UDACITY

Machine Learning Engineer

Dog breed classifier

Author: Thiago Francisco Martins

June, 2020

Introduction

At first, the idea of classifying a dog seems simple, but even for humans that might be a difficult task. The idea of looking to a dog and knowing which breed it is sounds something simple but effective for owners that are looking to adopt. Users of such technology could look to a dog they like and apply the algorithm to retrieve not only which breed is the dog they are looking at, but also retrieve important information about that breed such as temperament, habits, abilities and special cares. That would increase the right choice when adopting a pet and owners would be much more informed about the breed.

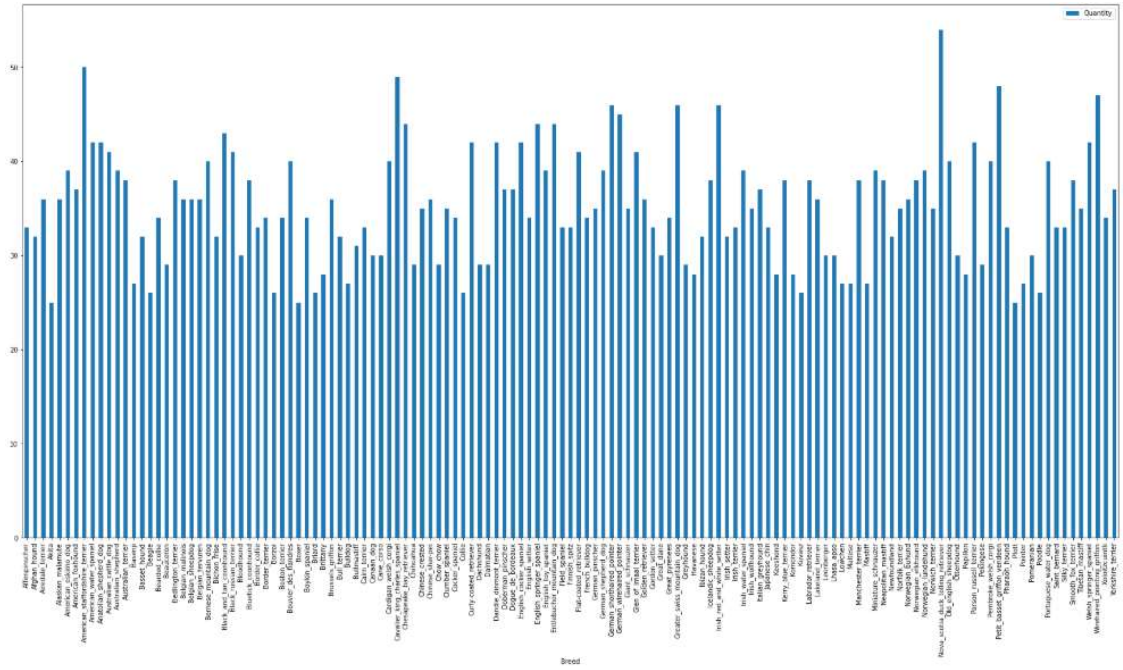
Problem

In this project, the goal is to build a machine learning algorithm that given an image of a dog, it will identify an estimate of the canine's breed. The project will have an API that will returns the name of the dog's breed given a file. If supplied an image of a human, the code will identify the resembling dog breed. The idea is to have a robust, fast and reliable algorithm that detects with high accuracy (metrics will be discussed later on) a dog's breed. Any developer that wants to attack this problem will be able to call this API and use it in his App or website.

Dataset and Inputs

To do that, we will use the dataset provided by the Udacity [1] notebook, analyzing the data, we can see that the data contains 133 dog breeds and has 8750 images. These number of instances usually is probably sufficient to the problem we want to tackle but another dataset may be used if the performance is not satisfactory. Please find below a graph representing how the number of instances are distributed across the different labels in the train folder of the dataset provided by the notebook.

Figure 1: Distribution across classes on the Udacity dataset (training)

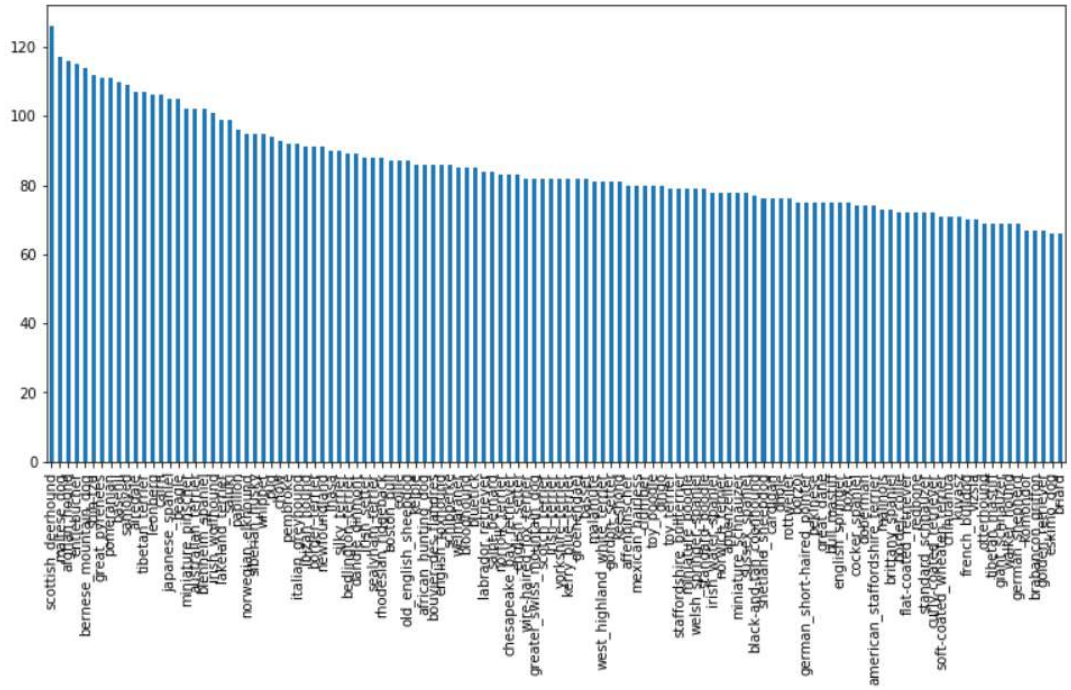


We can see that there is between 20 and 50 images for each class. That might be few images for a high accuracy classifier but we will first try it and if the performance is not enough we will add more data.

An additional human dataset will be used to detect if a human is present on the picture. The dataset has 5750 images from humans, which is pretty high to our goal.

If performance is not high enough, an additional dataset will be used: the [4] Stanford Dog Dataset. It has 20580 images in total and its train folder we has 10222 images. The figure below shows how the distribution on the training data is across different classes.

Figure 2: Distribution across classes on the Stanford dog dataset (training)



We can see that there is at minimum, 60 images per class, with an average of 80 images per class (only on the training data). This should be enough data if performance is not achieved on the first dataset.

[2] Dog dataset, 8750 images (6680 for training, 836 for testing, and 835 for validation).

Training set: 133 classes. Around 40 images per class.

[3] Human dataset, 5750 images, one image per person.

[4] Stanford Dog Dataset 20580 images (10222 training and 10358 for testing)

Training set: 120 classes. Around 80 images per class

Solution

Our solution will apply pre-process on the images of the dataset to increase their quality. Some computer vision process such as increase and decrease of exposure, rotation, and other methods will be applied to increase the quality of the data. The pre-process pipeline will also

has a resize method to feed into the network and a method to check if there is no discrepancy between labels from different datasets.

Once we pre-process the data, we can feed into the network. The type of the network will be a CNN (Convolutional Neural Network), which performs well in image recognition given this architecture allows the network to concentrate on small low-level features in the first hidden layer, then assemble them into larger higher-level features in the next hidden layer, and so on.

In a first step, we will create our own CNN architecture, with just a few layers and evaluate on the data. This step is important to know if our network is working with the labels and images provided and check the performance (though it is not expected to be high, around 5 times greater than a random classifier)

The network built in this project will make use of a technique called "transfer learning", which is the process of reusing a network trained in a similar task to our by changing the last layers and training the network on our dataset.

The [5] Image-net challenge evaluates algorithms for object detection and image classification, among them is a dog category which we can make use of in the process of transfer learning. The best way to do that is picking of the best performers in this competition and using it in our network to classify different dog breeds.

The network chosen is the [6] Xception, it merges the ideas of GoogLeNet and ResNet (winners of the 2014 and 2015 challenges respectively), but it makes some modifications to improve accuracy and performance. This type of network use less parameters, less memory and less computations than regular convolutional layers, and in general they even perform better. So it is a perfect fit given our circumstances of limited resources.

Finally we will deploy our model into an Endpoint through the steps taught in this course and we will be able to request a dog's breed name by given an image as input.

Benchmark model

The benchmark model is VGG-16 model. It was trained also on the [5] Image-net challenge but this network returns 1000 possible categories, among them it is several dog's breeds. The labels related to dogs in this network predicts an index between 151 and 268. As it was trained in a very large database, it is expected to have high performance and we will use it to compare with our model.

The categories of the VGG-16 network were trained using the [4] Image-net challenge. There are [7] 1000 categories of which (according with [1] Udacity notebook) indexes 151 to 268 are dogs. Image below was taken from the notebook:

to check to see if an image is predicted to contain a dog by the pre-trained VGG-16 model, we need only check if the pre-trained model predicts an index between 151 and 268 (inclusive).

Evaluation metrics

This project will be evaluated on Multi Class Log Loss:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

Where,

N	*No of Rows in Test set*
M	*No of Fault Delivery Classes*
Y_{ij}	*1 if observation belongs to Class j; else 0*
P_{ij}	*Predicted Probability that observation belong to Class j*

A similar competition in Kaggle: [8] Dog Breed Identification, used the same kind of evaluation and it accurately represents not only if the classes are labeled correct but also how confident our model is when it predicted classes are equal to the target.

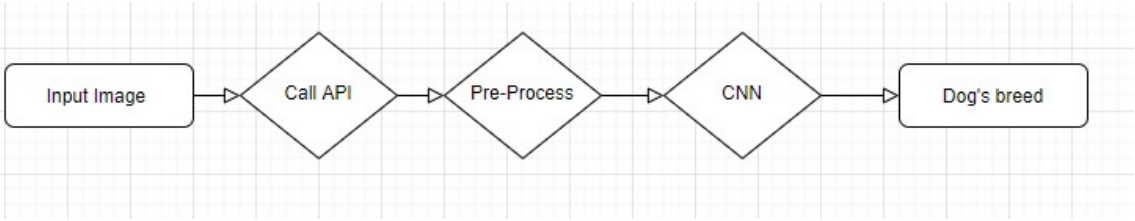
Project design

The project can be summarized in an API, pre-process and a convolutional network.

The API calls the notebook which will be running in an Amazon AWS instance, and a pre-process is applied to the image in order to resize it and detects if a human is present on the picture convert to usable parameters by the network.

The CNN, which will be built with transfer learning from an [6] Xception network will predict the correct dog breed and return to the user.

It is worth noting that unlike the notebook provided, this project will use Keras instead of the PyTorch framework.



References

[1] Udacity Dog classifier notebook: notebook

[2] Dog dataset (Udacity)

[3] Human dataset

[4] Stanford Dog Dataset

[4] Image-net challenge Image-net challenge

[5] Xception: Deep Learning with Depthwise Separable Convolutions Xception

[6] Image-net challenge categories categories

[7] Kaggle competition: dog breed identification Dog Breed Identification