Machine Learning Engineer Nanodegree, Udacity

Capstone Proposal

Christophe BOURGOIN
February 11th, 2020.

Proposal

Building a Dog Breed Classifier using Deep Learning

I. Project Definition: Overview, Problem Statement & Metrics

A. Project Overview

Even for a passionate & specialized human person on dogs as a veterinary, it's often difficult to identify perfectly the dog breed. A mobile or web app that would be able to determine the dob breed from an image would be therefore particularly useful for a lot of people.
With the incredible development of deep learning & computer vision in the last years, it's now possible to create this kind of app. This is what I propose to do in my capstone project.

More precisely the purpose of this project is to build algorithms that are able to detect & then classify automatically the dog breed from a picture. Such algorithms could then deployed easily in an app.

Using 2 datasets which 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), I will develop 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'll build 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'll compute 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 will mainly use 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 should proceed to several data preprocessing.
Probably I should resize & crop each image to 224x224 pixels.
Moreover, for the sake of performance, I'll apply data augmentation with a random horizontal flip on my training data.
Then, I'll convert each image to a tensor because I want to be able to use them in my Pytorch CNN models.
Finally, because of potential differences in the image described above, I'll also normalize them.

B. Solution statement : Implementation & Refinement

My solution will be composed of 2 steps:

1) in a first step, I'll start to build algorithms to detect human or dog faces.

To build an algorithm of human face detection I'll use OpenCV & Dlib libraries. My second version of the model from the Dlib library should perform very well on our human dataset while reducing the false positive of detected dog face in the dog dataset.
Then to detect the dogs in images, I'll develop a dog detector from a pre-trained model. I used a 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. In our project, I'll use the VGG16 model to identify & detect a dog in an image.
To sum up, the goal of this step will be able to detect if a dog or a human is present in an image.

2) In a second step, I'll develop an algorithm to predict the most probable dog breed from images.

More precisely I decided that I'll build convolutional neural networks (CNN) to predict & classify dog breed from images because the CNNs represent the state-of-art for this kind of image classification task.
In this second step, I'll start to build a **first convolutional neural network from scratch** based on the following architecture : 3 conv2d layers & 2 linear layers. It's a simple & relatively standard architecture for a CNN but in machine learning, it is always a good idea to remember the principle of parsimony.

This first CNN will be therefore my **benchmark model.** As our dog breed classification task is complex, it's probable that this first model performs not very well. However, it will be very useful to track the refinement & improvement that I could then do.

To improve the performance of my algorithm, I'll modify the architecture of my CNN to use transfer learning. As explained by Jason Brownlee in his blog which is called "Machine Learning Mastery", transfer learning is a machine learning method where a model developed for a task is reused as the starting point on a second task[1].
I'll create a **new CNN** that can identify dog breed from images and I'll use one of the most efficient models on the image classification task, which is called **Resnet[2]**
I'll try to apply a deeper & more complex architecture for my second CNN based one of Resnet's architectures. Here below the different Resnet architectures :

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

I should modify the last layers of Resnet to adapt the architecture to our classification problem with 133 classes.
The performance of this last version of my CNN should perform better & obtain a higher accuracy than my baseline model.

C. Project design :

Several iterations will be necessary for my workflow for approaching a solution with a satisfactory performance given our complex task of dog breed classification.
The most critical step in our workflow will be the building of a CNN to predict & classify the dog breed from an image. For this step, I'll start from my baseline model, a CNN that I'll build from scratch. I'll start to try a standard architecture for this CNN but however, I'll try to optimize & tune its hyperparameters to obtain the highest possible accuracy. I'll test different optimizers, learning rates or size of conv2d layers. I'll pay a lot of attention to its training & its validation.
For my second version of CNN, which will have a deeper and more complex architecture, I'll like to test Resnet152 which is the most complex Resnet architecture because our dog

[1] https://machinelearningmastery.com/transfer-learning-for-deep-learning/
[2] *"Deep Residual Learning for Image Recognition"*, He & al (2015), https://arxiv.org/abs/1512.03385

breed classification task is really hard. If it's possible to use this deep architecture in the Udacity's workspace, I'll tune its hyperparameters. However, this kind of architecture requires a lot of computational resources so probably that I should also test a Resnet architecture that is less complex like Resnet50. It'll be not a real problem because Resnet50 should already perform very well on our dog bread classification task.
After having trained & optimized my second version of CNN, the performance of my model in terms of accuracy should be improved. Then I'll test on new images of dogs & humans.

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