Capstone Project
Machine Learning Engineer Nanodegree

Lucas Micol
October 9th, 2020

Dog Breed Classifier

Project Overview

In this project, we will combine models of human and dog recognition in a dog breed classification algorithm, capable of detecting if an image contains a dog and providing its breed, or detect a human and estimate the dog breed that is most resembling.

Problem Statement

The goal of this project is to combine methods of image recognition to solve the classification of dog breeds. The idea is to identify if a photo contains a dog or a human and estimate the breed of dog or the dog breed that most resembles the person.

For this objective, multiple machine learning techniques will be combined into one final algorithm, including Convolutional neural networks[1], OpenCV[2], VGG-16 Model[3] and PyTorch transferred learning models[4]. The expected end result is an algorithm that has an accuracy above 60% and that can be later deployed on an AWS Endpoint.

Data Exploration

To solve this problem we have two datasets, one with photos of humans and other with dog pictures. The formers contain 13233 images that are organized in 5750 folders with the name of that person. The latter contains 8351 images distributed in 133 folders corresponding to the dog breeds.

For both cases the data is not balanced. However, the images are standardized and have the size of 250x250 pixels. Below we present an example in Figure 1 and Figure 2 of those datasets.

[1] https://cs231n.github.io/convolutional-networks/
[2] https://opencv.org
[3] https://neurohive.io/en/popular-networks/vgg16/
[4] https://pytorch.org/docs/stable/torchvision/models.html

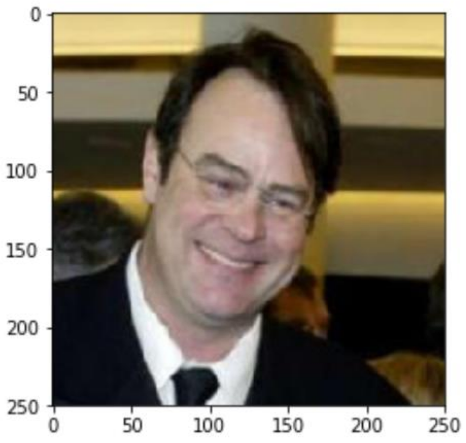

Figure 1: Human photo example

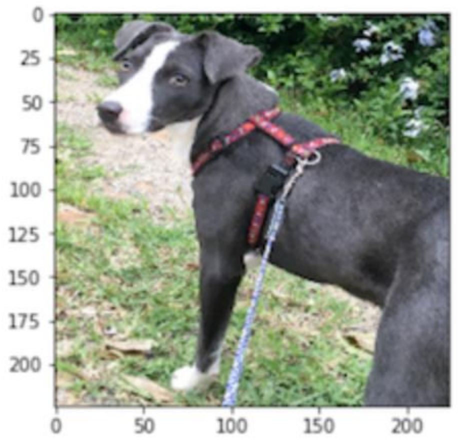

Figure 2: Dog photo example

Evaluation and Benchmark

For this project there will be two implementations. The first being a CNN made from scratch that must reach at least **10%** of accuracy. And the second one being a CNN using transfer learning that need to achieve **60%** or above.

$$Accuracy = \frac{(true\ positives + true\ negatives)}{dataset\ size}$$

In the evaluation part, considering that we have multiple categories, the indicated loss function is the **cross-entropy loss**, also known as log loss.

Development Stages

The main stages of development of this work will be described in this section.

1. **Human detection**
   The first step was to find human faces on images, this part of the project was implemented using the library CV2 from OpenCV. Testing this implementation, I got 98% of accuracy for human faces. However, when testing with the dog dataset I got 17% of false positive.

2. **Detect Dogs**
   For the dog detection I used the pre-trained model VGG-16. In this case I got 100% of accuracy for the dog pictures, while only having 1% of false positives.

3. Create a CNN to Classify Dog Breeds (from Scratch)

   The third step was also the longest. In this part the idea was to build a whole CNN from scratch. So, I ended up trying different approaches to solve this problem.

   The first part was to pre-process the images that are going to run on the CNN. For this, I resized the images to 224x224 pixels and normalized the pictures. This size and normalization metrics are used in other models, as VGG-16 or ImageNet. So, I followed the same first steps. I also used some random transformations to prevent overfitting.

   The second part was the creation of the architecture of the CNN. Here is where I invested more time compared to all the other tasks. Not only finding an ideal architecture to solve the problem was difficult, but tuning its parameters was extremely time-consuming.

   The final result was composed with one convolutional layer that takes the 224x224 image inputs and passes through the neural network, until the final layer produces a 128-size output. I used ReLU as the activation function, and pooling layers (2,2) to reduce dimension. Then the 128 output goes into a two fully connected layers to produce a 133-dimensional output. In between each layer a dropout of 10% was applied to avoid overfitting.

   What we need here is at least 10% of accuracy, with the model described we reached 15% (127/836) within 20 epochs.

4. Create a CNN to Classify Dog Breeds (using Transfer Learning)

   The following objective was to enhance the results by using transferred learning. For that resnet101 was used. Because resnet101 is trained to identify features in images the procedure to get a better accuracy was straight forward.

   First, I imported the model, then replaced the last layer for a linear with 133 outputs, configured the Cross-entropy loss function and the optimizer. And finally, after executing the training, within just five epochs the model reached 68% of accuracy.

5. **Write and test of the algorithm**

   The last step was to put all the algorithms together by writing the function that call all the previous implemented functions.

   Thus, the image to be tested go through a pre-processing where they are adjusted to the ideal size, and then are applied to the dog detector, if it is not recognized, it passes to the human face detector, if the latter also fails, the program shows an error saying that It was unable to identify the image.

Improvement

I believe the algorithm by Transfer Learning can do much better accuracy, I will probably try different trained models and parameters to improve the results.

With the completion of this work, I would like to test my learning by deploying the developed model on an endpoint on AWS. And together with that, host a website on GitHub that communicates with a REST API and a Lambda function on Amazon server to apply an image sent to the neural network and return the result to the user in a web application. However, due to the schedule of the nanodegree, this idea remains as future work.

Conclusion

With the ending of this project I feel much more confident to combine different machine learning techniques in my daily work. This project instigated me to build new projects in the field of machine learning.

References

1. Source code of the project: https://github.com/udacity/dog-project
2. PyTorch documentation: https://pytorch.org/docs/stable/index.html