Udacity Capstone Project Report
Jill Shah

Dog Breed Classifier using CNN

1. Definition

Project Overview:

Image classifier is one of the hot topics in the Machine Learning field. There are potentially n- number of categories in which a given image can be classified. Manually checking and classifying images is a very tedious process. The task becomes near impossible when we're faced with a massive number of images, say 10,000 or even 100,000. How useful would it be if we could automate this entire process and quickly label images per their corresponding class?
Self-driving cars are a great example to understand where image classification is used in the real-world. To enable autonomous driving, we can build an image classification model that recognizes various objects, such as vehicles, people, moving objects, etc. on the road.

Image classification is a supervised learning problem: define a set of target classes (objects to identify in images), and train a model to recognize them using labeled example photos. Early computer vision models relied on raw pixel data as the input to the model. Raw pixel data alone doesn't provide a sufficiently stable representation to encompass the myriad variations of an object as captured in an image. The position of the object, background behind the object, ambient lighting, camera angle, and camera focus all can produce fluctuation in raw pixel data; these differences are significant enough that they cannot be corrected for by taking weighted averages of pixel RGB values.

One of the most popular Udacity projects across machine learning and artificial intelligence nano degree programs. The goal is to classify images of dogs according to their breed. If the Human face is given as input it will suggest dog breed.

Python libraries like OpenCV and Tensorflow are used for all the processing. Opencv used for face detection and Tensorflow used for model building and improvement of the model.

Problem Statement:

The goal of the project is to build a machine learning model that can be used within web app to process real-world, user-supplied images. The algorithm has to perform two tasks:

Dog face detector: Given an image of a dog, the algorithm will identify an estimate of the canine's breed.

Human face detector: If supplied an image of a human, the code will identify the resembling dog breed.

We will try to solve this problem using 2 methodologies. First we will use CNN building it from scratch. That will serve as our benchmark model to beat. Then we will use transfer learning to improve our model.

Metrics:

The goal here is to compare the performance of my model with that of the benchmark model. Therefore, I would use accuracy as an evaluation metric. Also because the benchmark model only specifies the accuracy. After exploring the data, it has been noticed that there is some imbalance in the data set. Few dog breeds have more sample images than others. To counterbalance the imbalance in datasets one should use other metrics like precision, recall and F1-score to evaluate the model properly. However the imbalance is not that prominent and we will also be using transfer learning which has been trained on large number of varied images. Hence I have just accuracy here which will provide a decent metric.

2. Analysis

Data Exploration:

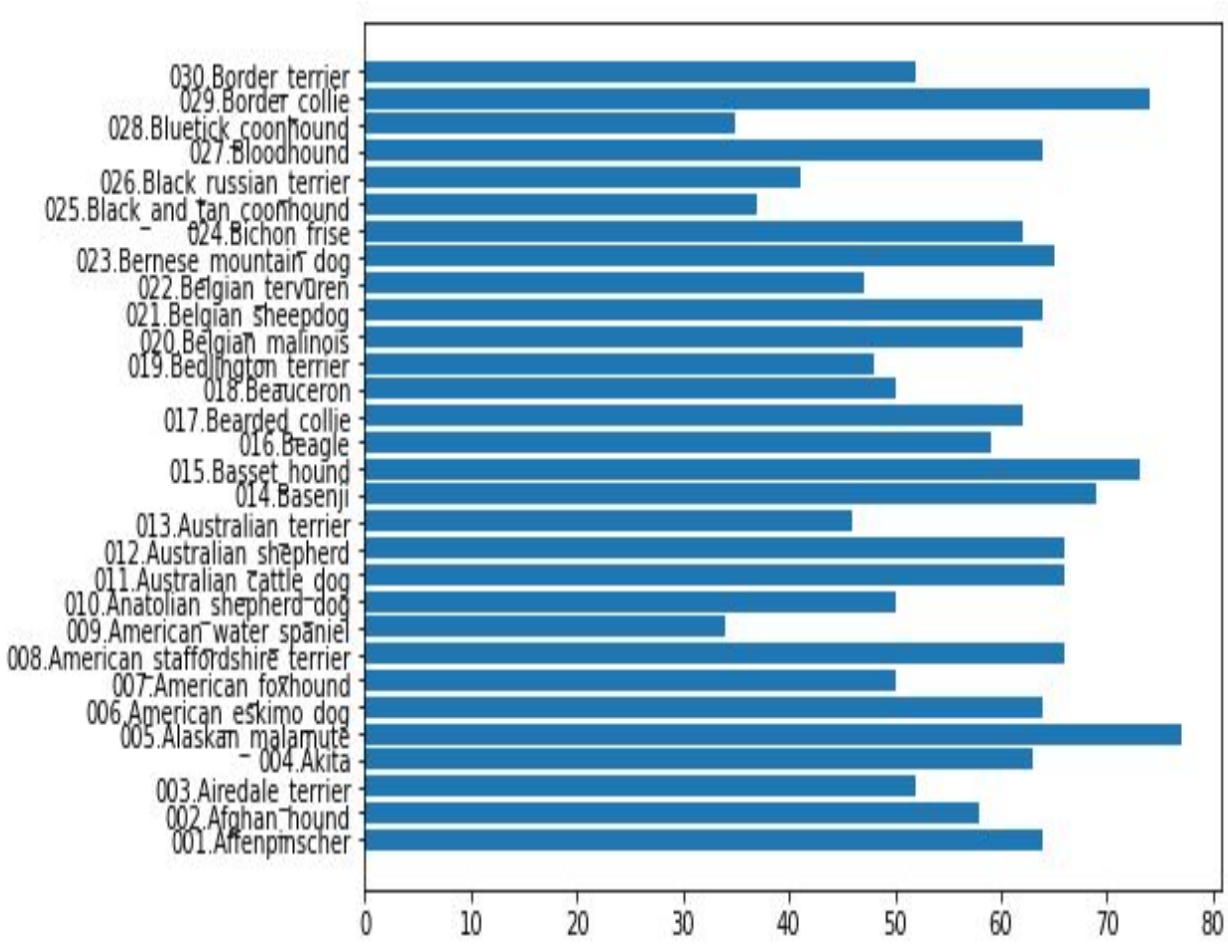
Input Format : Image type

Dataset information: We need two types of images to test and train the model.

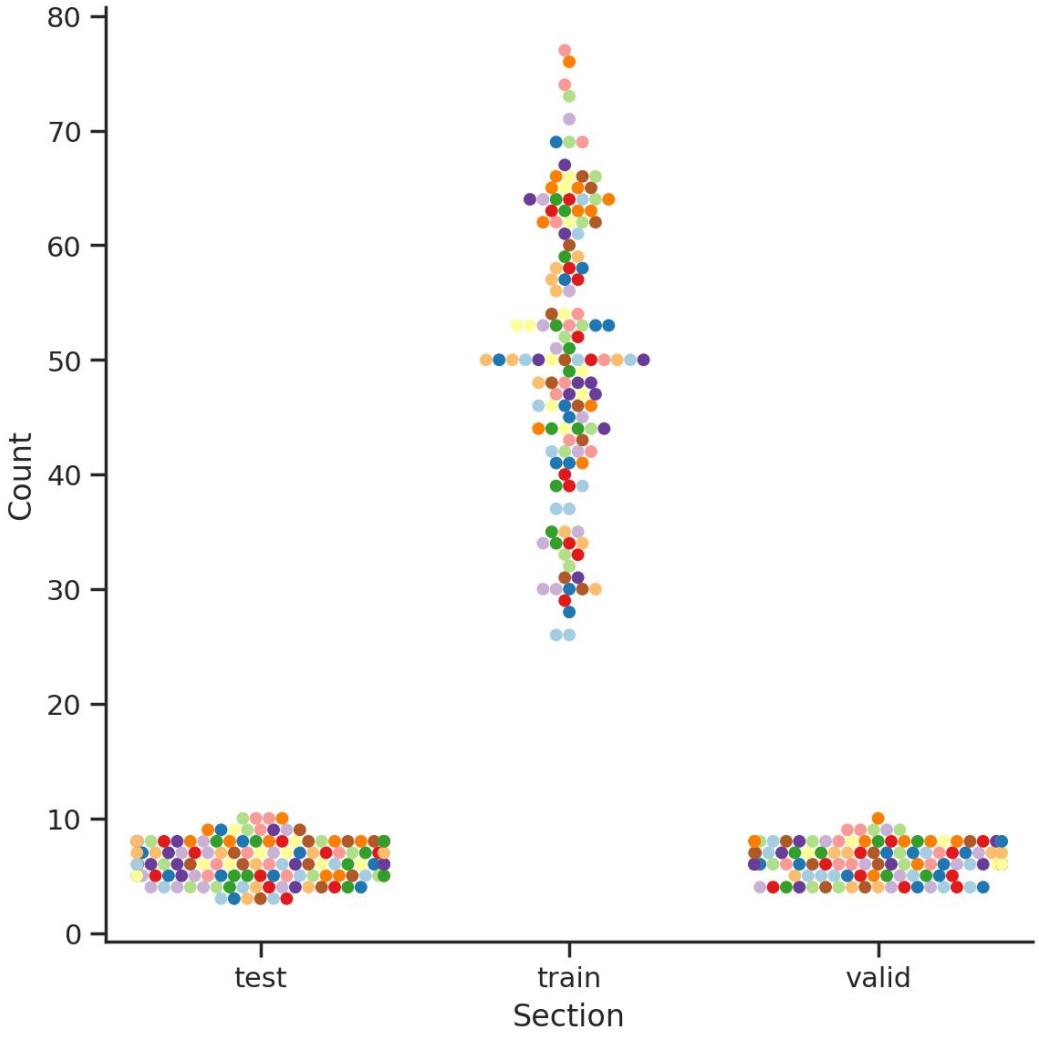
Dog Dataset: The dogImages folder contains 3 folders: test, train and valid. Each folder here contains 133 subfolders for dog breeds we need to classify for. There are a total of 8351 images. The data is not balanced because the number of images provided for each breed varies. Few have 4 images while some have 8 images.

Sample Images from dog dataset:

Plot of number of images in first 30 folders:



Below is the distribution of count of dog images of each breed across train, test and validation datasets:



Human Dataset: Human pictures are sorted by name of each human. The human dataset contains 13233 total human images which are sorted by names of humans (5750 folders). All images are of size 250x250. Images have different backgrounds and different angles. The data is not balanced because we have 1 image for some people and more than one for some.

Sample Images from the Human Images Folder:


Algorithms and Techniques

For image classification problem that we have at hand, we will be using CNN. CNN is a class of deep neural networks. CNN has been commonly used in image classification and it finds its application in major problems like photo tagging in Facebook and self-driven cars. CNN is quite fast and efficient. It takes in an input image, analyses it's features and then classifies an image into categories based on probability.

Human Face Detector: Here I will use Cascade Classifiers(HaarCascade) from OpenCV to recognize the face of a person on an image. Haar Cascade classifier is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses "integral image" concepts to compute the "features" detected. Haar Cascades uses the Ada-boost learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers then use cascading techniques to detect the face in an image. Haar cascade classifier is based on the Viola-Jones detection algorithm which is trained in given some input faces and non-faces and training a classifier that identifies a face.

If a face is recognized this means that a person is present in the picture.

Dog detector: To recognize dogs in the picture, I used pre-trained VGG-16 model. VGG-16 is a convolutional neural network that is 16 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

Dog Breed Classifier from Scratch: Here I will use PyTorch to design, train and test a model from scratch. I am building this model and running it on my PC which has severe

hardware limitations. I have simplified the architecture so as to not overheat my PC. My model is as below:

- Convolutional layer with 3 input and 32 output dimensions, kernel size 3.
- Batch Normalization with size 32
- Relu Activation Function
- Pooling layer
- Convolutional layer with 32 input and 64 output dimensions, kernel size 3.
- Relu Activation Function
- Pooling layer
- Convolutional layer with 32 input and 64 output dimensions, kernel size 3.
- Relu Activation Function
- Pooling layer
- Flatten Image to 1d vector
- Fully Connected Linear Layer
- Relu Activation Function
- Fully connected linear layer to output number of classes

Dog Breed Classifier with transfer learning: In this section I will apply the technique transfer learning. In transfer learning, a pretrained model is used which is then only modified according to the intended use. In the PyTorch framework I used ResNet101. In pre-trained models, we only need to train the classification section.

Benchmark

CNN Model from Scratch: As a benchmark model we will use a CNN model created from scratch which should have an accuracy greater than 10 percent. It is still better off than a random chance which has a probability of less than 1%.

Transfer Learning using PyTorch: For the transfer learning model, my benchmark to beat is 60 percent as provided in the assignment notebook.

3. Methodology

Data Pre-Processing:

For the data pre-processing, we have used PyTorch's transform method. The images in the training dataset do not have the same resolution. Also there is a slight imbalance in the number of images available across various dog breeds. To counterbalance this we are randomly, rotating, flipping and cropping the image. I am also resizing the image to 224*224 pixels. This helps the algorithm train for images which are no in the standard format.

Implementation:

Below are the steps I took to build this project:

1. Download Udacity images for this project into lfw and dog images folder
2. Checked for imbalance in datasets.
3. Used OpenCV's HaarCascade classifier to identify human faces in dataset and checked for model's accuracy. Used VGG16 to identify dog pictures.
4. After this I built a CNN model from scratch and trained it and validated it. Model has 3 conv. Layers and 1 hidden layer. Pooling has been used before each layer.
5. Model when deployed on test data gives an accuracy of 10 percent and it serves as the benchmark model.
6. Finally we will build the new model using transfer learning using ResNet101. It gave us an accuracy of 68 percent.

Hyper Parameters used for Model from Scratch:

Loss Function Used: Cross Entropy Loss

Optimizer: Stochastic Gradient Descent

Learning rate : 0.01

Epochs : 20

Batch_size = 10

Hyper Parameters used for Transfer Learning Model:

Loss Function Used: Cross Entropy Loss

Optimizer: Adam

Learning rate : 0.002

Epochs : 10

Batch_size = 10

Refinement:

Initially I used ResNet50 for transfer learning, but it was just giving me a 60 percent accuracy on the test set. Then I switched to Resnet101, which gave me an accuracy of 68 percent.

4. Results

Model Evaluation and Validation:

1. Model From Scratch: The model architecture has been defined earlier. Here I used the learning rate of 0.01, stochastic gradient descent as the optimizer and 20 epochs to train the model. After each epoch, the model was validated using validation set. The training loss decreased from 4.85 to 3.79 while validation loss decreased from 4.8 to 4.04. The model when tested on test set of 800 odd images gave an accuracy of 10 percent which though not impressive, was well enough to satisfy the provided requirement.
2. Transfer Learning with ResNet 101: The model architecture has been defined earlier. Here I used the learning rate of 0.002, Adam as the optimizer and 10 epochs to train the model. After each epoch, the model was validated using validation set. The model when tested on test set of 800 odd images gave an accuracy of 68 percent which though not impressive, was well enough to satisfy the provided requirement.

Justification: With 68% accuracy, the model performed much better than the benchmark model.

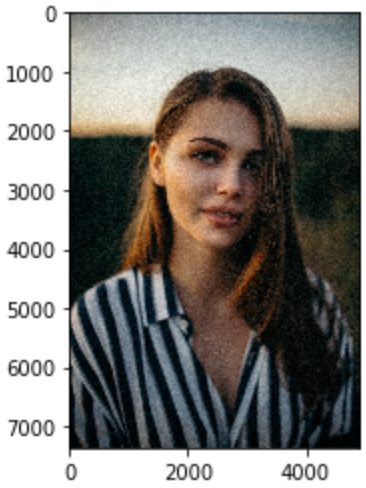
5. Conclusion

Free Form Visualization:

Below are the output on some of the images outside of the ones provided:

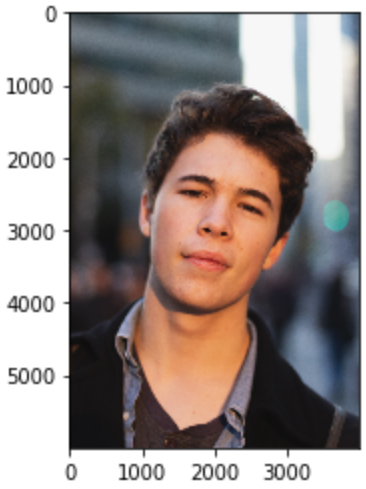
Human / resembing dog breed is Pekingese
./myHumans/lucas-sankey-bXq8pVfP-fY-unsplash.jpg

Human / resembing dog breed is Chinese crested
./myHumans/stefan-stefancik-QXevDflbl8A-unsplash.jpg

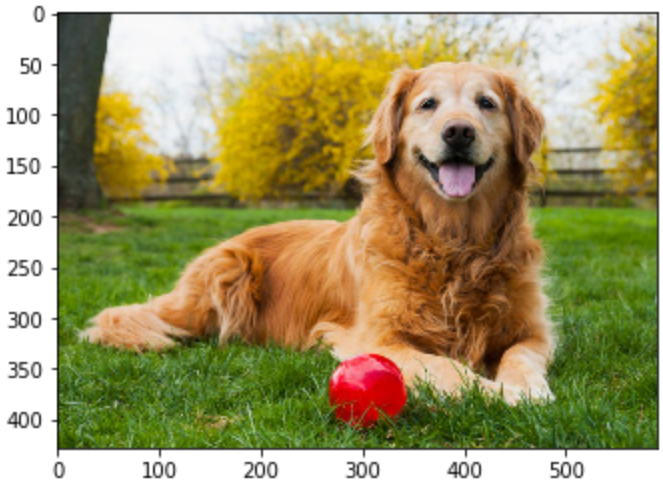


Human / resembing dog breed is Maltese
./myHumans/harishan-kobalasingam-8PMvB4VyVXA-unsplash.jpg

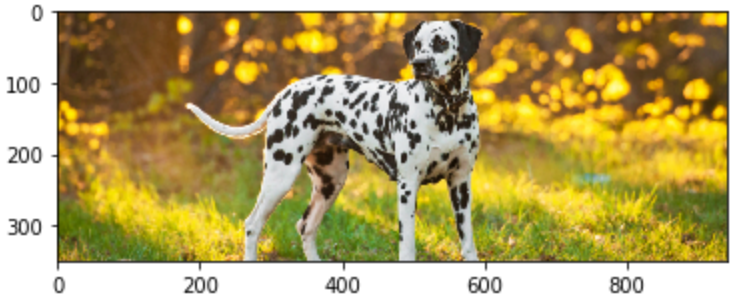
Udacity Capstone Project Report
Jill Shah



Dog / dog breed is Golden retriever
./myDogs/senior-golden-retriever-with-ball-picture-id488657289.jpg



Dog / dog breed is Dalmatian
./myDogs/shutterstock_205844974.jpg



Human / resembing dog breed is Golden retriever
./myDogs/michelle-huber-jkeRT6XDBGY-unsplash.jpg

Udacity Capstone Project Report
Jill Shah

Reflection:

To summarize the entire process, we started off with downloading the image sets. Then we built/used algorithms to detect human images and dog images using HaarCascade and VGG-16. Then we moved on to build our own algorithms to classify the dog images into various breeds. This was done first by using a CNN model from scratch and then by using transfer learning.

This project gave me a general idea of how the image classification solutions are built and deployed. The most difficult aspect in this project for me was calibrating the model architecture such that it fits my hardware specification.

Improvement:

There is still a lot of scope to increase model accuracy with the following techniques: Image Augmentation, Increasing Dense layers and Increasing no of epochs with Dropout to decrease the chances of model overfitting. Following the above areas, I'm sure we could increase the testing accuracy of the model to above 95%.