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 and human face detectors. However, this was a first step to build an app that makes predictions of dog breed based on an image. The next approach that we wanted to use was through applying a convolutional neural network algorithm to train a model to associate breeds of dog to their image. We also want to achieve a certain benchmark of model performance accuracy to build an app that recognizes the correct dog's breed successfully in as many cases as possible.

Metrics

We split the data into train, test and validate. Common practice is to use a 60/20/20 approach for that. Accuracy is our what we use as a suggested in the notebook. Accuracy is the ratio of correctly classified objects over the total number of classified objects. Using test and validation dataset allows to gauge the performance over a faster runtime to allow for tuning the model parameter prior to deploying the model. Log loss is also a measure that we look at as a measure of uncertainty in model prediction.

Data Exploration

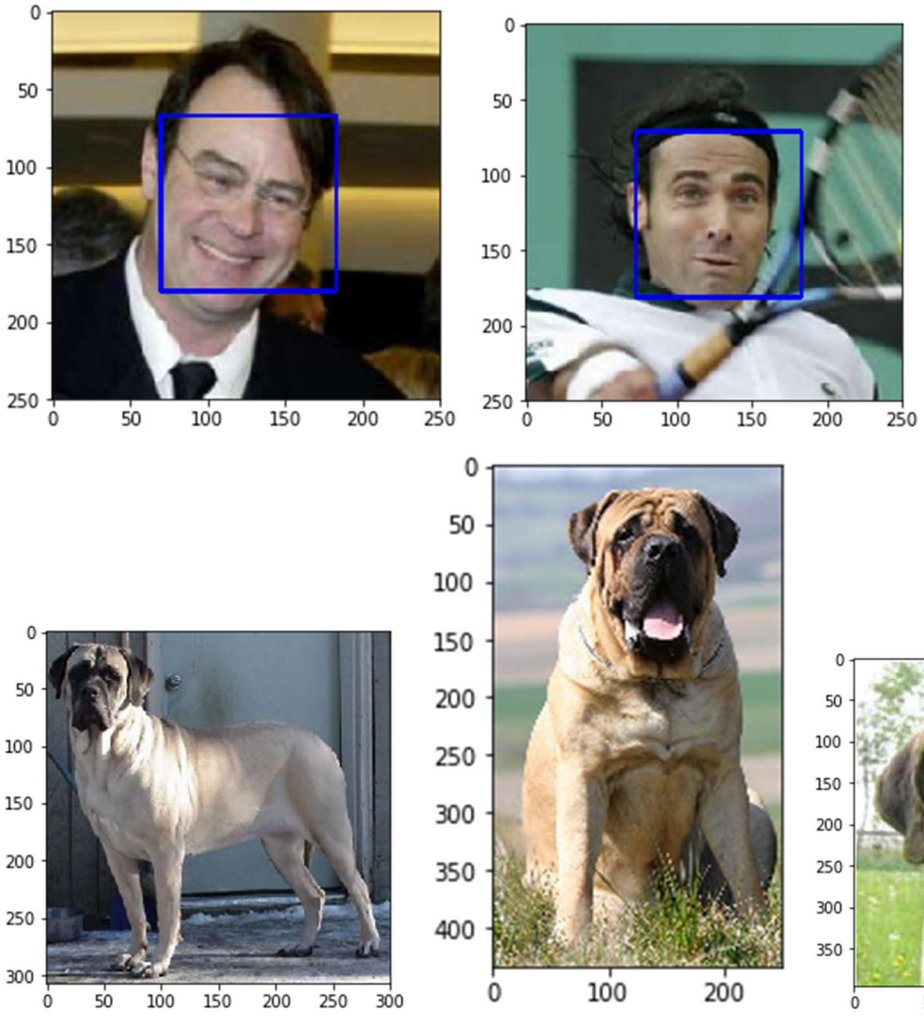
The input dataset format differed for humans and dogs. Each dataset came into different format and size which required to use normalization.

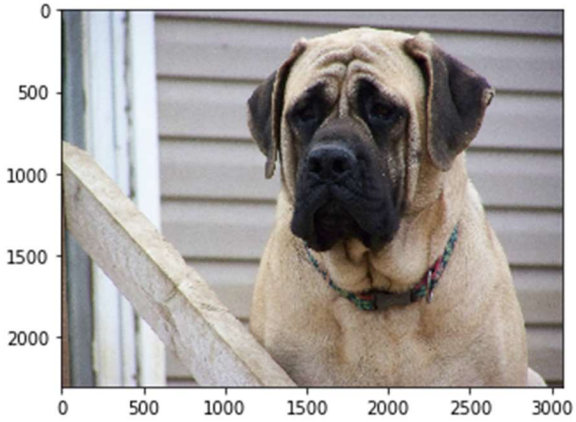
Dog Dataset: 8351 total images. The dog's datasets were split into train (6680), test (836) and validate (835). The dog's breed was each stored in a folder, hence totaling 133 folders. One of the issues was that the dog's images came in different sizes (not fully sized in some cases) and with different backgrounds.

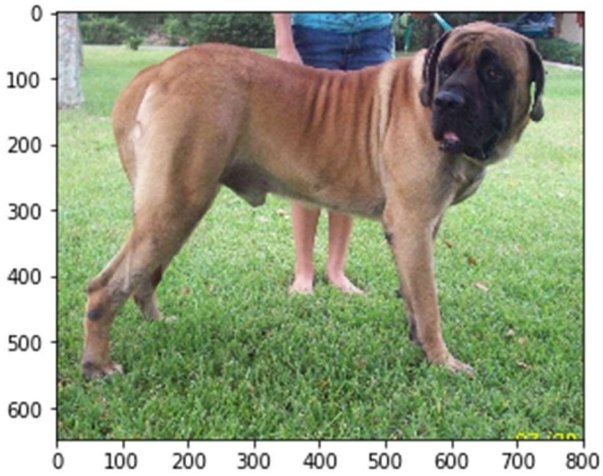
Human dataset: 13233 total images sorted by name and stored in 5750 folders. The human dataset was more uniform, and all came in 250x250 size.

Both datasets are relatively unbalanced. For instance, the number of images associated with each breed varies for the dog's dataset, and sometimes there were several more image for the same person compared to others in the case of the human dataset.

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







Algorithm and Techniques:

Convolution is the process of adding each element of an image to its local neighbors, weighted by the kernel. Convolutional Neural Network or CNN is used, in this case to solve a multiclass classification problem. CNN is an algorithm that is frequently used in the context of deep learning. Its advantage relies on the fact that in can take an image as input, analyze features and objects that are contained inside it, and use ones specific to each image to differentiate between images. In machine learning, CNN's are used for image classification and recognition because of its high accuracy.

To train the CNN algorithm, we first used Open CV's cascade classifier to detect features in human faces dataset. We then use a pre-trained Pytorch model (VGG16) to detect dog's images. We then use the trained CNN model to process and predict dog's breed in the input image of human or dog.

Benchmark

The CNN that was built from scratch had to perform with an accuracy of at least 10%. This would guarantee a relatively good starting level compared to a random guess (1 over 133 ~1%).

The CNN model created using transfer learning must have accuracy of 60% and above. In other words, 2 out 3 dog breeds identified correctly seems like a reasonable predictor.

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.

Data preprocessing:

We had to conduct some data preprocessing prior to building and training our model. All images were resized to 224x224 and normalized to the train, test and validation datasets. We also used image augmentation to minimize overfitting. We then apply random data rotation and horizontal flip. The last step consisted in converting the data into tensor prior to transferring to the model.

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 (10%). 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.

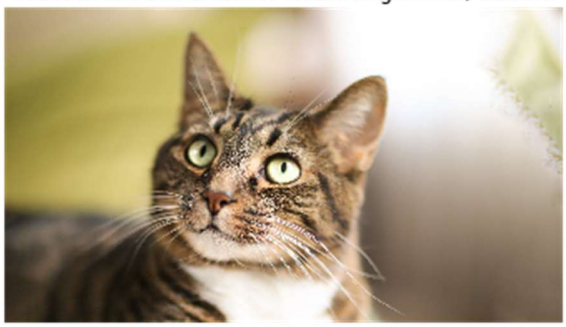
Justification
The model performance is satisfying. BY using transfer learning, we improved the accuracy to 69% from the previous 16% obtained with the model created from scratch. In both situation the benchmark was outperformed.

Improvement
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

Nice doggie!
You are a French bulldog, aren't you?!?

Hello, hoooooman!
You look like a...Pembroke welsh corgi. Hmm, weird!!!



Hello, hoooooman!
You look like a...Pharaoh hound. Hmm, weird!!!

Nice doggie!
You are a Labrador retriever, aren't you?!?

Hello, hoooooman!
You look like a...Mastiff. Hmm, weird!!!

Hello, hoooooman!
You look like a...Bearded collie. Hmm, weird!!!

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/