Project Title

**Dog Breed Identification using CNN**

College Advisor: Smt. Smita Ponde

Session: 2022-23

Submitted by:

**Rutuja U. More (46062)**

**Abhishek A. Sapkal (46129)**

B.Tech. Computer Science

**April 2022**

**Organization: YBI Foundation, Delhi**

**College Name: Deogiri Institute of Engineering and Management Studies, Aurangabad**

**ABSTRACT**

Our project is about identifying breeds of dog from its image. The model accepts an image of dog and identify its corresponding breed.Our dataset containing almost 120 different breeds of dog.

Following are some of the characteristics of our project:

* We have implemented this model using Convolutional Neural Network (CNN)
* As CNN works in layer format so it takes time to train model
* So for improving accuracy of our model we have used Resnet-50 with technique called transfer learning
* We have got nearly 83 percent accuracy for our model

**ACKNOWLEDGEMENT**

We would like to thank my college faculty and advisor Smt. Smita Ponde for guiding and providing the opportunity to complete the internship. We would also like to than out Training and Placement In charge Pravin Mahadik for arranging the internship at YBI Foundation.

We have been privileged to learn and upskill at YBI Foundation for two months period. The admin staff and academic team is great and provide in-depth knowledge with hands-on practice to master the concepts.

|  |  |  |
| --- | --- | --- |
| Contents | | |
|  | List of Figures | I |
|  | List of Screen Shots | II |
| 1. | INTRODUCTION  * 1. Introduction   2. Problem Statement   3. Objectives   4. Applications | 1 |
| 2. | DATA COLLECTION AND ANALYSIS | 2 |
| 3. | FINAL DESIGN AND IMPLEMENTATION | 4 |
| 4. | PERFORMANCE ANALYSIS | 10 |
| 5. | CONCLUSION AND FUTURE SCOPE | 14 |
| **Annexure** Project Code | |  |

**List of Figures**

**Fig No Title of the Figure Page No**

Fig2.1 No of Images per dog breed in dataset 2

Fig2.2 Most Popular Dog Breeds 3

Fig2.3 Distribution of Dataset 3

Fig3.1 Block Diagram of the project 9

Fig4.1 Confusion Matrix 13

**List of Screenshots**

**Screenshot No Title of Screenshot Page No**

Screenshot 3.1a CNN from scratch 6

Screenshot 3.1b CNN from scratch 6

Screenshot 3.1c CNN from scratch 7

Screenshot 3.2 Accuracy of the mode 7

Screenshot 4.1a Resnet50 Model with FTRL Optimizer 10

Screenshot 4.1b Accuracy of Ftrl Resnet50 Model 10

Screenshot 4.2a Resnet50 model with Nadam optimizer 11

Screenshot 4.2b Accuracy of Nadam Resnet50 Model 11

Screenshot 4.3a Resnet50 model with Adamax optimizer 12

Screenshot 4.3b Accuracy of Adamax Resnet50 model 12

Screenshot 4.4 Accuracy of the model 13

**1.Introduction**

1.1 Introduction

Our project is to identify dog breeds if a photo is given to us using a machine learning model. This can be done by using the neural networks in machine learning. A machine learning approach to image classification involves identifying and extracting key features from images and using them as input to a machine learning model. Image classification, technically speaking, is a machine learning method and it is designed to resemble the way a human brain functions. With this method, the computers are taught to recognize the visual elements within an image. By relying on large databases and noticing emerging patterns, the computers can make sense of images and formulate relevant tags and categories.

1.2 Problem Statements

* Design a machine learning model which helps NGO worker’s to identify different dog breeds

1.3 Objectives

* A NGO is working for betterment of stray animals
* They are facing difficulties while identifying their breeds
* Design a machine learning model to help them identify breeds of different animals
  1. Application

The main application of our project is to identify different breeds of dogs if the image of a dog is provided.

**2. Data Collection and Analysis**

We have taken data set from kaggle

Link for dataset: https://www.kaggle.com/c/dog-breed-identification/data

the following were some of the insights about the data:

1. Total categories of Dog Breeds: 133
2. Total Dog Images: 8351 (Train: 6680, Valid: 835, Test: 836)
3. Most popular breeds

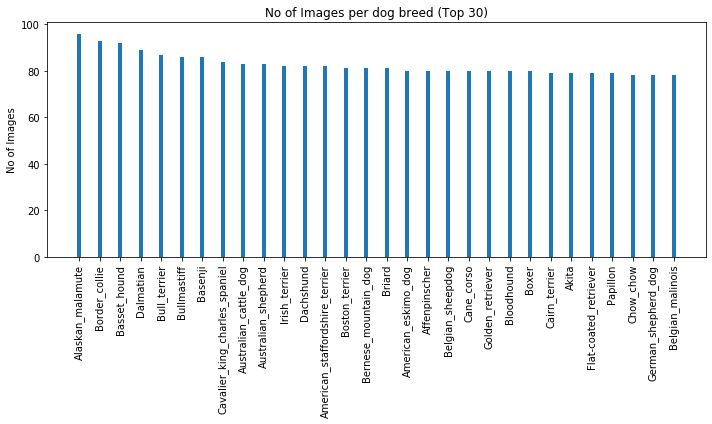


Fig 2.1: No of Images per dog breed in dataset



Fig2.2: Most Popular Dog Breeds

The division of data set in test train and valid is done as follows

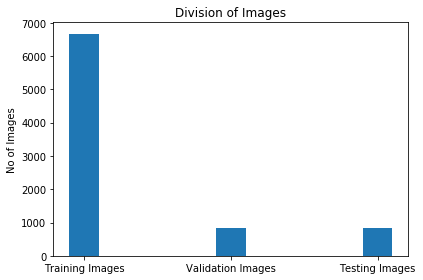


Fig 2.3: Distribution of Dataset

**3.Final Design and Implementation**

Step 1: import the dataset

We import the dataset of dog images for further modeling. We populate a few variables by using the load\_files function from the scikit-learn library.

Import dog data:

train\_files,test\_files,valid\_files - numpy arrays containing file paths to images

train\_target,test\_target,valid\_target - numpy arrays containing onehot-encoded classification labels

dog\_names - list of string-valued dog breed names for translating labels

Step 2: Detect Dogs

We use a pre-trained ResNet-50 model to detect dogs in images. Our first line of code downloads the ResNet-50 model, along with weights that have been trained on ImageNet, a very large, very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories. Given an image, this pre-trained ResNet-50 model returns a prediction (derived from the available categories in ImageNet) for the object that is contained in the image.

Pre-process the Data

When using TensorFlow as backend, Keras CNNs require a 4D array (which we’ll also refer to as a 4D tensor) as input, with shape where nb\_samples corresponds to the total number of images (or samples), and rows, columns, and channels correspond to the number of rows, columns, and channels for each image, respectively.

The path to tensor function below takes a string-valued file path to a color image as input and returns a 4D tensor suitable for supplying to a Keras CNN. The function first loads the image and resizes it to a square image that is pixels.

Next, the image is converted to an array, which is then resized to a 4D tensor. In this case, since we are working with color images, each image has three channels. Likewise, since we are processing a single image (or sample), the returned tensor will always have shape

The path to tensor function takes a numpy array of string-valued image paths as input and returns a 4D tensor with shape

Here, nb\_samples is the number of samples, or number of images, in the supplied array of image paths. It is best to think of nb\_samples as the number of 3D tensors (where each 3D tensor corresponds to a different image) in your dataset!

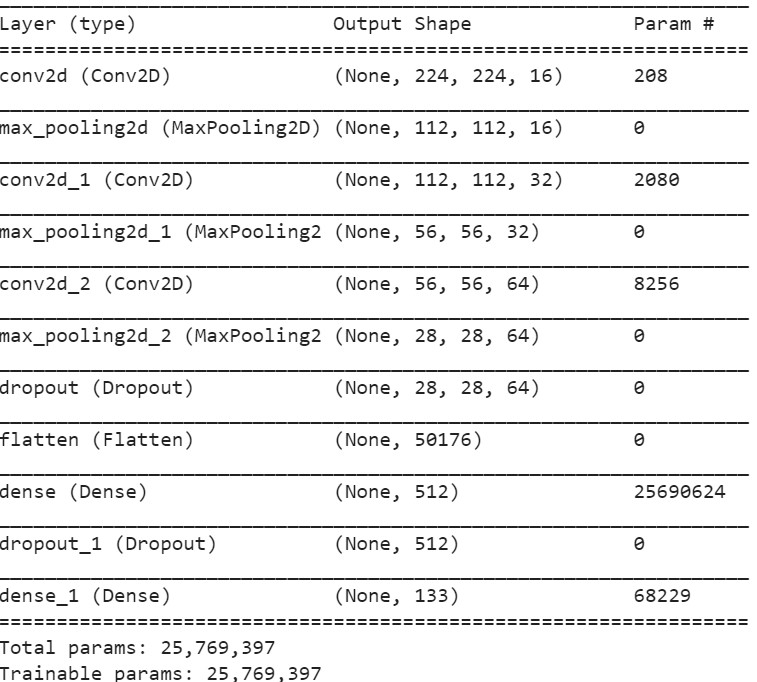
Building a CNN from Scratch

The Idea behind CNN is that local understanding of the image is good enough. Having lesser amount of parameter reduces the time required to train the model.

We build a CNN from Scratch but the accuracy of the model was very low



Screenshot 3.1a: CNN from Scratch

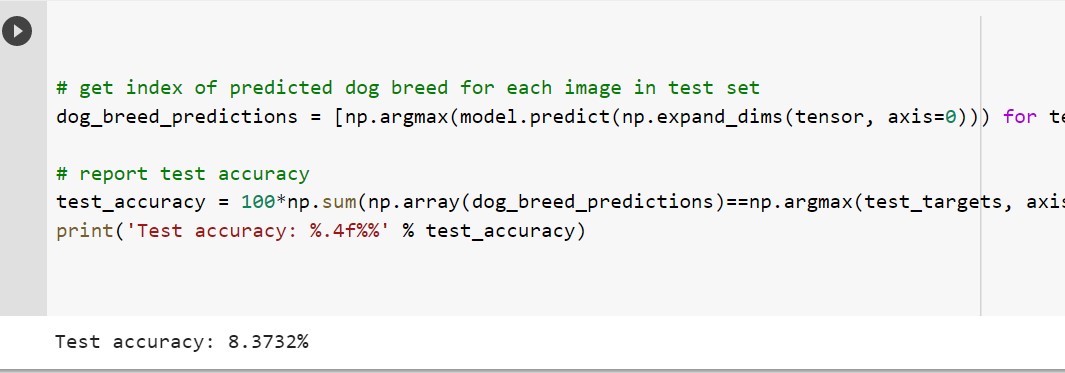


Screenshot 3.1b: CNN from Scratch



Screenshot 3.1c: CNN from Scratch

Accuracy of the above model was 8 percent so to increase the accuracy we used the pretrained resnet50 model.



Screenshot 3.2: Accuracy of the CNN model

Making Predictions with ResNet-50

Getting the 4D tensor ready for ResNet-50, and for any other pre-trained model in Keras, requires some additional processing. First, the RGB image is converted to BGR by reordering the channels. All pre-trained models have the additional normalization step that the mean pixel (expressed in RGB as and calculated from all pixels in all images in ImageNet) must be subtracted from every pixel in each image. This is implemented in the imported function preprocess\_input . If you're curious, you can check the code for preprocess\_input here.

Now that we have a way to format our image for supplying to ResNet-50, we are now ready to use the model to extract the predictions. This is accomplished with the predict method, which returns an array whose

-th entry is the model’s predicted probability that the image belongs to the

-th ImageNet category. This is implemented in the Resnet50\_predict\_labels function below.

By taking the argmax of the predicted probability vector, we obtain an integer corresponding to the model’s predicted object class, which we can identify with an object category through the use of this dictionary.

Block Diagram of the Project

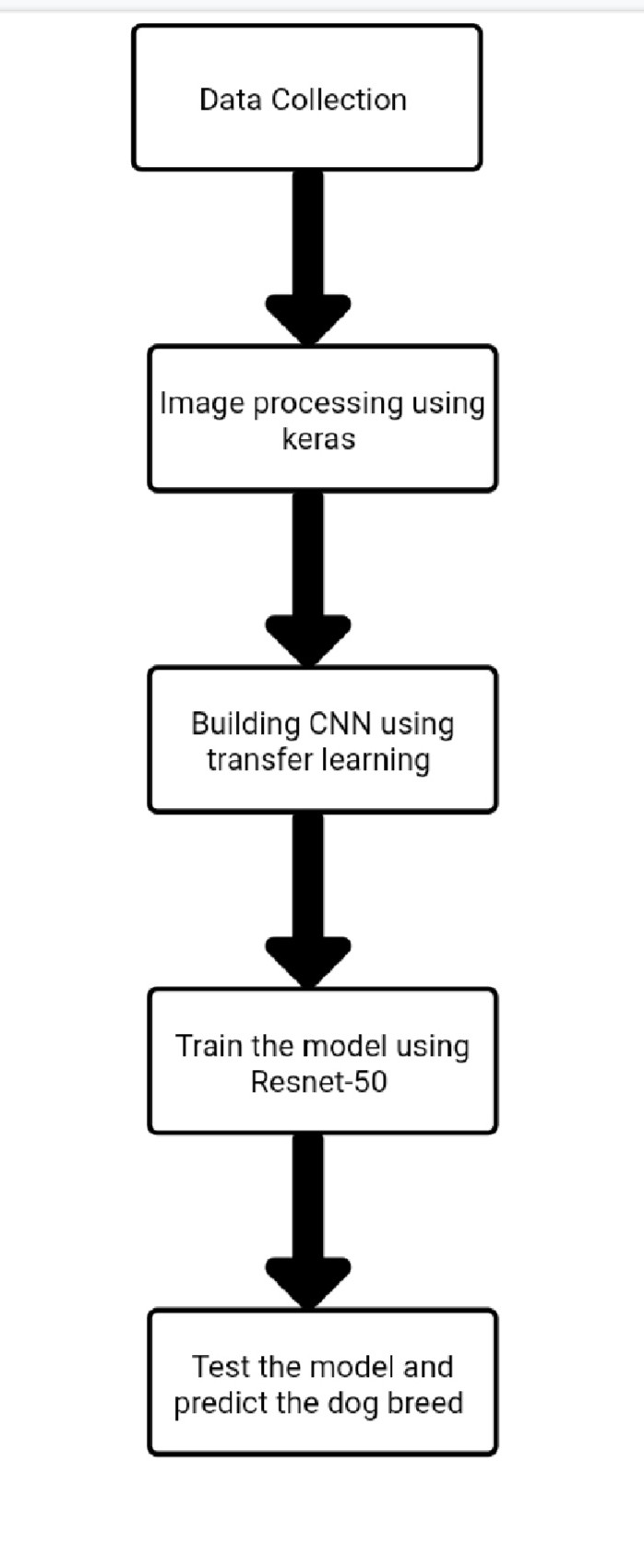
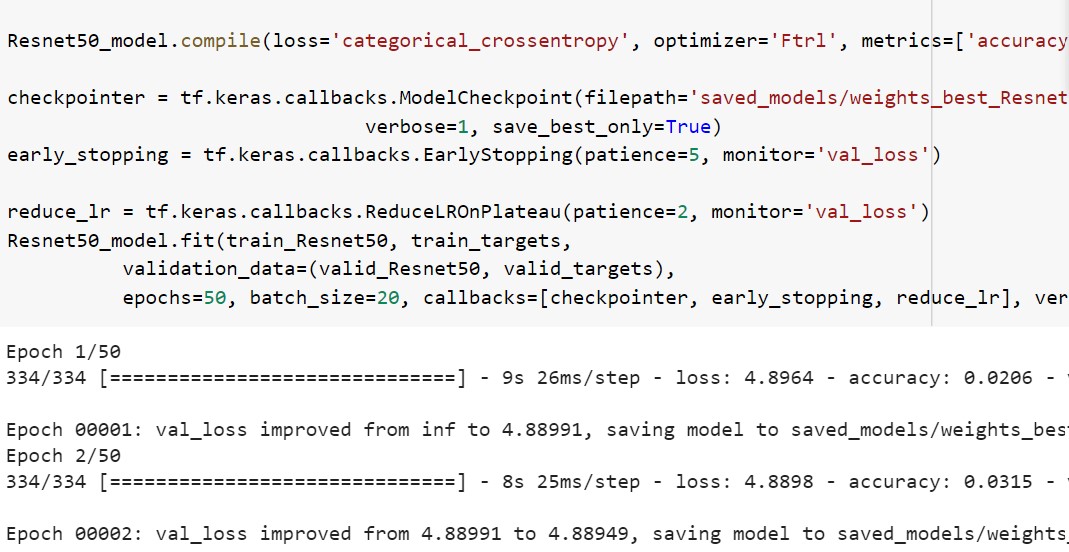


Fig 3.1: Block Diagram of Project

**4. PERFORMANCE ANALYSIS**

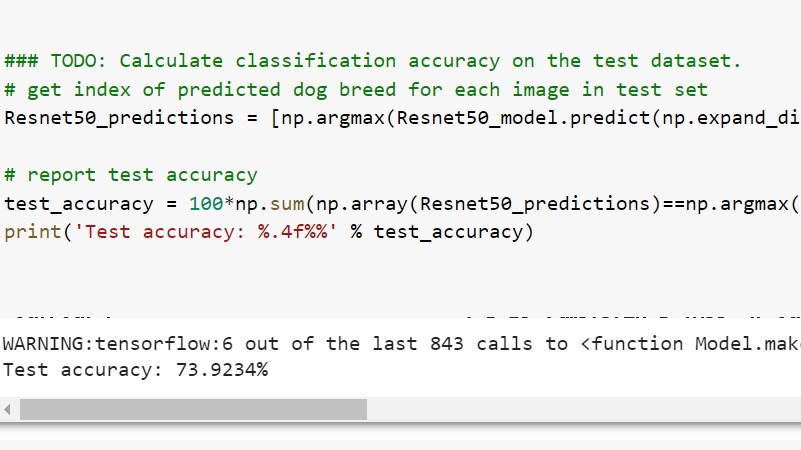
As we have used Resnet50 pre trained model it increases the accuracy of the model. The Pre-trained model contain different types of optimizers to optimize the accuracy of the model more and more we have tried different optimizers and all of them give slightly accuracy although the accuracy did not differed very much.

Optimizer Follow the regularized Leader (FTRL)



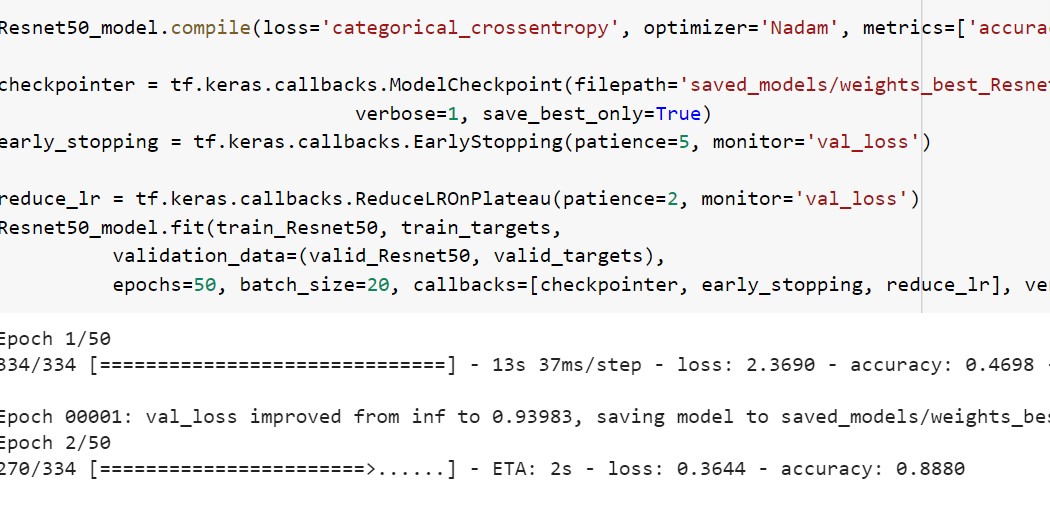
Screenshot 4.1a: Resnet50 Model with FTRL Optimizer

Accuracy of the Ftrl optimized model comes out to be 73.9 percent



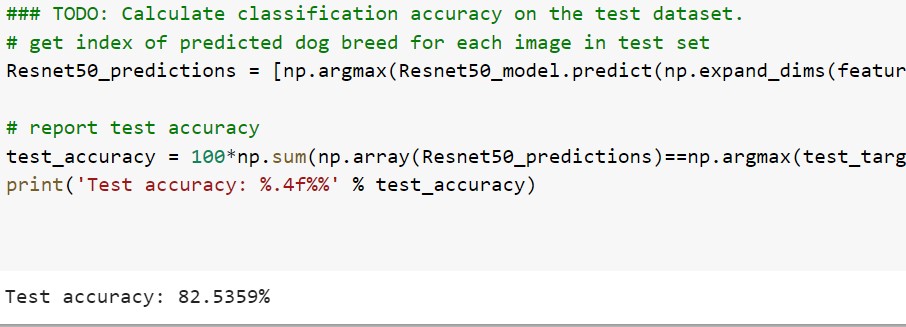
Screenshot 4.1b: Accuracy of Ftrl Resnet50 Model

Optimizer Nadam



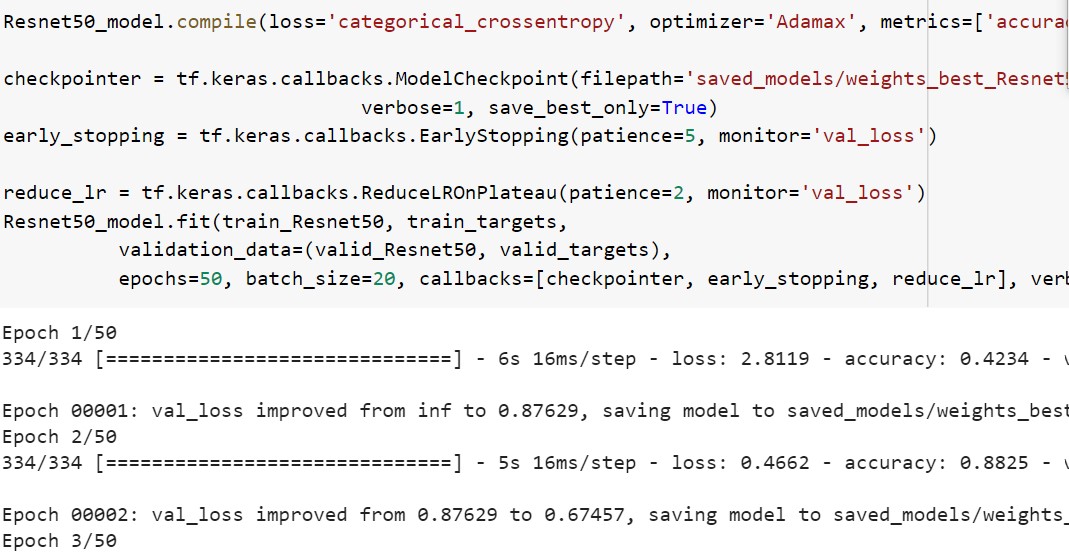
Screenshot 4.2a: Resnet50 model with Nadam optimizer

Accuracy of the Ftrl optimized model comes out to be 82 percent



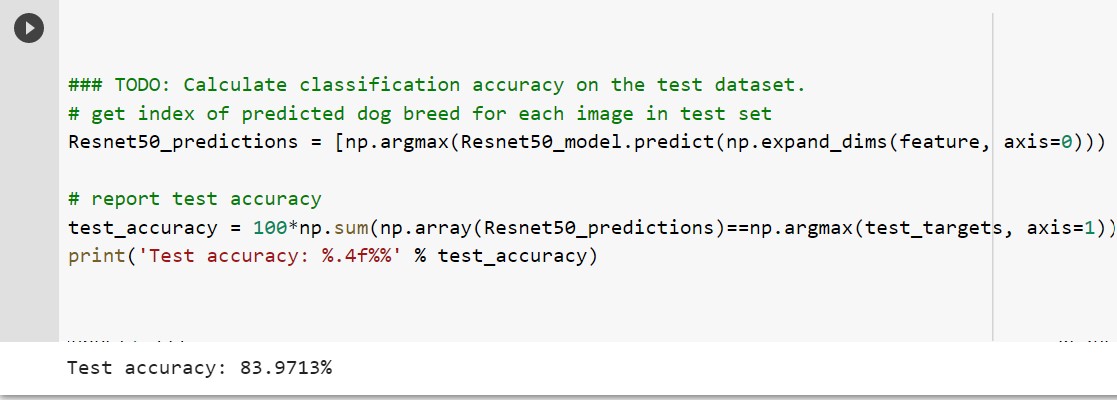
Screenshot 4.2b: Accuracy of Nadam Resnet50 Model

Optimizer Adamax



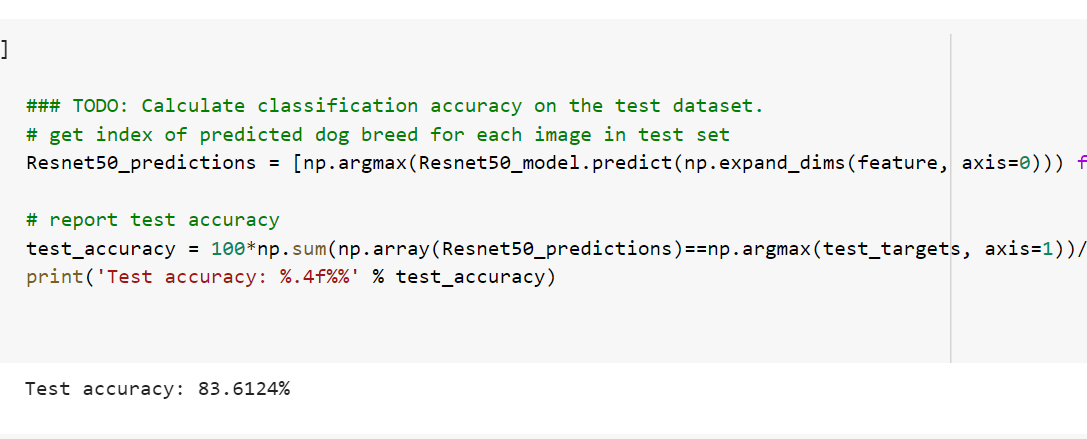
Screenshot 4.3a: Resnet50 model with Adamax optimizer

Accuracy of the Adamax optimized model comes out to be 83 percent



Screenshot 4.3b: Accuracy of Adamax Resnet50 model

As all of the above optimizers shows different accuracies that is between 73 and 84 we have used rmsprop optimizer as it has consistently given accuracy over 83 percent. Accuracy of the model after using the transfer learning comes out to be about 83 percent.



Screenshot 4.4: Accuracy of the model

Confusion Matrix :

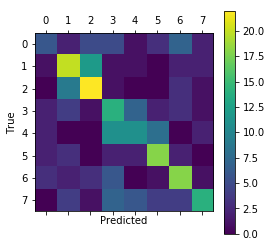


Fig 4.1: Confusion Matrix

**5. CONCLUSION**

Neural Networks are very useful in detecting the images. It can be used to identify animals or different breeds of a same animal. In our project we used neural networks to identify the breed of a dog if a image of the dog is given to us. We did our project in following stages.

1. We analyzed and pre-processed the dataset. Separate train, test and valid sets for algorithm.
2. We used a CNN from scratch that performed poorly as it failed to extract the features.
3. We then used transfer learning and the accuracy increased a lot.

Future Scope:

Future work should further explore the potential of convolutional neural networks in dog breed prediction. Given the success of our key point detection network, this is a promising technique for future project.

**Project Code**

from sklearn.datasets import load\_files

from keras.utils import np\_utils

import numpy as np

from glob import glob

def load\_dataset(path):

data = load\_files(path)

dog\_files = np.array(data['filenames'])

dog\_targets = np\_utils.to\_categorical(np.array(data['target']), 133)

return dog\_files, dog\_targets

train\_files, train\_targets = load\_dataset('/content/drive/MyDrive/train')

valid\_files, valid\_targets = load\_dataset('/content/drive/MyDrive/valid')

test\_files, test\_targets = load\_dataset('/content/drive/MyDrive/test')

len(train\_files)

len(valid\_files)

len(test\_files)

print('There are %s total dog images.\n' % len(np.hstack([train\_files, valid\_files, test\_files])))

print('There are %d training dog images.' % len(train\_files))

print('There are %d validation dog images.' % len(valid\_files))

print('There are %d test dog images.'% len(test\_files))

from keras.applications.resnet50 import ResNet50

# define ResNet50 model

ResNet50\_model = ResNet50(weights='imagenet')

from keras.preprocessing import image

from tqdm import tqdm

def path\_to\_tensor(img\_path):

# loads RGB image as PIL.Image.Image type

img = image.load\_img(img\_path, target\_size=(224, 224))

# convert PIL.Image.Image type to 3D tensor with shape (224, 224, 3)

x = image.img\_to\_array(img)

# convert 3D tensor to 4D tensor with shape (1, 224, 224, 3) and return 4D tensor

return np.expand\_dims(x, axis=0)

def paths\_to\_tensor(img\_paths):

list\_of\_tensors = [path\_to\_tensor(img\_path) for img\_path in tqdm(img\_paths)]

return np.vstack(list\_of\_tensors)

from keras.applications.resnet50 import preprocess\_input, decode\_predictions

def ResNet50\_predict\_labels(img\_path):

# returns prediction vector for image located at img\_path

img = preprocess\_input(path\_to\_tensor(img\_path))

return np.argmax(ResNet50\_model.predict(img))

### returns "True" if a dog is detected in the image stored at img\_path

def dog\_detector(img\_path):

prediction = ResNet50\_predict\_labels(img\_path)

return ((prediction <= 268) & (prediction >= 151))

from PIL import ImageFile

ImageFile.LOAD\_TRUNCATED\_IMAGES = True

# pre-process the data for Keras

train\_tensors = paths\_to\_tensor(train\_files).astype('float32')/255

valid\_tensors = paths\_to\_tensor(valid\_files).astype('float32')/255

test\_tensors = paths\_to\_tensor(test\_files).astype('float32')/255

bottleneck\_features = np.load('/content/drive/MyDrive/DogResnet50Data.npz')

train\_Resnet50 = bottleneck\_features['train']

valid\_Resnet50 = bottleneck\_features['valid']

test\_Resnet50 = bottleneck\_features['test']

import tensorflow as tf

Resnet50\_model = tf.keras.models.Sequential()

Resnet50\_model.add(tf.keras.layers.GlobalAveragePooling2D(input\_shape=train\_Resnet50.shape[1:]))

Resnet50\_model.add(tf.keras.layers.Dense(1024, activation='relu'))

Resnet50\_model.add(tf.keras.layers.Dense(133, activation='softmax'))

Resnet50\_model.compile(loss='categorical\_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

checkpointer = tf.keras.callbacks.ModelCheckpoint(filepath='saved\_models/weights\_best\_Resnet50.hdf5',

verbose=1, save\_best\_only=True)

early\_stopping = tf.keras.callbacks.EarlyStopping(patience=5, monitor='val\_loss')

reduce\_lr = tf.keras.callbacks.ReduceLROnPlateau(patience=2, monitor='val\_loss')

Resnet50\_model.fit(train\_Resnet50, train\_targets,

validation\_data=(valid\_Resnet50, valid\_targets),

epochs=50, batch\_size=20, callbacks=[checkpointer, early\_stopping, reduce\_lr], verbose=1)### TODO: Train the model.

### TODO: Calculate classification accuracy on the test dataset.

# get index of predicted dog breed for each image in test set

Resnet50\_predictions = [np.argmax(Resnet50\_model.predict(np.expand\_dims(feature, axis=0))) for feature in test\_Resnet50]

# report test accuracy

test\_accuracy = 100\*np.sum(np.array(Resnet50\_predictions)==np.argmax(test\_targets, axis=1))/len(Resnet50\_predictions)

print('Test accuracy: %.4f%%' % test\_accuracy)

import tensorflow as tf

import os

import numpy as np

from keras.preprocessing import image

import cv2

def label\_to\_category\_dict(path):

'''Returns a dictionary that maps labels to categories'''

categories = os.listdir('/content/drive/MyDrive/train')

label\_to\_cat = map(lambda x: (int(x.split('.')[0]) - 1, x.split('.')[1]), categories)

label\_to\_cat = {label: category for label, category in label\_to\_cat}

return label\_to\_cat

label\_to\_cat = label\_to\_category\_dict(train\_files)

def predict\_breed(img\_path):

'''Predicts the breed of the given image'''

# extract bottleneck features

bottleneck\_feature = extract\_Resnet50(path\_to\_tensor(img\_path))

bottleneck\_feature = tf.keras.models.Sequential([

tf.keras.layers.GlobalAveragePooling2D(input\_shape=bottleneck\_feature.shape[1:])

]).predict(bottleneck\_feature).reshape(1, 1, 1, 2048)

# obtain predicted vector

predicted\_vector = Resnet50\_model.predict(bottleneck\_feature)

# return dog breed that is predicted by the model

return label\_to\_cat[np.argmax(predicted\_vector)]

def extract\_Resnet50(tensor):

'''Returns the VGG16 features of the tensor'''

return tf.keras.applications.resnet50.ResNet50(weights='imagenet', include\_top=False).predict(tf.keras.applications.resnet50.preprocess\_input(tensor))

predict\_breed('/content/drive/MyDrive/Akita\_00258.jpg')

predict\_breed('/content/drive/MyDrive/Collie\_03849.jpg')

**REFRENCES**

* https://nouman10.medium.com/dont-know-the-breed-of-your-dog-ml-can-help-6558eb5f7f05
* https://medium.com/nanonets/how-to-easily-build-a-dog-breed-image-classification-model-2fd214419cde