Gojko Galonja
22.04.2020.

CAPSTONE PROPOSAL

Convolutional Neural Networks

Project: Write an Algorithm for a Dog Identification App

Domain background

The main goal of this project will be to build a model that can detect and classify dogs and humans. If a dog image is given as an input to the application, the purpose of the model is to classify the breed of a dog. If a human image is given as an input, then as an output, the user should get the resembling dog breed corresponding to the image of the dog.
In this project, Deep learning, more specifically Convolutional Neural Networks technology will be used in order to finish and execute the given problem.
Using larger quantities of images as our main source of data, we will create an image classification algorithm that will give us certain output.
This project is quite known in the Machine learning world. It's been used and practiced in universities, bootcamps and courses, and with a good reason - the value after finishing this problem will be great, as a lot of knowledge will be gained.

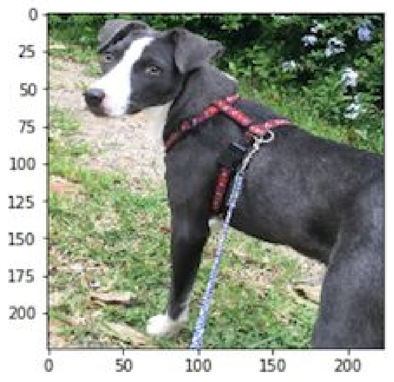
I am mostly passionate about the computer *vision* area of deep learning, so I chose this project. On top of building a CV related model, I will get to deploy it to production, which is another great knowledge accumulation for the real-world things I am working on right now.

Problem - Why, What, How?

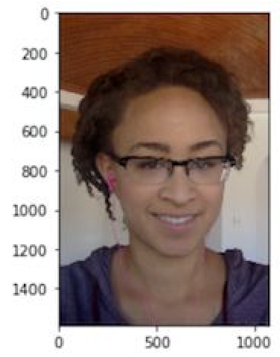
The goal is to build a web or mobile application that will detect whether a dog or a human face is detected in a given image.
The dataset used for training consists of two classes - dogs and humans.

We are building this application in order to detect if a dog is on the image, and if so, which breed is it. If the human face is detected, then try to figure out which dog breed the human face is similar to. Human face detection will be done with OpenCV technology, more specifically with its builtin Haar Cascade algorithm.
The reason behind the question why we are building this is simply to detect dog breed and on the human face part, to bring some fun in it.
In order to build this algorithm, **supervised learning** methods will be applied.
Below are the examples of both outputs of the detection and classification.

```
hello, dog!
your predicted breed is ...
American Staffordshire terrier
```



```
hello, human!
```



```
You look like a ...
Chinese_shar-pei
```

Datasets

As stated, the dataset will consist of two classes - human and dog images.
The number of human images is 13233 and the number of dog images is 8351.
The testing images are real-world, user-like images that mimics the real input of the users of the application.
The dataset is already contained in the Udacity data folders.

1. Dog images folder - **dog_image**

    - Dog image folder consists of train, valid and test folder with each containing 133 dog breed folders with it's own pictures in it

    - Number of dog pictures in general is 8351, so roughly the number of images across the dog breed folders can be 62.78 which represents mean value

    - Distribution between Train,Test and Validation sets is:
        - Train - 6680 images, which makes 80%
        - Test - 836 images, which makes 10%
        - Validation - 835 images, which makes 10%

2 Human images folder - **/lfw**

- Human image folder consists of names of people containing their image (one per each person/folder)

- Number of images in human folder in total is 13233

The solution proposal

Key of solving this problem will be in applying CNN architecture in order to make a model that will do the detection/classification. First, the model will be created from scratch. Then another model will be created using transfer learning technique to create another model.
The reason we use transfer learning is because the dog dataset is insufficient in order to create a model that will give good dog breed prediction.
VGG-16 model is a state-of-the-art model that is trained on an exponentially larger dataset.

Benchmark

Via **face detection** function we can determine how many faces occur in both human faces and dog breed folders.
With this method we raise **two questions**:

1. What percentage of the first 100 images in human_files have a detected human face?
2. What percentage of the first 100 images in dog_files have a detected human face?

Model trained from scratch needs to meet requirements of minimum 10% accuracy.
Model trained with transfer learning needs to meet requirements of minimum 60% accuracy.

"We also mention that random chance presents an exceptionally low bar: setting aside the fact that the classes are slightly imbalanced, a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%."

(*** *Questions and quoted sentence are taken from the Udacity* ***Dog Project Workspace*** ***)

Evaluation

To solve previously stated questions, we iterate through 100 images in each folder and find out if there are errors in detection.
By running **face_detector** function we get next results:

1. Number of human faces detected in the human files is **98, out of 100**.
2. Number of human faces detected in the dog files is **17, out of 100**.

This approach shows few inaccuracies, which results in **error** involved in detection.
Evaluation metric used to evaluate the accuracy of the model will be.

For evaluation of the model, function **test** will be created, which will perform following segments:

- Monitor
- Test loss
- Check accuracy

Mainly focusing on the **Mult-Class Log Loss** evaluation method, because we have an imbalanced dataset, so logging the loss will give us insight of the result difference between prediction and the actual label.

Design

It will be broken down into the following steps:

- Create human face detector / Detect humans - Use OpenCV Haar Cascade which returns positive if human face is detected and negative on the opposite

- Create dog detector - Use VGG-16 (pretrained model) in order to detect the dogs

- Write three separate data loaders for the training, validation, and test datasets of dog images (located at dog_images/train, dog_images/valid, and dog_images/test)

- Create a model from the scratch
    - Create a CNN model architecture to classify dog breed
    - Specify Loss Function and Optimizer
    - Train, Validate and Test the Model

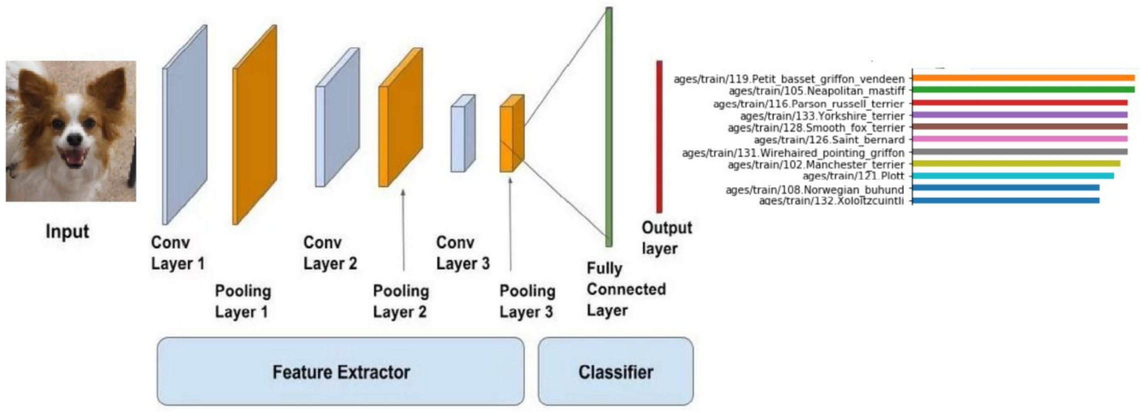
- Create a model with **Transfer Learning** technique (With VGG-16 pretrained model) to train new model
    - Create a CNN model architecture to classify dog breed
    - Specify Loss Function and Optimizer
    - Train, Validate and Test the Model

- Predict the dog breed / Test the model **accuracy**

- Write an algorithm that will:
    - if a **dog** is detected in the image, return the predicted breed
    - if a **human** is detected in the image, return the resembling dog breed
    - if **neither** is detected in the image, provide output that indicates an error

- **Test** algorithm on sample images

Visual representation of CNN network applied for dog breed classification (*** *Image taken from Google images search* ***):



Input
Conv Layer 1
Pooling Layer 1
Conv Layer 2
Pooling Layer 2
Conv Layer 3
Pooling Layer 3
Fully Connected Layer
Output layer
ages/train/119.Petit_basset_griffon_vendeen
ages/train/105.Neapolitan_mastiff
ages/train/116.Parson_russell_terrier
ages/train/133.Yorkshire_terrier
ages/train/128.Smooth_fox_terrier
ages/train/126.Saint_bernard
ages/train/131.Wirehaired_pointing_griffon
ages/train/102.Manchester_terrier
ages/train/121.Plott
ages/train/108.Norwegian_buhund
ages/train/132.Xoloitzcuintli
Feature Extractor
Classifier