Capstone Project Report

Dog Breed Detection using Convolutional Neural Networks

Guilherme Baldo Carlos, January 20th, 2022

I. Definition

Project Overview

There are hundreds of dog breeds all around the world. The current project aims to develop convolutional neural networks (CNNs) to identify dog breeds with input images. The algorithm produced is composed of a pipeline of image classifiers. This algorithm works by identifying if an image provided by the user contains a person, a dog or neither. In the case of a dog, the algorithm responds with the dog breed. In the case of a person the algorithm responds with the dog breed the person in the picture most resembles. Finally in the case of neither a dog nor a person in the image the algorithm will respond with invalid input.

To develop the solution two datasets are used. First, to evaluate the face detector the LFW (Labeled Faces in the Wild) public dataset is used. This dataset contains 13 thousand images of people. The second dataset is provided by Udacity and corresponds in a dataset with 133 dog breeds and 8351 dog images with varying image size. Those dog images are splitted in train set (6680 images), validation set (835 images) and test set (836 images). One third dataset is indirectly used and it is the ImageNet which contains 1000 classes and it is used to perform transfer learning, so we use CNNs that are pretrained on those models.

Problem Statement

To address the challenge proposed, three image classifiers needed to be used.

- face detector: used to identify faces in images in order to classify an input image as a human or not.
- dog detector: a CNN responsible for performing classification and used for telling if the input image corresponds to a dog or not.
- dog breed classifier: a CNN developed and trained to perform dog breed classification and telling the dog breed the dog in the input image is or the dog breed the person in the input image most resembles.

The project was developed using a base jupyter notebook which breaks down the project in the following steps:

1. Import datasets: two links are provided and the datasets must be download the datasets and extracted it in the predefined location
2. Detect humans: in this step the Haar Cascade face detector provided by OpenCV is used to perform face detection. A function to wrap the face detector should be developed.
3. Detect dogs: a pretrained VGG-16 which is a state-of-the-art CNN classifier must be used to perform dog detection. Since this model was trained to perform 1000-class classification, a function to wrap this model and filter dog and non-dog detections should be developed.
4. Create a CNN to classify dog breeds from scratch: in this step it is necessary to develop and train a CNN model from scratch.
5. Create a CNN to classify dog breeds using transfer learning: in this step an existing CNN provided by pytorch must be chosen to perform dog breed classification. Transfer learning must be used.
6. Write your algorithm: the final algorithm is built using the face detector to identify if the input image corresponds to a person, the dog detector to identify if the input image corresponds to a dog and the dog breed classifier to tell what is the dog breed or what dog breed most resembles the person in the input image.
7. Test your algorithm: in this final step the algorithm must be tested with different dog and human images externally provided.

Evaluation Metrics

Since we are talking about classification, we want to measure how many of the human images the face detector is able to detect. How many dog images the dog detector is able to correctly predict. And finally how many dog breeds the dog breed classifier is able to predict correctly.

The metric used for that is the accuracy and for classification purposes it is calculated as below:

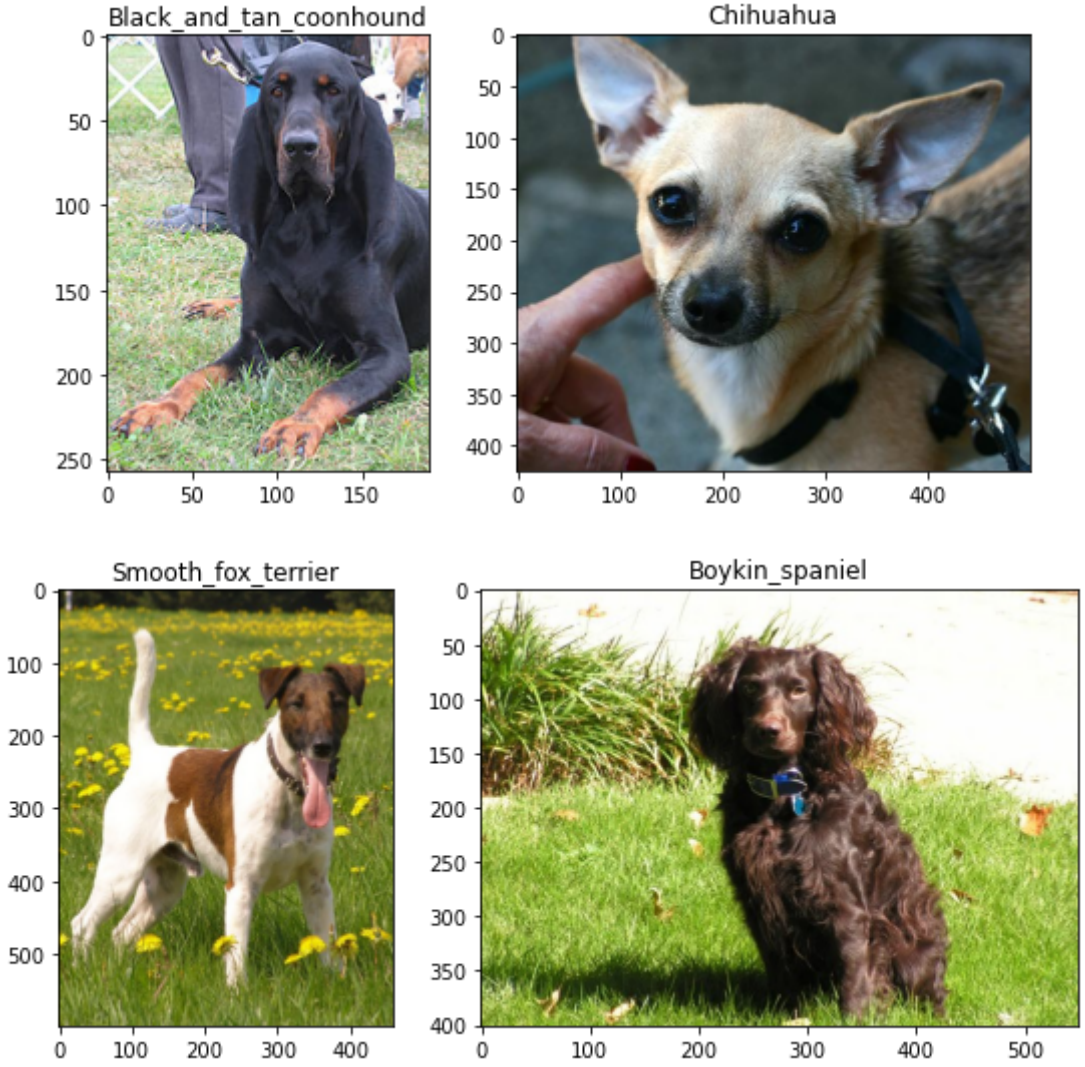
$$accuracy = \#\ of\ correct\ predictions\ /\ \#\ of\ predictions$$

The accuracy requirement for the dog breed classifier built from scratch is minimum 10% and the accuracy for the dog breed classifier using transfer learning is 60%.

II. Analysis

Data Exploration

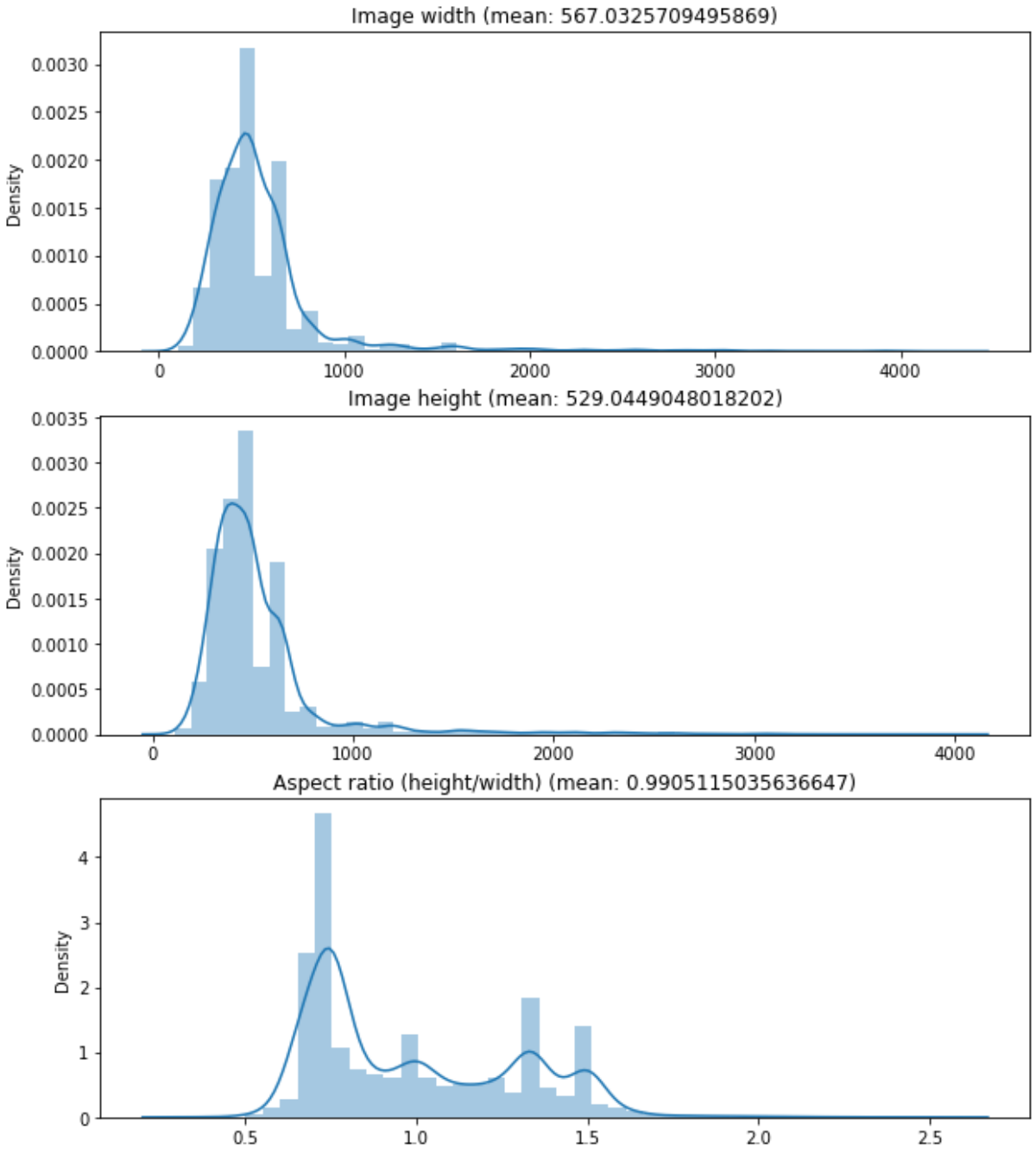
In this project we train and evaluate the dog breed classifier in a dataset provided by Udacity. This dataset contains 133 dog breeds and 8351 dog images with varying image size. Those dog images are splitted in train set (6680 images), validation set (835 images) and test set (836 images). We see below some examples of images from the dataset with its labels.



As we note, the images come in different sizes. That is explored further in the exploratory visualization section.

Exploratory Visualization

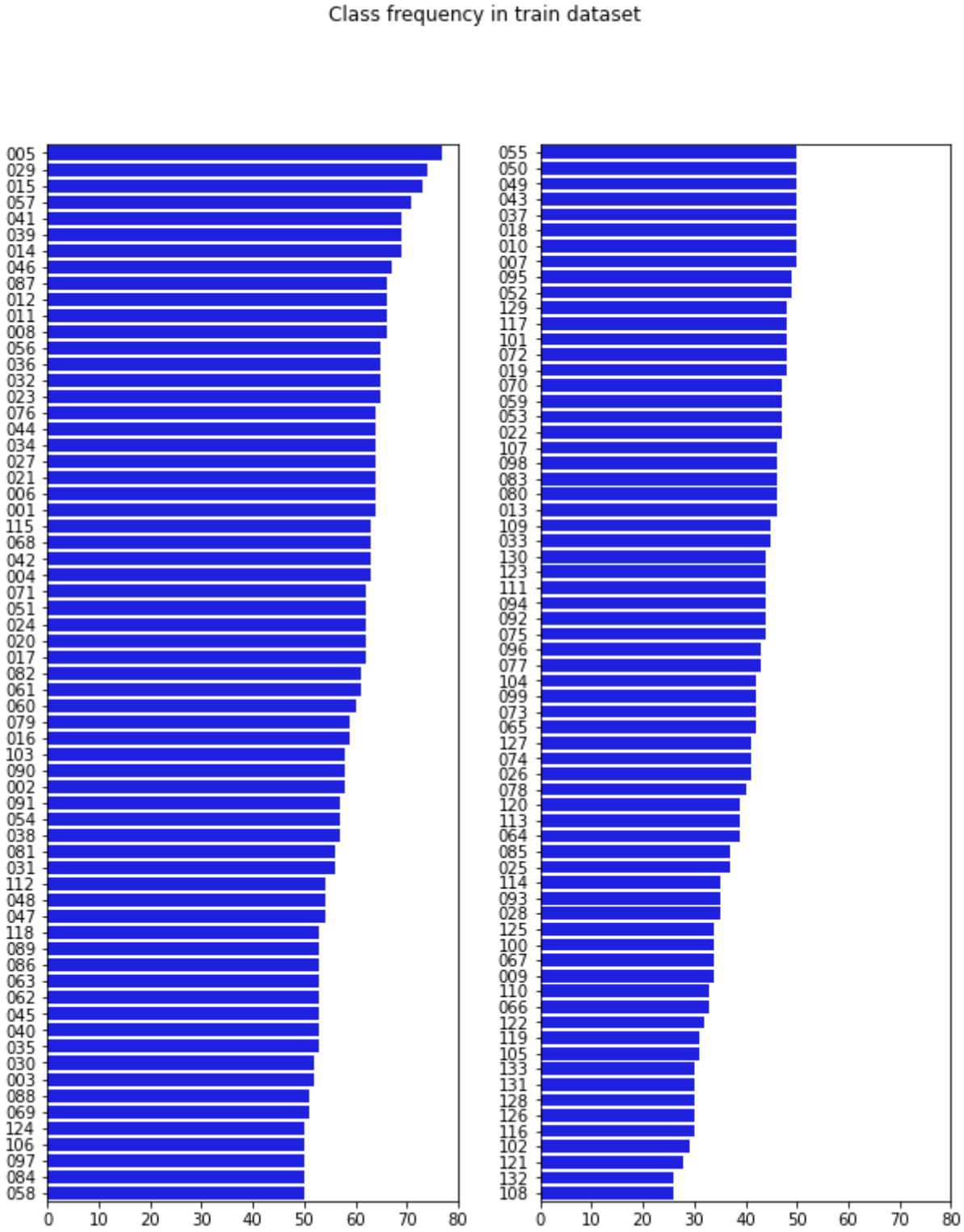
Some visualizations are used to further understand the dataset. Regarding the aforementioned difference in image dimensions from the dataset, the plots below show how these dimensions are distributed. The width, height and aspect ratio (height/width) distributions are shown below.



It is possible to notice that for the image width and height there is a high frequency of images with dimensions around 550 pixels. As for the aspect ratio, the majority of images present an aspect ratio that goes from 0.5 to 1.5 (1 indicates a square image). That said, the images

range from a landscape aspect image with width twice the height to portrait aspect images with height 50% greater than width.

As stated before there are 133 dog breeds (classes) in the dataset. It is important for training that the dataset is not far from balanced. Therefore, the visualization below shows the frequency distribution for each class in the training dataset.



It is possible to see that there is a difference in distribution between different dog breeds. Ranging from dog breeds with 26 images to dog breeds with 77 examples. So the most frequent class in the dataset has around triple the frequency of the less frequent. That is not addressed in this project but may be an improvement point.

Algorithms and techniques

To solve the problem of human detection the face detector Haar Cascades provided in OpenCV is used.

To solve the problem of dog detection the VGG-16 model pretrained on ImageNet was used. This model is already able to detect a 1000 classes, so a function was built to binarize the output of this model from dog (class 151 to 278) and non dog (0 to 150 and 279 to 999).

To solve the dog breed classification per se two solutions are explored. The first solution consists of a model built and trained from scratch on the dog breed dataset following CNN design guidelines. That solution was built with inspiration in architectures from literature. The other solution was based on the selection of a state of the art model pretrained on ImageNet and fine tuned on the dog breed dataset.

Benchmark Model

The benchmark model that will be used to compare the solution built using transfer learning was the solution built from scratch. It is expected that the results achieved with transfer learning require fewer epochs for training and achieve better metrics.

There are imposed requirements for the accuracy of the models. The model built from scratch must achieve at least 10% accuracy and the model built with transfer learning must achieve at least 60% accuracy.

III. Methodology

Data preprocessing

The preprocessing pipeline used for the dog detector (pretrained VGG-16) was composed by the following transformations:

1. Resize the input image to 256x256.
2. Cropping 224x224 from the center of the image.
3. Converting the image to tensor.
4. Normalizing the tensor.

For the dog breed classifier two other data preprocessing pipelines were defined. One for the training dataset and the other for both the validation and test.
The data preprocessing pipeline define for training is described below:

1. Randomly flipping the image horizontally.
2. Randomly rotating the image 10 degrees.
3. Cropping 224x224 from a random position of the image.
4. Converting the image to tensor.
5. Normalizing the tensor.

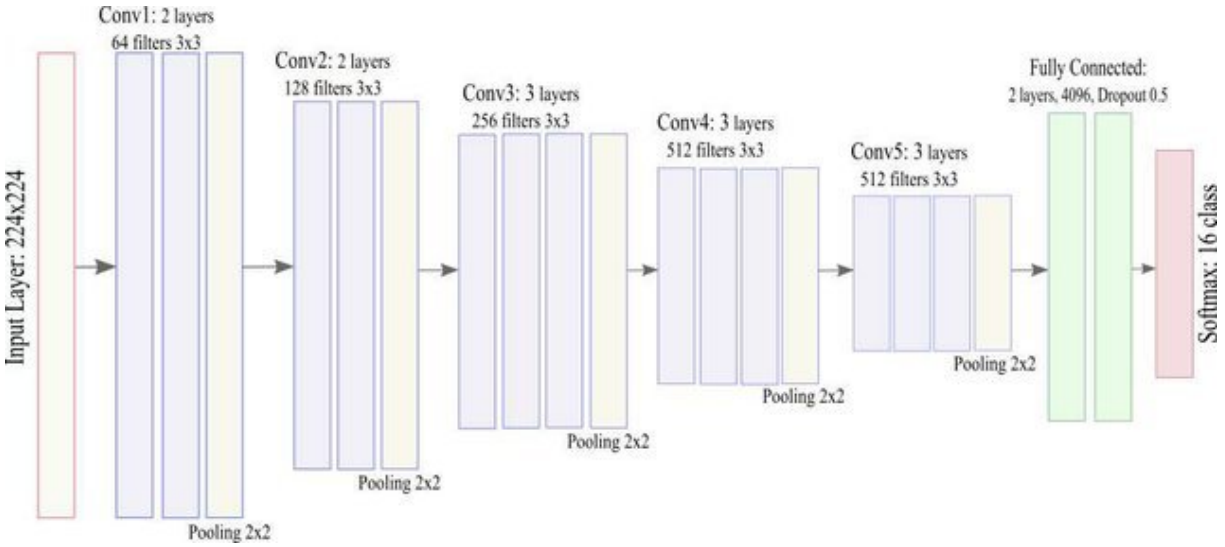
And for the validation and testing:

1. Resize the input image to 256x256.
2. Cropping 224x224 from the center of the image.
3. Converting the image to tensor.
4. Normalizing the tensor.

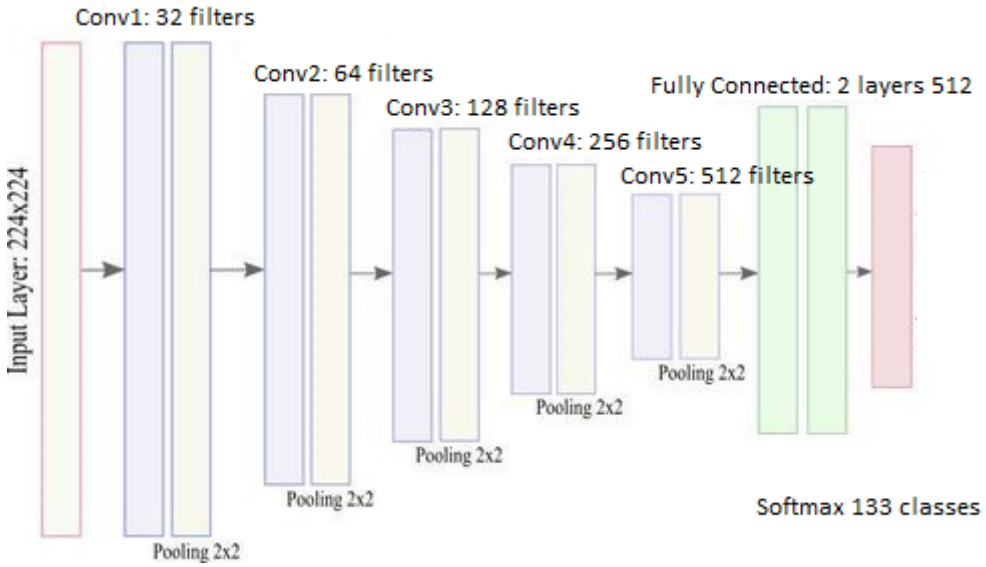
As for the dog detector.

Refinement

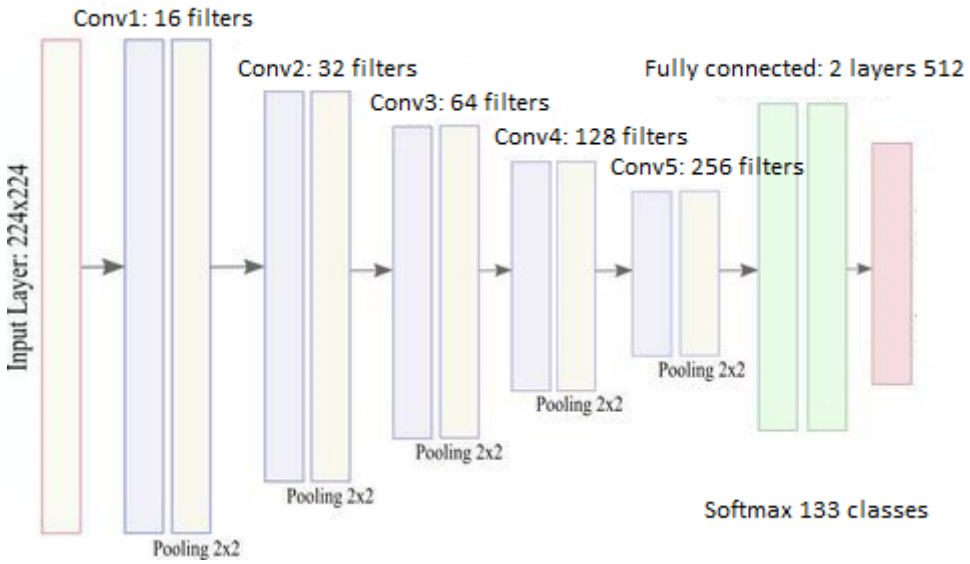
The starting point to design the dog breed detector from scratch was to take inspiration in the previously used model VGG-16 (used for the dog detector). The VGG-16 architecture is shown below.



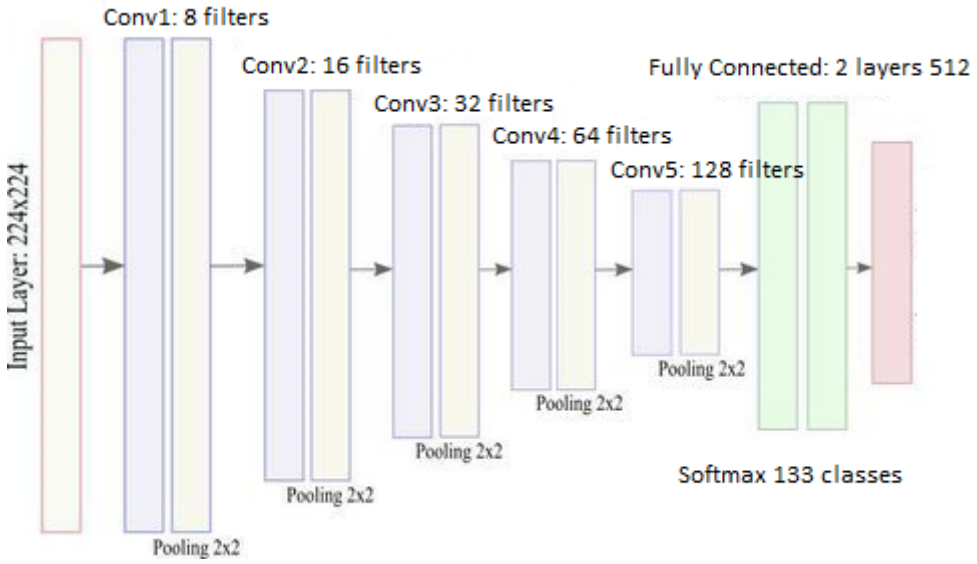
Since the training was going to be from scratch. The model was simplified by reducing the number of convolutions and the number of filters.



This architecture achieved an accuracy of 17% which was already above the threshold required of 10% accuracy. But trying to reduce even further the filter number in order to reduce the model complexity we produced the following architecture.

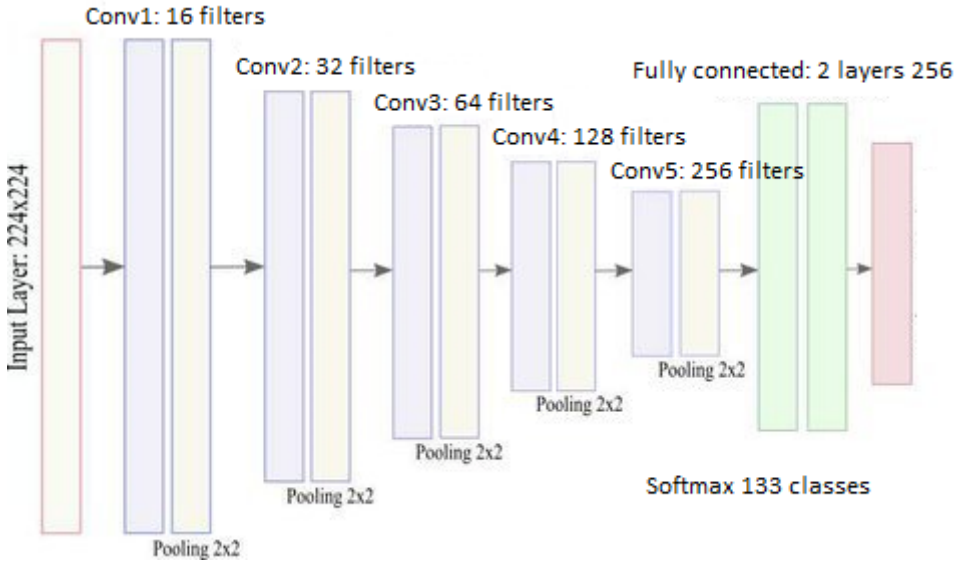


Despite the reduction in complexity this architecture achieved an accuracy of 31%. Which represented an improvement in comparison to the previous architecture. Trying to reduce the number of filters even more, the architecture below was produced.

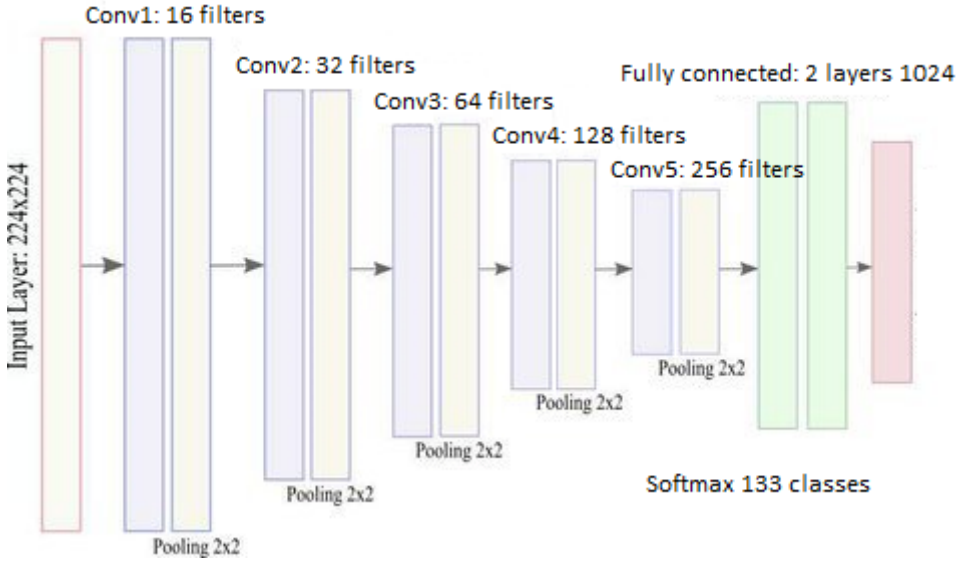


In its turn, the architecture above achieved a 26% accuracy, indicating that the previous architecture was superior. So the previous architecture is considered in the next tests.

Now trying to explore how the number of neurons in the fully connected layer affects the results. The next architecture is shown next.



The number of neuron reduction led to a reduction in accuracy to 24%. So next the increase in the number of neurons is explored.



The accuracy achieved by this model was 29% which is still less than the better model with 31% accuracy. So that was the one used.

In the next step of the project a model using transfer learning was used. And the ResNet family of models was chosen. The ResNet family of models consists of a family of classification models, in which each member of this group has a different depth (number of layers).

The following table shows the difference in architecture between the ResNet models.

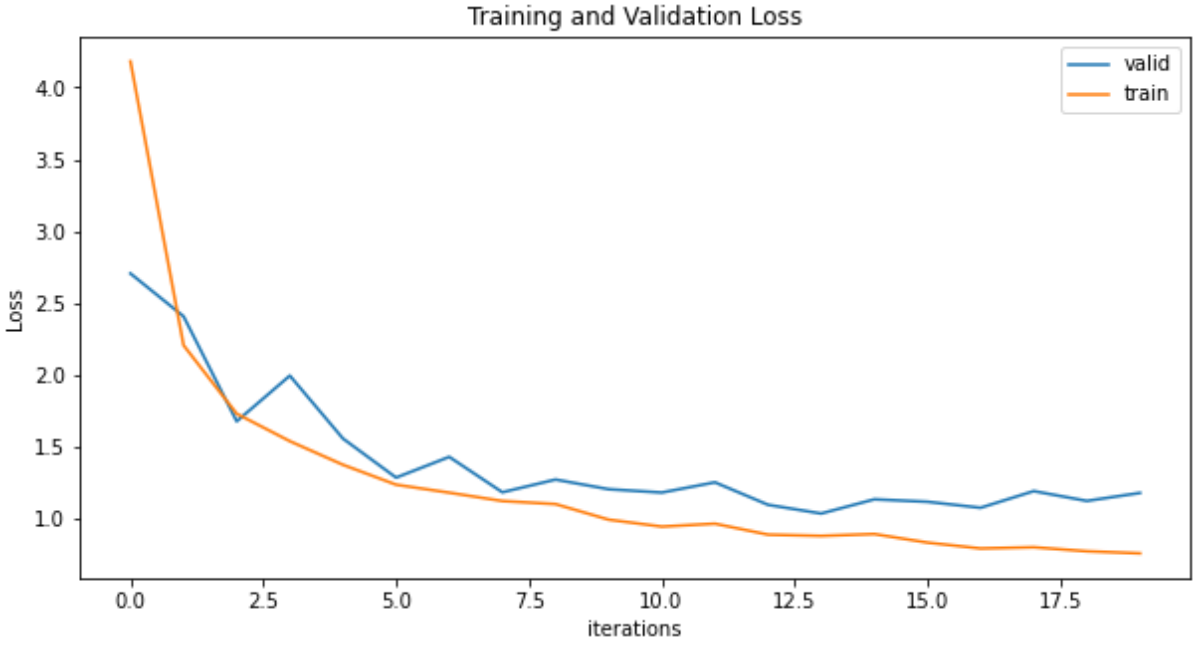
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Testing the ResNet18 the accuracy achieved was 70% which is already above the desired threshold of 60%. But exploring the larger models the ResNet34 yielded an accuracy of around 70% also, which indicates that using ResNet18 is better because a simpler model with the same accuracy is better. It was not possible to test ResNet50, ResNet101 and ResNet152 because the model occupied the whole machine memory.
Therefore, ResNet18 was selected.

Implementation

The models were implemented with the PyTorch framework and trained with SGD, cross entropy loss and adam optimizator. As stated before, accuracy was the metric used. The model trained from scratch was trained for 50 epochs with learning rate 0.001 and the model trained with transfer learning was trained for 20 epochs with learning rate 0.001.
The following graph shows the evolution of the training and validation loss during the training epochs for the model trained from scratch.

And the following graph shows the evolution of the training and validation loss during the training epochs for the model trained with transfer learning.



The next diagram shows the whole algorithm fluxogram.

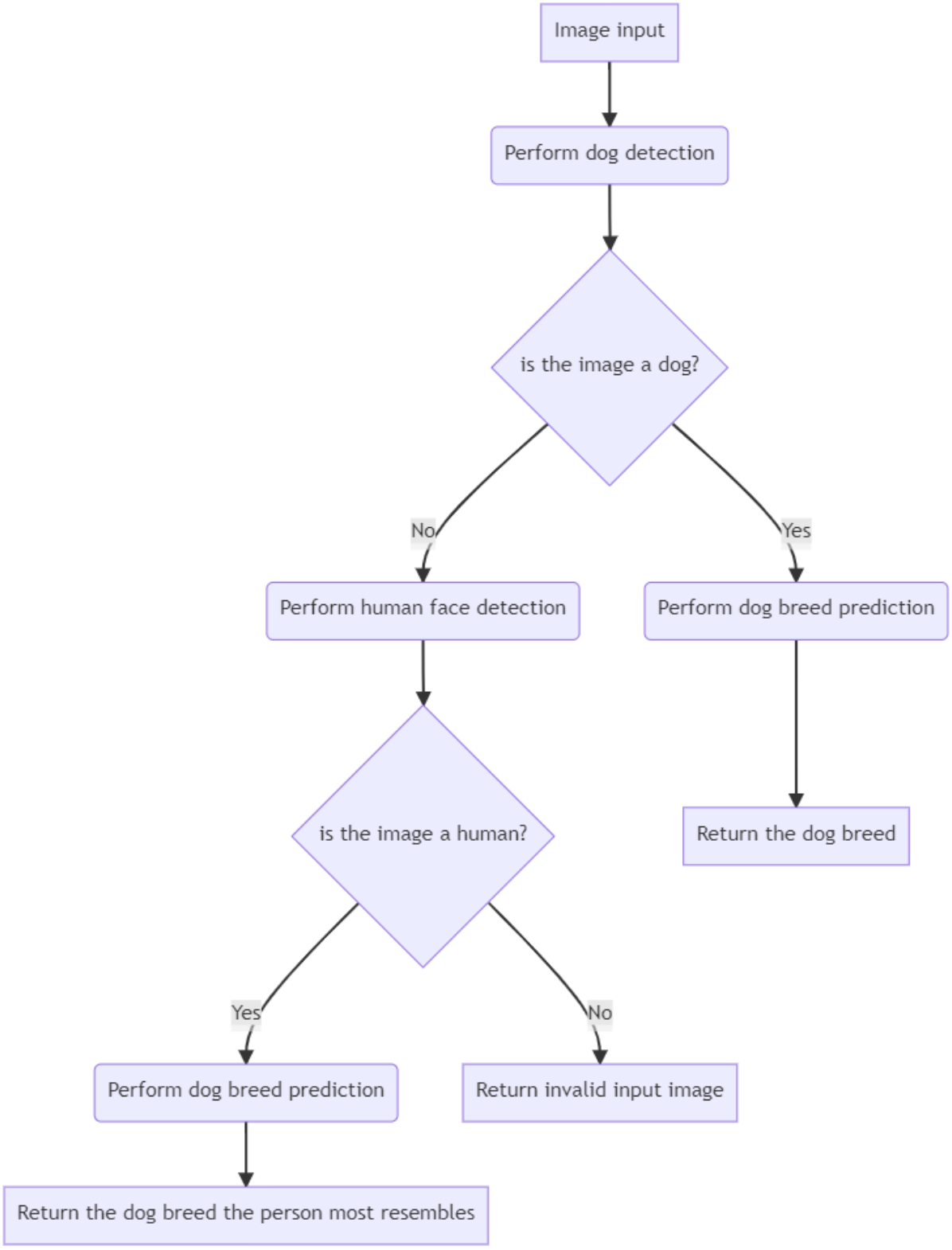


Image input
Perform dog detection
is the image a dog?
No
Yes
Perform human face detection
Perform dog breed prediction
is the image a human?
Return the dog breed
Yes
No
Perform dog breed prediction
Return invalid input image
Return the dog breed the person most resembles

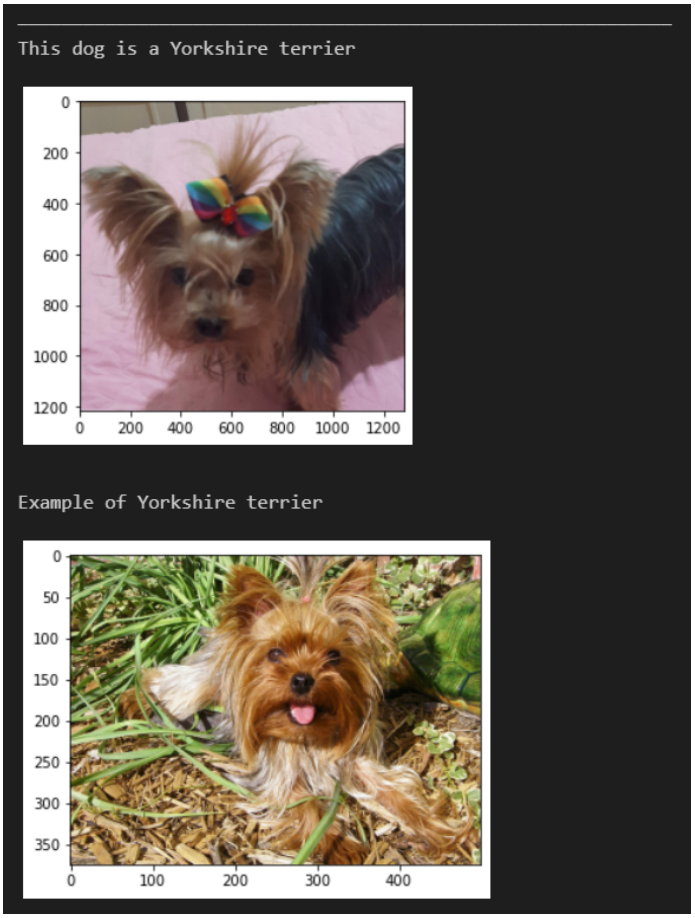
IV. Results

Justification

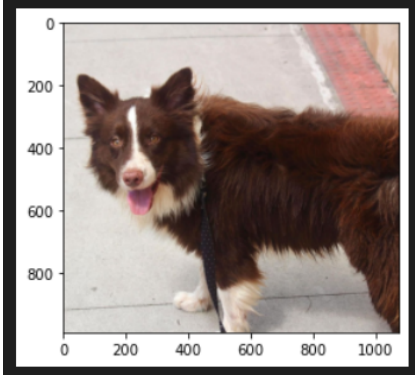
For the dog breed detector, both the model trained from scratch and the model trained with transfer learning were able to achieve an accuracy above the threshold required. Namely, the model trained from scratch achieved a 31% accuracy when 10% was required and the model trained with transfer learning (ResNet18) achieved a 70% accuracy when 60% was required. The final algorithm used the model trained with transfer learning to perform the dog breed detection. For the dog detector teh VGG-19 was used and the HaarCascades face detector provided by OpenCV was used as a face detector.

Model evaluation and validation

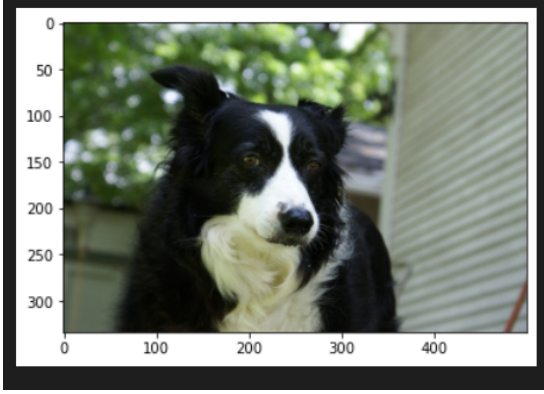
Running the algorithm for some dog images from outside the dataset we get the following results



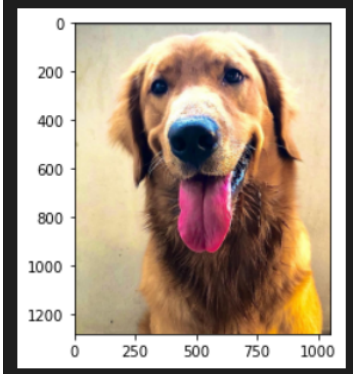
This dog is a Border collie



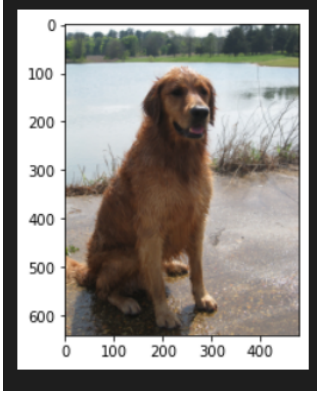
Example of Border collie



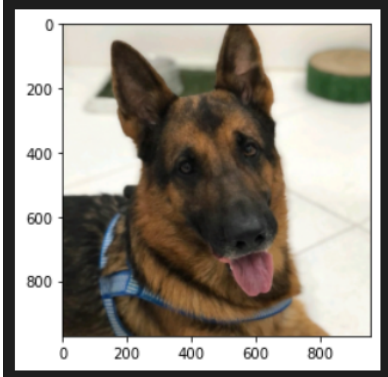
This dog is a Golden retriever



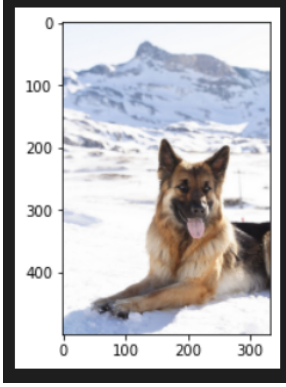
Example of Golden retriever



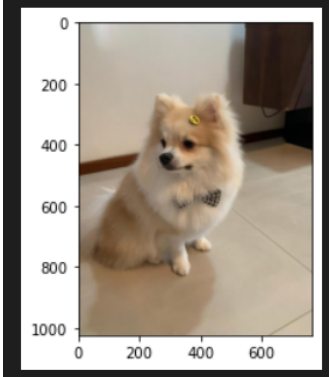
This dog is a German shepherd dog



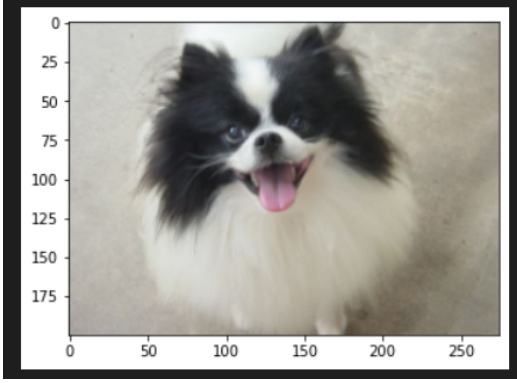
Example of German shepherd dog



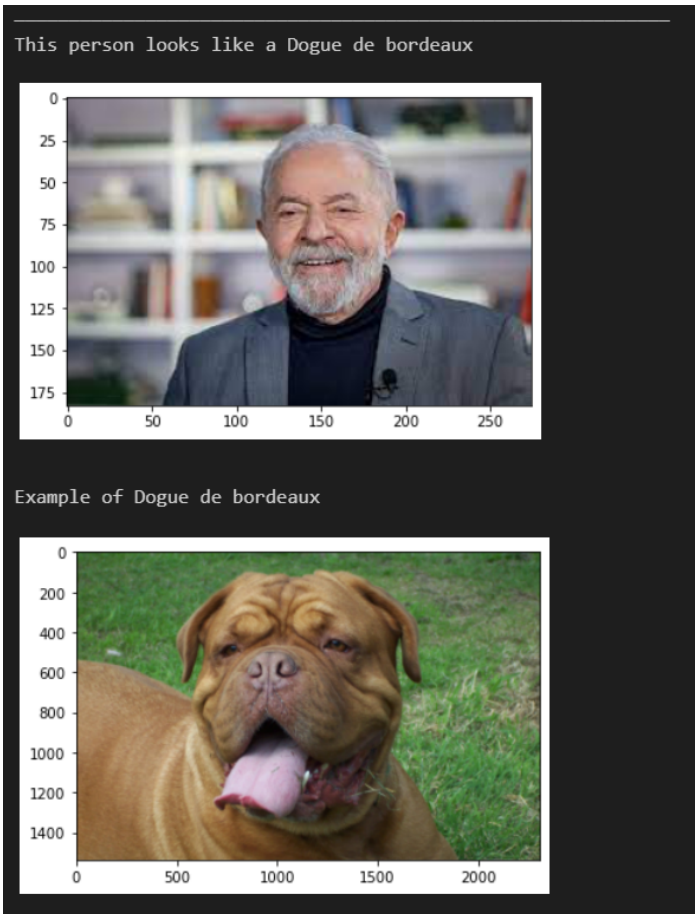
This dog is a Pomeranian



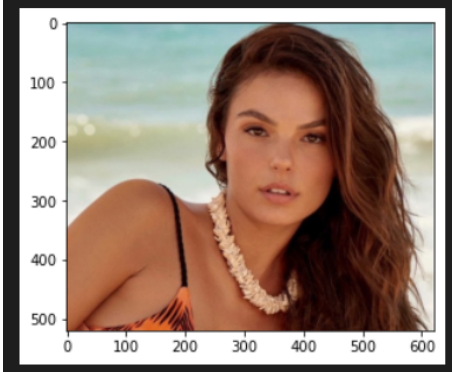
Example of Pomeranian



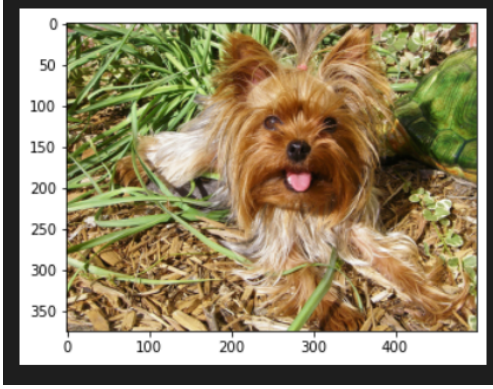
So it is possible to see that the model is succeeding in identifying the dog breed as at least in the examples provided it was able to guess correctly the dog breed. The examples below the input image are from the dog images dataset provided by Udacity.
When we apply the same algorithm to people we get the following example results.



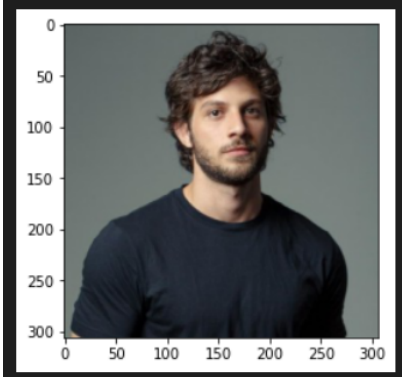
This person looks like a Yorkshire terrier



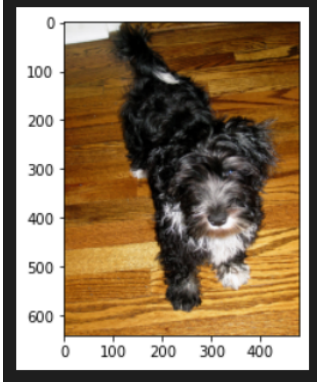
Example of Yorkshire terrier



This person looks like a Lowchen



Example of Lowchen



Despite being funny to experiment with, it is possible to notice that the output of the algorithm really resembles the human images.

And finally when we present the algorithm with a non-dog and non-human image it responds adequately.