Dog Breed Classification

A Design Project Report

submitted by

PRAVEEN B R (COE19B007)

in partial fulfilment of requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

Department of Computer Science and Engineering
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING, KANCHEEPURAM

DEC 2022

DECLARATION OF ORIGINALITY

I, **Praveen B R**, with Roll No: **COE19B007** hereby declare that the material presented in the Project Report titled **Dog Breed Classification** represents original work carried out by me in the **Department of Computer Science and Engineering** at the Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Praveen B R

Place: Chennai

Date: 30.11.2022

CERTIFICATE

This is to certify that the report titled **Dog Breed Classification**, submitted by **Praveen B R (COE19B007)**, to the Indian Institute of Information Technology, Design and Manufacturing Kancheepuram, in partial fulfilment of requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** is a bonafide record of the work done by him/her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Rahul Raman
Project Internal Guide
Assistant Professor
Department of Computer Science and Engineering
IIITDM Kancheepuram, Chennai - 600 127

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to acknowledge and give my warmest thanks to my supervisor Dr. Rahul Raman who made this work possible. His guidance and advice carried me through all the stages of my project. I would also like to give special thanks to my family and friends as a whole for their continuous support and understanding which really held my motivation. Your prayer for me was what sustained me this far.

Finally, I would like to thank God, for letting me through all the difficulties. I have experienced your guidance day by day. You are the one who let me finish my degree. I will keep on trusting you for my future.

ABSTRACT

This project deals with a fine-grained image recognition problem, one of multi-class classification, namely determining the breed of a dog in a given image. In this report, I will talk about data set used, some preprocessing and also about the training and testing with three networks individually. The combined model was able achieve a 91.8% accuracy after 15 epochs in detecting the breed of unseen dog images while they were individually reaching only around 88. %.

KEYWORDS: Transfer Learning; Breed Identification; Cattle Identification; Concatenation

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DSP	Digital Signal Processing

NOTATION

λ	wavelength

CHAPTER 1

INTRODUCTION

1.1 Overview

This project deals with a fine-grained image recognition problem, one of multi-class classification, namely determining the breed of a dog in a given image. We intend to expand the dog breed identification further for cattle recognition. In this project, I have concatenated some networks and trained them to achieve a higher accuracy.

1.1.1 Data Set

In this project, dog breed identification data set from kaggle has been used[1] [2]. The data set consist of 10222 images of dogs spread among 120 classes with each class having around 70 to 120 samples. There is also private test data set but it has not used in the scope of this project.

1.1.2 Transfer Learning

- As the data set had only 50 to 90 images for training, the best bet was to use transfer learning.
- It is a technique used to initialize weights for training.
- Reusing of a weights from previously trained model on a similar problem is called as transfer learning. It is popular in the field of deep learning.
- It gives decent result with a limited data set and also is faster rather the training millions of images.
- In problems of image classification, networks trained on the humongous Image-Net data set are usually preferred.

1.1.3 Network Ensemble

- ’Ensembling’ is very common and are known to give positive results. Random forests used to combine a variable number of decision trees is a classic example.
- Ensemble is technique of combining various models to arrive at single network and use it as solution. This is done because an ensemble tends to perform better than singles improving the generalization ability.
- They can generally be of three types,
  - Weighted Ensemble
  - Concatenating Ensemble
  - Average Ensemble

1.2 Objective

This project aims to extract features from images of a dog given as input by the user using neural networks and identify its breed. For training, multiple networks have been trained on Stanford Dog Dataset and finally the model combined concatenating ensemble has been used. The model gives out probabilities for 120 classes and argmax is used to determine the breed.

CHAPTER 2

METHODOLOGY

2.1 Data Processing

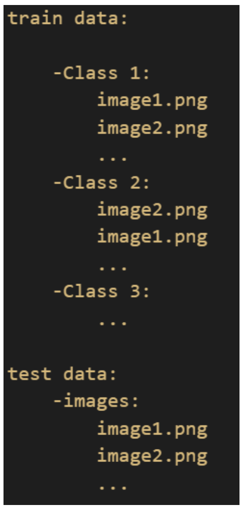
2.1.1 Training Data Preparation

The data is in a zip file with images in no specific order but with unique names and a csv file with names of files and their respective labels.

- Here, I am converting the data in the format as in figure 2.1
- By having the data set in the said format we can use imagedatasetfromdirectory() function which converts our image data into tf.data.Dataset format directly. The said format is memory time efficient while training model
- The training directory is created using the following code. It creates a folder for every breed and using csv file copies dog images into their respective folder. (figure 2.3)

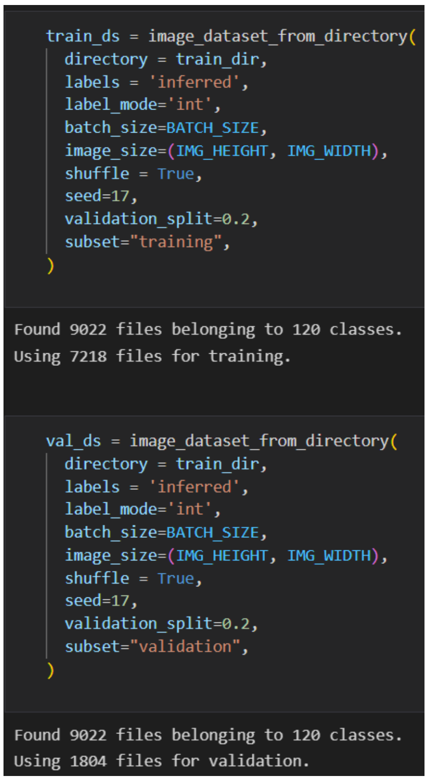
2.1.2 Validation and Testing data set

- 10 samples from each breed is moved to test directory. So in total we will have 1200 images for testing phase.
- Data for validation is created using imagedatasetfromdirectory() function by using training data set with a validation split of 0.2 and subset as validation.
- Validation set consist a total of 1804 samples picked in random way.
- Training data is also prepared in same way but with subset as training.
- Training set consists of the remaining total 7218 samples.
- Augmentation is used to generate some additional data which will also help generalize the model.
- Random horizontal flips, rotation, zoom are all used.



```
train data:

    -Class 1:
        image1.png
        image2.png
        ...
    -Class 2:
        image2.png
        image1.png
        ...
    -Class 3:
        ...

test data:
    -images:
        image1.png
        image2.png
        ...
```

Figure 2.1: Directory Format



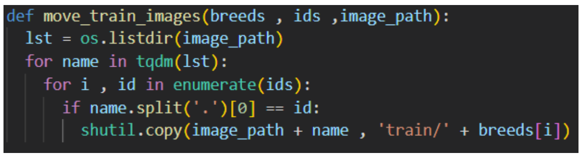
```
train_ds = image_dataset_from_directory(
    directory = train_dir,
    labels = 'inferred',
    label_mode='int',
    batch_size=BATCH_SIZE,
    image_size=(IMG_HEIGHT, IMG_WIDTH),
    shuffle = True,
    seed=17,
    validation_split=0.2,
    subset="training",
)
```

```
Found 9022 files belonging to 120 classes.
Using 7218 files for training.
```

```
val_ds = image_dataset_from_directory(
    directory = train_dir,
    labels = 'inferred',
    label_mode='int',
    batch_size=BATCH_SIZE,
    image_size=(IMG_HEIGHT, IMG_WIDTH),
    shuffle = True,
    seed=17,
    validation_split=0.2,
    subset="validation",
)
```

```
Found 9022 files belonging to 120 classes.
Using 1804 files for validation.
```

Figure 2.2: Creating training directory



```
def move_train_images(breeds , ids ,image_path):
  lst = os.listdir(image_path)
  for name in tqdm(lst):
    for i , id in enumerate(ids):
      if name.split('.')[0] == id:
        shutil.copy(image_path + name , 'train/' + breeds[i])
```

Figure 2.3: Training and Validation split

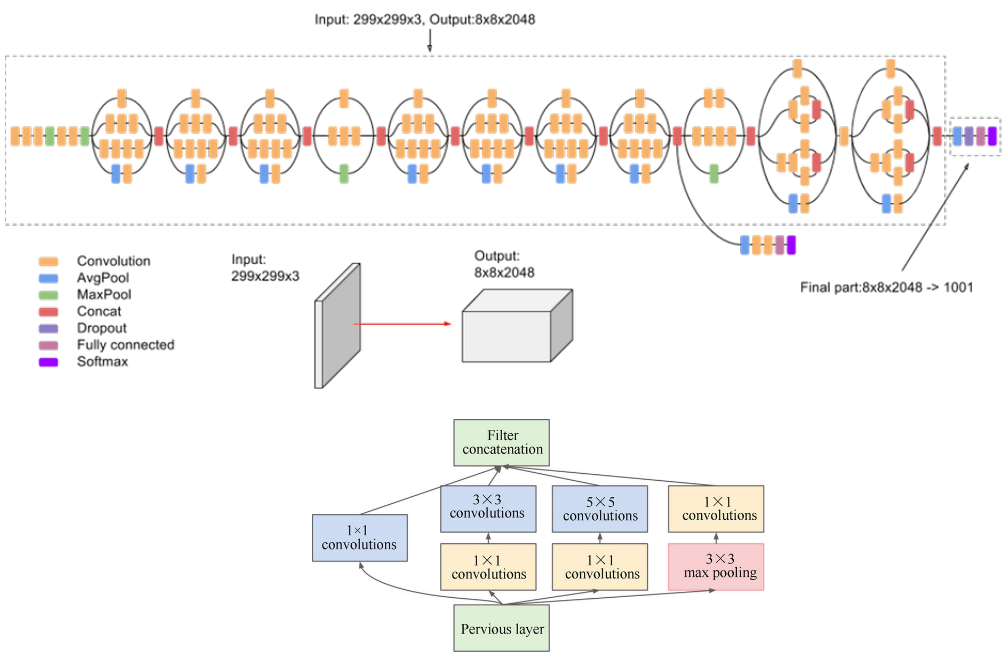


Figure 2.4: Architecture of Inception v3

2.2 Training and Testing

Now that we have both training and testing data sets in the intended formats we prepare models and start the intensive processes.

2.2.1 Models

- As in the paper [3], the author was able to achieve 88% accuracy in detecting cattle by using Inception and side views of the cattle. So, I tried implementing inception for this problem of dog breed classification.
- Also there have been researches done using resnet50 with favourable outcomes like in [4]
- In this project, I have used transfer learning with inceptionv3[5], xception[6], resnet152[7] networks pretraing on imagenet data set.

Inception v3

The Inception-v3 is an inception convolutional neural network architecture uses Label Smoothing, Factorized 7 x 7 convolutions, and the inclusion of an auxiliary classifer to strengthen back propagation.

xception

- Similar to inception model except that it uses depthwise convolutions in the place of normal convolution

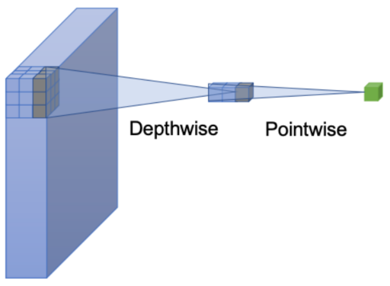


Figure 2.5: Depthwise convloution

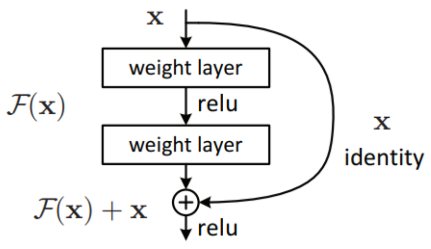

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.6: Building block for residual learning

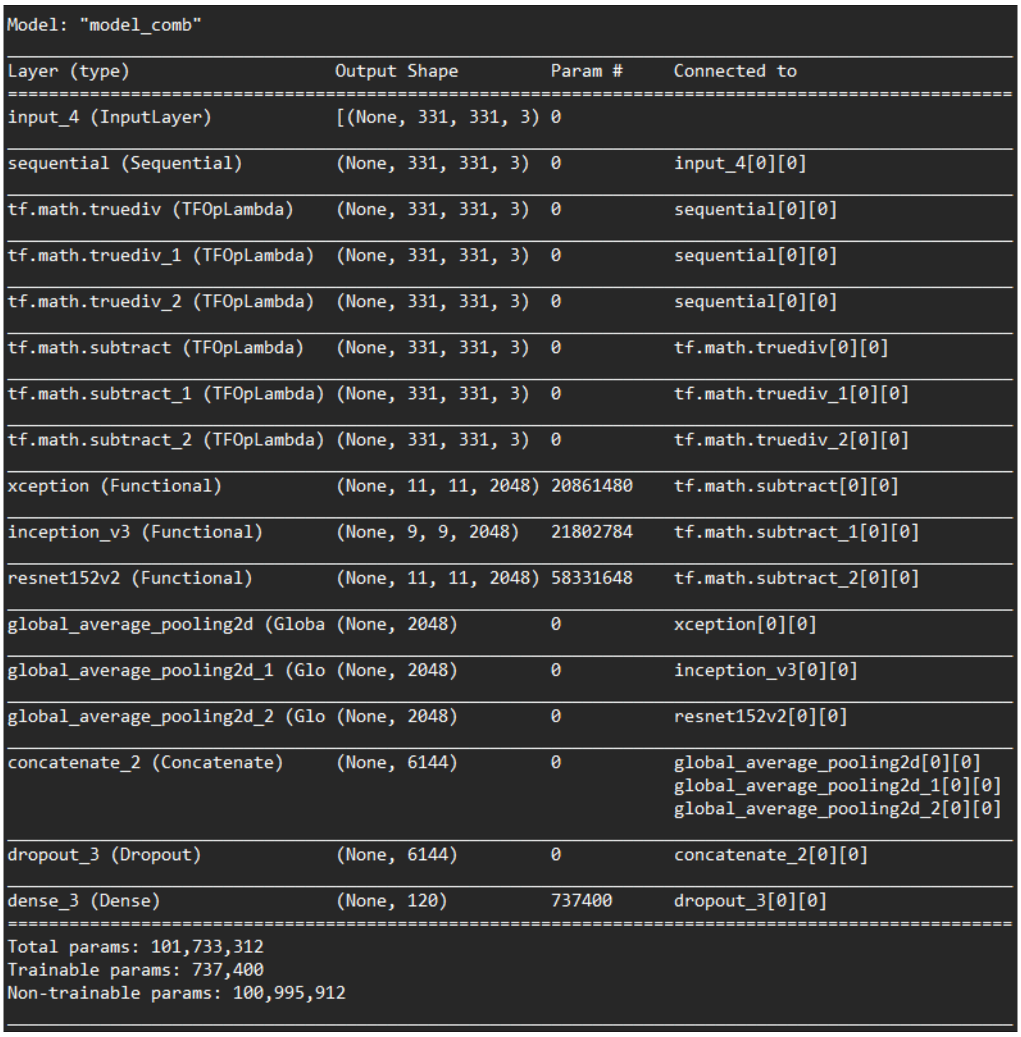
- Depthwise Separable Convolution divides the computation into two phases, as opposed to standard convolution, which does the channelwise and spatially-wise processing in one step. A single convolutional filter is applied for each input channel during depthwise convolution, and the output of depthwise convolution is then combined linearly using pointwise convolution.

Resnet

- Residual Networks, or ResNets, train residual functions with reference to the layer inputs as opposed to learning unreferenced functions. Instead than assuming that every few stacked layers will exactly match a specified underlying mapping, residual nets enable these layers to fit a residual mapping. By stacking residual blocks on top of one another, they can create networks; for instance, a ResNet-50 uses fifty layers of these blocks.
- They handle the problem of degradation and vanishing gradient. As a result just by increasing depth it will result in an accuracy increase without any problems.

2.2.2 Training

- The aforementioned models with weights are downloaded and the top fully connected layers are removed.
- Augmentation is added in pipeline after getting the inputs.
- Imported convolution layers are freezed with their weights and a pooling layer is added at the end of them.



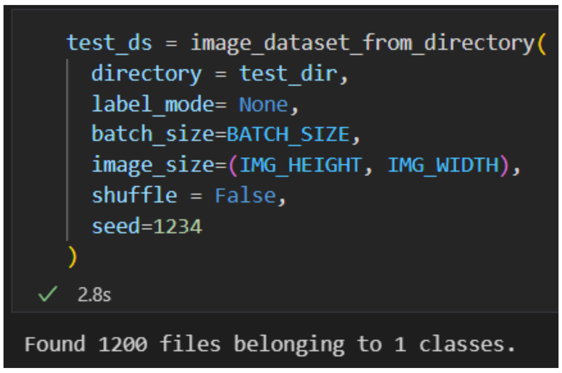
```
Model: "model_comb"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            [(None, 331, 331, 3) 0
__________________________________________________________________________________________________
sequential (Sequential)         (None, 331, 331, 3)  0           input_4[0][0]
__________________________________________________________________________________________________
tf.math.truediv (TFOpLambda)    (None, 331, 331, 3)  0           sequential[0][0]
__________________________________________________________________________________________________
tf.math.truediv_1 (TFOpLambda)  (None, 331, 331, 3)  0           sequential[0][0]
__________________________________________________________________________________________________
tf.math.truediv_2 (TFOpLambda)  (None, 331, 331, 3)  0           sequential[0][0]
__________________________________________________________________________________________________
tf.math.subtract (TFOpLambda)   (None, 331, 331, 3)  0           tf.math.truediv[0][0]
__________________________________________________________________________________________________
tf.math.subtract_1 (TFOpLambda) (None, 331, 331, 3)  0           tf.math.truediv_1[0][0]
__________________________________________________________________________________________________
tf.math.subtract_2 (TFOpLambda) (None, 331, 331, 3)  0           tf.math.truediv_2[0][0]
__________________________________________________________________________________________________
xception (Functional)           (None, 11, 11, 2048) 20861480    tf.math.subtract[0][0]
__________________________________________________________________________________________________
inception_v3 (Functional)       (None, 9, 9, 2048)   21802784    tf.math.subtract_1[0][0]
__________________________________________________________________________________________________
resnet152v2 (Functional)        (None, 11, 11, 2048) 58331648    tf.math.subtract_2[0][0]
__________________________________________________________________________________________________
global_average_pooling2d (Globa (None, 2048)         0           xception[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 2048)         0           inception_v3[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_2 (Glo (None, 2048)         0           resnet152v2[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 6144)         0           global_average_pooling2d[0][0]
                                                                 global_average_pooling2d_1[0][0]
                                                                 global_average_pooling2d_2[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 6144)         0           concatenate_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 120)          737400      dropout_3[0][0]
==================================================================================================
Total params: 101,733,312
Trainable params: 737,400
Non-trainable params: 100,995,912
__________________________________________________________________________________________________
```

Figure 2.7: Final Model's summary

- Then multiple networks are concatenated together and a dense layer with 120 neurons representing each class is added.
- Images are all resized to dimension 331x331.
- Model uses sparse categorical crossentropy as error measure
- Adam Optimizer is used with a learning rate of 0.001
- Then the network is trained for 25 epochs with batch size as 10 with call back for early stopping if the model is not get trained.
- Training and validation data sets are used.

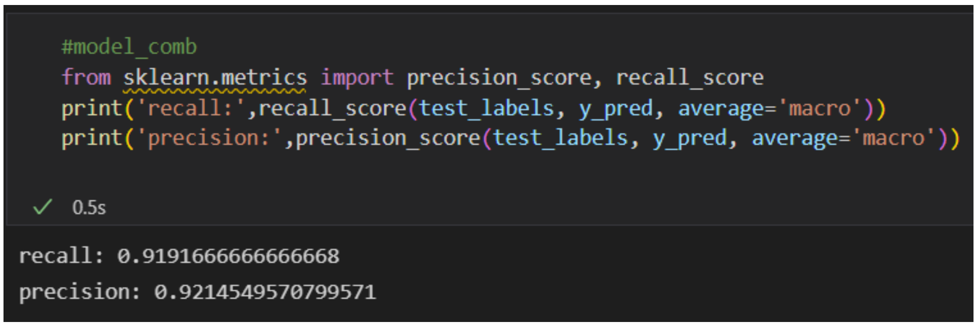
2.2.3 Testing

- Image_Dataset_From_Directory function used to access the test dataset which is in the form of a single directory consisting of 10 images from each of 120 classes adding up to 1200 samples.



```
test_ds = image_dataset_from_directory(
    directory = test_dir,
    label_mode= None,
    batch_size=BATCH_SIZE,
    image_size=(IMG_HEIGHT, IMG_WIDTH),
    shuffle = False,
    seed=1234
)
✓ 2.8s
Found 1200 files belonging to 1 classes.
```

Figure 2.8: Testing data set



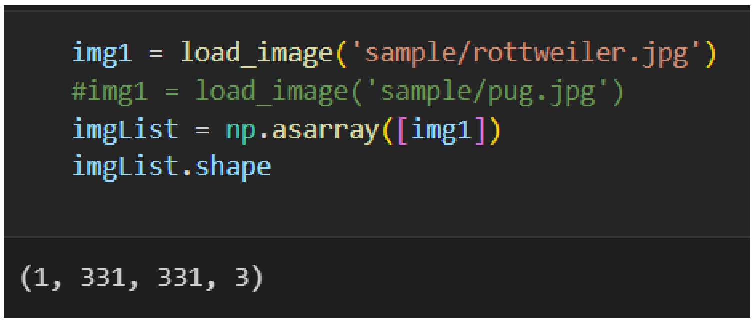
```
#model_comb
from sklearn.metrics import precision_score, recall_score
print('recall:',recall_score(test_labels, y_pred, average='macro'))
print('precision:',precision_score(test_labels, y_pred, average='macro'))

✓ 0.5s
recall: 0.9191666666666668
precision: 0.9214549570799571
```

Figure 2.9: Final Model's score

- Images are resized to the predetermined dimension of 331x331.
- The feed forward network is give the probability of each and every class in the final layer using softmax activation.
- In the end, argmax is used to get predicted class with highest probability
- Ground truth can be accessed by using test_truth.txt

2.3 Inference

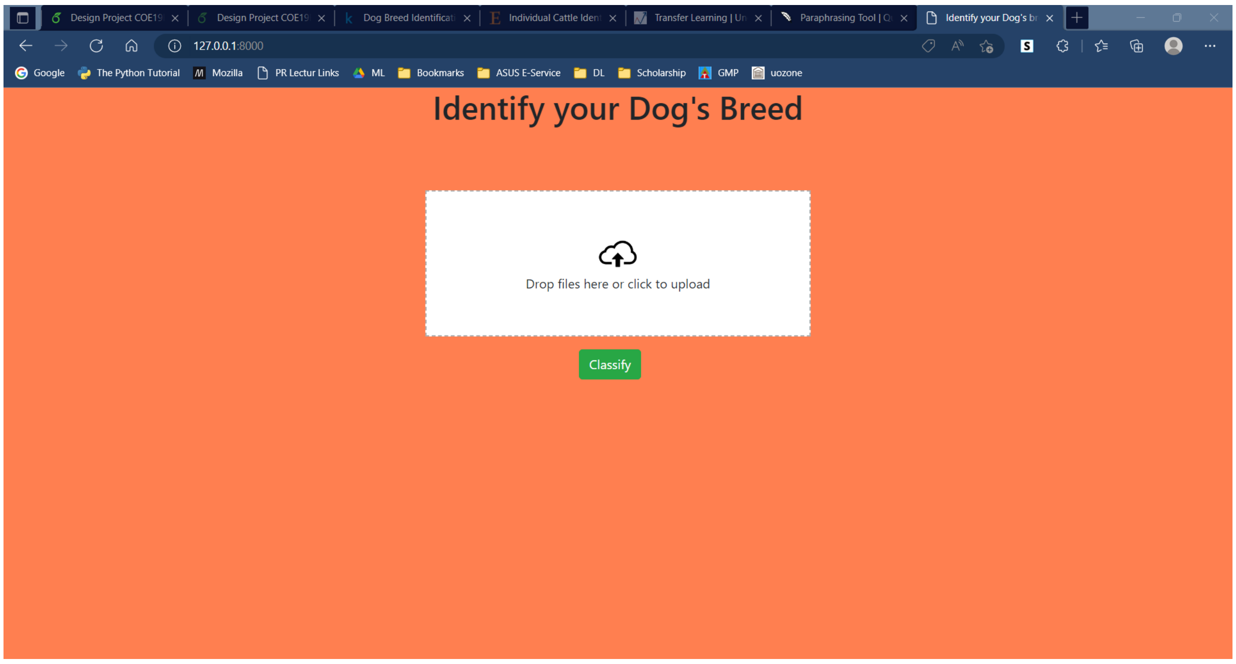
- Inference phase happens almost similar to testing.
- The model expects inputs in batches so when inferring with a input image it is necessary to convert in required format.
- And also the output of the model will be a vector of size 120 with probabilities of the image being the respective class
- Argmax function is used arrive at the class with highest probability.



```
img1 = load_image('sample/rottweiler.jpg')
#img1 = load_image('sample/pug.jpg')
imgList = np.asarray([img1])
imgList.shape
```

```
(1, 331, 331, 3)
```

Figure 2.10: For inferring



2.4 GUI

- A simple website is made using the model with django in the backend.
- User will be allowed upload an image of dog in the website and model in the backend will predict the breed and it will be displayed on the website.
- Java script is used to make website responsive.
- After classifying, automatically result will be displayed in the screen along with the probability of top three predictions.

CHAPTER 3

RESULTS AND ANALYSIS

3.1 Transfer Learning

- As a result of Transfer Learning of weights pre-trained on imagenet data set, we can see that during training both training and validation accuracies are pretty high right from the beginning.(Fig 3.1)
- It enables the model to learn effectively with relatively lesser data and lesser training.

3.2 Concatenating Ensemble

- Inception, xception, resnet all of them trained on same data and tested on same data gave accuracy in the range 86% to 90%.(Fig 3.2)
- While the ensembling of the three models by concatenating and using 120 neurons with softmax activation resulted in accuracy of 91.5%, approximatelty 3% boost in the accuracy. (Fig 3.3)

3.3 GUI Output

- I have used javascript to make a responsive website with the model using django as the back-end.
- Website is very simple with options for the user to upload image and and a button to classify the uploaded image.
- Image is then processed and the result is displayed on the screen along with top 3 class probabilities in a table. (Fig 3.4)

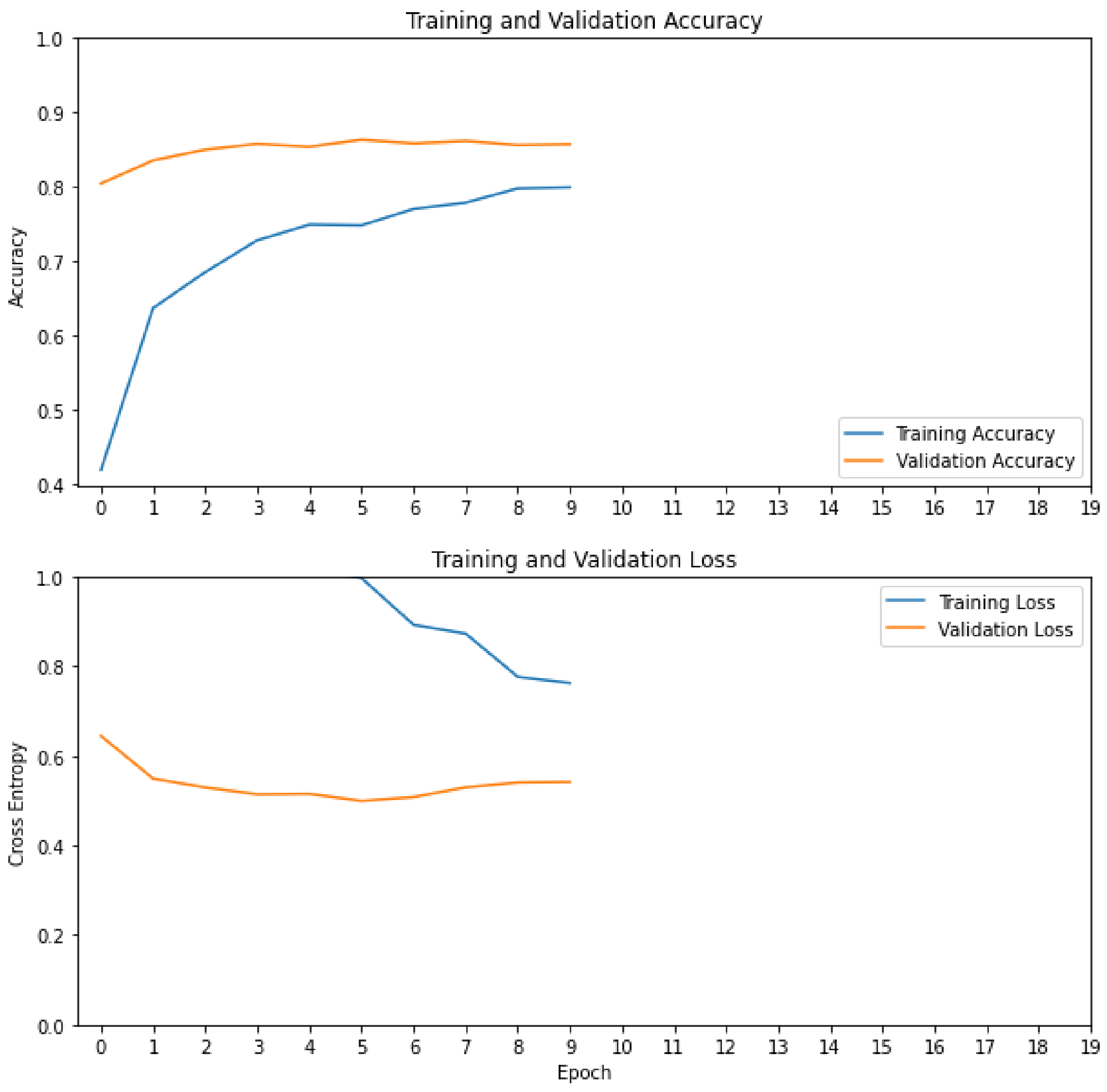
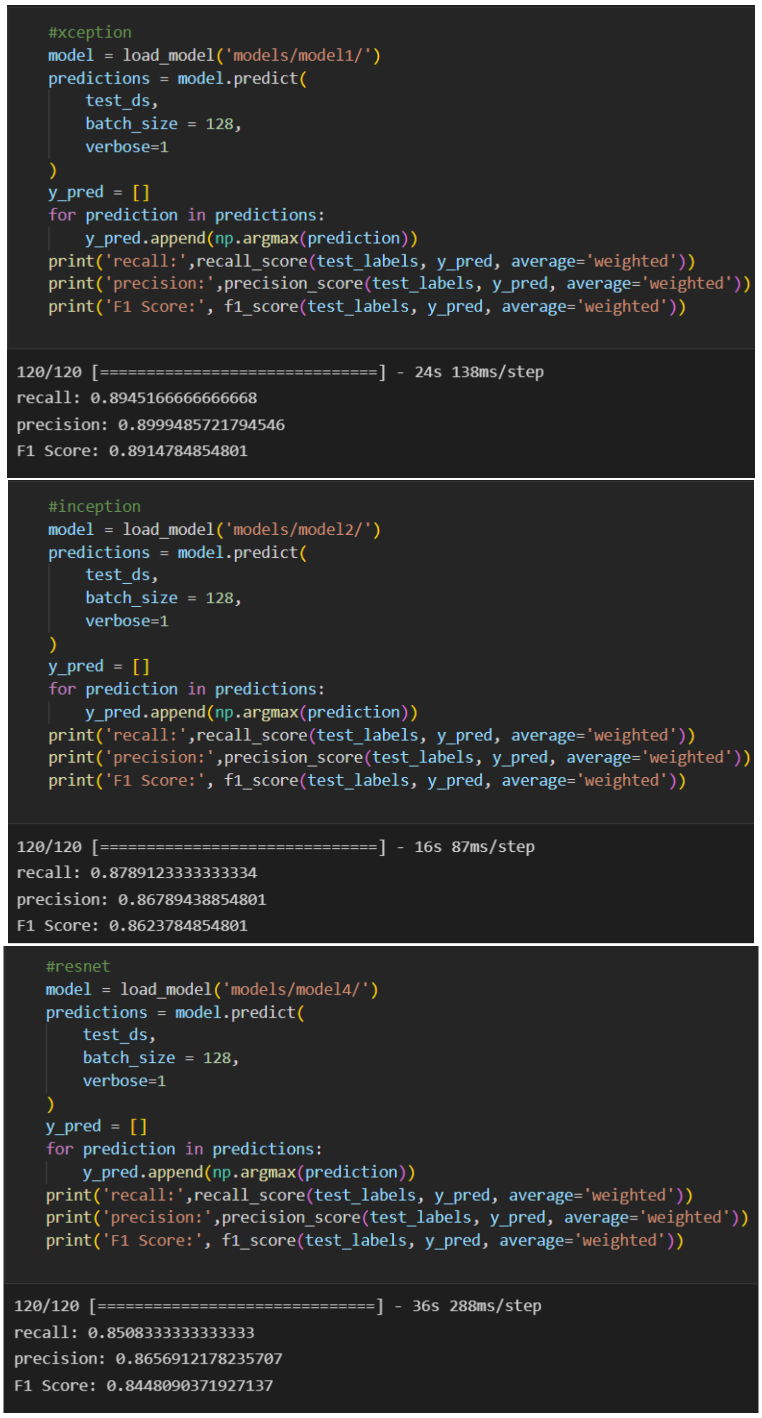


Figure 3.1: Accuracy and loss plots



```
#xception
model = load_model('models/model1/')
predictions = model.predict(
    test_ds,
    batch_size = 128,
    verbose=1
)
y_pred = []
for prediction in predictions:
    y_pred.append(np.argmax(prediction))
print('recall:',recall_score(test_labels, y_pred, average='weighted'))
print('precision:',precision_score(test_labels, y_pred, average='weighted'))
print('F1 Score:', f1_score(test_labels, y_pred, average='weighted'))
```

```
120/120 [==============================] - 24s 138ms/step
recall: 0.8945166666666668
precision: 0.8999485721794546
F1 Score: 0.8914784854801
```

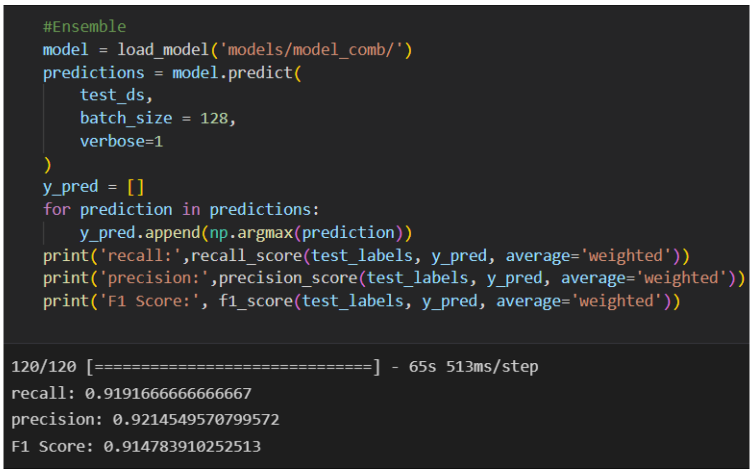
```
#inception
model = load_model('models/model2/')
predictions = model.predict(
    test_ds,
    batch_size = 128,
    verbose=1
)
y_pred = []
for prediction in predictions:
    y_pred.append(np.argmax(prediction))
print('recall:',recall_score(test_labels, y_pred, average='weighted'))
print('precision:',precision_score(test_labels, y_pred, average='weighted'))
print('F1 Score:', f1_score(test_labels, y_pred, average='weighted'))
```

```
120/120 [==============================] - 16s 87ms/step
recall: 0.8789123333333334
precision: 0.86789438854801
F1 Score: 0.8623784854801
```

```
#resnet
model = load_model('models/model4/')
predictions = model.predict(
    test_ds,
    batch_size = 128,
    verbose=1
)
y_pred = []
for prediction in predictions:
    y_pred.append(np.argmax(prediction))
print('recall:',recall_score(test_labels, y_pred, average='weighted'))
print('precision:',precision_score(test_labels, y_pred, average='weighted'))
print('F1 Score:', f1_score(test_labels, y_pred, average='weighted'))
```

```
120/120 [==============================] - 36s 288ms/step
recall: 0.8508333333333333
precision: 0.8656912178235707
F1 Score: 0.8448090371927137
```

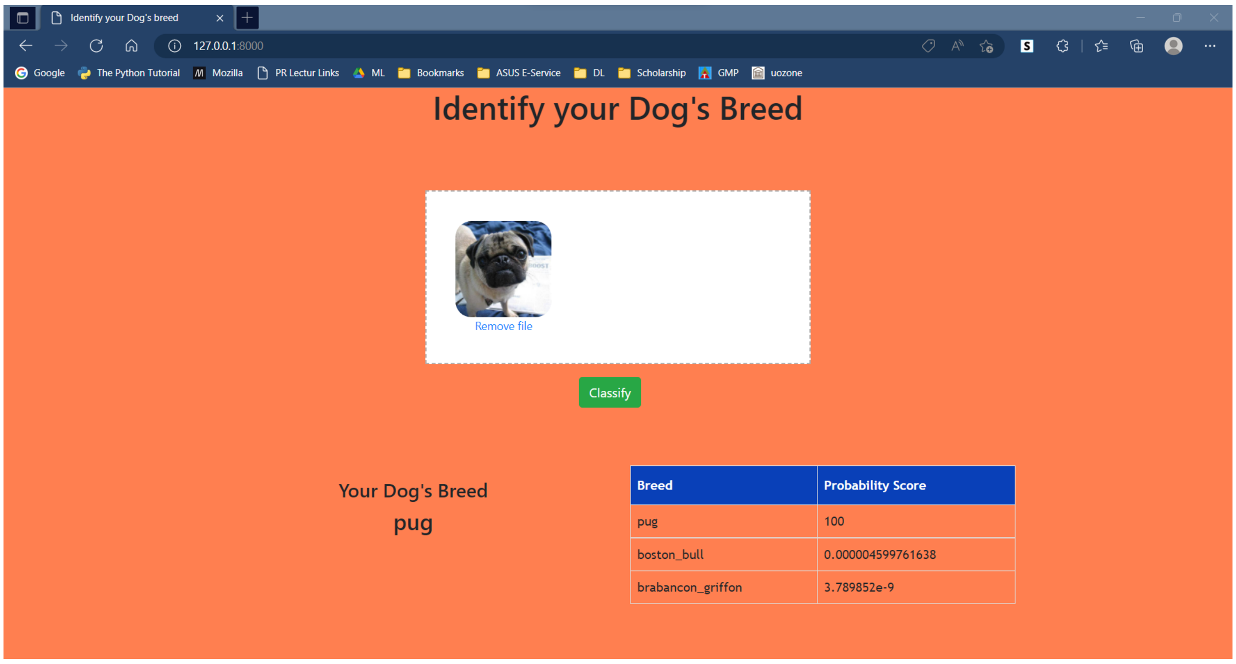
Figure 3.2: Accuracy of individual models



```
#Ensemble
model = load_model('models/model_comb/')
predictions = model.predict(
    test_ds,
    batch_size = 128,
    verbose=1
)
y_pred = []
for prediction in predictions:
    y_pred.append(np.argmax(prediction))
print('recall:',recall_score(test_labels, y_pred, average='weighted'))
print('precision:',precision_score(test_labels, y_pred, average='weighted'))
print('F1 Score:', f1_score(test_labels, y_pred, average='weighted'))
```

```
120/120 [==============================] - 65s 513ms/step
recall: 0.9191666666666667
precision: 0.9214549570799572
F1 Score: 0.914783910252513
```

Figure 3.3: Accuracy of combined network



Identify your Dog's Breed

Your Dog's Breed

pug

Breed	Probability Score
pug	100
boston_bull	0.000004599761638
brabancon_griffon	3.789852e-9

Figure 3.4: Result screen

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

4.1 Conclusion

With constant improvements being made in field of computing equipment, deep learning based algorithms have had a lot of success. Aiming to achieve a high accuracy and efficiency, people develop new architectures, extract rich features, solve sophisticated issues, combining various algorithms, develop better loss functions. Given these, the applications of object detection is so wide and new fields arise everyday. So there is always a scope of improvement.

The combined model achieves an accuracy of 92% while testing which is 5% higher than any of the three models attained when testing individually. And a web app has been to perform the task of dog breed identification with Django in the back-end.

4.2 Future Scope

Right from the beginning this project has been about classification. This project can be further expanded and tried to be implemented for cattle recognition or any other animal recognition task.

Since the model performed pretty decent for a 120 class classification, we can try expanding it for cattle identification with some alterations like in error atleast in a small scale.

In this field of study, there is always a room for improvement.

REFERENCES

[1] D. B. I. Challenge, “Data set,” *Kaggle*, 2017.

[2] B. Y. Aditya Khosla, Nityananda Jayadevaprakash and L. Fei-Fei, “Novel dataset for fine-grained image categorization.” *IEEE CVPR*, 2011.

[3] Y. Qiao, D. Su, H. Kong, S. Sukkarieh, S. Lomax, and C. Clark, “Individual cattle identification using a deep learning based framework,” *IFAC-PapersOnLine*.

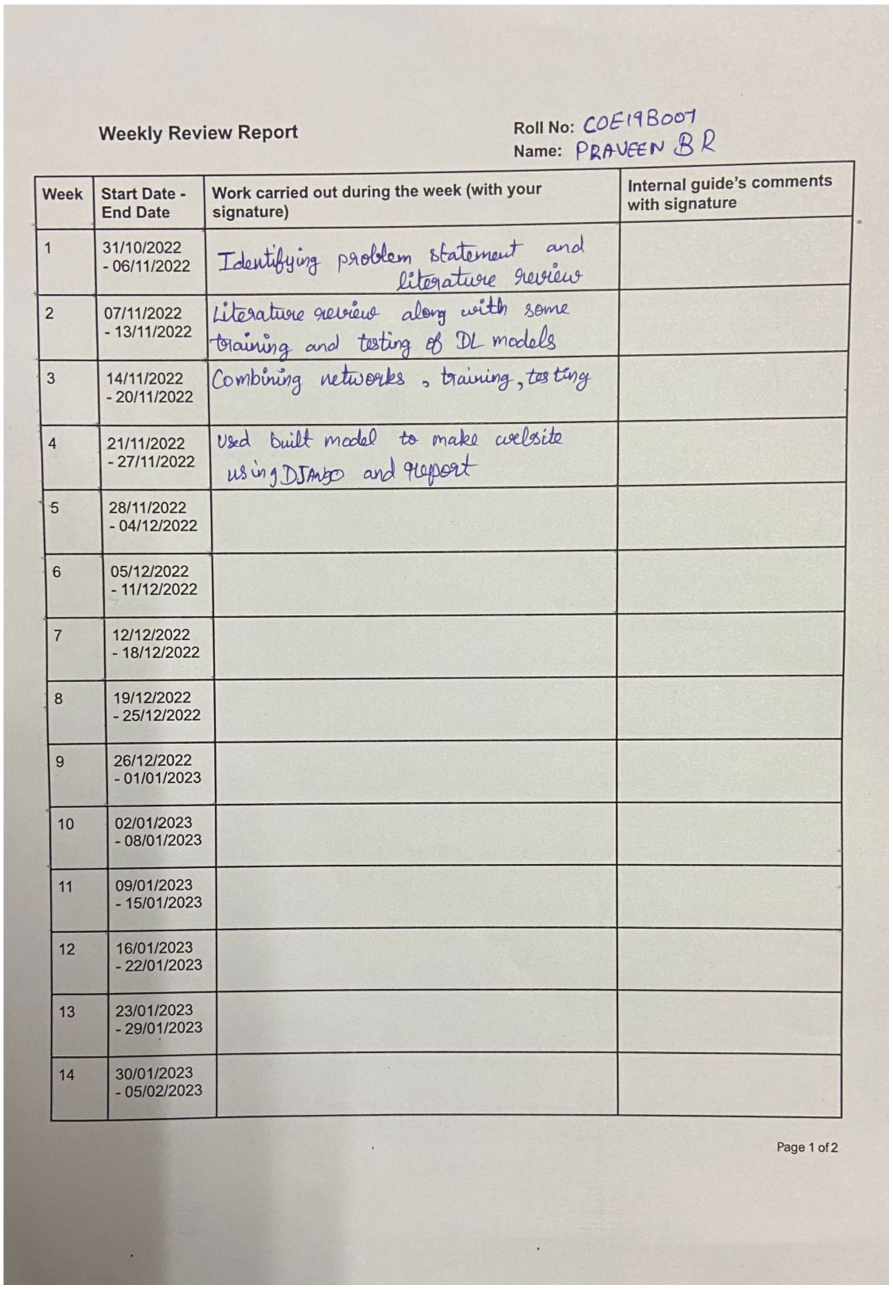
[4] H. Gong, H. Pan, L. Chen, T. Hu, S. Li, Y. Sun, Y. Mu, and Y. Guo, “Facial recognition of cattle based on sk-resnet,” *Scientific Programming*, vol. 2022, pp. 1–10, 11 2022.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[6] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” 07 2017, pp. 1800–1807.

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

Weekly Review Report



Weekly Review Report

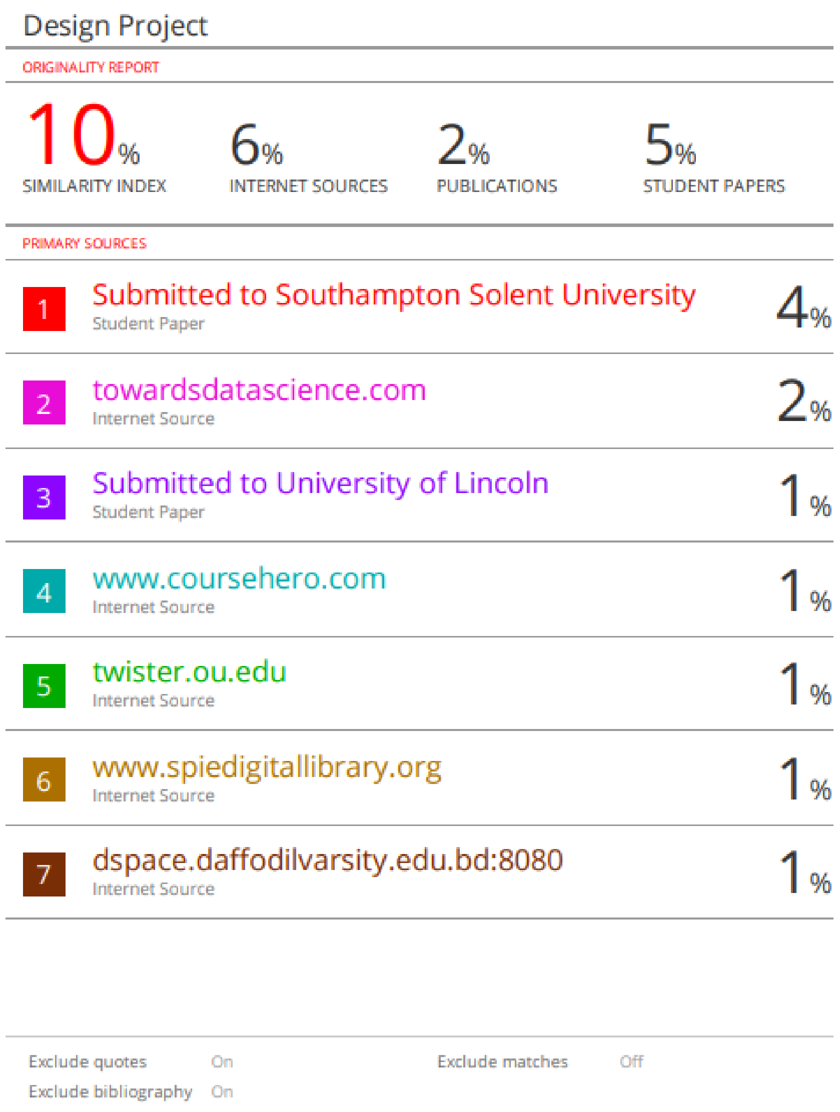
Roll No: COE19B007
Name: PRAVEEN B R

Week	Start Date - End Date	Work carried out during the week (with your signature)	Internal guide's comments with signature
1	31/10/2022 - 06/11/2022	Identifying problem statement and literature review	
2	07/11/2022 - 13/11/2022	Literature review along with some training and testing of DL models	
3	14/11/2022 - 20/11/2022	Combining networks, training, testing	
4	21/11/2022 - 27/11/2022	Used built model to make website using DJANGO and report	
5	28/11/2022 - 04/12/2022		
6	05/12/2022 - 11/12/2022		
7	12/12/2022 - 18/12/2022		
8	19/12/2022 - 25/12/2022		
9	26/12/2022 - 01/01/2023		
10	02/01/2023 - 08/01/2023		
11	09/01/2023 - 15/01/2023		
12	16/01/2023 - 22/01/2023		
13	23/01/2023 - 29/01/2023		
14	30/01/2023 - 05/02/2023		

Page 1 of 2

Figure 4.1: Weekly Report

Plagiarism Report



Design Project

ORIGINALITY REPORT

10%	6%	2%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Southampton Solent University Student Paper	4%
2	towardsdatascience.com Internet Source	2%
3	Submitted to University of Lincoln Student Paper	1%
4	www.coursehero.com Internet Source	1%
5	twister.ou.edu Internet Source	1%
6	www.spiedigitallibrary.org Internet Source	1%
7	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Figure 4.2: Plagiarism Report