Udacity Capstone Project
Machine Learning Engineer Nanodegree

Manuel Blanco Murillo
October 16th, 2020

CNN Project: Dog Breed Classifier

Project Overview

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.

Metrics

The data is split into train, test and valid dataset. The model is trained using the train dataset. We use the testing data to predict the performance of the model on unseen data. We will use accuracy as a metric to evaluate our model on test data.

This is a classification problem, because our data is an unbalanced, simple accuracy score is not very good here. For example, on Kaggle they use multi-class log loss metrics to evaluate models. I will also use this metric so I can compare it to results on Kaggle. I will also use Accuracy score testing because it considers precision and recall and it is easier for me to understand results.

Accuracy = Number of items correctly classified/ All classified items

Also, during model training, we compare the test data prediction with validation dataset and calculate Multi class log loss to find the best performing model. Log loss takes into the account of uncertainty of prediction based on how much it varies from actual label and this will help in evaluating the model.

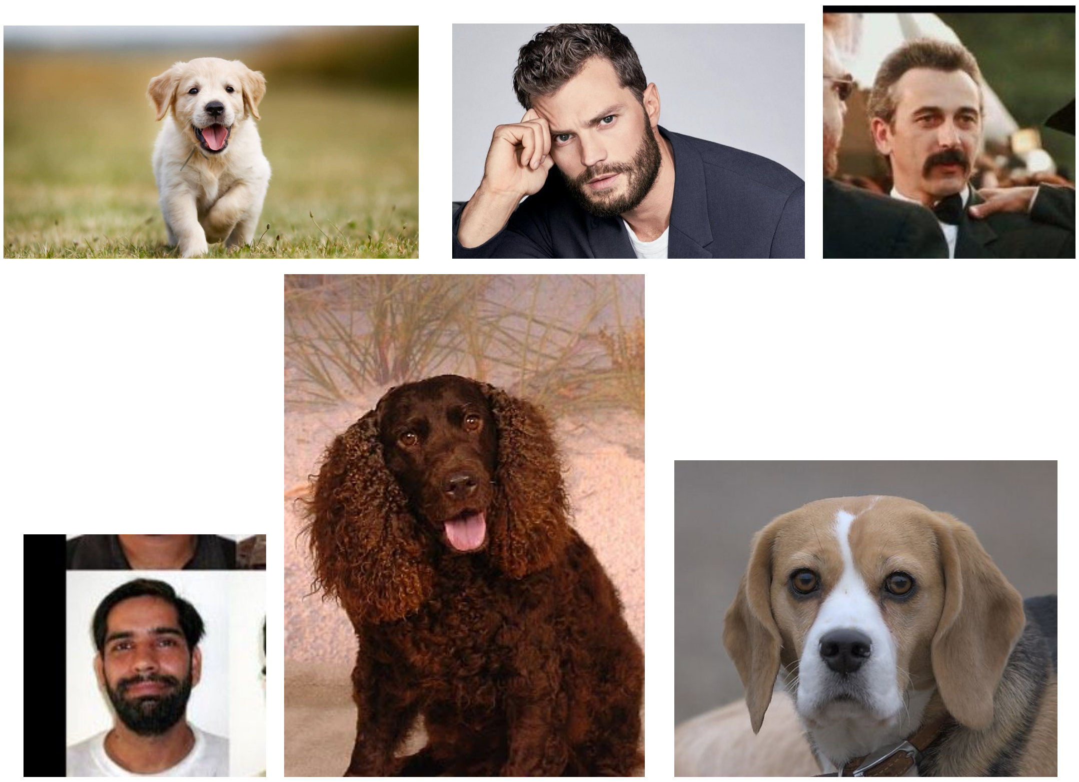
Data Exploration

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.

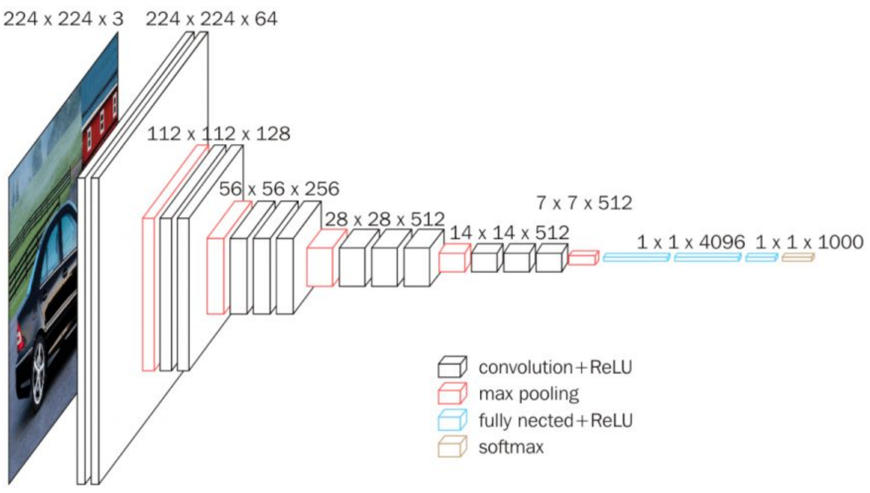
- *Extra Images*: In order to better test the model, I have included in this project a series of images of humans, dogs and other objects. In this way, we will not only analyze if the algorithm predicts well humans / dogs, but also if it is capable of detecting that there are images that do not contain either of the two elements, and therefore, the model warns us of this. I have included a folder called test_images that contains 14 files(3 humans, 3 dogs, 3 cats, 3 ducks and 2 houses) with these images(source: https://www.freeimages.com).


Example images from the datasets.

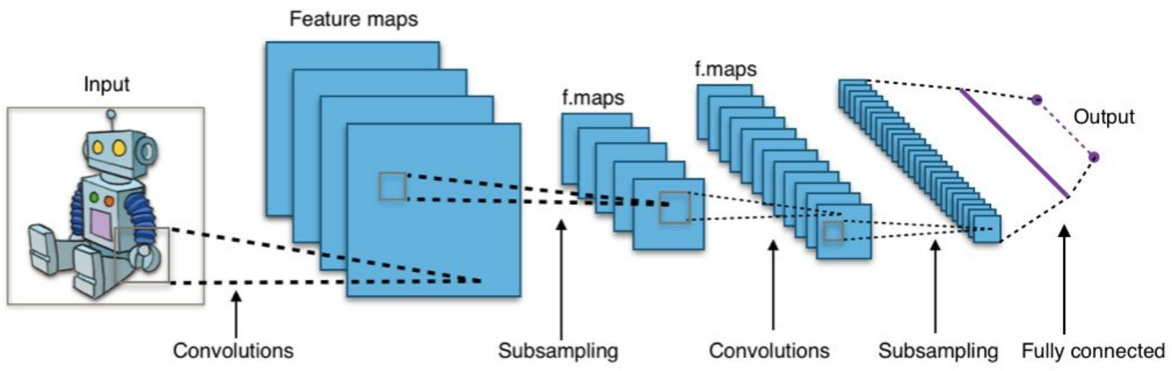
Algorithms and techniques:

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.



VGG-16 model (source: https://neurohive.io/en/popular-networks/vgg16/)



CNN architecture (source: https://en.wikipedia.org/wiki/Convolutional_neural_network)

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

Benchmark

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.

Data Preprocessing

All the images are resized to 224*224, then normalization is applied to all images (train, valid and test datasets). For the training data, Image augmentation is done to reduce overfitting. The train data images are randomly rotated and random horizontal flip is applied. Finally, all the images are converted into tensor before passing into the model.

Implementation

To solve this problem, a CNN model has been built from scratch, which is composed of 3 convolutional layers. All layers have a kernel size of 3 and stride 1. The first conv layer (conv1) takes the 224*224 input image and the final conv layer(conv3) produces an output size of 128.

I used the ReLU activation function and I used the pooling layer of (2,2) in order to reduce the input size by 2. The dimensional out is 133, produced for the two fully connected layers that we have here. A dropout of 0.25 is added to avoid over overfitting. After creating a model, we train and test it to meet the specification of a test accuracy of at least 10%.

Refinement

Once an acceptable precision is obtained from the CNN that I have created from scratch (14%), these results can be improved, for which we will use the transfer learning. Our CNN must attain at least 60% accuracy on the test set.

I will use Resnet101 architecture which is pre-trained on ImageNet dataset, the architecture is 101 layers deep. The last convolutional output of Resnet101 is fed as input to our model. We only need to add a fully connected layer to produce 133-dimensional output (one for each dog category). The model performed extremely well when compared to CNN from scratch. With just 5 epochs, the model got 83% accuracy.

Model Evaluation and Validation

The results obtained in this project can be summarized according to the different phases of the project:

- *Human Face Detector*: The human face detector function was created using OpenCV's implementation of Haar feature based cascade classifiers. 99% of human faces were detected in the first 100 images of human face dataset and 14% of human faces detected in first 100 images of dog dataset.

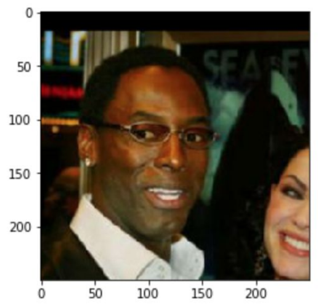
- *Dog Face Detector*: The dog detector function was created using a pre-trained VGG16 model. 94% of dog faces were detected in the first 100 images of dog dataset and 0% of dog faces detected in first 100 images of human dataset.

- *CNN Using Transfer Learning*: The CNN model created using transfer learning with ResNet101 architecture was trained for 5 epochs, and the final model produced an accuracy of 83% on test data. The model correctly predicted breeds for 680 images out of 836 total images.

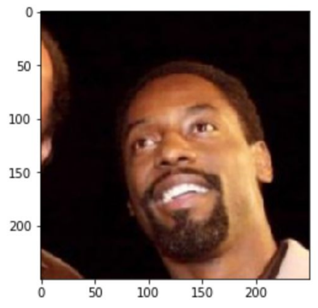
Accuracy obtained in the test data for the model: **83%** (695/836)

Here are some examples of model output:

```
Hello Human!
Predicted breed:  American water spaniel
```



```
Hello Human!
Predicted breed:  Dogue de bordeaux
```

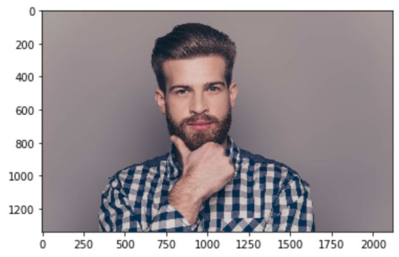


```
Hello Dog!
Predicted breed:  Poodle
```

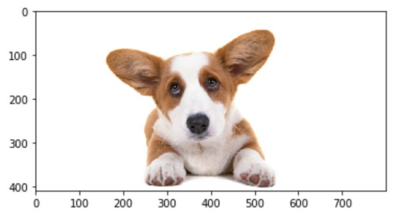
```
Hello Dog!
Predicted breed:  Poodle
```

Example of model output with the images provided in the project. All correctly identified.

```
Hello Human!
Predicted breed:  German shorthaired pointer
```

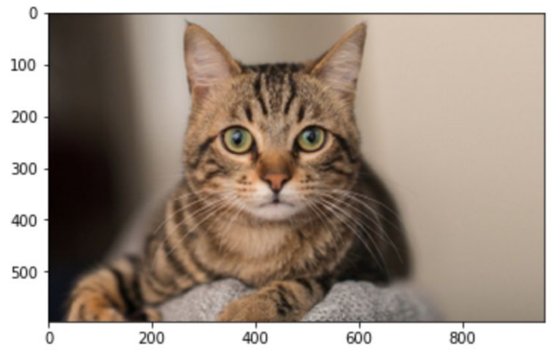


```
Invalid image
Invalid image
Hello Dog!
Predicted breed:  Cardigan welsh corgi
```



Example of model output with my test images. Some images are not recognized as humans or dogs, so it warns us that it is an invalid image.

```
Hello Human!
Predicted breed:  Cardigan welsh corgi
```



Example of a misclassified image, where the model has detected some human features and has classified it as such.

Justification

The results obtained show how the performance of the model has exceeded expectations. While in the model created with Transfer Learning the precision obtained is 83% (695/836), in the CNN model created from scratch, the precision is barely 14% (123/836). A big difference that can also be improved with greater refinement when adjusting the model.

Improvement

Although the results have been satisfactory, the model could be improved by adding more data, both to the training and to the test, which could also mitigate other added problems, such as balanced or unbalanced data, capable of influencing the final result. . In this project 133 dog breeds have been used, but increasing the breeds and the number of images per breed, would obtain more consistent results. Also, by performing more image augmentation, we can avoid overfitting and improve the accuracy.

In this case, it used only the ResNet 101 architecture for feature extraction, perhaps it would be interesting to try a different architecture in order to obtain an improvement in the results. Some of them could be ResNet - 50 or Inception v3, which in a more comprehensive exercise could be used and the results compared with ResNet 101.

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

https://www.freeimages.com