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.

Udacity Capstone Project Report
Jill Shah

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.

2. Analysis

Data Exploration:

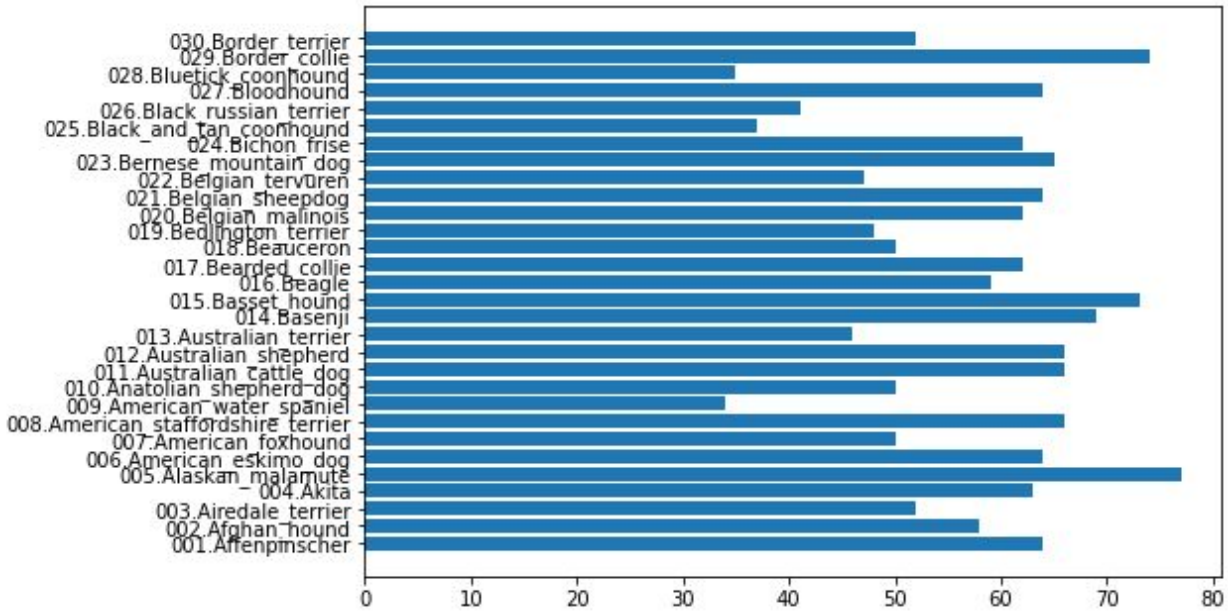
Input Format : Image type

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:

Plot of number of images in first 30 folders:



Scatter plot for images in train folder:

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.

Algorithms and Techniques

Human Face Detector: Here I will use Cascade Classifiers(HaarCascade) from OpenCV to recognize the face of a person on an image. 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.

Dog Breed Classifier from Scratch: Here I will use PyTorch to design, train and test a model from scratch.

Here, we input the image into a convolution layer. This helps extracts features from the input images while preserving the relationship between pixels by learning image features using small squares of input data. We also use pooling. Pooling reduces the

number of parameters when the images are too large. After this we flatten the image and feed it into fully connected layer.

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 Model: 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%.

Evaluation Metrics: As a monitoring metric we will consider accuracy. It gave us an accuracy of 68 percent.

3. Methodology

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 ResNet50. It gave us an accuracy of 68 percent.

Loss Function Used: Cross Entropy Loss

Optimizer: Stochastic Gradient Descent with learning rate of 0.01.

Data Pre-Processing: For the data pre-processing, we have used PyTorch's transform method. Here we are randomly, rotating, flipping and cropping the image. This helps the algorithm train for images which are no in the standard format.

4. Results

Reflection: 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%.

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