Udacity Machine Learning Engineer Nanodegree

Capstone Report

Classifying Dog Breeds

Rapolu Pavan Kumar

September 2020

1.Definition

1.1 Project Overview

The visual sense is one of the five core senses humans have. It has always been a challenge to computer programmers to teach computers the way humans perceive the world through their senses. Image recognition is a relatively easy task for humans and yet a very difficult task for computers to perform.

To create a better algorithm for image recognition, in 2010, ImageNet Large Scale Visual Recognition Challenge proposed an annual competition for research teams to evaluate their algorithms on a large dataset with 1000 categories and over 1 million images [1]. In the 2010s, the error rates of misclassifying images fell drastically, in large part due to the introduction of deep neural network optimized for spatial variations, called convolutional neural networks [2]. The ImageNet competition is still held every year.

By now, convolutional neural networks (CNN) are widely accepted for image recognition and image classification tasks. Image recognition is also a thriving research field with applications limited by our imagination: everything humans see, an algorithm could potentially be trained to see.

There is a plethora of real-world application for this research, such as:

• Car lane recognition for enabling self-driving cars

• Handwriting recognition for archiving documents faster

• Action recognition for supporting industrial line workers

1.2 Problem Statement:

As were stated in the Udacity page the problem will be " to learn how to build a pipeline to process real-world, user-supplied images. Given an image of a dog, your algorithm will identify an estimate of the canine's breed. If supplied an image of a human, the code will identify the resembling dog breed."

1.3 Metrics:

The data is split into train, test and valid dataset. The model is trained using the train dataset. We use the testing data to predict the performance of the model on unseen data. We will use accuracy as a metric to evaluate our model on test data.

$$\boldsymbol{Accuracy} = \ \#\ \boldsymbol{of\ correct\ classifications}\ /\ \#\ \boldsymbol{of\ total\ classifications}$$

Also, during model training, we compare the test data prediction with validation dataset and calculate Multi class log loss to find the best performing model. Log loss takes into the account of uncertainty of prediction based on how much it varies from actual label and this will help in evaluating the model.

2.Analysis

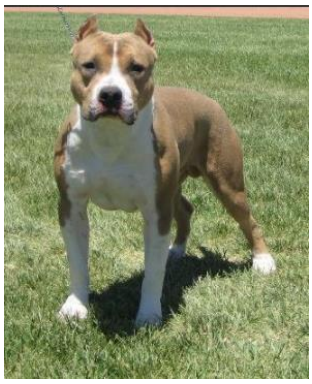
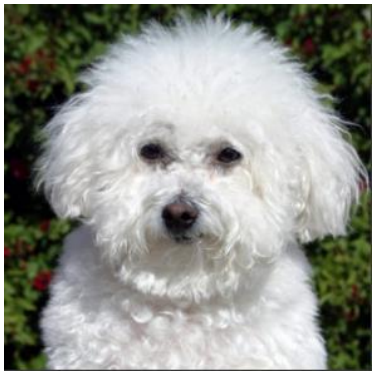
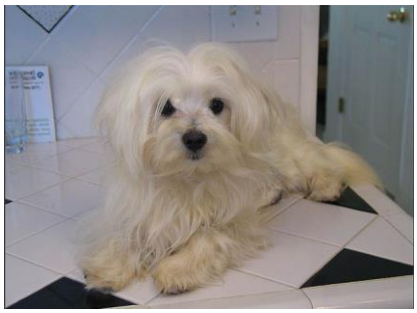
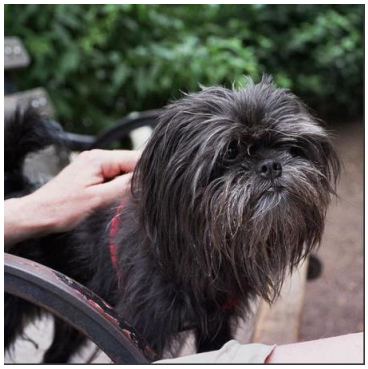
2.1 Data Exploration

For this project, the input format must be of image type, because we want to input an image and identify the breed of the dog. The dataset has pictures of dogs and humans.

Dog images dataset: The dog image dataset can be downloaded here https://s3-uswest1.amazonaws.com/udacity-aind/dog-project/dogImages.zip

Dog images dataset summary: The dog image dataset has 8351 total images which are sorted into train (6,680 Images), test (836 Images) and valid (835 Images) directories. Each of this directory (train, test, valid) have 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds, some images are not full-sized. The data is not balanced because the number of images provided for each breed varies. Few have 4 images while some have 8 images.

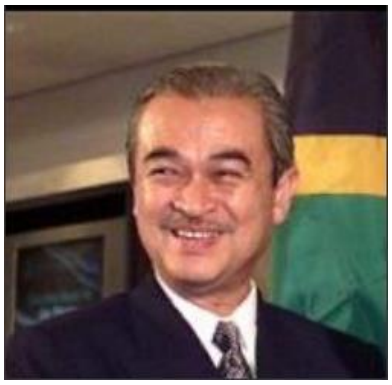
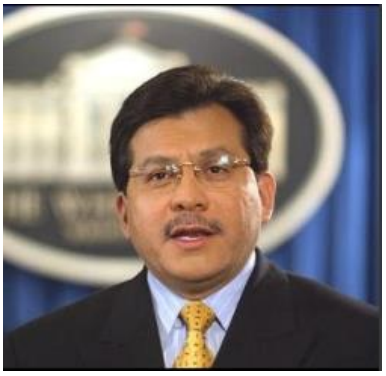
A set of example images from the dog dataset:

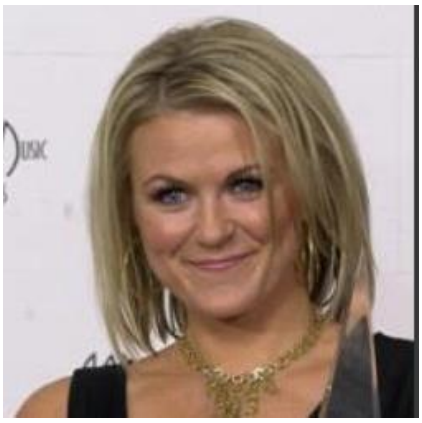






Human images dataset: The human dataset can be downloaded here https://s3-uswest1.amazonaws.com/udacity-aind/dog-project/lfw.zip

Human images dataset: The human dataset contains 13233 total human images which are sorted by names of human (5750 folders). All images are of size 250x250. Images have different background and different angles. The data is not balanced because we have 1 image for some people and many images for some.

A set of example images from the human dataset:

2.2 Algorithms and techniques:

For performing this multiclass classification, we can use Convolutional Neural Network to solve the problem. A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The solution involves three steps. First, to detect human images, we can use existing algorithm like OpenCV's implementation of Haar feature based cascade classifiers. Second, to detect dog-images we will use a pretrained VGG16 model. Finally, after the image is identified as dog/human, we can pass this image to an CNN model which will process the image and predict the breed that matches the best out of 133 breeds.

2.3 Benchmark

The CNN model created from scratch must have accuracy of at least 10%. This can confirm that the model is working because a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

3.Methodology

3.1 Data Pre-processing

All the images are resized to 224*224, then normalization is applied to all images (train, valid and test datasets). For the training data, Image augmentation is done to reduce overfitting. The train data images are randomly rotated and random horizontal flip is applied. Finally, all the images are converted into tensor before passing into the model.

3.2 Data Augmentation

We have also applied several transformations to augment the data in the train set for a more robust algorithm. We have used:

- A random vertical flip with a 30% likelihood.
- A random horizontal flip with a 30% likelihood.
- A random rotation by 30 degrees.

3.3 Implementation

I have built a CNN model from scratch to solve the problem. The model has 3 convolutional layers. All convolutional layers have kernel size of 3 and stride 1. The first conv layer (conv1) takes the 224*224 input image and the final conv layer (conv3) produces an output size of 128. ReLU activation function is used here. The pooling layer of (2,2) is used which will reduce the input size by 2. We have two fully connected layers that finally produces 133-dimensional output. A dropout of 0.25 is added to avoid over overfitting.

3.4 Refinement

The CNN created from scratch have accuracy of 15%, Though it meets the benchmarking, the model can be significantly improved by using transfer learning. To create CNN with transfer learning, I have selected the Resnet101 architecture which is pre-trained on ImageNet dataset, the architecture is 101 layers deep. The last convolutional output of Resnet101 is fed as input to our model. We only need to add a fully connected layer to produce 133-dimensional output (one for each dog category). The model performed extremely well when compared to CNN from scratch. With just 10 epochs, the model got 83% accuracy.

4. Results

4.1 Model Evaluation and Validation

Human Face detector: The human face detector function was created using OpenCV's implementation of Haar feature based cascade classifiers. 98% of human faces were detected in first 100 images of human face dataset and 17% of human faces detected in first 100 images of dog dataset.

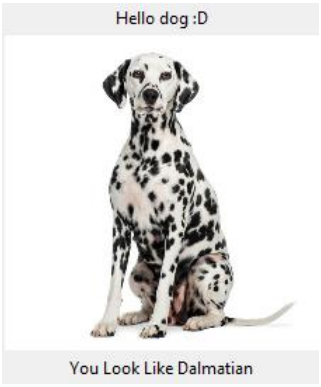
Dog Face detector: The dog detector function was created using pre-trained VGG16 model. 100% of dog faces were detected in first 100 images of dog dataset and 2% of dog faces detected in first 100 images of human dataset.

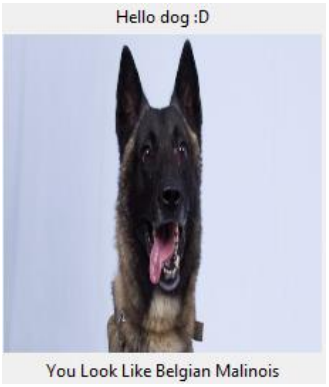
CNN using transfer learning: The CNN model created using transfer learning with ResNet101 architecture was trained for 10 epochs, and the final model produced an accuracy of 83% on test data. The model correctly predicted breeds for 699 images out of 836 total images.

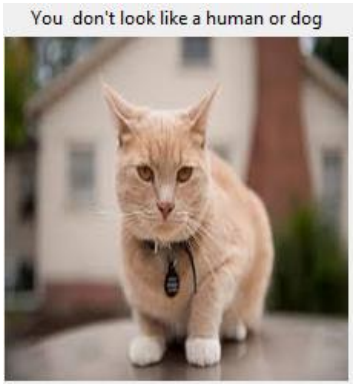
```
Accuracy on test data: 83% (699/836)
```

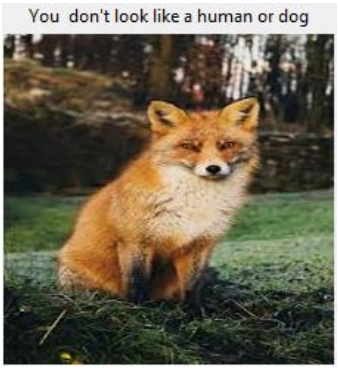
After training both the basic CNN and transferred ResNet101 model on the dog dataset, it's time to test if the classifier also performs well with new data. We run the classifier on new images and perform one of the following three tasks:

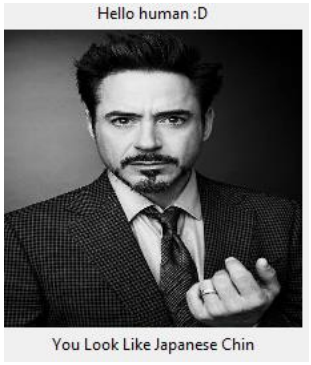
- If the image contains a dog, print the classified dog breed.
- If the image contains a human, print the lookalike dog breed.
- If the image contains neither, print the lookalike dog breed.

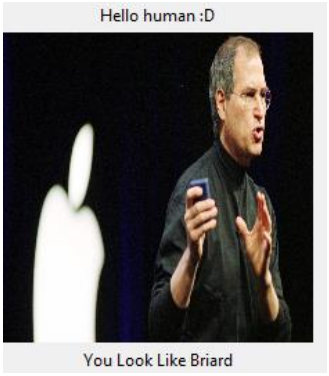












The model performs well on the new set of images of dogs, humans and others. The dog breed labels are accurate. The labels of humans are interesting but accurate and going through the training images of these classes one can see where the features come from. For the image of the robot it's completely arbitrary in my opinion. I don't see the features to expect a bull terrier, expect perhaps the slim torso and thick head.

4.2 Justification

I think the model performance is better than expected. The model created using transfer learning have an accuracy of 83% compared to the CNN model created from scratch which had only 15% accuracy.

5. Conclusion

In this project, we have built an algorithm to detect and classify dogs in images according to their breed. First, we tried a basic CNN architecture, which performed poorly with 15% classification accuracy. Second, we tried a more complex CNN architecture based on a pretrained ResNet101 model with a custom final fully-connected layer. This model performed extremely well with a classification accuracy of 83% on our test set of 836 dog images. The result of the transfer learning is much better than I had expected and probably much better than a human (without a trained eye) could classify dog breeds.

Personally, I have not even heard of some of the dog breeds before, so having an algorithm tell me the type of dog is magical.

5.1 Reflection

The process used for this project can be summarized with the following steps:

1. The initial problem was defined, and an appropriate dataset was obtained.

2. The data was explored and analyzed.

3. The data was preprocessed, and features were extracted.

4. A basic model was trained and evaluated.

5. A more advanced model was trained and evaluated.

5.2 Improvements

If we were to continue with this project, there are a number of additional areas that could be explored to improve the algorithm:

- Increase the dataset's size – currently, we are training on only 6680 images of dogs.
- Include more dog breed classes as well as mixed dog breeds.
- Tune hyperparameters further and more carefully.
- Experiment with other transfer model architectures, such as Inception V3 or Xception.

6. References:

1. Original repo for Project - GitHub: https://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification/dog_app.ipynb

2. Resnet101: https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html#resnet101

3. Imagenet training in Pytorch: https://github.com/pytorch/examples/tree/master/imagenet

4. Pytorch Documentation: https://pytorch.org/docs/master/

5. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neuralnetworks-the-eli5-way-3bd2b1164a53