Project Report

Capstone Project

Name : Shrey Amin

Course: Machine Learning Engineer Nano Degree

Topic: Dog Breed Classifier

Table of Contents

Domain Background

Image classification is one of the fundamental problems in field of Machine Learning and Deep Learning. This forms core of several computer vision and real-world image processing problems. Before introduction of deep algorithms image classification was mostly performed using processing techniques. But in last 10 years due to advancement in computation resources such like GTU and TPU is possible to train deep neural networks for to solve real world problem such as Image Recognition, Object Detection and Multi Class Image Classification. Extensive research in the field of Deep Learning and Computer Vision has made possible to solve several Artificial Intelligence problems. Further, this has even had a positive impact in Healthcare and Automobile industries. The scope of this project is confined and focuses on creating an image classifier. In this project, I will develop a Deep Learning model which will classify an input (i.e. human face or dog) into predefined breeds of dog.

Problem Statement

The objective of this project is to train a deep learning model and accurately predict a breed of dog given an image as an input. But an input image can ideally belong to numerous class. So there few constraints based on which the project will work. If input is an image of a human being then the classifier will predict the label that closely resembles a canine breed of the dog. Initially, a human face predictor will be used to determine whether the input image is of a human being or not. Then, the next step is to check whether the image of a dog or not. This will be done using a binary dog classifier. Thus, these steps will prevent any erroneous or invalid input image into the classifier to predicting breed of the dog.

Moreover, the main aim of this project is to create a robust classifier that can successfully determine breed of the dog image. So, it is equally important to create a classifier with acceptable accuracy so that the trained model can be used to classify real world images of dogs. So, the trained model can be deployed and integrated any web or mobile application to classify to predict breed of a input image of a dog.

The trained model will predict a score that indicates the probability of an input image belonging to a specific breed of a dog. But, before we move into final stage of creating a dog breed classifier, we will need a human face detector and a dog detector to valid the input image. Thus, the project is initially divided primarily divided into following steps:

1. Create a Dog detector to check image is of a dog or not.
2. Create a human face detector that draws a bounding box indicating the portion of a human face.
3. If the input is dog image, then determines its breed.
4. If input is a human image, then find a closely resembling breed of the dog.
5. Else term the input image as invalid.

Below image shows the ultimate scope of this project and what exactly the problem statement is.

Figure 1 Sample Input Dog Image

Metrics

For this project, the dataset is divided into 3 parts.

1. The model is trained using the train dataset.
2. The validation data is used to improve the training of the model. Validation data helps us to know whether the model is overfitting or not.
3. Finally, the testing data is used to predict the performance of the model on unseen data.

Further, accuracy score is the main metric used for the validating the performance of the dog breed classifier. Below is the formula to find accuracy.

Accuracy = Number of items correctly classified/ All classified items

Also, accuracy score will be used determine the performance of human face detector and dog detector.

Further, during model training, we compare the test data prediction with validation dataset and calculate. However, accuracy helps us to know the final score of classification, but we also want to evaluate model in probability of an image belonging to a specific breed of the dog. So, Multi class log loss or cross entropy loss is used 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. Thus, while training log loss will be used to calculate the error and then validation data will be used to improve the training performance.

Data Exploration

For creating a dog detector and dog breed classifier a labelled images dataset of dogs is required. There are 2 datasets used in the project which are as follows:

Dog images dataset

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.

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. This dataset is also not balanced. The data is not balanced because we have 1 image for some people and many images for some.

Exploratory Visualization

Following image show the variation in the datasets. All the images belong to RGB channels and the resolution and quality of the input images is good. This helps in learning more complex features like texture and color composition and this can help to determine breed of two closely matching images of the dogs.

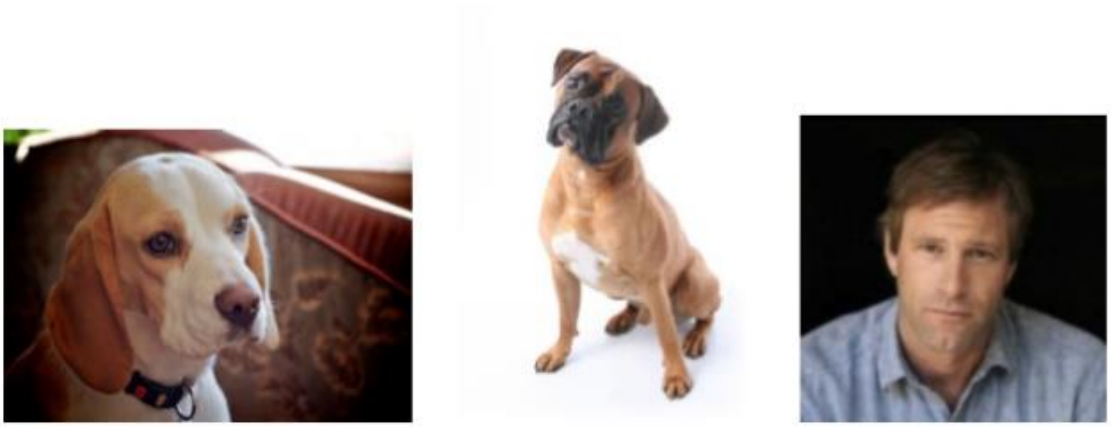


Figure 2 Sample Images in the dataset

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. For solving this multiclass problem, Convolutional Neural Network (CNN) is used which is a state-of-the-art algorithm for image classification problem. The approach include following includes 3 steps:

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.

1. Detect human image using OpenCV which in turn uses a haar feature based cascade classifiers.

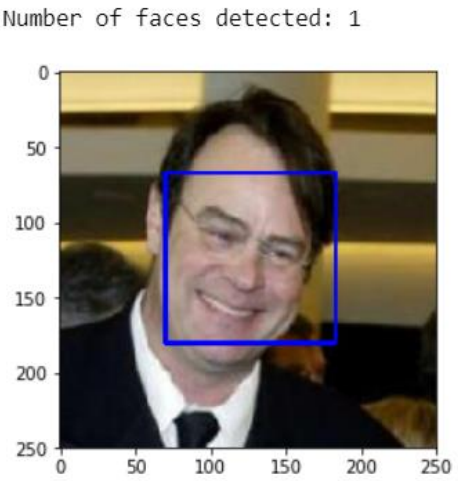


Figure 3 Output of Face Detector

2. Create a dog detector using a pretrained VGG-16 model. VGG-16 is one the CNN architecture which was trained initially on ImageNet dataset.
3. Train a custom model to predict breed of the dog given an input image.
4. Finally, train a CNN model using transfer and compare the results.

The goal is to pass an image as input to a CNN model which will process the image and predict the breed that closely matches the best out of 133 breeds.

Benchmark

The problem statement of this project is extremely difficult because in real world it is at times difficult for even humans to correctly a breed of a dog given from image. As we have comparatively less dataset for this project, the custom CNN model will have less accuracy. 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%. Moreover, we will be also using transfer learning technique to improve the performance of this model. Thus, the benchmark for creating CNN model using transfer learning should be at least 70% and this much higher than a custom CNN model.

Data Pre-processing

For creating any machine learning or deep learning-based model, the foremost step is to pre-process the dataset to improve the results of training and create a robust feature sets than can accurately solve a given problem.

Following are the steps that are involved in data pre-processing:

1. Resize the images in the dataset to fixed size 224*224 pixels.
2. Perform augmentation of the images to include variation in the dataset.
3. The train data images are randomly rotated, and random horizontal flip is applied.
4. Normalization the pixel values of the images.
5. Convert the images into appropriate tensor the feed in the model for training.

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

To summarize, the architecture of custom CNN model is as follows:

- First Layer is the Convolutional layer which takes image as the input.
- Then there two 2 convolutional layers followed by maxpooling layer.
- Each layer uses Relu as the activation function.
- Then there are two fully connected layer.

Further, dropout hyperparameter is used to avoid overfitting. However, this custom CNN model performs poorly, and accuracy obtained is just 18%. To improve the performance transfer learning technique is used.

Refinement

The CNN created from scratch have accuracy of 18%, Though it meets the benchmarking, the model can be significantly improved by using transfer learning. For transfer learning, Resnet101 architecture is used and initial layers are fined tuned and extra two fully connected layers are added on top of this architecture. Then, this model was trained for 5 epochs and there is significant improvement in the result. The accuracy obtained is 80%. This really shows that if we have small dataset it is better to used transfer learning technique than training the custom CNN model.

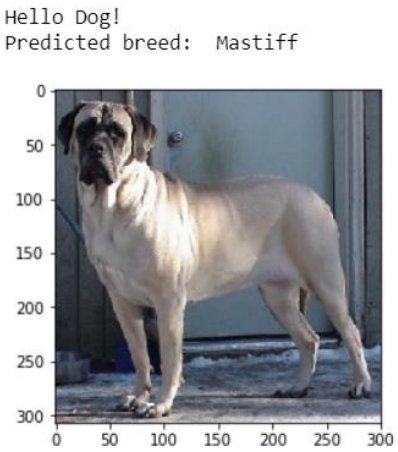


Figure 4 Output of model created using transfer learning

Model Evaluation and Validation

For training the model cross entropy loss function and stochastic gradient descent is used. Further, 3 models are used in this project which are as follows:

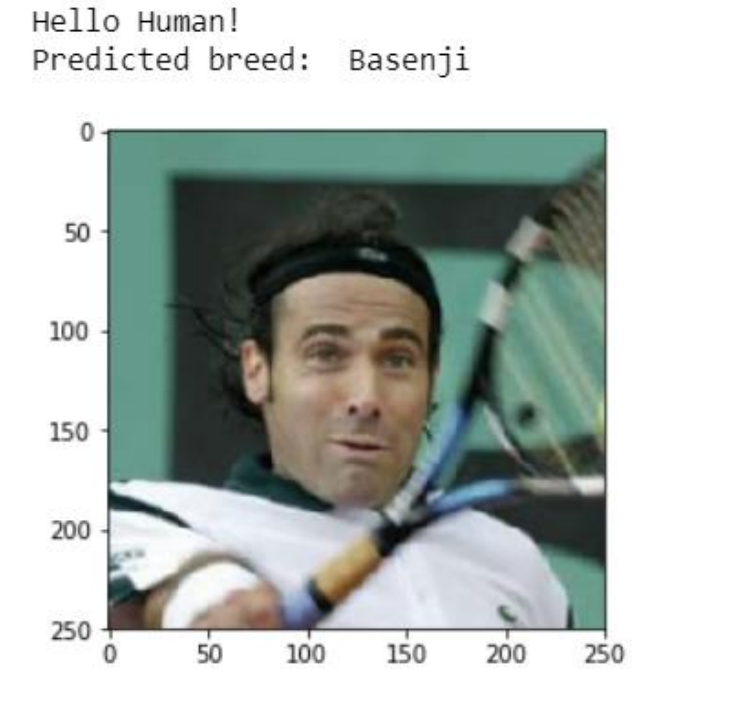
1. Human Face Detector: Its used OpenCV's haar cascade classifier to detect face of human from an input image. The accuracy obtained for human dataset was 98% and for dog dataset it was 17%.
2. Dog Classifier: A pretrained VGG16 model was used to classify the input image as dog or not. The accuracy of this model was 100% for dog dataset and 1% for human dataset. This clearly shows that this model is much more robust than human face detector.
3. Resnet101 CNN model: Transfer learning technique was used to create a custom dog breed classifier and the accuracy obtained was 80% which far better than training a CNN model from scratch.

Justification

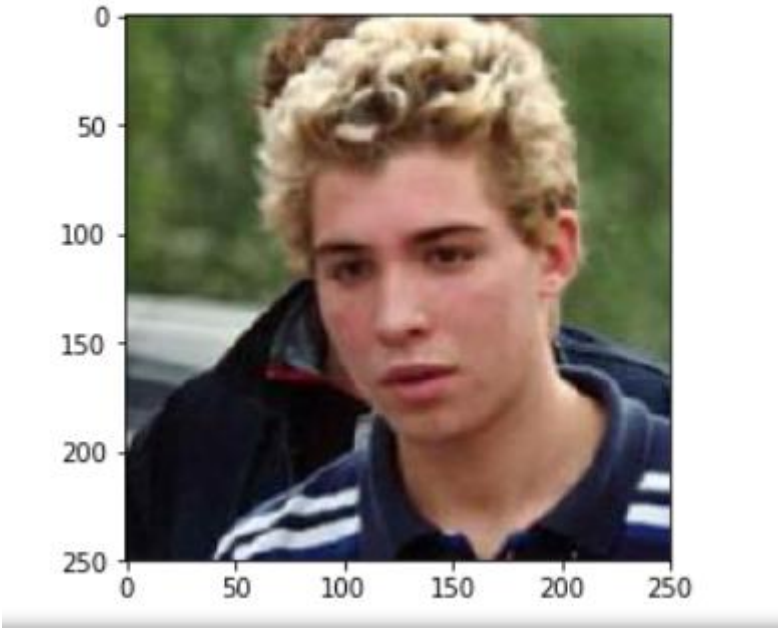
I think the model performance is better than expected and the performance is appreciable for a complex real-world problem. The model created using transfer learning have an accuracy of 80% compared to the CNN model created from scratch which had only 13% accuracy. Following are some of the results of the final dog breed classifier created using transfer learning.

Future Work and Improvements

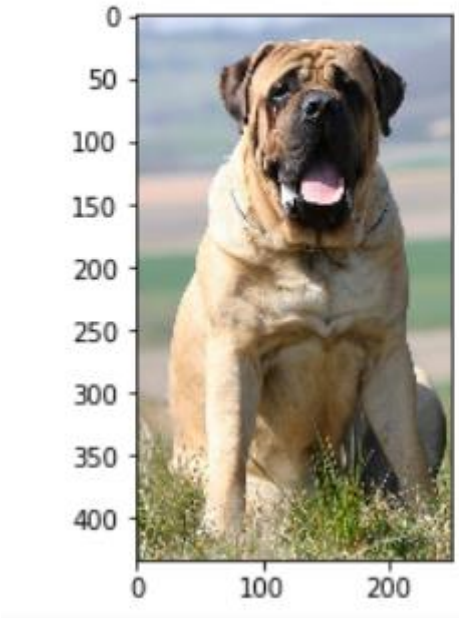
The final model had an accuracy of 80%. For a small dataset, the results are pretty good. But there is still room for improvement. More training data is needed to learn the complex features that can be used in differentiating breed of the dogs. Variation can also be included by increasing the rate of augmentation. Moreover, results of several CNN architecture can be compared to select the best model.



```
Hello Human!
Predicted breed:  Cane corso
```




```
Hello Dog!
Predicted breed:  Mastiff
```



References

1. Original repo for Project - GitHub: https://github.com/udacity/deep-learning-v2pytorch/blob/master/project-dog-classification/

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

3. Imagenet training in Pytorch: https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f6173/imagenet/main.py#L197-L198

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

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

6. http://wiki.fast.ai/index.php/Log_Loss