Dog Breed Classifier using CNN- Final Report

Project Overview

The dog classifier problem consists of identifying a dog breed using machine learning algorithms to train a model based on a database of images of dogs and humans. The idea is to rain a model to distinguish dog's breed when presented with a random image of a dog but also associate a dog's breed based on similarities with human features when presented an image of a human face.

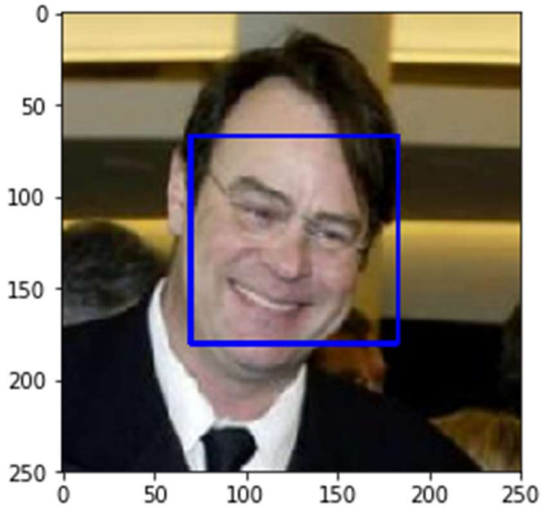
Problematic

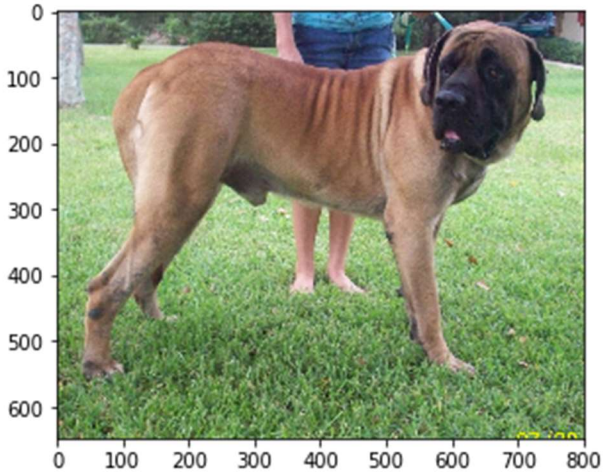
There are two parts to the problem as described above. First, we need to create an algorithm that detects dog's and their breed in each input image. Second, we need another one that is trained to associate dog's breed that resembles features of human faces. To achieve this, we built dog face and human face detectors.

Data Exploration

The input dataset format differed for humans and dogs. Each dataset came into different format and size which required to use normalization. The dog image dataset had 8351 images whereas the human dataset accounted for 13233 total images. The dog's datasets was split into train (6680), test (836) and validate (835).

Below are examples of an image from each dataset to describe some of these issues:





Methodology

The problem required the use of multiclass classification and we used one of the CNN algorithms available in Pytorch to achieve this. We first built a human face detector using a cascade classifier available in OpenCV. We then used VGG16 available in Pytorch to train the dog face detector on the dog image input dataset. We finally passed on the processed image (dog or human) onto a CNN model which will identify the most dog's breed that most resembles the input image. Some model accuracy minimum requirements were set to be at 10%. This is to maintain a reasonable level of prediction accuracy.

To achieve our objective, we use the conventional approach to split the data into train, test and validate segments. Our model performance was then tested using the test dataset portion and we used accuracy as a metric to evaluate our CNN model. Hyperparameter tuning was achieved by comparing cross validation and testing datasets to obtain a model with good performance.

Implementation

The architecture is composed from a feature extractor and a classifier.

The feature extractor has 3 CNN layers in to extract features. Each CNN layer has a ReLU activation and a 2D max pooling layer in to reduce the amount of parameters and computation in the network.

After the CNN layers we have a dropout layer with a probability of 0.5 in to prevent overfitting and an average pooling layer to calculate the average for each patch of the feature map.

The classifier is a fully connected layer with an input shape of 64 x 14 x 14 (which matches the output from the average pooling layer) and 133 nodes, one for each class (there are 133 dog breeds). We add a softmax activation to get the probabilities for each class.

We use the VGG16 network that was pretrained on the ImageNet dataset. The ImageNet dataset contains similar images with our own dataset so we can keep the feature extractor part.

Because we have a small dataset and we don't need to identify dogs but dog breeds we replace the classifier with a fully connected layer with 1024 nodes, with ReLU and a dropout with a 0.4 probability, connected to an output layer with 133 nodes (same as the number of classes).

We then train the model but we freeze the parameters for the feature extractor so only the classifier parameters get backpropagated.

Results

The human face detector identified 98% of humans correctly in human input images whilst 17% of humans were confused with dogs. The dog detector on the other hand detected only 1% of humans as dogs whilst detecting dogs with 100% accuracy.

As far as results from our train and tested CNN model, a 16% accuracy was achieved (140/836) with Test Loss 3.67, therefore above the accuracy minimum set requirements. The results were improved by using the transfer learning classifier approach and the tested model performed better with a accuracy of 69% (577/836) and test loss of 1.06.

When testing the algorithm on our own images, we noted that dogs were detected correctly in 2 of the 3 dog images provided. Humans were detected correctly almost on all images, except for the cat image detected incorrectly as human face.

Examples from own image tested with algorithm

Conclusion

We constructed detectors for dogs and humans with the objective to attempt to successfully associate a dog's breed based on the input dog or human face image. The model was then trained, tested and validated and refined to progressively increase prediction accuracy. The model performed well in line with accuracy level of prediction when tested with our own images. Further improvement to the model can of course be made to increase accuracy by using perhaps more layers in the neural network architecture and/or train the model with further iterations or parameters.

References

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

2. OpenCV: Haar feature-based cascade classifiers

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

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

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