Capstone Project

Dog Breed Classifier

Bousbiat Hafsa
Hafsa.bousbiat@gmai.com

Data Science Nano Degree
Udacity Learning Platform

March 11, 2022

Abstract

The current manuscript investigates the problem of Dog Breed recognition in images. In the scope of this problem, instances from different classes share common parts but have wide variation in shape and appearance. This domain is especially challenging since the appearance of corresponding parts can vary dramatically, e.g., the faces of bulldogs and beagles are very different.

We explore different CNN approaches and show case the power of transfer learning in solving such challenging problems. An accuracy of 80% was achieved where the transfer learning from the ResNet50 improved the performance with a factor of 10. The finale recognition algorithm is deployed on a flask web application for a user-friendly interaction.

Convolutional Neural Networks, Image classification, Transfer Learning

Contents

1. Introduction

1.1 Project Overview

Images and photos are omnipresent in the digital era and are yet to increase with the large popularity that social media is gaining. Thus, the images are becoming an important information support not only in daily life but in many scientific fields also. Animal species preserving for example rely in huge part on recording images of different animals and following their development. It is thus of paramount importance to develop models that provide acceptable performance in recognising an detecting different species and breeds. Traditional image processing algorithms used to rely on hand generated features which made it a hard problem to solve. Nonetheless, as the hardware industry thrived throughout the years, several deep models have seen the light. These models do not only offer automatic feature learning with gradient descent algorithm but also allow to transfer learning and use the learned features on different problems.

In this project, Convolutional Neural Networks are used to detect and recognise Dog's breed in images. The power of transfer learning is demonstrated through the use of several state-of-the-art models. The ResNe50 model provided the best accuracy and was deployed in a flask web-application.

1.2 Problem Statement

The problem tackled in the current manuscript is the "Dog Breed Classification". The goal is to end up with a final application where the user can upload an image and the model will provide him with the appropriate class. The project will take into consideration only the following scenarios:

1. **The input image contains a human**: In this case, the model directly suggests that a human face is present in the image.
2. **The input image contains a dog**: When the image contains a dog. The image is forwarded to the breed classification model that will return the breed of the dog.
3. **The input image does not contain neither a human nor a dog**: In this case the model should be properly able to state that the image contain no faces and no dogs.

The previous scenarios thus require the combination of several models in order to achieve the goals of our project. The following steps have been made in order to develop an appropriate algorithm that would allow the user to easily detect the presence of a dog or a human face in an image:

1. The data provided by Udacity was downloaded and analyzed to understand the shapes of the images as well as the distribution of the number of images per class to avoid biased models.
2. The human face detector is then designed using cascade features and open CV2.
3. The dog face detector is built using the pre-trained RESNET model. The main goal of this detector is to determine if a dog is present or not in an image.
4. The final part of our algorithm is the dog breed classification model. To develop this model, we explore similar alternatives. First, we build a CNN model from scratch and evaluate its performance on the data set. Second we use pre-trained models, publicly available in keras, and evaluate their performance for the task of dog breed classification. We compare between all these models and the best performing model (with the best accuracy) is saved for deployment.
5. We build a flask web application that allow a user to upload an image and return the prediction of our algorithm.

Figure 1. A sample image from the human dataset.

2. Background

2.1 Dataset

On the other hand, dog's breed dataset can be described as follows:

1. Size of the Dataset
    - 133 total dog categories.
    - 8351 total dog images.
2. Size of data after splitting
    - 6680 training dog images.
    - 835 validation dog images.
    - 836 test dog images.

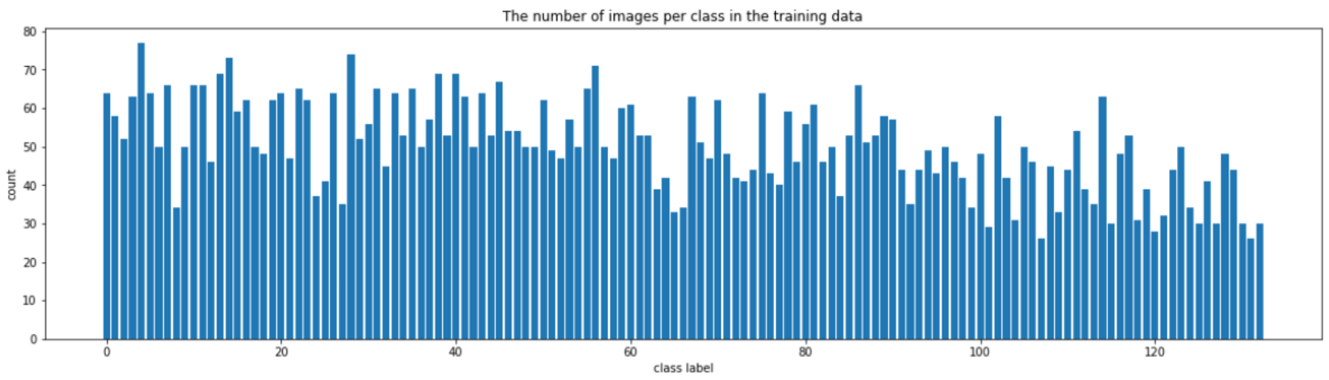


Figure 2. The distribution of images across classes

The training data is more or less homogeneously distributed across the 133 classes and therefore we don't need to fix unbalanced classes. This would mean that using the accuracy to evaluate the model would not be a problem.

2.2 Metrics

The metric used during the current project is the accuracy which is given by the following formula:

$$accuracy = \frac{the\ number\ of\ correctpredictions}{the\ data\ size} \tag{1}$$

The accuracy is appropriate to be used in the case of the current problem since the dataset contains a more or less balanced distribution of images across different classes.

3. Methodology

3.1 Pre-processing

Since we rely on the combination of several models to solve the problem at hand, each part would have different pre-processing steps depending on the requirements of the model used.

- **Face Detector** Before using the face detector algorithm, it is standard procedure to convert the images to grayscale and then use the cascade classifier pre-downloaded.
- **Dog Detector and Breed detection** both of these algorithms use the ResNet50 model. Thus, the images are reshaped and normalised accordingly. We first load the image and resize it to a square image that is 224×224 pixels. Next, the image is converted to an array, which is then resized to a 4D tensor.

3.2 Implementation

3.2.1 Face Detector

We use OpenCV's implementation of Haar feature-based cascade classifiers[1] to detect human faces in images. OpenCV provides many pre-trained face detectors, stored as XML files on github[2]. The provided model resulted in the following:

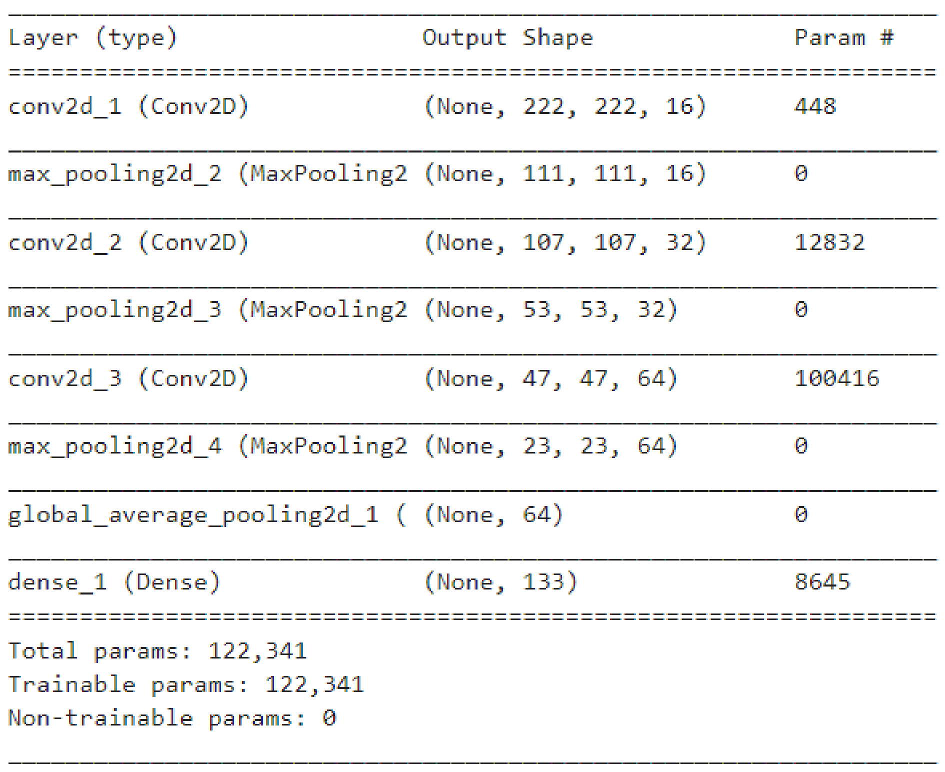
- The percentage of the first 100 images in human files, that have a detected human face, is: 100%
- The percentage of the first 100 images in dog files, that have a detected human face, is: 11%

This algorithmic choice necessitates that we communicate to the user that we accept human images only when they provide a clear view of a face (otherwise, we risk having unnecessarily frustrated users!). However, expecting the user to always provide a clear image is not reasonable and will make them abstain from using the app. It would rather be more efficient to take this into consideration during the development of the model. We can use instead a CNN model that is already trained on an augmented data containing a portion of unclear images(e.g., adding blur), which will allow to algorithm to detect even unclear images. Nonetheless, we will stick with the cascade classifier for our algorithm since the most important task is to identify the dog's breed.

3.2.2 Dog Detector

we use a pre-trained ResNet-50[3] model to detect dogs in images. We download the model along with weights that have been trained on http://www.image-net.org/, a very large, very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories[4]. Given an image, this pre-trained ResNet-50 model

[1] http://docs.opencv.org/trunk/d7/d8b/tutorial$_{py_face_detection.html}$
[2] https://github.com/opencv/opencv/tree/master/data/haarcascades
[3] http://ethereon.github.io/netscope//gist/db945b393d40bfa26006
[4] https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 16)      448
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 107, 107, 32)      12832
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 53, 53, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 47, 47, 64)        100416
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 133)               8645
=================================================================
Total params: 122,341
Trainable params: 122,341
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. The architecture of the model

returns a prediction (derived from the available categories in ImageNet) for the object that is contained in the image. In order to check if an image is predicted to contain a dog by the pre-trained ResNet-50 model, we need only check if the class returned by the model is between 151 and 268 (inclusive). This model resulted in the following:

- The percentage of the first 100 images in human files, that have a detected dog, is: 0%
- The percentage of the first 100 images in dog files, that have a detected dog, is: 100

3.2.3 Breed Classification

The breed classification is the main problem that we are interested in investigating. Therefore, we consider different alternatives for this part, as follows:

1. **A CNN from scratch** we use in this first step the recommended architecture provided by Udacity. This architecture is based on convolutional layers that excel in image recognition tasks. The architecture is composed of three successive convolutional layers followed by a averaging layer and a dense layer with a sigmoid function.
2. **Transfer Learning** In the second step, we use transfer learning to create a CNN using bottleneck features from different pre-trained models. To make things easier, we use pre-computed features for all of the networks that are currently available in Keras for the following models: VGG-19[5], ResNet-50[6] and VGG16[7].

[5]https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogVGG19Data.npz
[6]https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogResnet50Data.npz
[7]https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogVGG16Data.npz

4. Evaluation

4.1 Classification performance

At first all of the considered models were trained only for 20 epochs to have a first idea about their performance with a learning rate of 0.001. Unfortunately, the accuracy obtained with the CNN developed from scratch is very low (around 7%) and show that this model will not perform properly if we use it in our application.

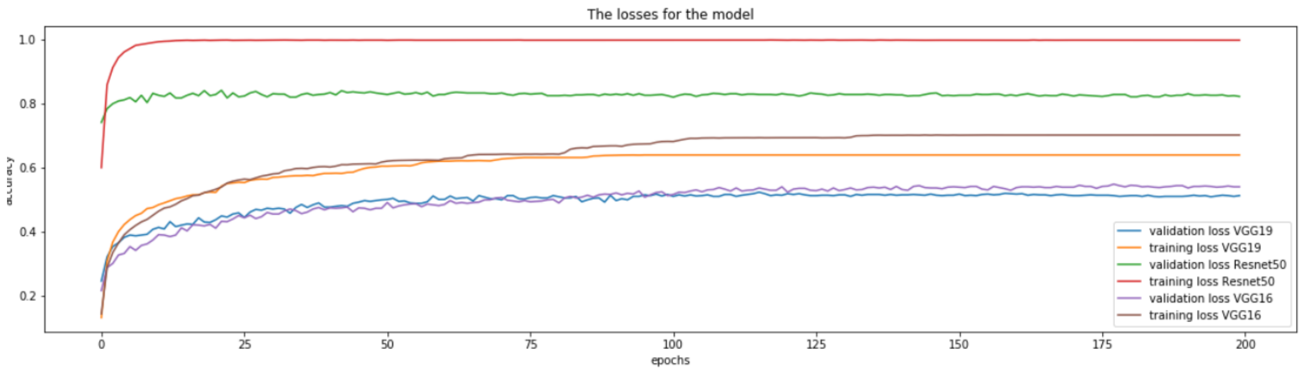


Figure 4. Training and validation losses for different models.

On the other hand, using transfer learning from different models shows that after only 20 epochs a minimum accuracy of 40% on validation data can be achieved. The plots of the losses shows that the ResNet50 have approximately a constant validation loss after the first few epochs. On the other hand, the VGG16 and VGG19 have an increasing losses that seem to still are improving. This ResNet50 demonstrated an accuracy of 81% on testing data. This last observation, encouraged the author to trained for more epochs, precisely 200 epochs, showed in the previous plot.

4.2 Computational Performance

The three different pre-trained models were evaluated and the one providing the best validation loss was selected. However, this model has the highest number of parameters and thus is more computationally consuming than the two other models. In real scenarios pruning techniques can be used to reduce the size of the model and enhance the performance. The use of pruning techniques is however out of the scope of the current project. We will thus select the model only based on best validation accuracy.

5. Conclusion

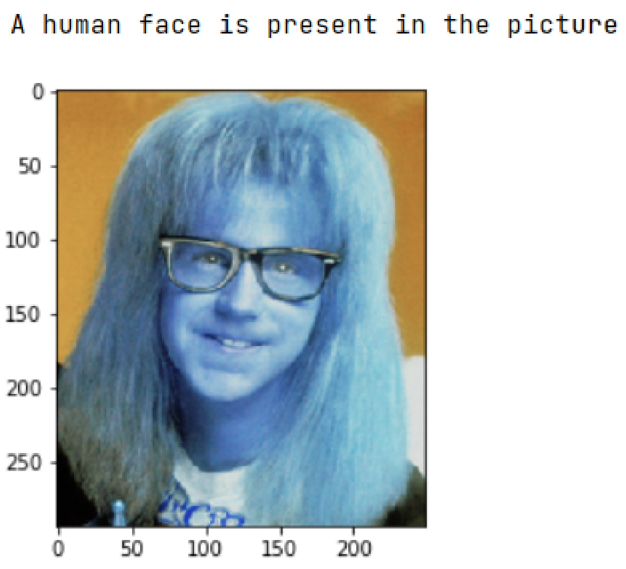
5.1 Sample of predictions

An example of the generated predictions by the developed model are presented in the following:

5.2 Potential Improvements

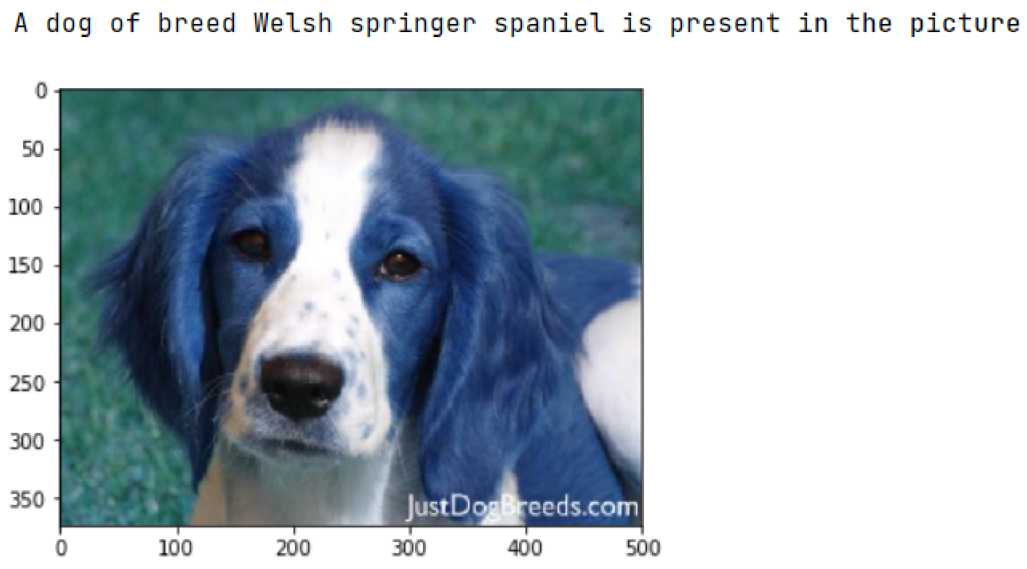
The results obtained are impressing. However, further improvements are still possible using:

1. The human detector can be improved using a CNN model.
2. Data augmentation: the images from each class can be augmented and this will improve the performance of the model.
3. Cross validation would allow to better assess the generalisability of the model



```
A human face is present in the picture
```

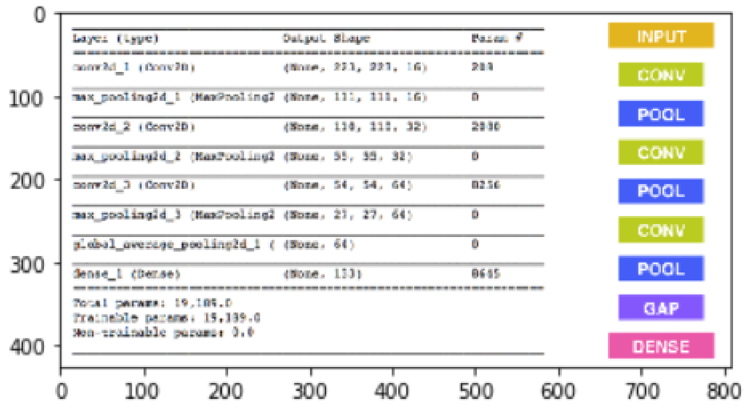
Figure 5. Example of prediction for scenario 1



```
A dog of breed Welsh springer spaniel is present in the picture
```

Figure 6. Example of prediction for scenario 2

```
The picture does not contain a human neither a dog face
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 110, 110, 32)      2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 54, 54, 64)        8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 133)               8645
=================================================================
Total params: 19,189.0
Trainable params: 19,189.0
Non-trainable params: 0.0
_________________________________________________________________
```

Figure 7. Example of prediction for scenario 3