Data Scientist For IoT project report

neural network image classifier

Abdullah Alrebdi
382122696

IT351
13042

Abstract

Would it be difficult for you if I showed you a picture to identify what animal that is? unless you don't know that species, it would be the easiest thing that you do in a glance without even thinking. In fact, humans identify things in the subconscious meaning that you don't actually think much. But, that doesn't apply in computers unfortunately.
How do computers work and identify then? A simple question can be answered as "they do what they are told to do". Computers are not that smart to work, build, learn and improve their own. Incointrary, a programmer writes a set of instructions to tell a computer what to do explicitly. However, this method tends to be complex when dealing with high demand tasks like objects identification. Instead, by combining math, statistics and other fields of science we make a computer learn to do or predict things.

Introduction

In this report, I will be showing you the image classifier that I built using neural networks which can identify not only a cat or a dog, NO, it can recognize 10 possible objects/animals.
How does it work entirely, the data that is used to train, procedure and algorithm(neural network) used and evaluation questioning how good my model is?

Image classifier

🔘 Data set

The data set is a set of pictures 32x32 pixiles labeled with a class such as a horse.

In order to train a model perfectly, you should provide an adequate and appropriate number of images to train the model. This is not an easy task especially, if you have a specification on the data set you want. Fortunately for my case, the Internet is full of those, not to mention some libraries that have a perfect number of them with high quality were chosen carefully and that's what I'm going to use. The data set from Tensorflow-keras and each data point is labeled meaning that this specific image, we know its class.

Making the whole data set for only training is not a good practice, we don't want the model to overfit -to be good only on training set-. Picking up the noise from the data is something we want to avoid by splitting the data into two parts traiing_data and testing_data the first one is ofcourse bigger in sake of avoiding underfitting -hasn't learned much yet-. We can also get the label for each for training and evaluation purposes.

🔘 Procedure

Just once an adequate and high-quality amount of data has been obtained and splitted. We make neural networks but it is worth noting that the pictures are represented by pixels; each pixel represents a color. It is important to break down the pictures into smaller pieces and figure out which colors are important -pattern- for recognizing different objects while also specifying how many we have in a picture.

Now the neural networks takes place which is a set of neural (imagine it as a small part that can have a number between 0 and 1) connected layer after layer. Each one gets data from the layer above and then passes it or returns it, to indicate there is something wrong just like saying "correct this". And the final one represents the output, in our case it is the class. After that we make the model learn and save it to save resources (like i did, meaning you don't have to wait until the model training ends) and then the evaluation starts, asking ourselves how good is the model?

⏺️ Evaluation

Two ways I did to evaluate the model, first, the evaluation provided by the library such as accuracy. It gives around 71% accuracy. It is not the best thing but it is very good considering depending only on accuracy is misleading.

Second I pulled some pictures from the Internet that are 32x32 pixels that are one of the following: 'PLANE', 'CAR', 'BIRD', 'CAT', 'DEER', 'DOG', 'FROG', 'HORSE', 'SHIP', 'TRUCK
And surprisingly it identifies all nine of them correctly! I'm not saying it gives 100% accurate results, no, nothing perfect. In fact, the first time I trained the model, it gave me a false identification on a clear picture of a deer. That is probably due to the split function. The first time I built this model, the splitting function split the data in a bad way. Of course if you do the cross validation method you'll get better results but the second time I run the model it gives me fantastic results even on the previous deer sample.

Summery:

I built a classifier using the power of neural networks to identify classes: a car, truck, deer…etc by training dataset of photos provided by the library of the size 32x32 and then clear to the algorithm the pattern of collors which class is which and then evaluate by evaluation measures or by yourself. Just drag the photo you want and test (see README file).

Python code:

It is attached, please read the README file
382122696@qu.edu.sa
Abdullah