Dog_Classifier

Table of Contents:

Project Overview:

This Project is related to learning and implementing image classification using Deep neural network. This project uses Convolutional Neural Networks (CNNs)! In this project, I built a pipeline to process real-world, user-supplied images. Given an image of a dog, the algorithm will identify an estimate of the canine's breed out of roughly 133 known breeds. If supplied an image of a human, the code will identify the resembling dog breed.

The Process flow steps followed in the python Note book "dog_app.ipynb" are mentioned below;

Step 0: Import Datasets

Step 1: Detect Humans

Step 2: Detect Dogs

Step 3: Create a CNN to Classify Dog Breeds (from Scratch)

Step 4: Use a CNN to Classify Dog Breeds (using Transfer Learning)

Step 5: Create a CNN to Classify Dog Breeds (using Transfer Learning)

Step 6: Write your Algorithm

Step 7: Test Your Algorithm

Problem Statement:

Here we will be dealing with a image classification related to Dog breed identification. Algorithm has to identify and classify the Dog and its breed through the image as an input.

Metrics:

Here Accuracy is the chosen metrics for the Model evaluation. I have used 3 Models (VGG16, VGG19 & ResNet50) for evaluation and then improved the accuracy of ResNet50 model by adding pooling layers and hyperparameter tunings.

Some of the key libaries used are listed below along with the paths for the imported libraries;

from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D from keras.layers import Dropout, Flatten, Dense from keras.models import Sequential from keras.callbacks import ModelCheckpoint from tensorflow.keras.callbacks import ModelCheckpoint from pathlib import Path import random from keras.preprocessing import image from tqdm import tqdm from PIL import ImageFile

Data Exploration:

Provided Dog and Human dataset was explored through the respective classifiers (dog_detector & face_detector). following were the findings through Data exploration

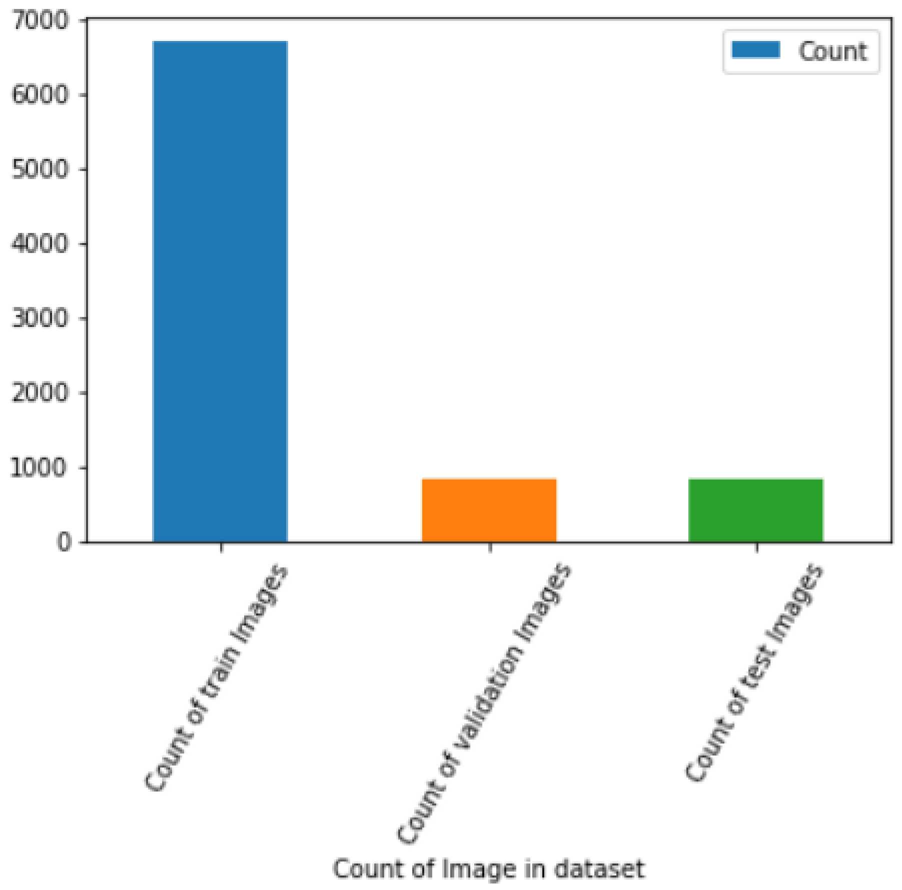
There are 133 total dog categories. There are 8351 total dog images.

There are 6680 training dog images. There are 835 validation dog images. There are 836 test dog images.

There are 13233 total human images.

Data Visualization:

I have covered the findings in above tab (Data Visualization).



Data Preprocessing:

The uploaded images were preprocessed using following steps in the notebook.

While using TensorFlow as backend, Keras CNNs require a 4D array (which we'll also refer to as a 4D tensor) as input, with shape (nb_sampls,rows,columns,channels), 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.

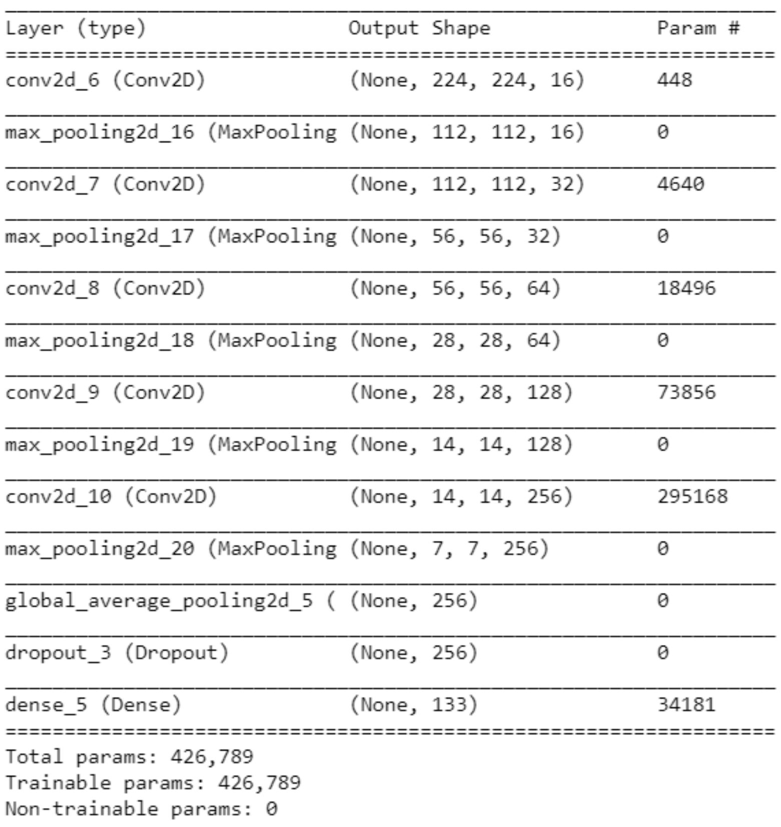
The path_to_tensor function below takes a string-valued file path to a color image as input and returns a 4D tensor suitable for supplying to a Keras CNN. The function first loads the image and resizes it to a square image that is 224×224 pixels. Next, the image is converted to an array, which is then 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 of (1,224,224,3).

The paths_to_tensor function takes a numpy array of string-valued image paths as input and returns a 4D tensor with shape (nb_samples,224,224,3).

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!

Implementation:

After writing algorithm for Dog and Human face Detectors I Created a CNN Model to Classify Dog Breeds (from Scratch). it needed help of GPU to get trained. The below mentioned CNN Architecture was used;



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_6 (Conv2D)            (None, 224, 224, 16)      448       
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 112, 112, 16)      0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 112, 112, 32)      4640      
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 56, 56, 32)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 56, 56, 64)        18496     
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 28, 28, 64)        0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 28, 28, 128)       73856     
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 14, 14, 128)       0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 14, 14, 256)       295168    
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 7, 7, 256)         0         
_________________________________________________________________
global_average_pooling2d_5 ( (None, 256)               0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_5 (Dense)              (None, 133)               34181     
=================================================================
Total params: 426,789
Trainable params: 426,789
Non-trainable params: 0
```

Refinement:

With above model , we got Test accuracy: 30.3828%.

To improve model, To reduce training time without sacrificing accuracy, I train a CNN using transfer learning. I used 3 pretrained models using Botlleneck features as a prasing parameter.

Model Evaluation and Validation:

I train a CNN using transfer learning. I used 3 pretrained keras models (VGG16, VGG19, ResNet50) with transfer learning technique to reduce training time without sacrificing original accuracy.

The Accuracy we got from the different models is mentioned below;

Base Model (Training CNN from scratch): Test accuracy: 30.3828%.

VGG16_predictions: Test accuracy: 47.4880%.

VGG19_predictions: Test accuracy: 49.4019%.

ResNet50_prediction: Test accuracy: 82.4163%

Justification:

I used ResNet50 model as my model for final predictions as its base accuracy was best out of 3 models I tried.

To enhance the Test accuracy and reduce execution time, Created a CNN to Classify Dog Breeds (using Transfer Learning). As the given dataset is small with 6680 entries and new data is similar to original training data, we use the following method as guideline for Architecture;

a) slice off the end of the neural network

b) add a new fully connected layer that matches the number of classes in the new data set.

c) randomize the weights of the new fully connected layer; freeze all the weights from the pre-trained network.

d) train the network to update the weights of the new fully connected layer.

With Resnet50 network the model Test accuracy has improved to 79.4258% from 42.8230%. This is a significant improvement. In Next step, will change few of the tuning parameters to check if the accuracy further improves.

Added a Dropout layer with 0.4 Dropout rate before output layer, improved Test accuracy to 81.93%. Next I will change the Epoch and check for the result.

With 0.4 Dropout rate, epochs=20 and Batch size=on Resnet50 network model, The Test accuracy improved to 82.4163%.

Improvement:

The output is better than I expected with 90% correct classifications. There are few improvements which I feel should help for improving the Test Accuracy.

1. Adding more images of dogs to increase the training dataset as current data set is relatively small to predict 133 variety of Dog Breed.

2. Further fine tune model parameters e.g. epochs, dropout rate & Batch size.

3. Train the model using other Transfer learning model using Inception, Exception networks.

4. Increase the depth of the network for identifying additional features by adding more set of Convolutional layer and pooling layer.

5. Validation loss is not increasing consistently with increasing the epochs. Need to experiment with reduced epochs for the Test accuracy improvement.

Conclusion:

After training the pretrained ResNet50 model by adding pooling layers and hyperparameter tuning, the accuracy of 82.4163% was achieved.

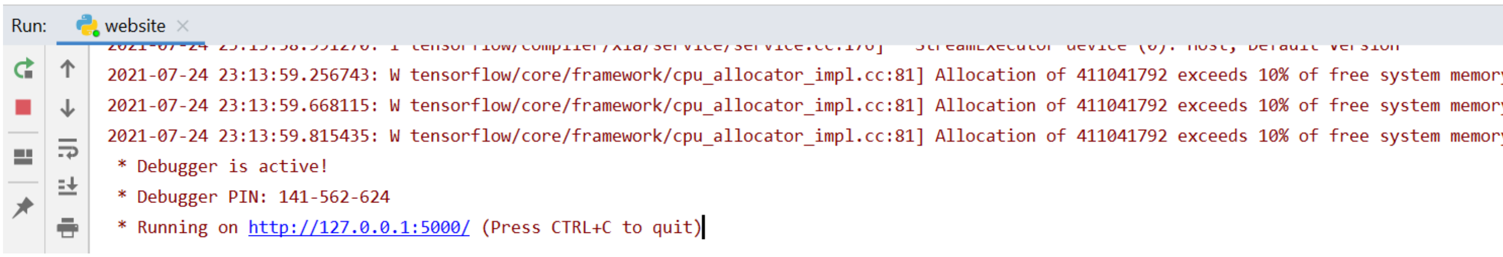
Web app for Dog breed classification

Key elements Web app for Dog breed classification:

1. Train.classifer.py: CNN model was trained and saved as "trained_saved_model.h5".
2. website.py : Flask app to predict the uploaded image as a dog breed classifier. VGG16 pretrained classifier was used for image classification.
3. index.html : This file is linked with flask app and coded in html which enables user to upload the image and then predict the dog breed with probability numbers.

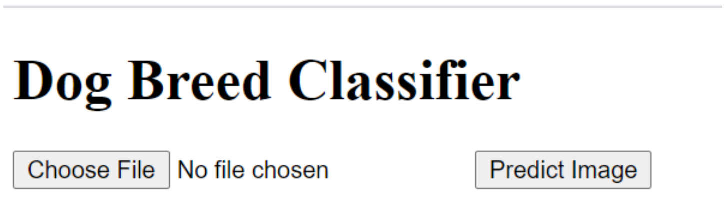
Instructions for running/testing the Web app:

Step-1: run website.py file in the environment. It will show as "Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)"



```
2021-07-24 23:13:58.991270: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
2021-07-24 23:13:59.256743: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 411041792 exceeds 10% of free system memor
2021-07-24 23:13:59.668115: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 411041792 exceeds 10% of free system memor
2021-07-24 23:13:59.815435: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 411041792 exceeds 10% of free system memor
 * Debugger is active!
 * Debugger PIN: 141-562-624
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

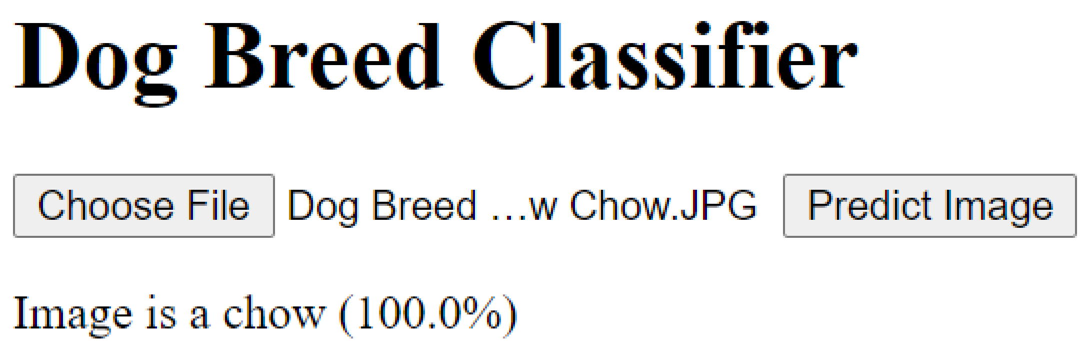
Step-2: Click on "http://127.0.0.1:5000/" link or manually type the address in the browser. Following page will show up.



Dog Breed Classifier

Step-3: Use "Choose File" button for uploading the image and "Predict Image" button for prediction.

Dog Breed Chow Chow.JPG



Dog Breed Classifier

Image is a chow (100.0%)