Capstone Project Abdelrahman Emad

Machine Learning Engineer Nanodegree April 8th,2021

Dog Breed Classifier

Table of content

Definition

Project Overview

A dog breed classifier is a computer vision classifier that uses Convolutional Neural Networks (CNN) to build a pipeline to process real-world, user-supplied images.

Given an image of a dog, The algorithm will identify an estimate of the canine's breed. If supplied an image of a human, the code will identify the resembling dog breed.

Problem Statement

The goal is to create a dog breed classifier based on a dog or human image. the tasks involved are the following:

1. Download and preprocess the input data
2. Create a Human Face detector
3. Create a general dog detector
4. Using transfer learning to create the dog breed classifier.
5. Take a raw image and return the dog breed.

Metrics

Accuracy is a common metric for multi-class classifiers; it takes into account both true positives and true
negatives with equal weight.
accuracy =(true positives + true negative) / size of dataset

Analysis

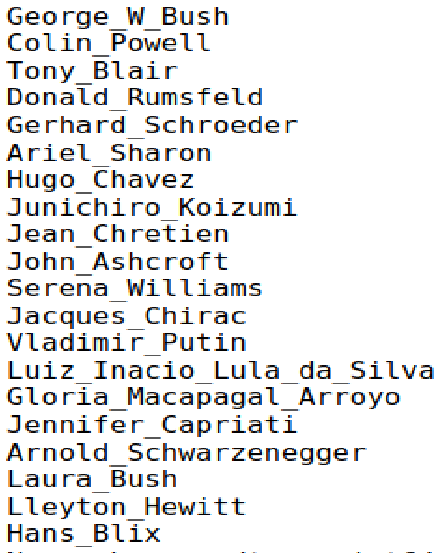
Data Exploration and Visualization

Human dataset

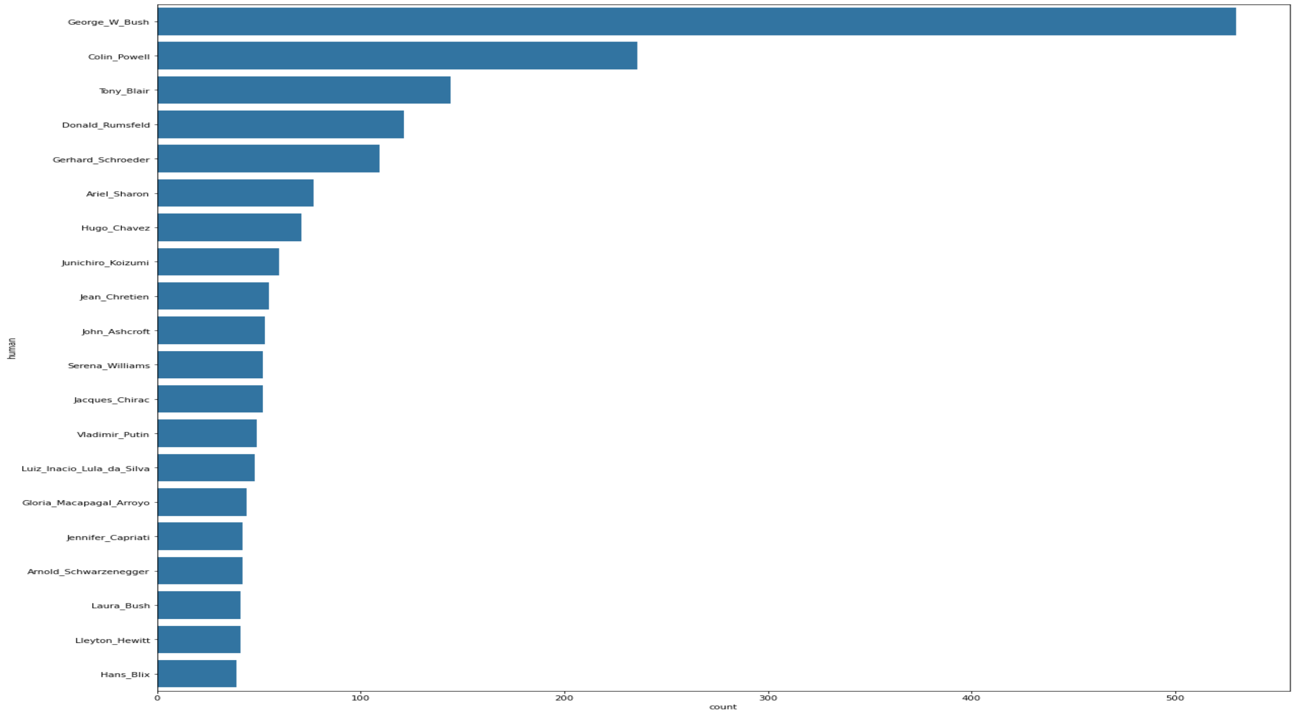
The human dataset has 13233 images with 5749 unique Humans. George_W_Bush has the most counts with 530 image

The top 20 Humans were:

Name # of images George.W.Bush



```
George_W_Bush                530
Colin_Powell                 236
Tony_Blair                   144
Donald_Rumsfeld              121
Gerhard_Schroeder            109
Ariel_Sharon                  77
Hugo_Chavez                   71
Junichiro_Koizumi             60
Jean_Chretien                 55
John_Ashcroft                 53
Serena_Williams               52
Jacques_Chirac                52
Vladimir_Putin                49
Luiz_Inacio_Lula_da_Silva     48
Gloria_Macapagal_Arroyo       44
Jennifer_Capriati             42
Arnold_Schwarzenegger         42
Laura_Bush                    41
Lleyton_Hewitt                41
Hans_Blix                     39
```



Sample images with labels

Nabil_Shaath

Brendan_Stai

Jennifer_Garner

Lisa_Murkowski

Jamie_Cooke

Hernan_Crespo

Cemil_Cicek

Allison_Janney

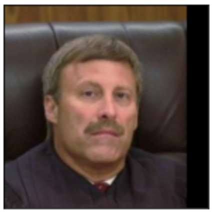
Travis_Rudolph

Demetrius_Ferraciu

Sanjay_Gupta

Brian_Cook

Hussam_Mohammed_Amin


LeRoy_Millette_Jr

Eddie_Lucio

Ward_Cuff

Nia_Vardalos

Don_King

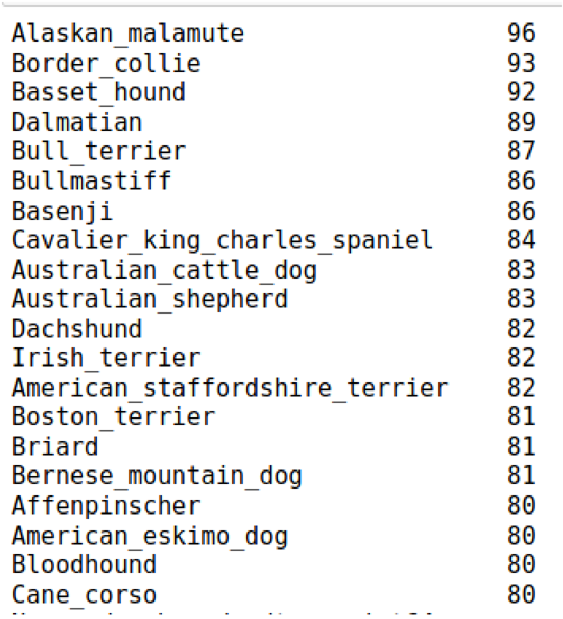
Bill_Bradley

Steve_Zahn

Dog dataset

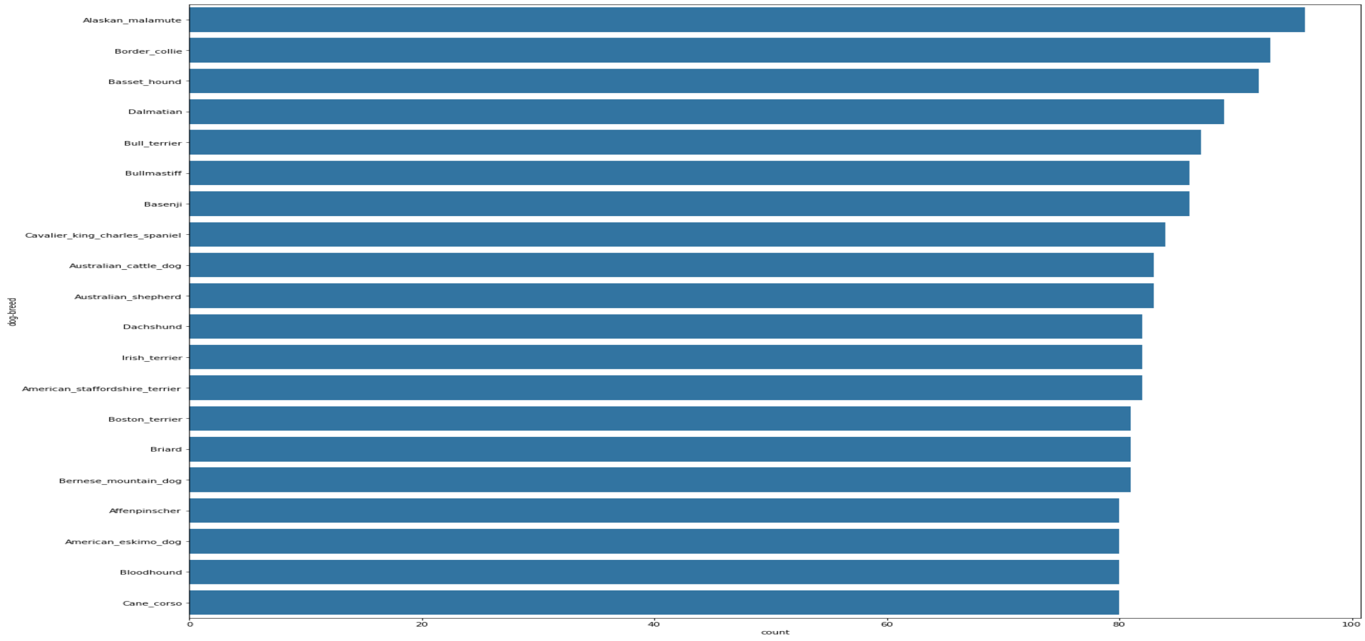
The Dog dataset has 8351 images with 133 unique dog breeds. the Alaskan malamute has the most counts with 95 image

The top 20 Dogs were:

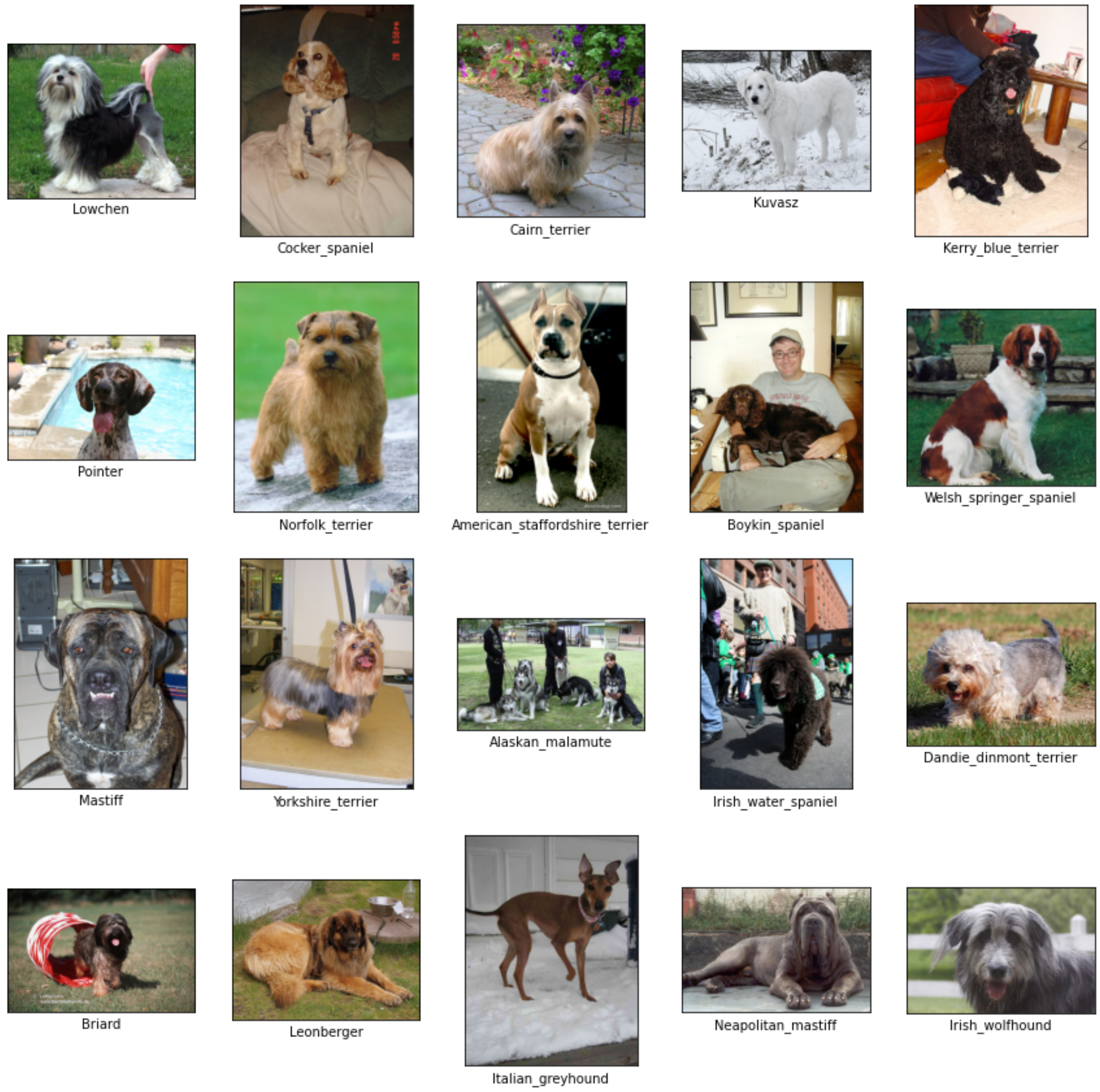


Dog breed	# counts
Alaskan_malamute	96
Border_collie	93
Basset_hound	92
Dalmatian	89
Bull_terrier	87
Bullmastiff	86
Basenji	86
Cavalier_king_charles_spaniel	84
Australian_cattle_dog	83
Australian_shepherd	83
Dachshund	82
Irish_terrier	82
American_staffordshire_terrier	82
Boston_terrier	81
Briard	81
Bernese_mountain_dog	81
Affenpinscher	80
American_eskimo_dog	80
Bloodhound	80
Cane_corso	80

Alaskan malamute



Sample images with labels



Algorithms and Techniques

The classifier is a Convolutional Neural Network, which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches.

The following parameters can be tuned to optimize the classifier:

1. Training parameters:
   - Training length (number of epochs)
   - Batch size (how many images to look at once during a single training step)
   - Learning rate (how fast to learn; this can be dynamic)
2. Neural network architecture:
   - Number of layers
   - Layer types (convolutional, fully-connected, or pooling)

3. Preprocessing parameters

During training, both the training and the validation sets are loaded into the RAM. After that, random
batches are selected to be loaded into the GPU memory for processing.

Benchmark

To create an initial benchmark for the classifier, I used my local machine (6 GByte GPU” NVIDIA Corporation TU116M [GeForce GTX 1660 Ti Mobile]”, 12 logical CPU “Intel® Core™ i7-9750H CPU @ 2.60GHz × 12”)

It took 5.42 Second to classify 200 Image so the average for one image was 27 ms

Methodology

Data Pre-processing

The preprocessing has the following steps:

1. Make augmentation for images (RandomRotation,RandomResizedCrop and RandomHorizontalFlip)
2. Normalize the input images
3. The images are divided into training, validation, and test sets
4. Create the data loaders by configuring the batch size and shuffle the training data

Implementation

The implementation process can be split into two main stages:

1. The classifier training stage
2. Using the trained model to classify dogs

The classifier training stage

During the first stage, the classifier was trained on the preprocessed training data using transfer learning.

The training steps were:

1. load both the training and validation images into memory using the data loaders that we have created
2. Define the network architecture and training parameters
3. Define the loss function, accuracy
4. Train the network, logging the validation/training loss and the validation accuracy
5. If the accuracy is not high enough, return to step 2
6. Save and freeze the trained network

The network architecture

I have used transfer learning using vgg16 model

```
 (features): Sequential(
   (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (1): ReLU(inplace=True)
   (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (3): ReLU(inplace=True)
   (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
   (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (6): ReLU(inplace=True)
   (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (8): ReLU(inplace=True)
   (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
   (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (11): ReLU(inplace=True)
   (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (13): ReLU(inplace=True)
   (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (15): ReLU(inplace=True)
   (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
   (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (18): ReLU(inplace=True)
   (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (20): ReLU(inplace=True)
   (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (22): ReLU(inplace=True)
   (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
   (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (25): ReLU(inplace=True)
   (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (27): ReLU(inplace=True)
   (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (29): ReLU(inplace=True)
   (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
 )
 (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
```

```
  (classifier): Sequential(
    (fc1): Linear(in_features=25088, out_features=4096, bias=True)
    (relu): ReLU()
    (Dropout1): Dropout(p=0.5, inplace=False)
    (fc2): Linear(in_features=4096, out_features=1024, bias=True)
    (Dropout2): Dropout(p=0.5, inplace=False)
    (fc3): Linear(in_features=1024, out_features=133, bias=True)
    (output): LogSoftmax()
  )
)
```

I have kept the features layers as It has already trained to detect shapes and patterns and replaced the classifiers layers with one that I need to classify the new 133 dog breed.

The new classifier becomes:

```
(classifier): Sequential(
    (fc1): Linear(in_features=25088, out_features=4096, bias=True)
    (relu): ReLU()
    (Dropout1): Dropout(p=0.5, inplace=False)
    (fc2): Linear(in_features=4096, out_features=1024, bias=True)
    (Dropout2): Dropout(p=0.5, inplace=False)
    (fc3): Linear(in_features=1024, out_features=133, bias=True)
    (output): LogSoftmax()
  )
```

Using the trained model to classify dogs

1. Create the model using the given layers
2. Load the pre-trained weights
3. Process the input Images
4. Using open cv to detect faces in Images
5. Using pre-trained VGG model to detect dogs
6. Predict the input Image using the model

Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model.

The final architecture and hyperparameters were chosen because they performed the best among the tried combinations.

For a complete description of the final model and the training process, refer to The network architecture section

I have trained the model after replacing the Classifier layers for 10 Epochs with the following train and Validation losses:

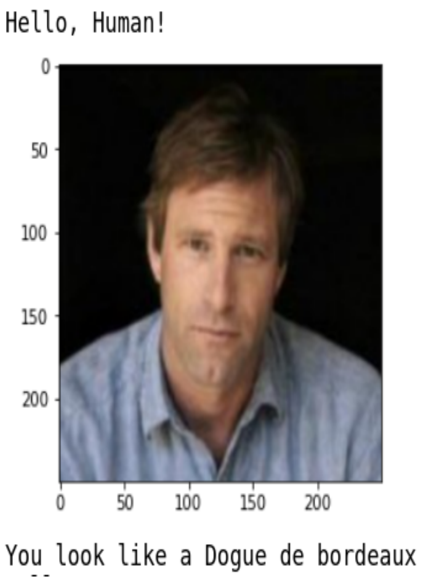
```
Epoch: 1    Training Loss: 1.343471      Validation Loss: 0.829716
Epoch: 2    Training Loss: 1.270921      Validation Loss: 0.739991
Epoch: 3    Training Loss: 1.208083      Validation Loss: 0.681313
Epoch: 4    Training Loss: 1.185337      Validation Loss: 0.700088
Epoch: 5    Training Loss: 1.175069      Validation Loss: 0.613085
Epoch: 6    Training Loss: 1.112700      Validation Loss: 0.649210
Epoch: 7    Training Loss: 1.109303      Validation Loss: 0.549573
Epoch: 8    Training Loss: 1.084382      Validation Loss: 0.585246
Epoch: 9    Training Loss: 1.055470      Validation Loss: 0.690492
Epoch: 10   Training Loss: 1.021630      Validation Loss: 0.687842
```

So as we can see the Validation was still decreasing so we can have better accuracy If we Increased the number of epochs but It will take a longer time.

Using the test set, the Test Accuracy was 80% (677/836)

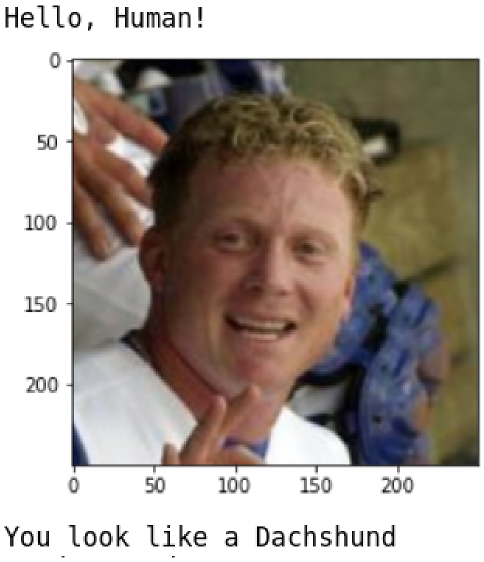
Conclusion

Using Convolution neural networks we can classify the dog breeds with an accuracy of more than 80 % and we can even make this classification for humans.



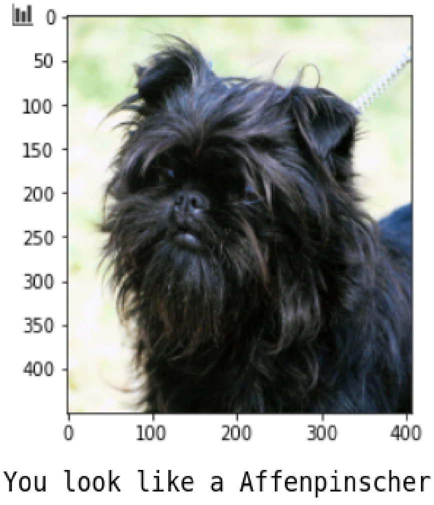
```
Hello, Human!
```

```
You look like a Dogue de bordeaux
```

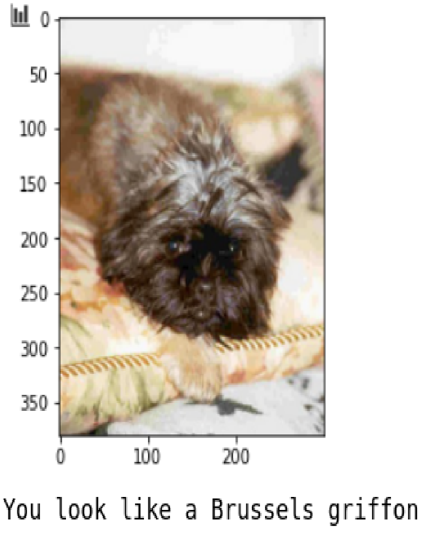


```
Hello, Human!
```

```
You look like a Dachshund
```



```
You look like a Affenpinscher
```



```
You look like a Brussels griffon
```

Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, public datasets were found
2. The data was downloaded and preprocessed (segmented)
3. A benchmark was created for the classifier
4. The classifier was trained using the data and transfer learning (multiple times, until a good set of parameters, were found)
5. The model was used to detect the dog breed on unseen images.

Improvement

To achieve better accuracy for the model

1- we can use different model features layers for our model

2- we can CNN to detect humans better than a face classifier

3- we can train for more epochs