Machine Learning Engineer Nanodegree

Capstone Project

Samuele Buosi

May 11th, 2020

Dog Breed Classifier

I. Definition

Project Overview

The purpose of this project is to classify the different dog breeds. In the world, exists a lot of breeds of dogs that differences one to another by a lot of parameters such as high, size, kind of employment, etc...

For this project, the classification is totally based on images and the goal will be to classify 133 dog breeds using state of art deep learning algorithms to teach the computer how to give an estimation of a particular dog breed from an dog image.

Image classification is one of the most important field in machine learning/computer vision and one other most important challenge is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) where each year review the new classification algorithms[1].

Problem Statement

The final purpose of this project is to write an algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither.

In general, after a detection, we could have:

- dog detection in the image, return the predicted breed.
- human detection in the image, return the resembling dog breed.
- neither detection in the image, provide output that indicates an error.

Metrics

In order to deal with a multi class classification problem, the evaluation metric that will be used is the negative log- likelihood loss function.

Using this matric, the algorithm will calculate each iteration the distance of a predicted output to the corresponding label.

In this way, the algorithm will learn from it and will adjust the predictions in order to minimize this distance (loss).

II. Analysis

Data Exploration

The datasets needed are two: dogs dataset and human dataset.

The dogs dataset is composed by:

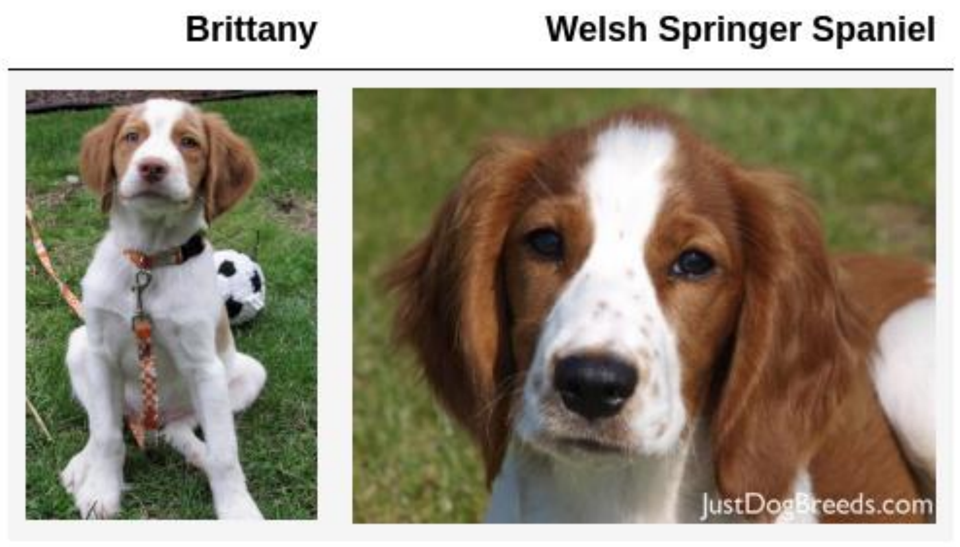
- Training: 6680 images
- Validation: 835
- Test: 836
- Total images: 8351
- Classes (dog breeds): 133

The human dataset is composed by 13233 human images.

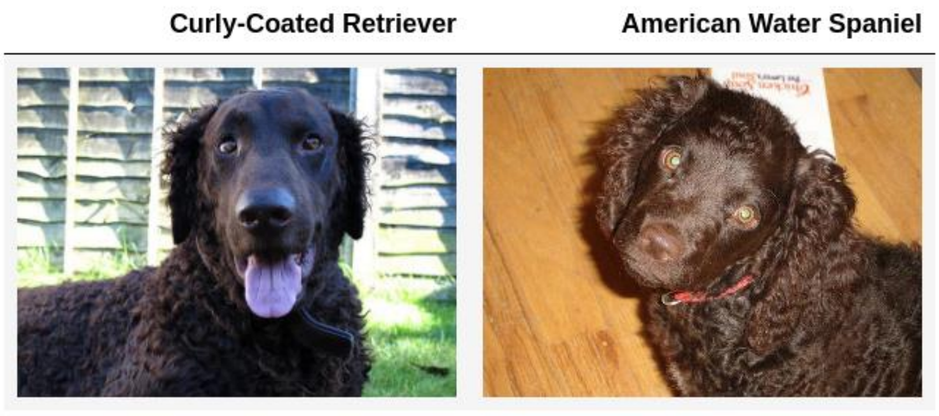
All the images are resized to 244x244 and normalized before being used with the model.

Exploratory Visualization

We mention that the task of assigning breed to dogs from images is considered exceptionally challenging. To see why, consider that even a human would have trouble distinguishing between a Brittany and a Welsh Springer Spaniel.



It is not difficult to find other dog breed pairs with minimal inter-class variation (for instance, Curly-Coated Retrievers and American Water Spaniels).



Likewise, recall that labradors come in yellow, chocolate, and black. The vision-based algorithm will have to conquer this high intra-class variation to determine how to classify all of these different shades as the same breed.

Yellow Labrador

Chocolate Labrador

Black Labrador

The random chance presents an exceptionally low bar: setting aside the fact that the classes are slightly imabalanced, a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

Algorithms and Techniques

The solution of the project follows six main steps.

First, we need to explore the datasets in order to understand how to use them and choose the proper algorithms.

Second, implement a Haar feature-based cascade classifier using OpenCV in order to detect faces in the human dataset.

In the third step, I will use a pre-trained VGG16 model in order to detect dogs in the dogs dataset

Fourth, I will create a LeNet[2] like architecture that uses CNN in order to classify the 133 dogs breeds and have an accuracy greater than 10%.

In the step five, I will use the transfert learning tecquinque in order to a pre-trained ResNet50 architecture and continue the training with the dogs dataset. The minimum accuracy required is 60% on the test set.

Seventh, I will write a custom algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither.

In the last step, I will Test the Algorithm with some random images found online.

Benchmark

The model will be compared with different benchmark models in a Kaggle competition (https://www.kaggle.com/c/dog-breed-identification/discussion)

III. Methodology

Data Preprocessing

For the Haar feature-based cascade classifier no data preprocessing is necessary.

For the pre-trained VGG16 model, is used only an image resize transformation to 224 x 224 pixels because is the input size of the network.

In order to increase the performance of the custom CNN model and the ResNet50 model, I have introduced an ETL pipeline to the dogs dataset where:

- For all the three datasets (train, validation, test) I put the data a mean to 0 and a standard deviation to 1 (Z-score normalization)
- For the training dataset is also used the following transformation:
  - Random rotation of maximum 30 degrees
  - Random resize crop of 224 x 224 pixels
  - Random horizontal flip
- For the validation dataset is also used the following transformation:
  - Image resize to 256 x 256 pixels
  - Centre crop of 224 x 224 pixels
- For the test dataset is also used the following transformation:
  - Image resize to 224 x 224

After this transformation, the data is putted to three different data loaders where a batch size of 64 is used.

Implementation

For the Haar feature-based cascade classifier is used a computer vision pre-trained face detector.

For the VGG16 pre-trained model, is used PyTorch in order to make the predictions.

For the custom CNN model, I followed the LeNet architecture and I divide the task in two main blocks:

- feature extraction
- classification

For the feature extraction, is used some convolution layers.

For classification, some fully connected layers are used.

I also used maxpooling in order to reduce the dimension of the layers and the dropout in order to prevent overfitting.

The loss function used is: Cross Entropy.

The optimizer used is: Stochastic gradient descent (SGD).

The learning rate is set to 0.05.

Here the net summary:

```
Net(

(conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)

(fc1): Linear(in_features=6272, out_features=500, bias=True)

(fc2): Linear(in_features=500, out_features=133, bias=True)

(dropout): Dropout(p=0.3))
```

For the last model (ResNet50), I have changed the last layer of the architecture to:

```
Linear(in_features=2048, out_features=133, bias=True)
```

This modification is made because the output nodes of ResNet50 is 2048 but we have 133 dog breeds.

The loss function used is: Cross Entropy.

The optimizer used is: Stochastic gradient descent (SGD).

The learning rate is set to 0.001.

Refinement

Especially for the custom CNN and ResNet50 models, some hypermeters toning is made during all the training and design process.

These are some parameters toned during the process:

- Training epochs, learning rate, dropout value, etc...

IV. Results

Model Evaluation and Validation

All the models trained meet the expectation end the results are:

- For the pre-trained VGG16 the dogs detected in dogs images is the 98.0%.
- For the custom CNN model, the accuracy is: 20% (174/836)
- For the ResNet50 model, the accuracy is: 72% (610/836)

Justification

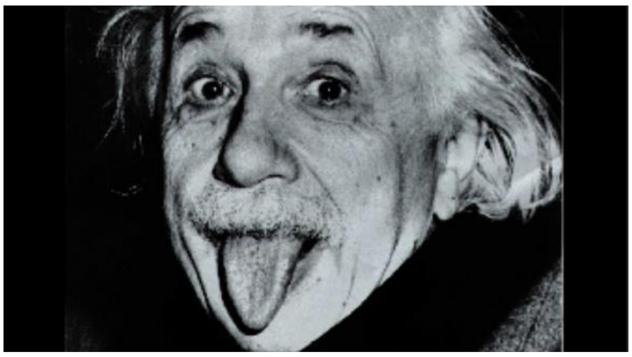
All the model trained meet the expectation of the project but there is a room for improvement in according to the benchmark.

V. Conclusion

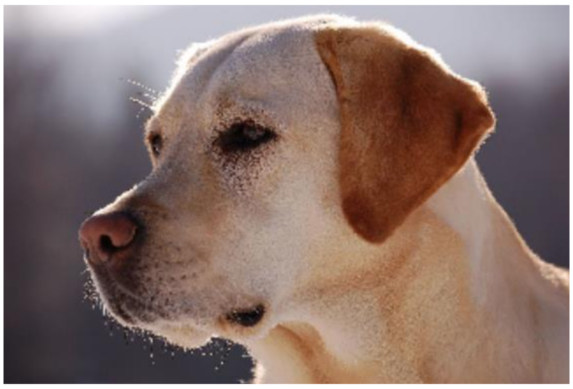
Free-Form Visualization

Here some results from the final algorithm that combine the face detector with the ResNet50 dog breeds model:

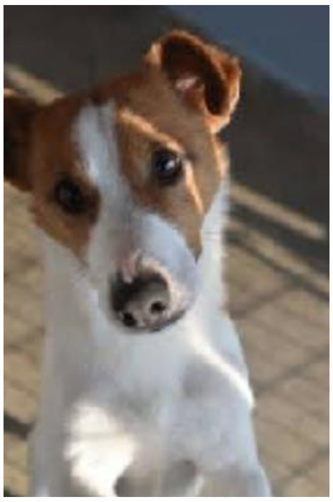
Face detected! Looks like a Dogue de bordeaux



```
Neither!
```



```
Dogs Detected: Golden retriever
```



```
Dogs Detected: Basenji
```

Dogs Detected: Pomeranian

Neither!

Reflection

So, in the end, the solution designed for this project has followed these steps:

- Step 0: Datasets exploration
- Step 1: Detect Humans using a Haar feature-based cascade classifiers
- Step 2: Detect Dogs using a pretrained network
- Step 3: Create a CNN to Classify Dog Breeds (from Scratch) using a LeNet[2] like architecture.
- Step 4: Create a CNN to Classify Dog Breeds using Transfer Learning and a using a ResNet50 architecture.
- Step 5: Write a custom Algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither.
- Step 6: Test the Algorithm with some random images found online.

In this project, we have used a lot of different approaches for manage the dog breeds classification problem and the final solution, with a little bit of fine toning, could be used in a general setting to solve these types of tasks.

Improvement

For this task, I think is possible to improve the model in several ways:

- try some different model architectures like resnext, etc...
- hyper-parameters fine tuning (change the optimizer, batch size, learning rate, etc...)
- improve de dataset with more images and more data augmentation

Reference

1. Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.
2. https://en.wikipedia.org/wiki/LeNet