Report: Write an Algorithm for a Dog Identification App

Carolina Abs da Cruz

```
Human detected :-) and the resembling dog breed is: Dogue de bordeaux
```

I. Overview

The main goal of this project is to develop an algorithm that could eventually be used as part of a mobile or web app. The code accepts any user-supplied image as input. If a dog is detected in the image, it provides an estimate of the dog's breed over 100 dog breed categories. In case a a human face is detected, it provides an estimate of the dog breed that is most resembling. I've utilized the Udacity training space with Pytorch which was GPU-enabled to allow for the models to run quickly.

II: Problem Statement

The main goal of the project is to use a convolutional neural network (CNN) to predict dog breeds. Initially, a CNN model from scratch was built but the results were poor (11% accuracy). Since image recognition requires more complex feature detection the results were acceptable.

A transfer learning approach, using a p**re-trained VGG-16 Model¶**, was a much more accurate approach with 85 % of accuracy in only 5 epochs!

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources

required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

The end goal of the project is to provide a predicted breed for dog images and for humans, the function will tell you which dog the human most resembles.

III. Project Structure

The project is divided in the following steps:

- Step 0: Import Datasets
- Step 1: Detect Humans
- Step 2: Detect Dogs
- Step 3: Create a CNN to Classify Dog Breeds (from Scratch)
- Step 4: Create a CNN to Classify Dog Breeds (using Transfer Learning)
- Step 5: Write your Algorithm
- Step 6: Test Your Algorithm

Step 0: Import Datasets

The files containing the dog images and the list of dog breed categories were provided by Udacity. In addition, some human faces were provided to test the algorithm.

There are in all 8351 dog images with 6680 of them being training images and more than 10 000 human faces.

A major project development stage was the data pre processing. All the images were reshaped to 224 in order to be used in both pre trainned and from scratch models.

Moreover, the trainning dataset was augmented. Image data augmentation is a technique that is used to increase the size of a training dataset by creating modified versions of images in the dataset.

The augmentation techniques produce variations of the images that can affect the ability of the fit models to generalize what they have learned to new images.

All datasets were normalized. The pixel values in images must be scaled prior to providing the images as input to a deep learning neural network model during the training or evaluation of the model. The standard normalization of mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225] have been used.

And all images have been converted to tensor what improves performance computation ;-) I spent most of the time here since I got many errors trainning the model...

Step 1: Detect Humans

OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images will been used. Thet percentage of the first 100 images in `human_files` having a detected human face will be calculated.

Here a generalized funtion count_detection_type has been created in order to be used with detect dogs and avoid code ducplication (both tasks are very similar).

```
number of faces detected in 100 human faces inputs =  98 which corresponds to 98  % of performance in this short data
set
number of faces detected in 100 dog picture inputs =  17 which means a bad performance if this value is greater than
zero...
```

Some dogs were identified as humans :-) what means interesting results in the last section.

Step 2: Detect Dogs

Among the transfer learning methods, the VGG-16 model hasl been used, along with weights that have been trained on Imagenet, 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. The percentage of the first 100 images in the dataset having a detected dog face will be calculated.

```
number of dogs detected in 100 human faces inputs =  0
number of dogs detected in 100 dog picture inputs =  100
```

The results are really good fot the dog detector, at least for the first 100 images.

Step 3: Create a CNN to Classify Dog Breeds (from Scratch)¶

Once the functions for detecting humans and dogs in images are ready, the dog breed from images has been be predicted. In this step, a CNN algorithm that classifies dog breeds has been created *from scratch* in order to get at least 10% in the test accuracy

This task was very interesting. The idea in this section is to develop a CNN model as simple as possible to avoid performance issues to get a minimal accuracy of 10 %. I did some searchs in internet about the subject and there are lots of differents solutions.

In my configuration I decided to have only 3 convolulation layers and 2 FC layers. I am obsiously using a Relu function and maxpool layers between the convolution layers.

I had some performance issues so fixed it was a **major project development stage.** I decided to use a stride of size 2 and it worked very nicely! It helped a lot. Moreover, I started with 32 filters in the first Convolution layer. In the figure bellow there is a print of the layers used and the parameters of the model.

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=6272, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=133, bias=True)
  (dropout): Dropout(p=0.3)
)
```

The accuracy of the model was not significative but it was around the asked value, 11 %. I only used 10 epochs and as you can see in the figure bellow both loss and validation accuracies were decreasing so it was completely possible to use more epochs to improve the results but since I didn't want to spend all my Gpu time I stopped there.

```
Epoch: 7        Training Loss: 3.849107         Validation Loss: 4.026291
Validation loss decreased (4.082339 --> 4.026291).  Saving model ...
Epoch: 8        Training Loss: 3.725503         Validation Loss: 3.996468
Validation loss decreased (4.026291 --> 3.996468).  Saving model ...
Epoch: 9        Training Loss: 3.611827         Validation Loss: 3.961194
Validation loss decreased (3.996468 --> 3.961194).  Saving model ...
Epoch: 10       Training Loss: 3.504261         Validation Loss: 3.941575
Validation loss decreased (3.961194 --> 3.941575).  Saving model ...
```

I used Cross entropy loss and Adam optimizer. Another **major project development stage was** the choice of the learning rate of the optmizer I chosed. I noticed the accuracy was more than 10 times better decreasing the learning rate!

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

In this step, transfer learning has been used to create a CNN that can identify dog breed from images in order to get at least 60% accuracy on the test set.

I just adapted the last layer of the pre trainned model vgg16 to have 133 outputs (number of dog breeds).

I used the same dataloader what was very pratical.

The accuracy was much better than the ones obtaines from the previous model from scratch as you can see bellow:

```
Epoch: 1        Training Loss: 0.593417         Validation Loss: 0.447194
Validation loss decreased (inf --> 0.447194).  Saving model ...
Epoch: 2        Training Loss: 0.534306         Validation Loss: 0.451095
Epoch: 3        Training Loss: 0.505782         Validation Loss: 0.424775
Validation loss decreased (0.447194 --> 0.424775).  Saving model ...
Epoch: 4        Training Loss: 0.477269         Validation Loss: 0.418957
Validation loss decreased (0.424775 --> 0.418957).  Saving model ...
Epoch: 5        Training Loss: 0.471790         Validation Loss: 0.402350
Validation loss decreased (0.418957 --> 0.402350).  Saving model ...
```

The accuracy of the model was 84 % and I only used 5 epochs and as you can see in the figure above both loss and validation accuracies were decreasing so it was completely possible to use more epochs to improve the results. I stopped there since there were no signs of overfitting and the accuracy was quite good.

Step 5: Write your Algorithm

In this step, an algorithm has been developed that takes a path and determines if the image contains a human face, a dog or neither. Then:

- 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.

In this step I used the pre trainned model to get the dog breed for obvious reasons.

Step 6: Test Your Algorithm¶

Test the algorithm with at least six images (from internet).

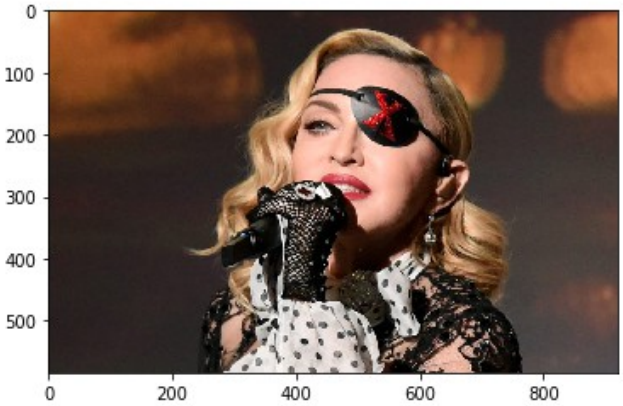
I download the images from internet and the images were stored locally so they could be used in the model since I needed a path.

I had interesting results:

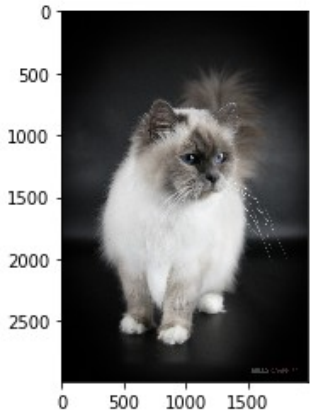
```
Dog detected :-) and the breed is:  Chihuahua
```


```
Human detected :-) and the resembling dog breed is: Poodle
```


```
Error, neither face or dog detected!
```



```
Human detected :-) and the resembling dog breed is: Japanese chin
```



The detection of dogs seems quite good since the cat was not detected as dog but the human face recognition is poor as expected from the performance obtained in the beginning of this project.

I tought it was fun and I don't know if I would install an app to do it by I think many people on facebook would try it if it was proposed in facebook wall.

From all the capstone project proposals this one was the only interesting and I know I could create my own model but... at least there was one interesting project :-)

IV: Project Closure

I enjoyed very much this nanodegree, I learned a lot, specially with the extra lessons udacity offered. I don't think this nanodegree is advanced level, let's say mediun-advanced but I had enough information to continue by myself and built my own project ideas :-)