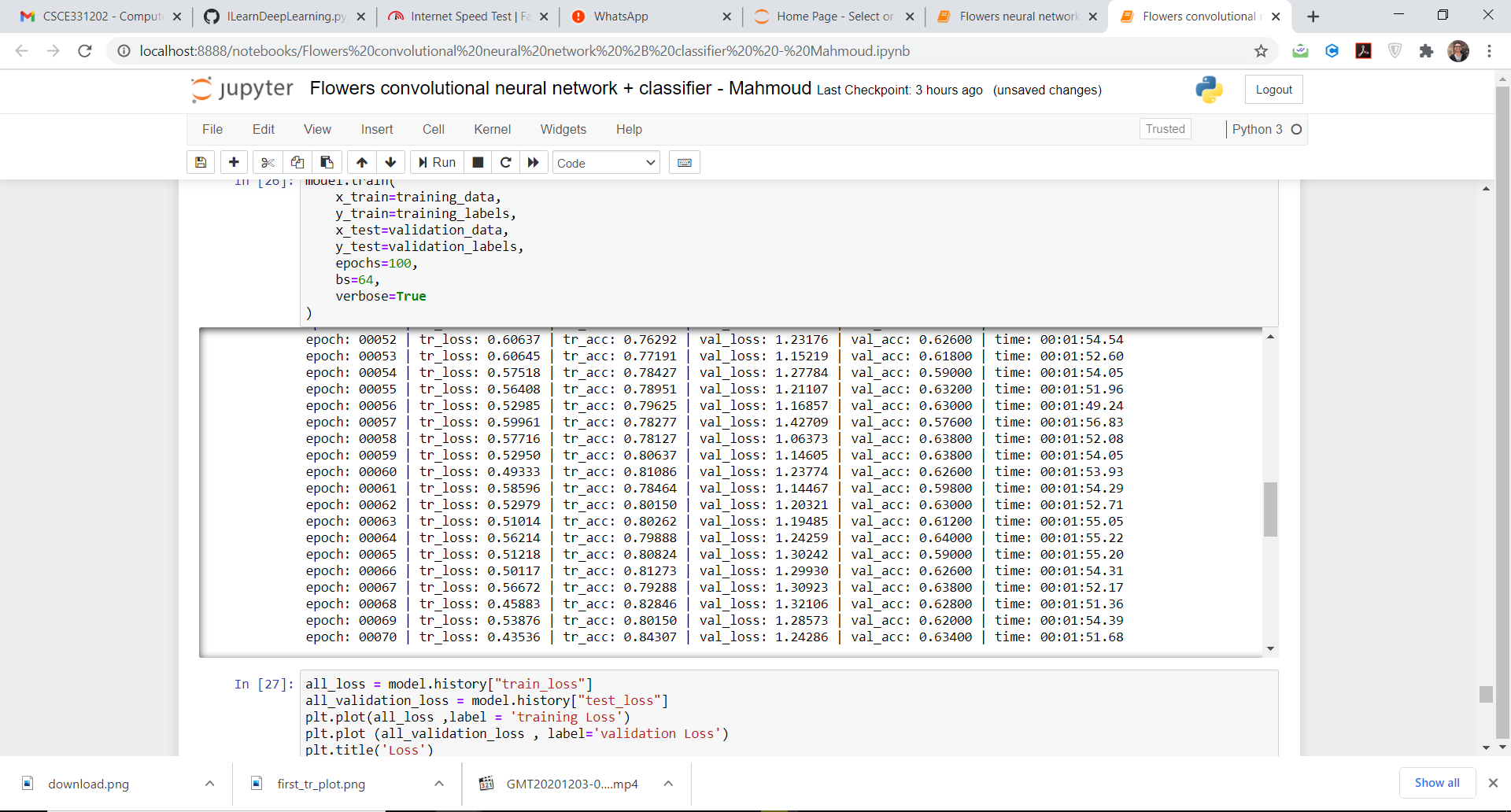
1. Convolution layer with 32 kernels with kernel size = (3,3,3) , stride = 1 and padding to maintain the size. # input (N, 32, 32, 3) out (N, 32, 32, 32)
2. Relu layer # input (N, 32, 32, 32) out (N, 32, 32, 32)
3. Convolution layer with 32 kernels with kernel size = (3,3,32) , stride = 1 and padding to maintain the size. # input (N, 32, 32, 32) out (N, 32, 32, 32)
4. Relu layer # input (N, 32, 32, 32) out (N, 32, 32, 32)
5. Max pool layer with pooling size = (2,2) and stride = 2 # input (N, 32, 32, 32) out (N, 16, 16, 32)
6. Dropout layer (keep\_prob = 85%) # input (N, 16, 16, 32) out (N, 16, 16, 32)
7. Convolution layer with kernel size = (3,3,32) , stride = 1 and padding to maintain the size. # input (N, 16, 16, 32) out (N, 16, 16, 32)
8. Relu layer # input (N, 16, 16, 32) out (N, 16, 16, 32)
9. Max pool layer with pooling size = (2,2) and stride = 2 # input (N, 16, 16, 32) out (N, 8, 8, 32)
10. Convolution layer with kernel size = (3,3,32) , stride = 1 and padding to maintain the size. # input (N, 8, 8, 32) out (N, 8, 8, 32)
11. Relu layer . # input (N, 8, 8, 32) out (N, 8, 8, 32)
12. Max pool layer with pooling size = (2,2) and stride = 2 . # input (N, 8, 8, 32) out (N, 4, 4, 32)
13. Dropout layer (keep\_prob = 85%) # input (N, 4, 4, 32) out (N, 4, 4, 32)
14. Convolution layer with kernel size = (3,3,32) , stride = 1 and padding to maintain the size. # input (N, 4, 4, 32) out (N, 4, 4, 32)
15. Relu layer # input (N, 4, 4, 32) out (N, 4, 4, 32)
16. Max pool layer with pooling size = (2,2) and stride = 2 # input (N, 4, 4, 32) out (N, 2, 2, 32)
17. Flatten layer # input (N, 2, 2, 32) out (N, 2\* 2\* 32)
18. Fully Connected layer # input (N, 2\* 2\* 32) , out ( N , 5 )
19. Softmax layer # input (N,5) out (N,5)

# Best Training:

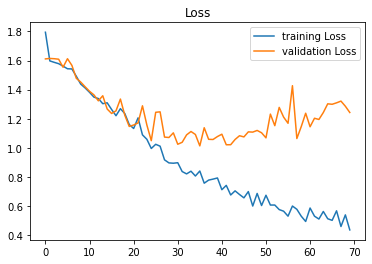
Best validation accuracy attained : 67%

Testing accuracy:



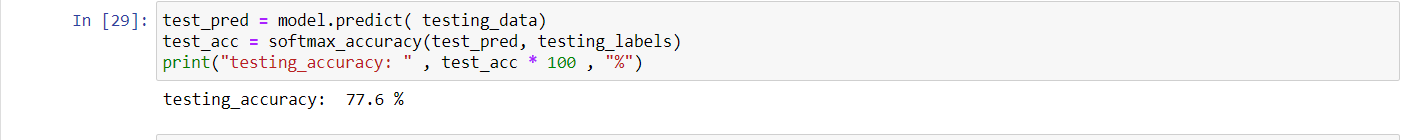
Stopping the training point :

* The training stopped here due to multiple signs of overfitting (increasing validation loss and decreasing training loss) in addition to limited time factors.

Best Training Loss vs epochs Graph:

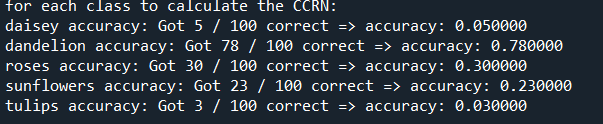
# Best ACCR:

An Accuracy of : 77.6% was obtained. (in comparison to 27.8% in the KNN Classifier and 55.4 with the fully connected)

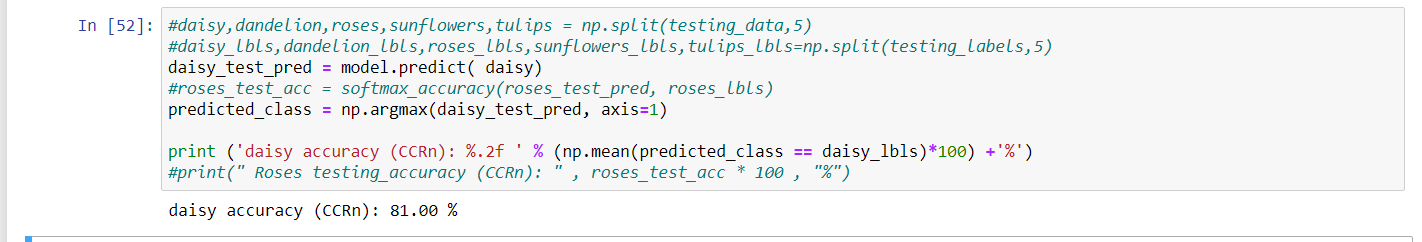


# Correct classification rate (CCRn) comparison:

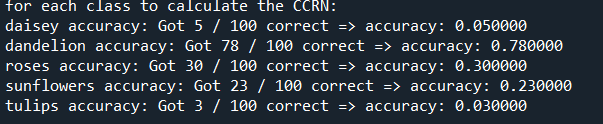
## For Daisy:

KNN : 5%

Neural network: 41%

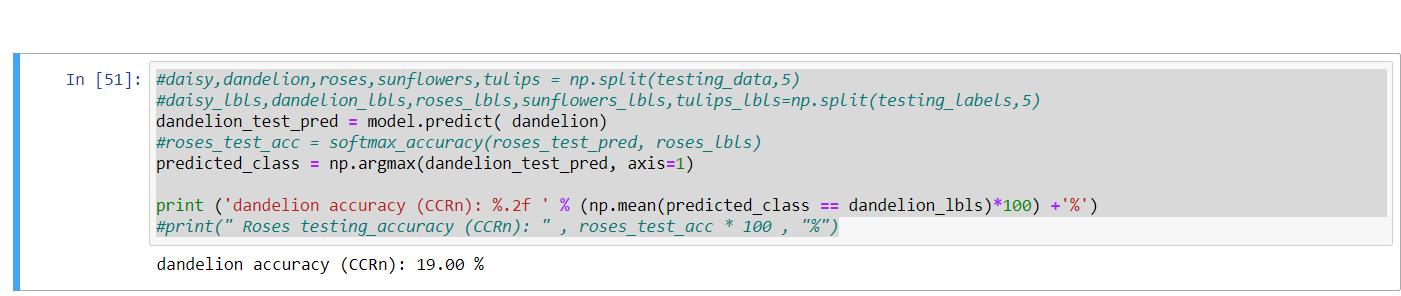
CNN : 81%

## For Dandelion:

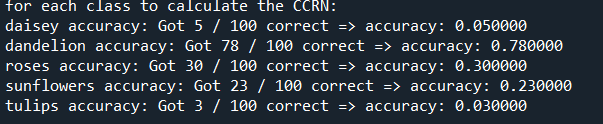
KNN : 78%



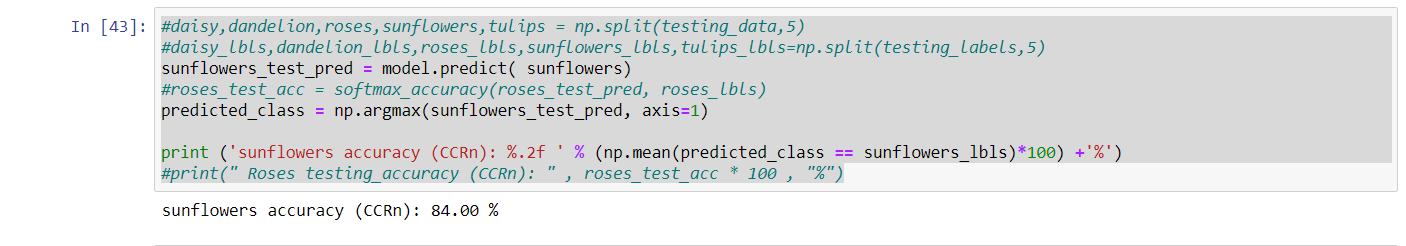
Neural network: 70%

CNN : 19%

## For Sunflowers:

KNN : 23%

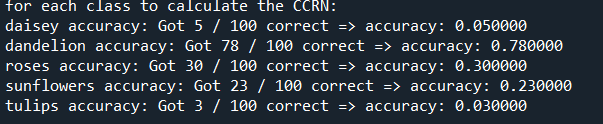
Neural network: 71 %



CNN : 84%

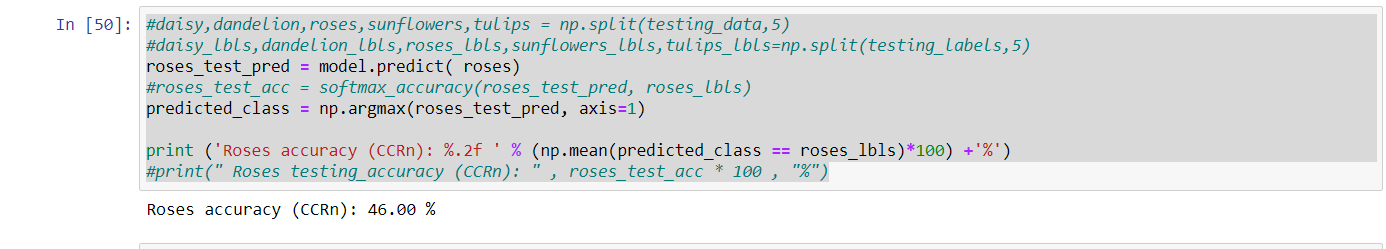
## 

## For Roses:



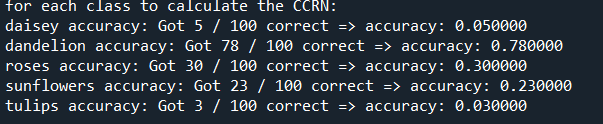
KNN : 30%

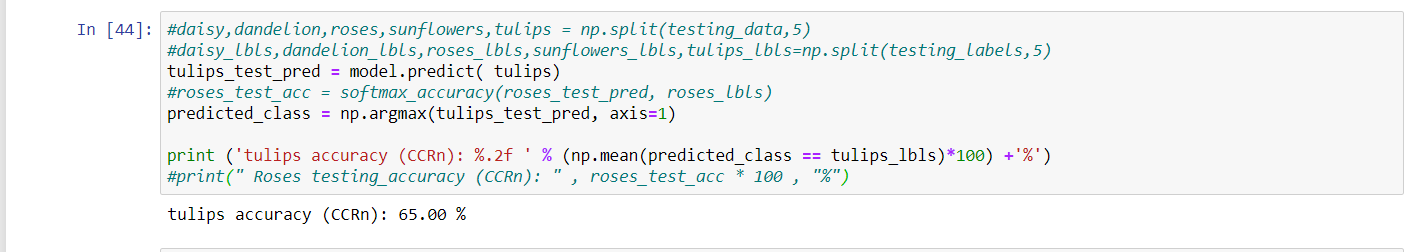
Neural network: 38%



CNN : 46%

## For Tulips:

  
KNN : 3%

Neural network: 57%

CNN:

The CNN generally performed better than the KNN classifier and the fully connected neural network in both the ACCR and the CCRn.

* The submitted code output is not for the best training. Multiple attempts were performed. However, the best training output was lost.