Christophe BOURGOIN
February 11th, 2020.

Capstone Proposal
Machine Learning Engineer Nanodegree, Udacity

Building a Dog Breed Classifier using Deep Learning

I. Project Definition: Overview, Problem Statement & Metrics

A. Project Overview

The purpose of this project is to build an algorithm that is able to detect & classify automatically the dog breed from a picture. It's a typical problem of computer vision, more precisely it's an image classification task. However, this task of dog breed classification is hard & complex because even for a passionate & specialized human person on dogs it's very difficult to identify perfectly the dog breed.
More precisely, in this project, 2 datasets are provided by Udacity & directly accessible in the Udacity's workspace: a first dataset composed a 13233 human images & a second dataset composed of 8351 dog images. From 2 datasets, I developed a series of models that could be used as part of a mobile or web app & that to perform if a dog or a human is detected in an image and then to provide an estimate of the dog's breed if a dog is detected or an estimate of the dog breed that is most resembling if a human is detected. If neither a dog or a human is detected in the user-supplied image, an error message will be sent.

B. Problem Statement

As explained above, in this project I built a series of algorithms that detect the presence of a dog or a human in an image & then provide an estimate of the dog breed (or the dog breed that is the most resembling if a human is detected). A critical part of this problem is, therefore, to classify correctly the dog breed.

C. Metrics

To measure the performance of my model, I computed the accuracy. In its crash course of machine learning, Google explained, for example, that "Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition":

Accuracy = Number of correct Predictions / Total number of predictions

II. Analysis & Methodology

A. Datasets & Data Preprocessing

I mainly used the second dataset, which is composed of 8351 dog images to build my dog breed classification algorithm. As this dataset is composed of dog images that have different characteristics (size, colors …), I proceed to several data preprocessing.
I resized & cropped each image tp 224x224 pixels.
Moreover, for the sake of performance, I also decided to apply data augmentation & to apply a random horizontal flip on my training data.
Then, I converted each image to a tensor because I wanted to be able to use them in my Pytorch CNN models.
Finally, because of potential differences in the image described above, I also normalized them.

B. Implementation & Refinement

My solution is composed of 2 steps: in a first step, I started to build an algorithm of human face detection with OpenCV & Dlib libraries. My second version of the model from Dlib library performed very well on our human dataset since it successfully detected 100% of human faces in our datasets while reducing the false positive of detected dog face in the dog dataset.
Then I developed a dog detector from a pre-trained model. I used a benchmark model that is called VGG16. VGG16 has been trained on ImageNet, which is a very large dataset used for image classification & other vision tasks. VGG16 model is able to classify an object in a picture in one of its 1000 categories. Thanks to the VGG16 model, my dog detector performs very well because it's able to detect 91% of dogs in the dog dataset.
To sum up, the goal of this step was to be able to detect if a dog or a human is in an image.

In a second step, I built convolutional neural networks to predict & classify dog breed from images. I started to build a **first convolutional neural network from scratch** based on the following architecture : 3 conv2d layers & 2 linear layers. I decided to apply a stride of 2 on the 2 first conv2D layers to accelerate the training of my model, padding of 1 & also to apply a dropout to limit overfitting. Even with a training of 50 epochs & an adaptative optimizer (Adamax), its performance was low (only 26% of accuracy, that is to say of dog breed correctly classified in %).
I then modified my strategy and I decided to used transfer learning to create a **new CNN** that can identify dog breed from images. I used one of the most efficient models on the image classification task, which is called **Resnet50** (Initially I started to try a more complex benchmark model, Resnet152 but I lacked computational resources in the Udacity's workspace even when my batch size was small). I only modified the last layers to adapt the architecture to our classification task of 133 classes. As expected, the performance of this last version of my CNN performed very well (86% accuracy), even with training of 20 epochs. For the sake of clarity, it's useful to recall that for the image classification task, one of the metrics frequently used is accuracy.

III. Results: Performance of my final model vs benchmark model

Relative to my benchmark model - my CNN built from scratch -, my CNN using transfer learning & Resnet50 performs very well. It's a great improvement in the accuracy (86% of

accuracy vs only 26%). For this kind of image classification task, which is hard, it's a great performance.

IV. Conclusion :

In this project, I built an app that determines whether an image contains a human, a dog or neither & then provides an estimate of the dog breed (or the dog breed that is the most resembling if a human is detected). This project was very interesting for me because I built a full pipeline to analyze & classify an image. It's a good project to develop my skills in computer vision.
The building of the dog breed classification was challenging.
The main difficulty in this kind of problem for me was to find the best neural network architecture & to tune the hyperparameters. It's often a long iteration process. In the present project, I proceed manually to tests different versions of architecture & I finally retained a version that performed well.

Finally, my last version of the model performs very well on this hard dog breed classification task. However, I think there are a lot of points for improvement of our algorithm.
Firstly, I think we could improve our process with a better image preprocessing. For example with a more complete data augmentation with vertical flipping or rotation, we should easily obtain better results.
Secondly, we could also try to clean better our images. In this notebook, we paid little attention to the quality of our images (what if an image represents a human & a dog or several dogs).
Thirdly, my CNN architectures don't be very optimized. So with more time to spend to optimize my architecture & to tune my hyperparameters, we'll improve the performance of our classification algorithm. In general, I used the Python library which is called Hyperopt to add an autoML step to optimize my CNN architectures. With hyperopt I can easily try a lot of hyperparameters range, I can test other optimizers or deeper layers or add more layers.

https://github.com/ChrisBg/Machine-Learning-Engineer-Nanodegree-Udacity/tree/images/my-dog-breed-classifier