Capstone Project — Achintya Sen

Machine Learning Nanodegree — November 1st, 2020

Definition

Project Overview

The dog breed classifier project is well known in the Computer Vision domain. The first competition for this was first raised in Kaggle in 2017 (Kaggle, n.d.), where the primary objective was to distinguish the breed of a dog given an input image. As the primary objective of this project, I would be building an interface through which an image can be pushed into a data pipeline to pass into a Convolutional Neural Network (CNN) which will output the breed of the input image. There are many types of breeds against the dataset, which makes it a multi-class classification problem.

A fun aspect of this project is to input images of humans and see the breed against which they match the most. The interface would take as input any image a run pre-trained model to classify a human for breed classification based on the model.

Problem Statement

The goals of the problem are as follows:

- Build an initial classifier for identifying a human or a dog. For this step, we will be using a pre-trained model using the OpenCV python package.
- Build the Convolutional Neural Network (CNN) defining the hyperparameter and train the model with training and the validation data. Evaluate the model using the testing data based on the metrics defined in the project
- Improve the metrics by experimenting with pre-built models using transfer learning. Train on training and validation data.
- Determine the best model among them to build a pipeline which inputs an image and performs the following function:
  - **Human Image:** Process the image and pass through the model to determine which breed resembles the human image
  - **Dog Image:** Process the image and determines the breed of the dog image.

Metrics

For evaluating the model, we will be determining the accuracy scores. The accuracy can be calculated using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

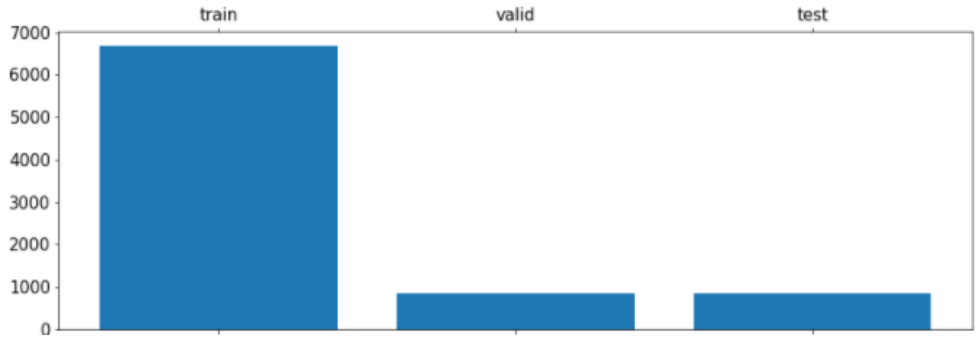
where: TP = True positive; FP = False positive; TN = True negative; FN = False negative

Analysis

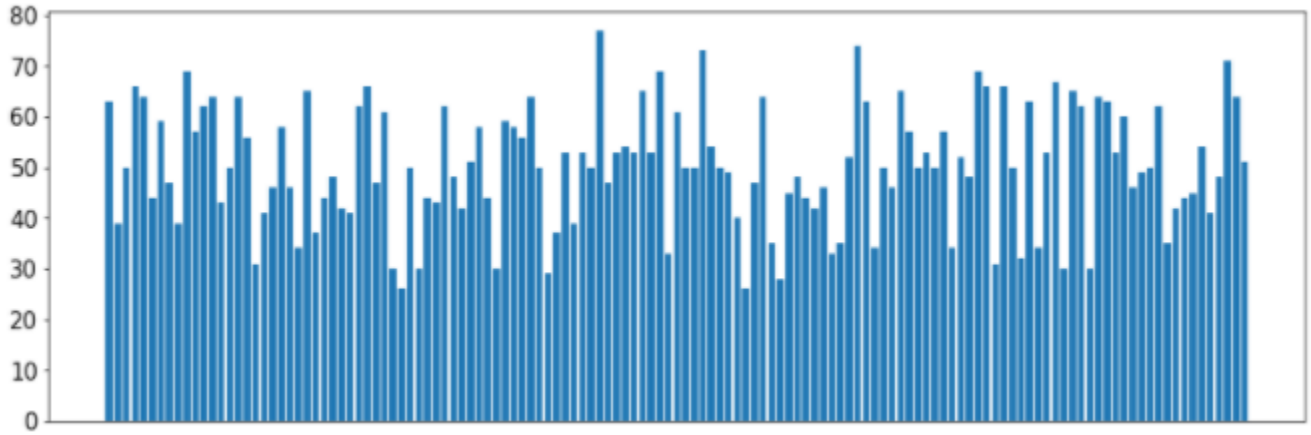
Data Exploration

Dog Images

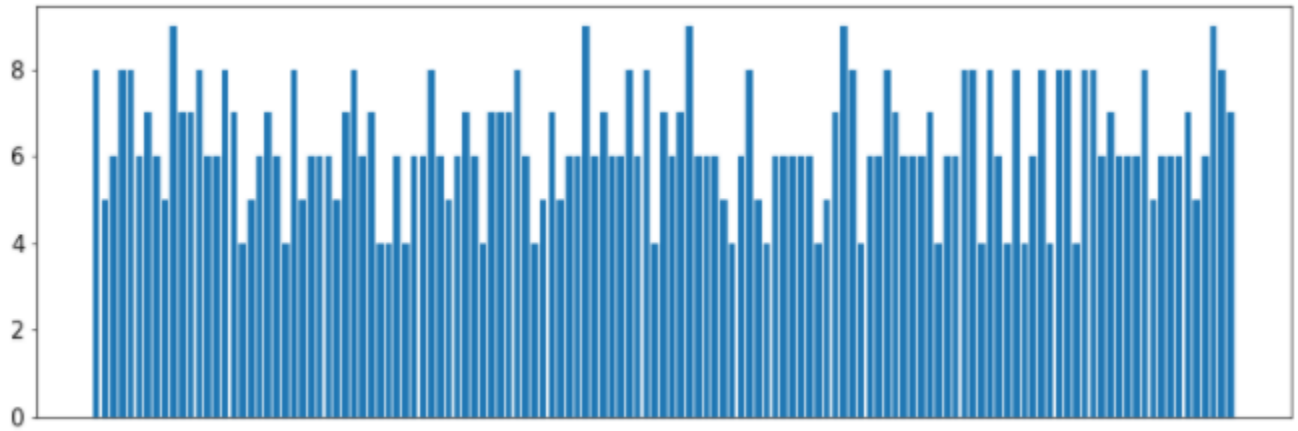
The dataset used in solving this problem is a set of images which consist of dogs with respective dog's breed in directories. This set is provided by the Udacity Machine Learning Nanodegree program and has the same structure as the training and validation set. The test dataset is a mix of dog breeds against which we will be testing the CNN model. There is a total of 133 breeds of dogs and a total of 8351 images of dogs. The number of images in each breed is not uniform which might cause an issue with the learning of the features of some of the breeds. Additionally, the image size is also not uniform among the dog images, hence will require cropping or resizing.



Distribution in Train, Valid, and Test



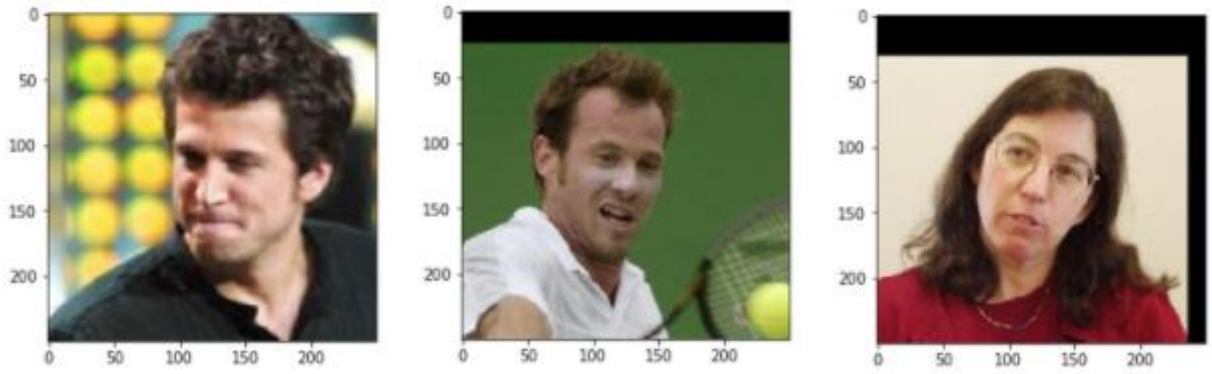
Distribution in dog breeds count in the training dataset



Distribution in dog breeds count in valid dataset

Human Images

Apart from the list of the dog images, we have a set of human images that will be used in determining the second classifier which will derive the dog breed resemble the input human image. There is a total of 13233 human images which are stored in 5750 directories against their name. In this case, as well the data is not uniform hence few directories have as few as 1 image.



Sample Human Images

Algorithms and Techniques

The solution defined for the problem is divided into 3 sections below:

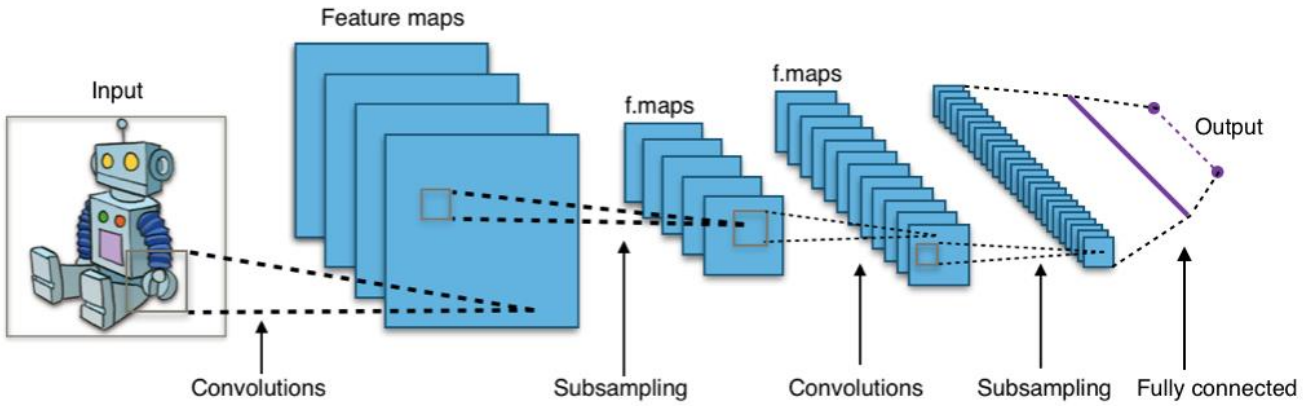
Human & Dog Detector

The first step of the process is to build a detector for determining a dog or human. For determining dogs, we will be using a pre-trained image classifier viz, Resnet50, VGG16. These classifiers have been built to classify many more classes than dog breeds. Each of the classifiers has different properties that can classify certain sets of dog

breeds. For determining humans, we will be using OpenCV's implementation of Haar feature-based cascade classifiers. *It is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.* (Open Source Computer Vision, n.d.).

Build Model

The second section of the problem is to build a model for classifying the dog breed. To identifying the features of the image we will be using the **Convolution Neural Network** (**CNN**, or **ConvNet**), which is a deep learning model. In recent years, there has been an increase in the popularity of the CNN image classification model in various domains of the industry. The model has an input layer, hidden layers, and a fully connected output layer. The hidden layer consists of many convolution layers that convolve the image to determine various patterns in the image. The fully connected layer will be connected to the classes of the dog breed. In this case, there will be 133 output nodes for the fully connected output layer.



(Wikipedia, n.d.)

Build Model using Transfer Learning

The third section is to improve on the model using the transfer learning technique which will improve the accuracy of the application. The transfer learning method is using to import the model architecture of the renowned pre-built image classifier namely, Resnet101, VGG-19, etc., which must be modified and processed on the available dataset. As these models have their properties, they must be modified to for out input images and output classes.

Benchmark

The benchmark for the Convolution Neural network model built from scratch should have an accuracy score above 10%. From the total set of 836 test images, the model should be able to accurately classify a minimum of 87 dog breeds. The second benchmark is using the transfer learning model architecture where we will be targeting 60% accuracy.

Methodology

Data Preprocessing

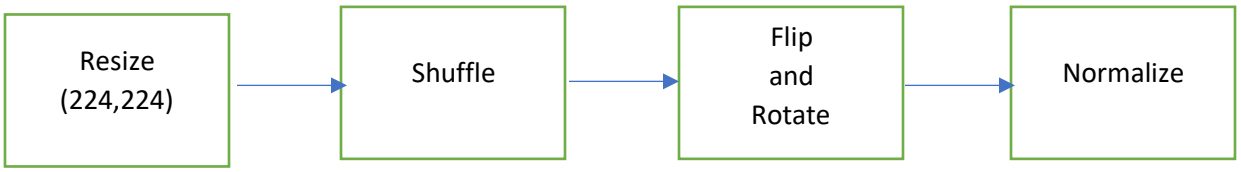
Data processing a crucial step in the problem due to the following factors:

- Imbalance in training image classes
- Variety in image size
- Variety in the positioning of the image feature (the body is not at center)
- Variety in color saturation in the image

For tackling the problems, we must experiment with multiple image processing transformations before the classification. Following are the steps I took for the same:

- The initial problem which needs to be tackled is the imbalance that can be handled with shuffling of the images. The *dataloader* class of the PyTorch has a built-in argument for adding shuffling in images for better learning.
- The second issue is the variation in image size which can be resolved using some form of resizing or cropping mechanism. As the position of the subject of an image cannot be determined, we will be proceeding to do center cropping of the image and resizing it to size 224X224. This resolution is chosen as most of the popular image classifiers like VGG 16 and VGG19 use this size for their learning.
- For tackling the third problem we must add rotation or flipping the image which will help in learning features of the image.
- The color saturation is an important aspect of the image as there are images taken with various contrast or brightness. Normalizing the depth of each pixel brings them in a uniform range, which will help the CNN model in better learning.

This leads to building an image processing pipeline as shown in the below diagram:



Implementation

The initial implementation is to build a face or dog detector for determining the type for the application.

- **Face Detector:** A method was built to input an image path that is read using the OpenCV native reader. The image is converted to grayscale and then processed using the face cascade classifier for determining a human image.
- **Dog Detector:** I have used the pre-trained VGG-16 image classifier for determining dogs. The VGG 16 model is trained on the ImageNet dataset which encompasses 120 dogs' breeds. There is a possibility that it would misclassify a few of the dog detection but should capture > 90% of the dog images.

In the next step in building the CNN model from scratch, I have used 5 convolution layers of kernel size 3, padding 1, and stride 1. The convolution layers are separated by a max-pooling layer of (2,2) size which will help in reducing the size of the image by a factor of 2. The input vector to the very first layer is the image vector of size (224,224) which is preprocess from the image processing pipeline. Each convolution layer is processed using the RelU activation function.

The output of the final convolution layer is an image dimension of 7X7X512, which fed into a linear neural network with 256 node output. That is followed by the batch normalization which normalizes the input vector, followed by a RelU activation function. After this stage, the final fully connected layer is added which output the 133-node output layer for the 133 dog classes. Below is the various component of the CNN network:

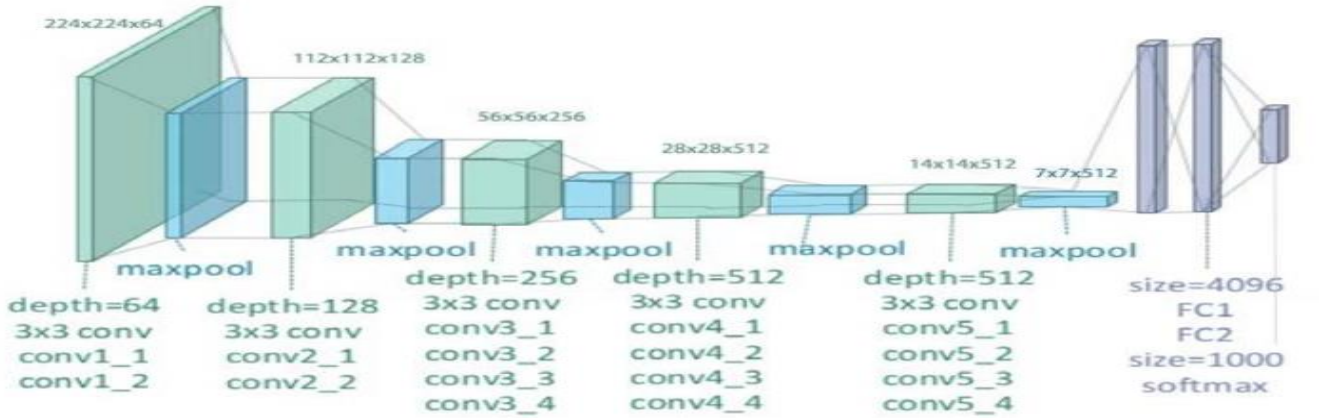
```
Net(
  (conv1): Conv2d(3, 36, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(36, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0,dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=25088, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=133, bias=True)
  (dropout): Dropout(p=0.25, inplace=False)
  (batch_norm): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_runnin
g_stats=True)
)
```

Refinement

For improving the model accuracy score, I have refined it using the transfer learning process. The transfer learning process in which a model which built for another Image set is imported in learning the feature of a different set of images. There are various types of image classification models built by researchers across the globe, namely Resnet models, VGG models, Inception model. They were initially built to train on the ImageNet dataset which is an image dataset with 1000 categories that can vary from any objects.

After thorough experimentation on available image classification models, I have decided to use the VGG19 pre-trained model for transfer learning. VGG-19 is a convolutional neural network that is 19 layers deep, which is pre-trained on the ImageNet database. The output of the model 4096 node layer, which is connected to a custom Linear fully connected layer with a 133-node output layer for the 133 dog breeds.

VGG-19 Architecture (ResearchGate, n.d.)



Results

Model Evaluation and Validation

Evaluation of the initial face and dog detector gave the following result:

- **Face Detector:** With the implementation of Haar feature-based cascade classifiers on a set of 100 input images for humans, the detector was able to classify 99% of the images positively.
- **Dog Detector:** With the implementation of **the** VGG-16 model on a set of 100 input images f dogs, the detector was able to classify 99% of the images positively.

With the initial implementation of the Convolutional Neural Network from scratch, I was able to achieve 34% accuracy using the test dataset. The model was trained for 50 epochs, which reduced the error on the training set to 2.1023. There would have been a slight improvement in the model after training for a longer duration, but the learning rate would have been slower. After 40 epoch cycles, I observed there was flat-lining in the learning with a minor reduction in the error.

Using transfer learning, I heavily experimented with the VGG model family. Below are stats on them:

- **VGG16**
  Test Loss: 1.021881
  Test Accuracy: 70% (586/836)
- **VGG19**
  Test Loss: 1.100511
  Test Accuracy: 72% (605/836)

I chose to use the VGG19 model which was able to give a score of 72% after 5 epoch cycles.

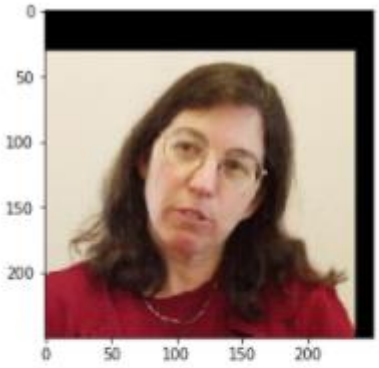
Justification

The detection method for determining dog or human reach a high benchmark with 99% accuracy, along with an error of 9% only in case of human detection.

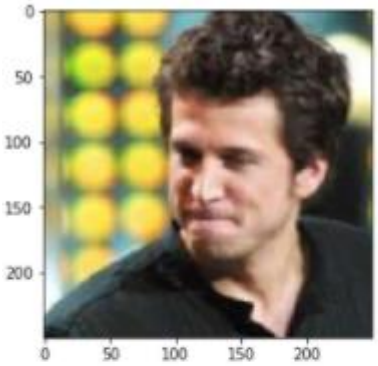
The intermediary model is built from scratch and the final model reaches the benchmark initially set. The intermediary model has performed well with 34% accuracy, where 10% was set as the initial goal. The final model using transfer learning has also performed well with 72% accuracy where the initial benchmark was set at 60%. Using the transfer learning model, the application can be built to give an accurate prediction of the dog breed.

There can further improvement that can be researched using the Resnet model family which is a newer model and can give higher accuracy with the dataset. The primary factor which led to higher accuracy in the scratch model is the data processing pipeline which can be further improved by adding better image extraction.

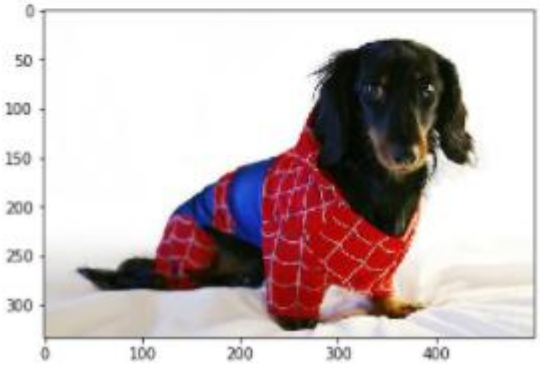
Outcome



Human Predicted!
You look like a English toy spaniel :)



Human Predicted!
You look like a Dachshund :)



Dog Predicted !
Predicted breed: Cairn terrier

Dog Predicted !
Predicted breed: Cairn terrier

References

Kaggle. (n.d.). *Kaggle*. Retrieved from kaggle.com: https://www.kaggle.com/c/dog-breed-identification

Open Source Computer Vision. (n.d.). Retrieved from https://docs.opencv.org/: https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

ResearchGate. (n.d.). Retrieved from researchgate.net: https://www.researchgate.net/figure/llustration-of-the-network-architecture-of-VGG-19-model-conv-means-convolution-FC-means_fig2_325137356

Wikipedia. (n.d.). Retrieved from wikipedia.com: https://en.wikipedia.org/wiki/Accuracy_and_precision

Wikipedia. (n.d.). Retrieved from https://en.wikipedia.org/: https://en.wikipedia.org/wiki/Convolutional_neural_network