Udacity Capstone Project
Machine Learning Engineer Nanodegree

Manuel Blanco Murillo
October 12th, 2020

CNN Project: Dog Breed Classifier

Domain Background

Undoubtedly, the development of more powerful computers, together with the rise of technologies such as Machine Learning (ML) or Artificial Intelligence (AI), as well as the large amount of data produced today by all kinds of devices, have fostered the development of all kinds of models and the ML in general.

The concept of Machine Learning coined by Arthur Samuel in 1959 has nothing to do with it, when its applications were quite limited, and where the amount of digital information generated was minimal, compared to the current boom.

One of the applications of ML in today's world is found in the dog breed classifier, which concerns us in this project, and which has been dealt with countless times in ML models. This model tries to identify a breed of dog through an input image, and in the case that the image is of a human, it must identify the most similar breed of dog.

It is about building a pipeline that can process a series of images provided by the user, allowing the model to identify the breed to which it belongs, through an estimate. Given the number of existing breeds, it is a multi-class classification problem, where we will use supervised machine learning, since we are interested in giving an estimate of which breed it belongs to.

In order to interact with this model, I am going to create a web application where the user can enter images, in order to obtain an estimate of the breed to which a dog or human belongs.

It should be said that I have chosen this project because it is very interesting to me, since it could be applied to any other type of images, such as the use of satellite images in order to classify rural or urban areas in them, or in meteorology models, where the types of clouds could be identified, fields in which I am currently working.

Problem Statement

The objective of this project is that the user can enter an image through a web application, and the Machine Learning model can identify what breed of dog it is, or if it is an image of a human, tell us what breed looks more. Which leads us to implement two different tasks:

- *Dog Face Detector*: When entering a dog image, the algorithm must identify an estimate of what breed it belongs to.
- *Human Face Detector*: When entering an image of a human, the algorithm must identify which breed of dog it resembles the most.

These two tasks answer the questions: What breed of dog is it? and What breed of dog does this human resemble ?, which in turn must first answer if there is a dog or a human in the image, in order to continue with the other questions.

As I said before, this is a supervised learning problem, in which we try to assign the dog to one of the classes (breeds) that we have, so we will use a predictive classification model, such as the multi-class predictive model.

Datasets and Inputs

The type of input data used in this project will be images, since it is through this image that the application will classify the breed to which the dog belongs. To do so, we will have a dataset provided by Udacity, with images of dogs and humans, with which we will train and test the model.

- Dog Images Dataset: There are a total of 8351 images of dogs, which are arranged in directories of train (6680 images), test (836 images) and valid (835 images). Each of these directories has 133 folders, corresponding to the different breeds of dogs. Since we must have a great variety of images, because we do not know what the user's image will be like, we have very varied images in terms of size, background, color, brightness, etc. Given that the images provided for each breed vary in number, the data is not balanced, since we have breeds with 8 images and others may have only 4.

- Human Images Dataset: The dataset for human images contains 13233 files, ordered by human names (5750 folders). Here the size of the images is the same, 250x250, although they have different backgrounds and shooting angles. As in the dog dataset, the data is not balanced, as we have one image for some people and many for others.

Perhaps, a way to balance the data could be eliminating images of the same breed type, both for dogs and humans, in order not to have some breeds with many images and others with very few, the data would be equalized and they would be seen if they change the results obtained.

Solution Statement

To perform this multi-class classification, we will use Convolutional Neural Network (CNN), which is a Deep Learning algorithm that will take an input image and assign it a weight and bias to different aspects of the image, which will differentiate it from another. This is a great option when analyzing images.

To detect images of humans, we are going to use algorithms like those of the OpenCV model. To detect the dogs in an image, we can use a VGG16 pre-trained model. Finally, once the image is classified as human or dog, it will be passed to a CNN that will process the image and predict the breed that matches the best of 133 possible.

We will create our CNN model using transfer learning because we need far fewer images this way, without giving up good results.

Benchmark Model

We will use Convolutional Neural Networks (CNN) created from scratch with an accuracy of more than 10%. This will confirm that it is working because a random assumption would be 1 in 133 breeds, which is at least 1%, without considering the unbalanced data for the dog images.

This CNN model created by transfer learning should have an accuracy of 60% or better.

Evaluation Metrics

Since what we are trying to solve is a classification problem, we will use the Multi class log loss. Since there is imbalance in the data set, precision is not a good indicator here for measuring performance. Log loss takes into account the prediction uncertainty based on how much it varies from the current label, which would help to evaluate the model.

Project Design

When designing and executing our project, we will follow a series of steps:

- Step 1: Import the necessary information from dataset and libraries to start the project. The data will be pre-processed, and even if necessary, an image will be cleaned and deleted if you want to balance the data. With this pre-processing we can create train, test and validation dataset.

- Step 2: We will use OpenCV’s implementation of Haar feature based
- cascade classifiers to detect the faces and see if they are human or not, the flow to detect them being the following:

    - Initialize the pre-trained face detector
    - Load the image to analyze
    - Convert the image to grayscale
    - Look for faces in the image
    - Returns a true boolean if the number of heads is greater than 0, false otherwise

- Step 3: We create the dog detector, for which we use the pre-trained model VGG16, the flow being:

    - Define the VGG16 model
    - Use GPU
    - Load and pre-process the image to be analyzed
    - Send an image to the model
    - The model returns an index between 0 and 999
    - Returns true if the index is between 158 and 268

- Step 4: We created a Convolutional Neural Network to classify the breed of dogs from scratch. Our data is already divided into train, test and validation, so we can use it in the model.

- Step 5: When we get an accuracy above 10%, we build a new CNN model using Transfer Learning with resnet101 architecture. We train and test the model. We can build the model with less information to get better results.

- Step 6: We write an algorithm to combine the dog and human detectors. Following this flow:

    - If a dog is detected in the image, the breed is returned
    - If a human is detected in the image, the closest breed is returned

- If it does not detect any of the above, an error will be indicated in the output

References

https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification

https://en.wikipedia.org/wiki/Convolutional_neural_network

https://www.kdnuggets.com/2018/04/right-metric-evaluating-machine-learning-models-1.html

https://pytorch.org/docs/master/

http://wiki.fast.ai/index.php/Log_Loss

https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

https://neurohive.io/en/popular-networks/vgg16/

https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html