Dog Breed Classifier Project Overview

Definition

Classification is the process of classifying things based on the similarity of features. It is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observations. Classification is one of the several methods intended to make the analysis of very large datasets effective.

Some of the examples of classification using machine learning are:

- Sign Language Identification
- Speech Recognition
- Object Classification

Problem Statement

Dogs are people's best friends. However, there is literally countless number of dog breeds in the world. How can one tell what the breed of a dog is if first met or given a picture/video clip? This project aims to develop a deep leaning model using convolutional neural network framework that can distinguish a breed of a dog given a picture of the

dog. The finished model should firstly feature the ability to distinguish whether the supplied picture is a dog or not. Secondly, the model should accurately identify dog breeds.

Metrics

The metrics for judging how the CNN model performs are validation loss values and prediction accuracy against test dataset. The validation loss value is defined by crossentropy loss, which is also called the log loss or multi-class loss in some platforms (e.g. Kaggle). The loss value measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy values increases if the predicted probability diverges from the actual labels.

Introduction

In this venture, I constructed and prepared a neural system model with CNN (Convolutional Neural Networks) move getting the hang of, utilizing 8,351 canine pictures of 133 varieties. CNN is a kind of profound neural systems, which is normally used to examine picture information. Regularly, a CNN engineering comprises of convolutional layers, actuation work, pooling layers, completely associated layers and standardization layers. Move learning is a method that permits a model created for an undertaking to be reused as the beginning stage for another assignment.

The prepared model in this venture can be utilized by a web or portable application to process genuine world, client provided pictures. Given a picture of a canine, the calculation will foresee the variety of the pooch. On the off chance that a picture of a human is provided, the code will recognize the most taking after pooch variety of that individual.

Fascinating and troublesome things in the task:

It was astounding to perceive how CNN calculation functions so well in pictures. There are 133 classifications and CNN does so well in anticipating those classes. The most troublesome part is building my own CNN calculation. Neural system has such a large number of parameters that it is hard to tune them. Be that as it may, on account of udacity's example model, which helped me to assemble my model with more prominent than one percent exactness.

Load Datasets

The full dataset utilized by this task contains 8,351 pictures of 133 classifications of mutts. The information is isolated into three envelopes for preparing, approval, and test set. The load_files work from the scikit-learn library is utilized to import the datasets.

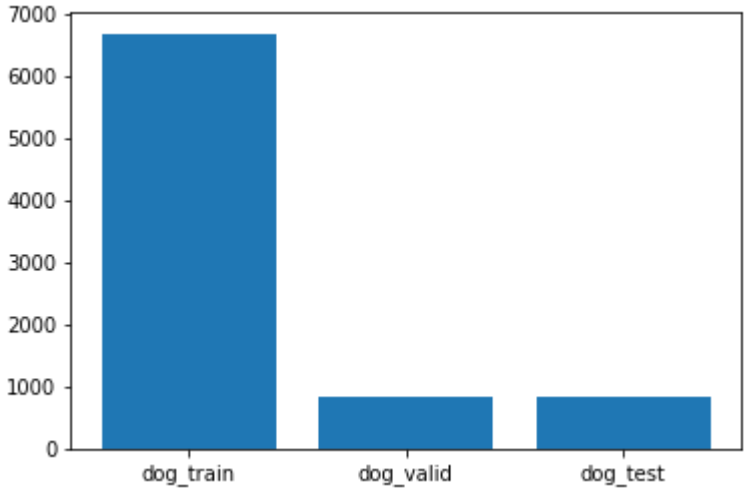
```
from sklearn.datasets import load_files
from keras.utils import np_utils
import numpy as np from glob import glob# define function to load train, test,
and validation datasetsdef load_dataset(path):  data = load_files(path)
 dog_files = np.array(data['filenames'])
 dog_targets = np_utils.to_categorical(np.array(data['target']), 133)
 return dog_files, dog_targets# load train, test, and validation datasetstrain_files, train_targets =
load_dataset('../../../data/dog_images/train')
valid_files, valid_targets = load_dataset('../../../data/dog_images/valid')
test_files, test_targets = load_dataset('../../../data/dog_images/test')# load list of dog names
dog_names = [item[20:-1] for item in sorted(glob("../../../data/dog_images/train/*/"))]
```

Analyze :

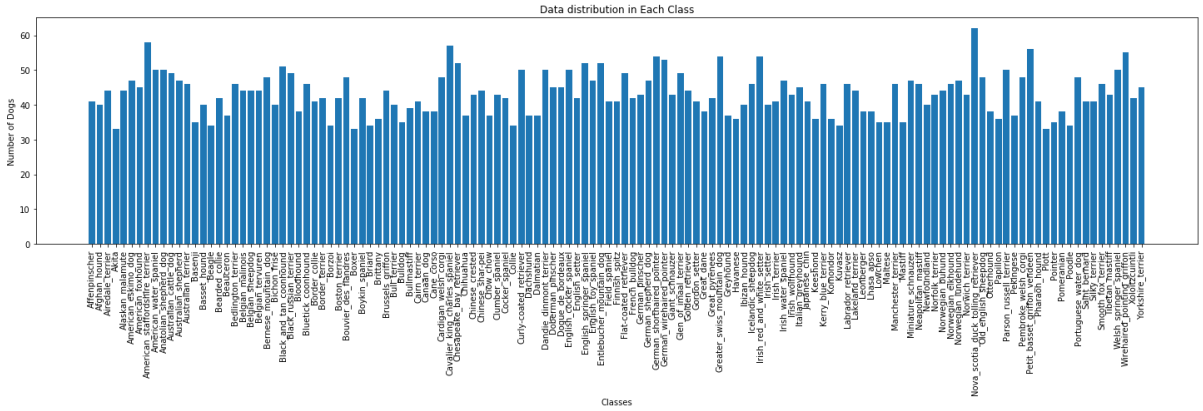
Here, in the Dog Breed Classification, the dataset contains the pictures of Dogs and Humans. There are a sum of 133 varieties, 8351 pictures

for hounds. Utilizing these pictures as information, it must be handled by our requirements and a model must be intended to prepare our machine.

On making the investigation on the information, we see that the goals of the pictures are not the equivalent for all pictures of pooches in thier particular varieties. The pictures have a shifted goals and they should be resampled dependent on the necessity of our model. Here are some expamples of the pictures examined as far as goals:



The investigation on the circulation of information gives us the data on equalization or irregularity in the information. In the event that there is a loose side in the information past a specific edge, we should see that the information is adjusted by including applicable pictures. On the off chance that the equalization in the information is relatively close to the edge, it is acceptable to convey forward with the activity. Let us perceive how it functions with our information in the beneath figure. The plot shows an away from on breed class with the quantity of mutts.



Benchmarks

The benchmark for the model can be referenced to the Kaggle leaderboard for dog breed identification competition. The target for this model is to reach a multiclass loss score less than 0.01, which is in the top 100 of the competition. The other benchmark will be 80% prediction accuracy, which will be used as the upper limit. The benchmark set by Udacity will be 60% prediction accuracy, which will be used as the lower limit. The final performance of the model will sit in between the two limits.

Detect Humans

Since we need to recognize the most taking after pooch breed for an individual, a capacity should be composed to identify whether a human face exists in a picture. This undertaking utilized a preprepared face locator gave by OpenCV. If it’s not too much trouble note that the information picture is changed over to grayscale before it is taken care of into the face course classifier.

```
 import
cv2
import matplotlib.pyplot as plt
%matplotlib inline# extract pre-trained face detector
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')# load color
(BGR) image img = cv2.imread(human_files[3]) # convert BGR image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)# find faces in image faces =
face_cascade.detectMultiScale(gray)# print number of faces detected in the image
print('Number of faces detected:', len(faces))# get bounding box for each detected
face for (x,y,w,h) in faces:
 # add bounding box to color image
 cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

# convert BGR image to RGB for plotting
cv_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)# display the image, along with bounding
box
plt.imshow(cv_rgb)
plt.show()
```

Classify Dog Breeds using Transfer Learning with CNN

The full dataset has 8,351 pictures of dog, which isn't sufficiently enormous to prepare a profound taking in model without any preparation. Subsequently, move learning with VGG-19 (a convolutional neural system that is prepared on in excess of a million pictures from the ImageNet database) is utilized to accomplish generally great exactness with less preparing time.

```
 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
```

Model Architecture

The last convolutional yield of VGG-19 is taken care of as contribution to the model. We just need to include a worldwide normal pooling layer and completely associated layers as pooch classifiers.

```
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Dropout, Flatten, Dense from keras.models import Sequentialmodel = Sequential() model.add(Conv2D(filters=32, kernel_size=2, strides=1, padding='same', activation='relu', input_shape=(train_tensors.shape[1], train_tensors.shape[2], train_tensors.shape[3])))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=65, kernel_size=2, strides=1, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=128, kernel_size=2, strides=1, padding='same', activation='relu')) model.add(MaxPooling2D(pool_size=(2,2)))
model.add(GlobalAveragePooling2D()) model.add(Dense(train_targets.shape[1], activation='softmax'))model.compile(optimizer='rmsprop',
loss='categorical_crossentropy', metrics=['accuracy'])from keras.callbacks import ModelCheckpointepochs = 5checkpointer = ModelCheckpoint(filepath='saved_models/weights.best.from_scratch.hdf5',
verbose=1, save_best_only=True)model.fit(train_tensors, train_targets, validation_data=(valid_tensors, valid_targets), epochs=epochs, batch_size=20, callbacks=[checkpointer], verbose=1)
```

Train Model

The model is prepared utilizing the pre-registered bottleneck includes as info. A model check pointer is utilized to monitor the loads for best approval misfortune. At the point when all ages are done, the model loads with the best approval misfortune are stacked into the VGG19_model, which will be utilized later for expectations.

```
from keras.callbacks import ModelCheckpoint # Train the model checkpointer = ModelCheckpoint(filepath='saved_models/weights.best.VGG19.hdf5', verbose=1, save_best_only=True) VGG19_model.fit(train_VGG19, train_targets, validation_data=(valid_VGG19, valid_targets), epochs=20, batch_size=30,
```

```
callbacks=[checkpointer], verbose=1)                    # Load the model weights with the best validation loss
VGG19_model.load_weights('saved_models/weights.best.VGG19.hdf5')
```

Predictions

At last, it is prepared to make forecasts. The VGG19_predict_breed work accepts a picture way as info, and returns the anticipated pooch breeds. The dog_breed_pred work is based on the past one, and profits anticipated outcomes depending for whether a pooch or a human is recognized in the info picture.

```
 from extract_bottleneck_features import *                    def
VGG19_predict_breed(img_path):                    # extract bottleneck features
bottleneck_feature = extract_VGG19(path_to_tensor(img_path))                    # obtain
predicted vector                    predicted_vector = VGG19_model.predict(bottleneck_feature)
# return dog breed that is predicted by the model                    return
dog_names[np.argmax(predicted_vector)]                    def dog_breed_pred(path):
# Detect dog or human and run prediction                    if dog_detector(path):
dog_breed = VGG19_predict_breed(path)                    result = 'This dog looks like a ' +
dog_breed + '.'                    elif face_detector(path):                    resemb_dog_breed
= VGG19_predict_breed(path)                    result = 'The most resembling dog breed of
this person is ' + resemb_dog_breed + '.'                    else:                    result = 'There
is no human or dog detected in this picture.'                    return result
```

Results

The exactness of the last model on test dataset is about **83%**, which isn’t terrible. Initially, I prepared a CNN model without any preparation without utilizing Transfer Learning. At that point, I made a CNN model utilizing move learning and VGG-19 with just one completely associated layer, and had the option to arrive at a precision of about 49%. At long last, I included a second completely associated layer to the classifier, and had the option to accomplish **83%** exactness.

The benchmark set was 80% of the accuracy and we got 83% so the benchmark is met.

Some of the predictions from the model are:

Conclusion

Because of the exchange learning strategy, I had the option to prepare a model with generally little dataset, and accomplished really great exactness. Moreover, the model was prepared inside a brief timeframe,

which is very proficient. The fundamental explanation is we can reuse the loads prepared by AI specialists utilizing a great many pictures.

There are a couple of potential enhancements for the model. To begin with, the parameters of completely associated layers, for example, number of layers, number of hubs, dropout rates, may be changed to show signs of improvement results. Second, utilizing an alternate analyzer or assessment metric may likewise improve model execution. Third, information expansion could likewise improve the last model exactness, as it will produce all the more preparing information.

Github Repo : https://github.com/avkolte/Dog-Classifier-Project