Dog breed Classifier

Capstone project report

Machine Learning Engineer Nanodegree

Capstone Proposal

Fernando de la Torre

February 24th, 2021

I. Definition

Project Overview

In 2012, a deep convolutional neural network architecture capable of outperforming the previous results on a difficult dataset such as imagenet, was released. Alexnet and the paper that is considered one of the most influencing in computer vision were born [1]. Only in 2020, the AlexNet paper has been cited over 70,000 times according to Google Scholar [2].

Since then, there have been more and more effective models for image classification, such as VGG16 [3], GoogleNet [4] and Resnet [5]. All of them have given the opportunity to use computer vision in different areas: medical diagnosis, social networks, self driven cars, surveillance security cameras, face recognition… Today, these convolutional neural networks are everywhere.

The aim of this project is to use some of these deep learning models to classify different dog breeds´ images.

Problem Statement

Let's suppose that a person wants to buy a dog, and he is determined to buy an expensive one because, someday, he plans to buy the perfect couple and sell the puppies. He was told that a french bulldog could be a good choice because of its price, but there is a dog that can be easily confused with it; the Boston terrier. Being able to choose between the two dogs can be the difference between a good purchase, a mistake or even a scam.

Boston terrier and French bulldog

American Kennel Club [6], the largest purebred registry in the world, currently lists 190 dog breeds in the United States alone [7]. Even for someone who knows about different kinds of dogs, like the buyer in the previous example, it can sometimes be hard to tell the exact breed of a dog.

Convolutional neural networks (as Alexnet), a class of deep neural networks most commonly applied to computer vision, are a good choice to help solve the problem. With the help

of CNNs, we will create a classifier capable of, firstly, identifying a dog or a person in an image.

Then, if there is a dog, it will classify its breed among 133 types. If there is a person in the image, the model will also come up with a dog breed; the one closest to the given face features. To achieve this, we will use different CNN models : pre-trained, built from scratch and lastly, using transfer learning. For the face recognition part we will use a Haar feature-based cascade classifier.

The last step is to write an algorithm that can detect a dog or a human in an image and recognise the dog breed or the resembling one if it is human.

Evaluation Metrics

We will use two metrics for evaluating the models: multi-class log loss metrics against the validation set during the training and accuracy against the test dataset when the training process has finished. Log loss is similar to the accuracy metric, but it will favor models that distinguish the classes more strongly. It is useful to compare models not only based on their output but also on their probabilistic outcome [8].

$$LogLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} * \ log(p_{ij})$$

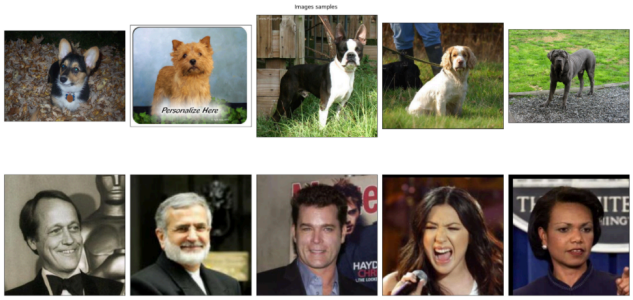
Accuracy is one of the most common metrics in image classification. It is the total number of correct predictions divided by the total number of predictions made for a dataset.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

II. Analysis

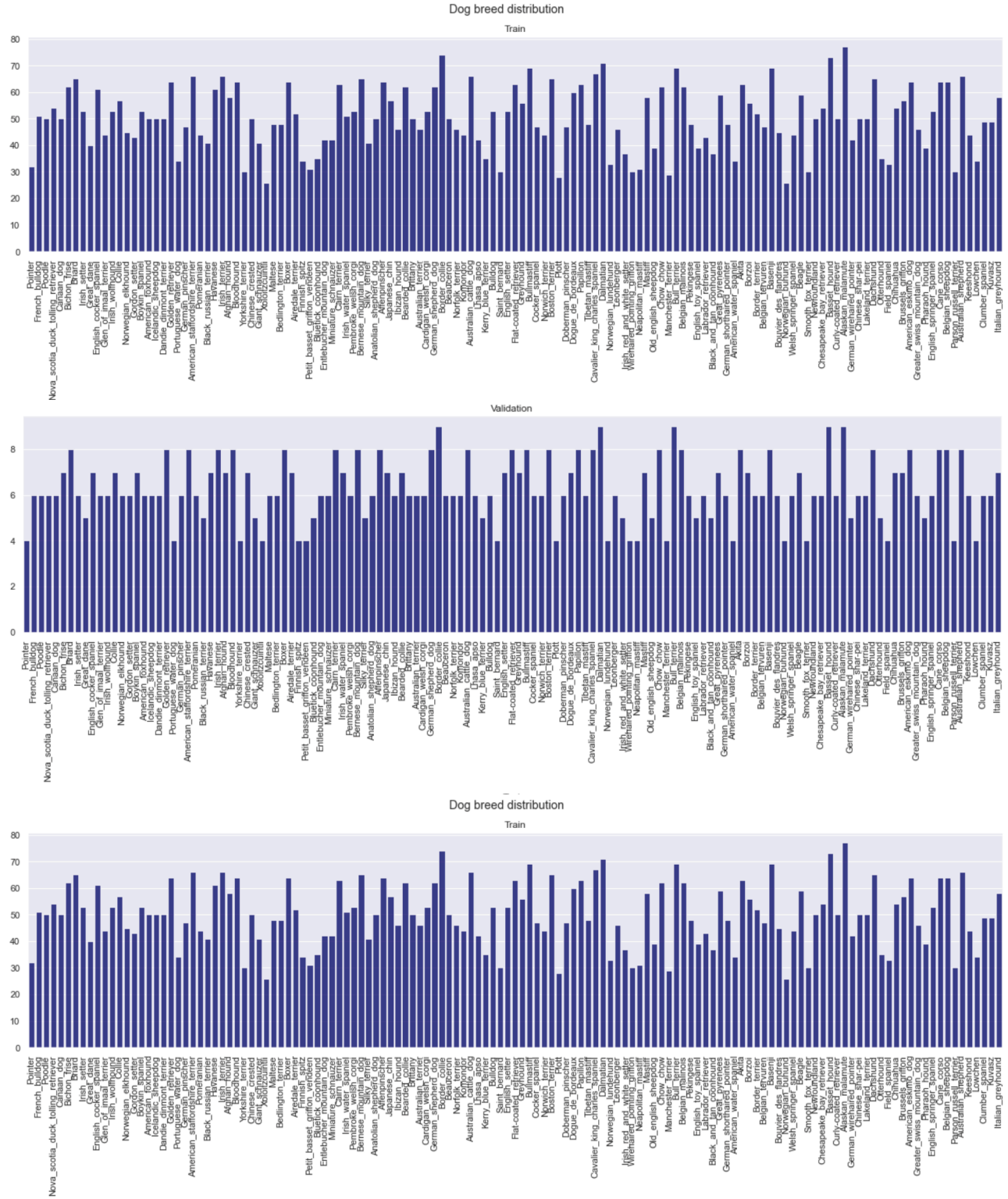
Data Exploration and Visualizations

The data is composed by 8351 dogs´ images and 13233 human faces, all of them RGB and jpeg format.



Dataset samples

The dog images are divided into 6680 for training, 835 for validation and 836 for testing. There are the same 133 breeds in each set: from 26 to 77 images of each in training set and from 3 to 10 in validation and testing sets. As we can see, the images in each class are not balanced.



The images themselves have different backgrounds, lights and the dog can appear in any pose and size. This is convenient for our model, because it can learn a lot of possibilities of what a dog is in a picture. This way we can assure that, later, it will be capable of recognizing a dog in any given image, no matter the angle or the light. The size of the images are different, too.

Different poses and lighting on the dogs dataset

The human dataset contains images of 5749 different people and from 2 to 530 images for each. All of them have the same size.

Algorithms and Techniques

We are going to use 3 different approaches using CNN for solving the problem of dog classification. First, we will use a VGG16 pre-trained model, which achieves 92.7% top-5 test accuracy in ImageNet, a dataset of over 14 million images belonging to 1000 classes [9]. The indexes that correspond to dog classification are between 151 to 268, inclusive. That's what we need to make a first dog detector. The process will be repeated with inception model.

The next model will be built from scratch in pytorch. This time, the architecture will be very simple compared to VGG16, but we can use it later to compare our accuracy against the next model.

Lastly, VGG16 will be used again, but this time we will use transfer learning to take advantage of the already trained layers and adjust it to our classification. The last layer will be replaced with a new one that will output 133 dog classes. The last layer is the only one that needs to be trained again, so the process will be faster than in the scratch model.

For the human detection we are going to use Haar feature-based cascade classifier.

Benchmark

The score proposed by Udacity for this project is 10% of accuracy for the scratch model and 60% for the transfer learning model. The goal of this project is to overcome these scores.

There is a dog breed competition at Kaggle [10], so just for reference I will compare the results with the mean score of the public leaderboard, 2.1695 log loss.

III. Methodology

Data preprocessing

All the images in the provided dataset are in different sizes, but our pre-trained model is expecting them to be a fixed size and normalized first. Just resizing all of them won't work, because the ratio of the image is different in each one, too. We might crop it wrongly or the dog could be distorted, changing his visual appearance. We could make the dog fatter or skinnier, so to speak. In order to avoid this problem, first we fix their aspect ratio to 1:1, expanding the shorter axis with black color.

Fixed aspect ratio preprocess

Now, the image is ready to be resized to the desired width and height of 224 pixels. We will make another series of transformations because it's better for the training process if the images show random variations along it. It doesn't give the model the opportunity to "learn" the data, so we prevent over fitting. The transformations are random horizontal flips and rotations. The last step is to translate to tensors and normalise the data.

The preprocess of validation and test don´t have the flips and rotations because it´s not needed and won't affect the final result.

Implementation

A. Person detection

It's a function that reads an image in a path and reduces the red, grey and blue channels to just one luminance channel, so it can be less compute demanding. OpenCv is a very popular computer vision library and provides its own implementation of pre trained Haar Feature-based Cascade classifiers, like face detection, eyes detection and others. It will be used in the last algorithm.

B. Dog detection

The VGG-16 pre-trained model is loaded. It needs the image to meet the same specifications as we described in the previous section for making a prediction: fix aspect ratio, resize to 224 x 224, transform to tensor and normalize. The prediction returns an index of the imagenet1000 classes. If it's between 151 and 268, it's stating that it is a dog. We will need this function in the last algorithm, too.

The same experiment will be repeated with the inception model.

C. CNN from scratch

The defined model has 5 convolutional layers, a pooling layer, two fully connected layers and a last dropout layer, as we can see in the code:

```
Net(
  (conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5): Conv2d(512, 512, 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=25088, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=133, bias=True)
  (dropout): Dropout(p=0.5)
)
```

The convolutional layers will extract the features, the pool layers will reduce the complexity and the dropout prevents over fitting. The loss function that will be used is crossentropy loss and the optimiser will be SGD. The learning rate was 0.02 and the momentum 0.6.

D. Transfer Learning

This time, VGG-16 will be used but without the last fully connected layer. We will maintain the rest of the networks unaltered with their weights and add a new final classification layer. The

output will be 133, our total dog breed classes. Then the model will be ready to train again this last layer.

E. Final algorithm

With the help of the human detector, the dog detector and the transfer learning model, we will build an algorithm that meets the project objective. It will take an image path and if it's a human image, it will predict the resembling dog breed. If there is a dog in the image, it will predict its breed.

Refinement

The scratch model showed signs of overfitting very soon, so I have adjusted the parameters of dropout, learning rate and momentum many times. The results weren´t as what I expected them to be, basing on the results of the Kaggle competition. I tried changing the whole model, but the results were always around 3.7 log loss at 20 epochs and from there, validation started to raise. There wasn't any point in enlarging the epochs because of that.

The last tuned model has had a 16% accuracy and 3.67 loss with 25 epochs.

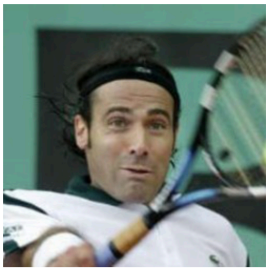
IV. Results

Both the human detector and dog detector threw good results in the tests. With just 100 images of dogs and a 100 human images, the dog detector predicted just 1% of the human images as a dog and a 100% of the dog images were predicted correctly as dogs. Inception had very similar results compared to VGG-16: 1% of dogs in human subset and 100% in the dogs subset.

Both trained models overcame the minimum proposed at the benchmark section. The scratch model has an unacceptable 16% accuracy, so it wouldn't be a serious image classifier in any case.

On the other hand, the transfer learning model reaches a very high score of 82% accuracy and 0.61 loss. It´s not only higher than the 60% proposed, but it´s better than the 2.16 mean result of the Kaggle competition we took as a reference.

There is a human in the picture that looks like a Portuguese water dog



There is a human in the picture that looks like a Ibizan hound

There is a dog in the picture that looks like a Mastiff

There is a dog in the picture that looks like a Affenpinscher

Some samples of the final algorithm results

Reflection

The result of the scratch model makes sense if we take into account that we have an unbalanced dataset, that some classes just had 3 images and that the architecture that I was defining was very simple comparing to the VGG-16. When I continued with the transfer learning model, it clearly showed that this was the way to go in this situation. The dataset was not as big as the one we needed to train a model from scratch, but it was enough to adjust a pre-trained model and obtain reasonable results. It was very revealing that just with 10 epochs we had such a good result.

For me, the project was a very good exercise to follow different CNN models applied to the same problem, even fun.

Improvement

The first thing I would try to improve is the dataset. If there are no more pictures available, the data imbalance has to be fixed. Maybe a good starting point is to remove all the classes with 3 or 4 pictures, because it´s really poor data.

With this fixed dataset, the scratch model can be improved by using a well known results net, such as resnet, and trying to simplify it. Fine tuning hyper parameters and enlarging the epochs could lead to better results.

Finally, the algorithm that wraps these models can contemplate the possibility that there is more than one person and more than one dog in the image. Haar cascade can draw the bounding boxes in the human faces, too, even print the dog breed predicted below it.

With an python WSGI server and this code inside, it can be encapsulated as an api that serves prediction for a mobile app. With the images converted to base64 string in the app, it can be sent easily and read from the api.

References

[1] https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[2] https://en.wikipedia.org/wiki/AlexNet
[3] https://arxiv.org/abs/1409.1556
[4] https://static.googleusercontent.com/media/research.google.com/es//pubs/archive/43022.pdf
[5] https://www.mygreatlearning.com/blog/resnet/
[6] https://www.akc.org/press-center/articles-resources/facts-and-stats/
[7] https://www.hillspet.com/dog-care/behavior-appearance/how-many-dog-breeds-are-there
[8] https://medium.com/@fzammito/whats-considered-a-good-log-loss-in-machine-learning-a529d400632d
[9] https://neurohive.io/en/popular-networks/vgg16
[10] https://www.kaggle.com/c/dog-breed-identification