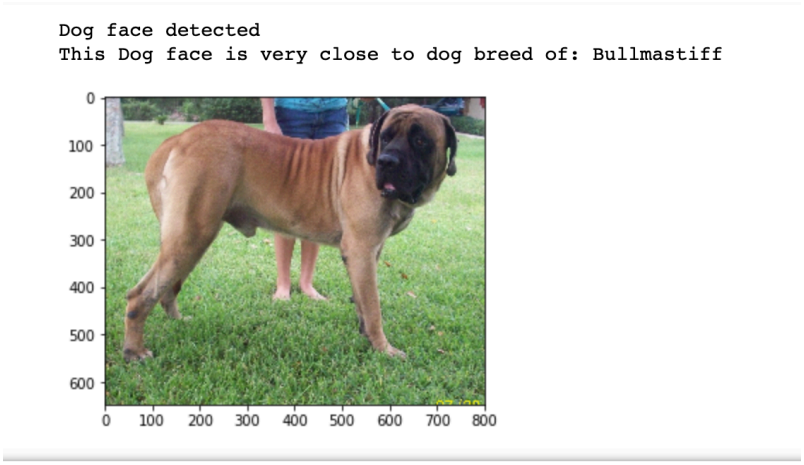
ASHOK MADISHETTI

Dog Breed Classifier using CNN

Prepared for: Udacity Machine Learning Engineer Nanodegree

Project Overview

The dog breed classifier is a machine learning model that has been built using one of the popular convolutional neural network (CNN) architectures including VGG16, ResNet etc. This will help to classify an image and identify a dog breed which is very close. If a dog is detected in the image, it will say ‘dog face is detected’ and provides predicted dog's breed name. If a human is detected, it will say ‘human face is detected’ and provides predicted dog breed name that is most resembling. The image below displays potential sample output of this project.



ASHOK MADISHETTI

Problem Statement

The aim of this project is to write an algorithm and create a model that could be used as part of a web application which can able to accept an image as input, process image by using model that has been built in this project and then provide detected face either human or dog and dog breed name which is very close to it.

The tasks involved are the following:

1. **Import datasets**: These are images of dogs and humans provided by Udacity
2. **Detect human face**: Algorithm classifies the image and predicts the dog breed name that is very close.
3. **Detect dog face**: Algorithm classifies the mage and then provides dog breed name as output
4. **Create a CNN architecture from scratch**: Create a CNN from scratch without using transfer learning.
5. **Create a CNN architecture using transfer learning:** Using one of the popular CNN imageNet architectures.

Metrics

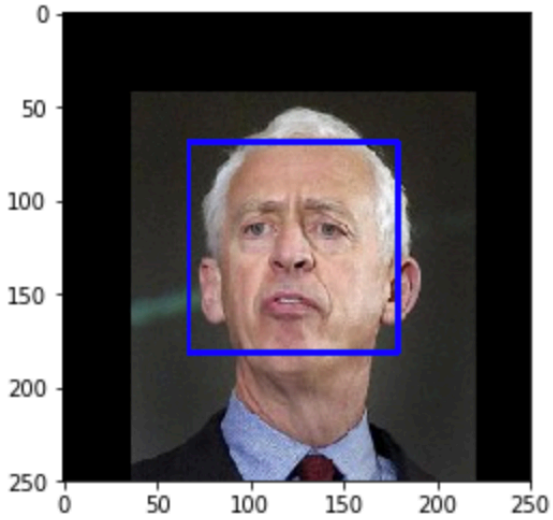
Using given datasets, created 3 sets of datasets for training, testing and validation. The training dataset has been used to train the model that has been built using a CNN architecture. The validation dataset has been used to evaluate the model and calculate the valid loss. This data was used to tune the model with number of epochs. Finally, the testing dataset of dog images was used to calculate and print the test loss and accuracy. Ensure that your test accuracy is greater than the expected value.

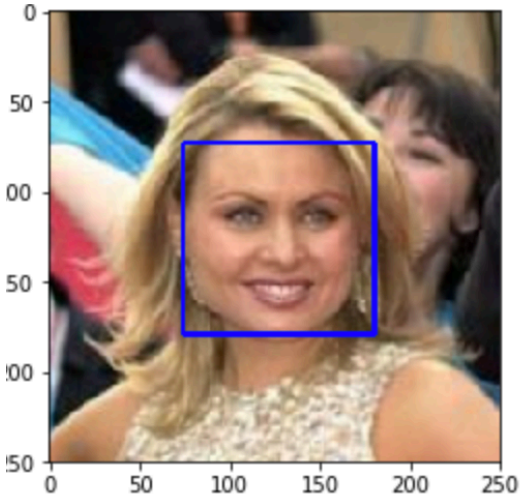
Accuracy = correctly classified items/ All classified items

Data Exploration

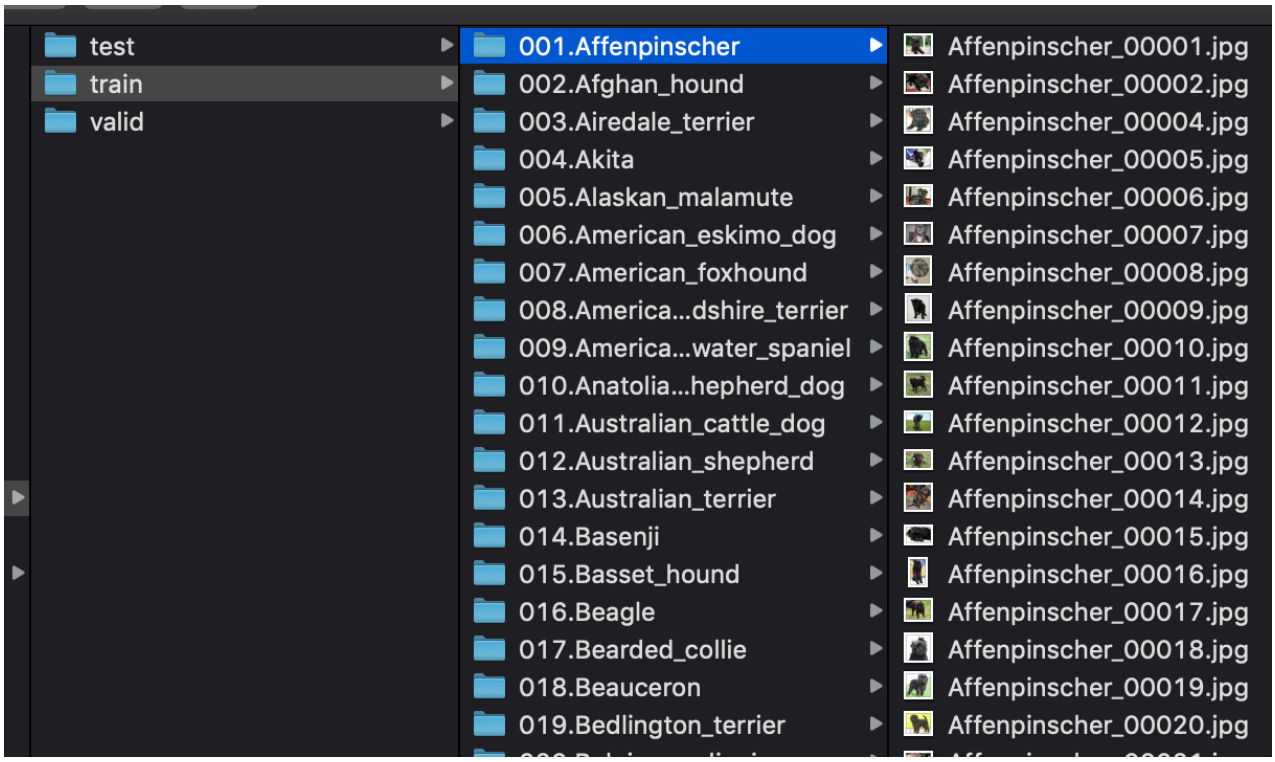
In this solution, the datasets are images of humans and dogs provided by Udacity.

Human dataset:
It has total of 13233 human face images of 250x250 size with various backgrounds, shapes and expressions. These files have been sorted by names. Here are sample human face images:





Dog dataset:
Dog images have been categorised into 3 types of datasets test, train and valid. Each dataset type has 133 folders with breed names in alphabetical order such as below:



test images: 836
train images: 6680
valid images: 835

Altogether, there are 8351 dog images that will be used in this project.

Following are few American eskimo dog face images from the train dataset:

Below one is from Beagle set:

By looking at above images, it is clear that images are not equal and have various backgrounds and different face styles. Also noted that image count for each breed is not same and it differs in numbers.

Algorithms and techniques

One of the Convolutional Neural Networks (CNN) model, VGG16, has been used to implement an algorithm which is an ideal for analysing images and predict the dog breed. VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”.

In the solution, at step 1, detects humans, OpenCV’s implementation of Haar feature-based cascade classifiers has been used to detect human faces in images.

At step 2, detect dogs, a pre-trained VGG16 model has been used to detect dogs in images.

At last step, once image has been detected as human or dog, it has been provided to CNN model that processes given image and predicts an output which is a dog breed name.

Benchmark

The CNN model created from scratch must attain a test accuracy of at least 10%. This will prove that newly created model is working as expected as a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

Data Preprocessing

Transformed all the images (test, train and valid) size close to the standard crop size 224 so that cropped image can contain most of the features. For training dataset, RandomHorizontalFlip has been applied to flip the image randomly with default probability of 0.5 and then RandomRotation has been applied to rotate the image up to 10 degrees angle.Then used transforms.ToTensor() to convert all the images to tensor before passing to model.

Implementation

I have followed the VGG16 configuration provided in https://neurohive.io/en/popular-networks/vgg16/ to create CNN model from scratch to implement the solution.

I have created 3 convolutional layers starting from 3, 36 in the first layer and then increased by a factor of 2 after each max-pooling layer, until it reaches 128.

The convolution stride is fixed to 1 pixel in each layer and Max-pooling is performed over a 2×2 pixel window, with stride 2.

Then created 2 Fully-Connected (FC) layers follow a stack of convolutional layers and the final layer produces 133 size output feature.

Refinement

As mentioned in the Benchmark section, the CNN model created from scratch achieved a good test accuracy 14 % (122/836) which is more than the expected minimum accuracy.

To improve the accuracy, I have used transfer learning to create a CNN that can identify dog breed from images and must attain at least 60% accuracy on the test set.

I have chosen the VGG16 image network that was used for dog detector which is pre-trained on ImageNet dataset. Later created a classifier using a sequential container. Then 3 Linear modules have been added to it along with 2 rectified linear unit functions and 2 dropouts to reduce the parameters. Sequential will make sure that added models will be executed in the order they are passed in the constructor.

After using this model architecture, I have received test accuracy 61% which is above the required accuracy.

Model Evaluation and Validation

Detect Humans: The human face detecter was created by using OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.

Out of 100 human files, 98% of human faces were detected.
Out of 100 dog files, 17% of human faces were detected.

Detect Dogs: The dog face detecter was created by using pr-trained VGG16 model.

Out of 100 dog files, 100% of dog faces were detected.
Out of 100 human files, 0% of dog faces were detected.

CNN Model from scratch: The CNN model created from scratch achieved a good test accuracy of 14 % (122/836).

CNN Model using transfer learning: The CNN model created using transfer learning with VGG16 architecture has achieved test accuracy of 61% (518/836).

Justification

The CNN model created using transfer learning has worked very well and achieved accuracy of 61% compared to the CNN model created from scratch which is 14%.

Improvement

The mode has achieved just above the expected minimum accuracy and can be improved more by using a different architecture such as ResNet 101. Training of model can be improved with more epochs execution