Convolutional Neural Network for Dog Breed Classification

Udacity Machine Learning Engineer Nanodegree Capstone Project

Written by **Alex Augusto Costa Machado** on **December 10th 2021**

1 Definition

1.1 Project Overview

For computer vision tasks, such as a dog breed classification, an algorithm mostly used is a Convolutional Neural Network, or CNN for short, which is a model that extracts features, like texture and edges, from spatial data.

The history behind this algorithm started during the 1950s, but it is around the year 2012 that CNNs saw a huge surge in popularity after a CNN called AlexNet achieved state-of-the-art performance labeling pictures in the ImageNet challenge. Alex Krizhevsky et al. published the paper "ImageNet Classification with Deep Convolutional Neural Networks" describing the winning AlexNet model.

1.2 Problem Statement

The purpose of this project is to use a Convolutional Neural Network to classify dog breeds using images as input. For images that contain a human instead of a dog, the algorithm should display which dog breed resembles the human in the picture.

1.3 Metrics

The metric to evaluate the quality of the classifier is the accuracy of the dog breed predictions. The accuracy can be used in this case because the training data is not imbalanced. The definition of prediction accuracy is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

2 Analysis

2.1 Data Exploration

There are two datasets for this project: one is a dog dataset containing 8351 images of 133 different dog breeds, and the other is a human dataset consisting of 13233 photos of 5749 different people. The dog dataset is divided in train, validation, and test datasets.

A sample of the dog dataset is presented in the Figure 1.

2.2 Exploratory Visualization

A way to explore this image dataset is to verify how many images there are for each of the dog breeds. The Figure 2 presents a graph of image count for each dog breed.

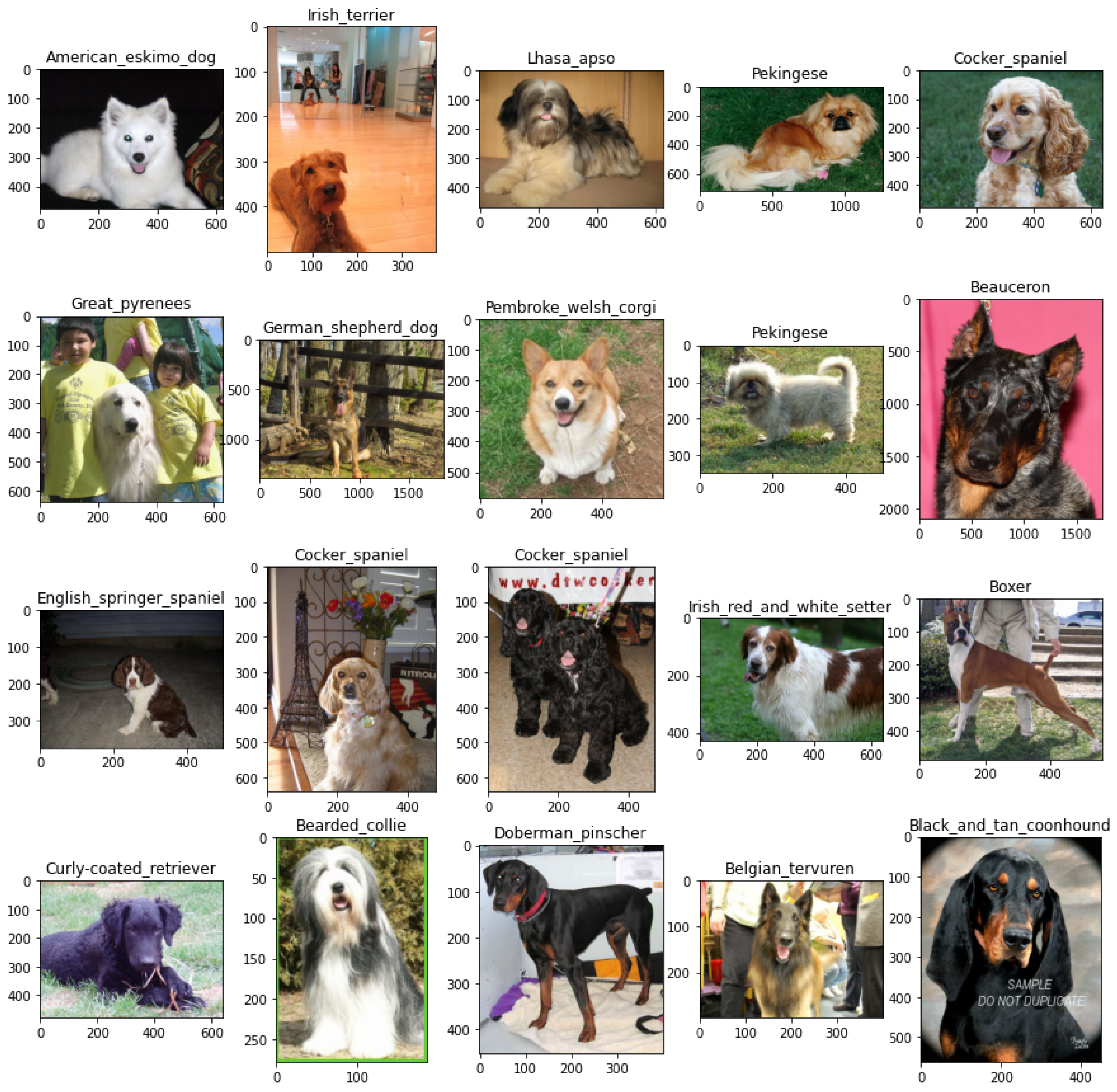


Figure 1: Dog dataset sample.

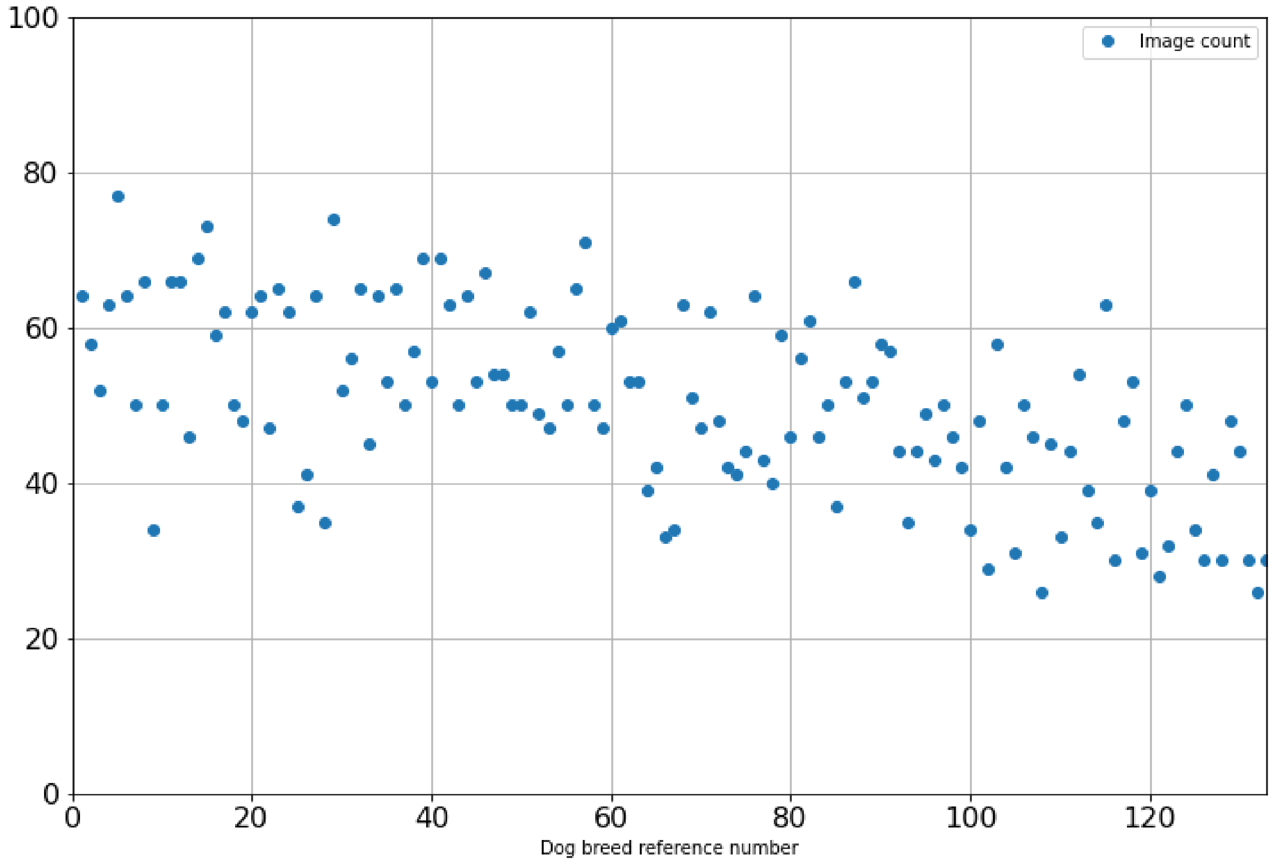


Figure 2: Number of images for each dog breed.

2.3 Algorithms and Techniques

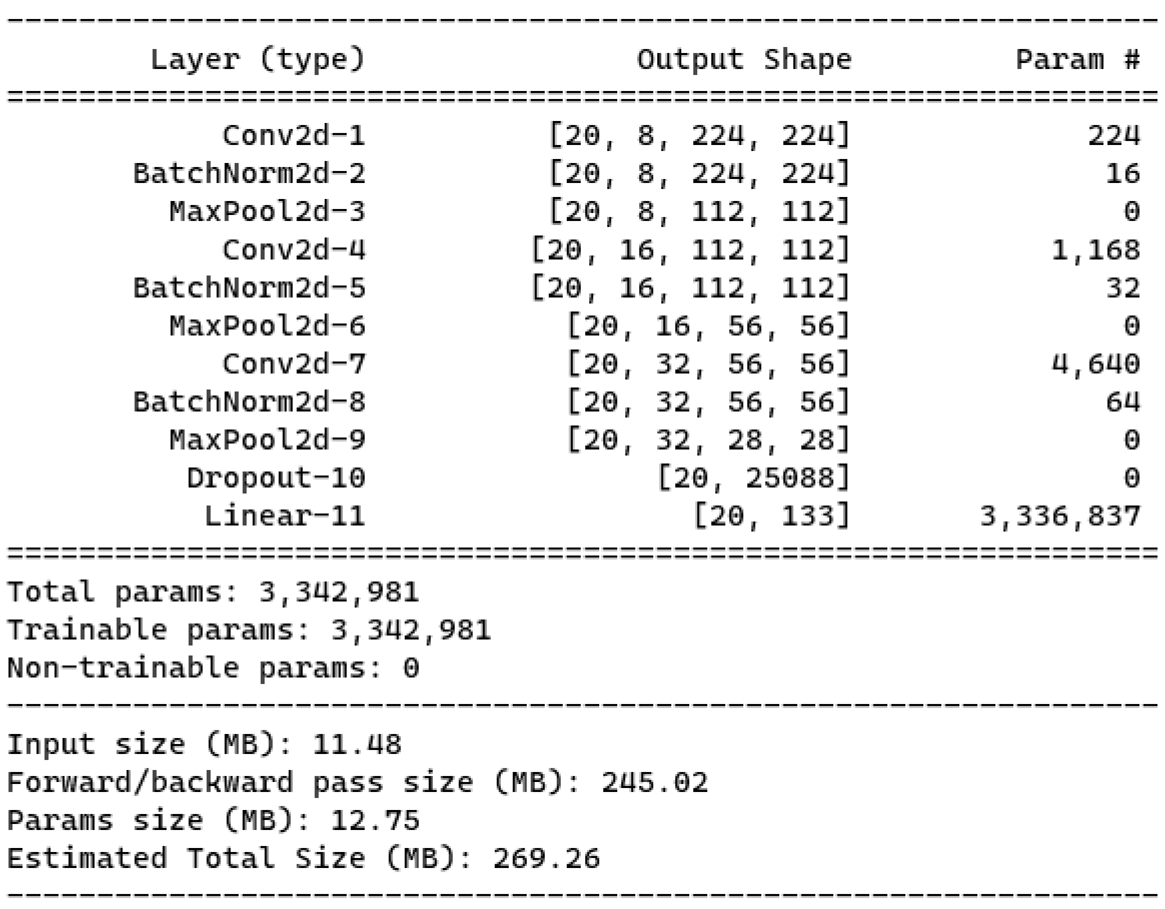
Three main algorithms will be used for this project: a human face detector, a dog detector, and a dog breed classifier. More about each of these algorithms is described below.

- Human Face Detector: an OpenCV's implementation of Haar feature-based cascade classifier to detect human faces in images. The pre-trained face detector is stored as XML file on GitHub.
- Dog Detector: a pre-trained VGG-16 model with weights that have been trained on ImageNet, a very popular dataset used for image classification and other vision tasks.
- Dog Breed Classifier: a pre-trained VGG-11 model on which the last layer is altered to output 133 classes and trained further using the dog dataset mentioned earlier.

2.4 Benchmark

The benchmark for the dog breed classifier will be a simple CNN model designed from scratch that will be compared to the final CNN obtained from transfer learning using a pre-trained VGG-11 model.

The baseline model contains three convolutional blocks, each presenting a convolutional layer, a batch normalization layer and a pooling layer. After these blocks, there is a dropout layer and a fully connected linear layer. A summary of the model is presented on Figure 3.



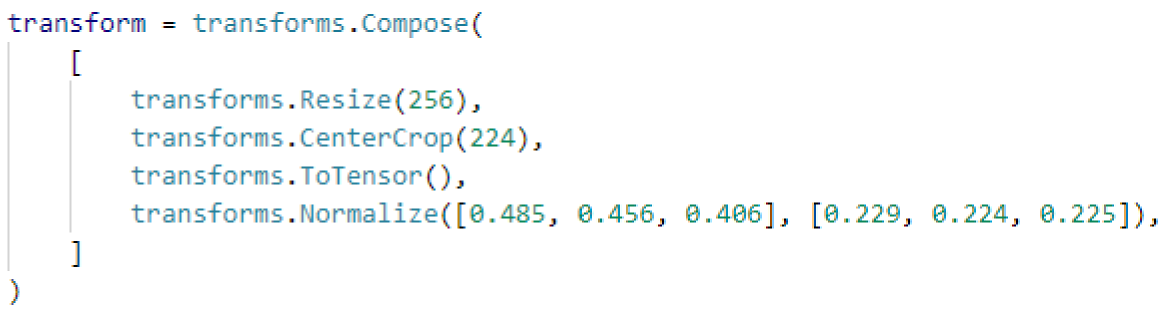
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [20, 8, 224, 224]             224
       BatchNorm2d-2          [20, 8, 224, 224]              16
         MaxPool2d-3          [20, 8, 112, 112]               0
            Conv2d-4         [20, 16, 112, 112]           1,168
       BatchNorm2d-5         [20, 16, 112, 112]              32
         MaxPool2d-6           [20, 16, 56, 56]               0
            Conv2d-7           [20, 32, 56, 56]           4,640
       BatchNorm2d-8           [20, 32, 56, 56]              64
         MaxPool2d-9           [20, 32, 28, 28]               0
          Dropout-10                [20, 25088]               0
           Linear-11                  [20, 133]       3,336,837
================================================================
Total params: 3,342,981
Trainable params: 3,342,981
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 11.48
Forward/backward pass size (MB): 245.02
Params size (MB): 12.75
Estimated Total Size (MB): 269.26
----------------------------------------------------------------
```

Figure 3: Summary of the benchmark model.

3 Methodology

3.1 Data Preprocessing

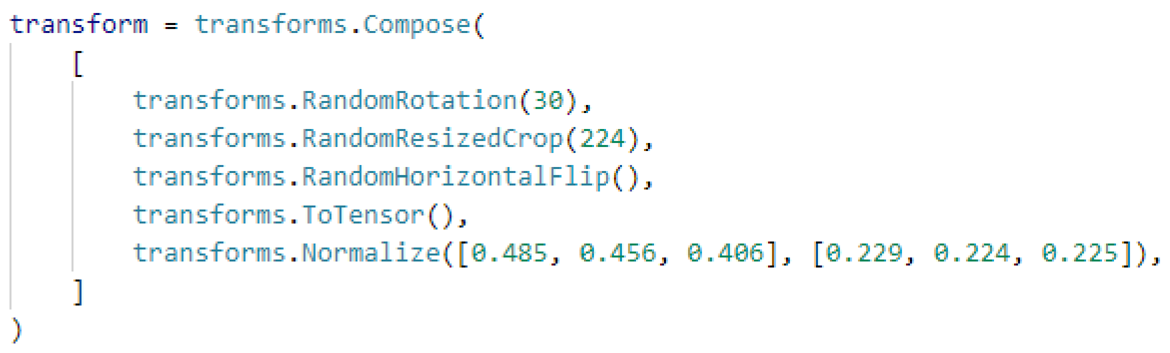
Since the images have all different sizes, it was necessary to resize them. The images on the validation and testing datasets were resized to 256 pixels and cropped to square shapes of 224x224 pixels about the center and transformed into tensors, then the resulting tensors were normalized. A code snippet is presented on Figure 4.



```
transform = transforms.Compose(
    [
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
    ]
)
```

Figure 4: Code for validation and testing transform.

For the training data, in addition the resizing, cropping and normalization, data augmentation was applied, so the images were also randomly rotated and horizontally flipped. The Figure 5 presents the code used for this transformation.



```
transform = transforms.Compose(
    [
        transforms.RandomRotation(30),
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
    ]
)
```

Figure 5: Code for training transform.

3.2 Implementation

The steps required for this implementation are as follow:

- Exploration of both the dog and human datasets in order to verify the quality of the data and if any cleaning was necessary.
- Making all the images the same size and normalizing of images.
- Apply data augmentation on training dataset such as random horizontal flip and random rotation.
- Configuration of pretrained human face detector using a pretrained OpenCV algorithm.
- Configuration of pretrained dog detector using a VGG-16 model.
- Design of a Convolutional Neural Network to utilize as benchmark model.
- Training, validating and testing benchmark model.
- Change a pretrained VGG-11 model to output 133 classes, which is the number of dog breeds on the training dataset.
- Train this model using the dog dataset to improve its accuracy.
- Test the new model and verify final accuracy.
- Write an algorithm that receives an image and, using the models described above, verify if it is that of a dog or a human.
- Predict which dog breed is in the image or with which dog breed the human in the picture looks like.

3.3 Refinement

When using transfer learning with the VGG-11 pretrained model, at first it only reached 46% of accuracy. Even trying to change the hyperparameters and batch size would make no difference on the final accuracy. What really made it reach 86% was shuffling the training data before feeding it to the model.

4 Results

4.1 Model Evaluation and Validation

The model achieved a prediction accuracy of 86% when exposed to the test dataset. Using images of dogs chosen randomly on the Internet, I obtained the results presented on Table 1. Only two cases did not obtain the expected result (pictures 5 and 6), but the predicted dog breed is very similar to the target breed.

On Figure 6 we can see the images for both cases. A Shih tzu (Figure 6a) was provided as input, but a Lhasa apso (Figure 6b) was predicted. The same happened for the Staffordshire bull terrier (Figure 6c) used as input and the American staffordshire terrier (Figure 6d) received as prediction.

	Target	**Prediction**
Picture 1	Alaskan malamute	Alaskan malamute
Picture 2	French bulldog	French bulldog
Picture 3	Afghan hound	Afghan hound
Picture 4	Boxer	Boxer
Picture 5	Shih tzu	Lhasa apso
Picture 6	Staffordshire bull terrier	American staffordshire terrier

Table 1: Comparison between target values and prediction results.

4.2 Justification

The Table 2 presents a comparison between the results obtained for the proposed model and the benchmark model.

	Benchmark model	**Proposed model**
Epochs	80	20
Training loss	3.437396	0.830221
Validation loss	3.561250	0.534802
Test accuracy	20%	86%

Table 2: Comparison between benchmark and proposed models.

It is possible to see that the proposed model performs a lot better than the benchmark model. As we used transfer learning, the model only required 20 epochs to reach an accuracy of 86%, which is a satisfactory result.

5 Conclusion

5.1 Reflection

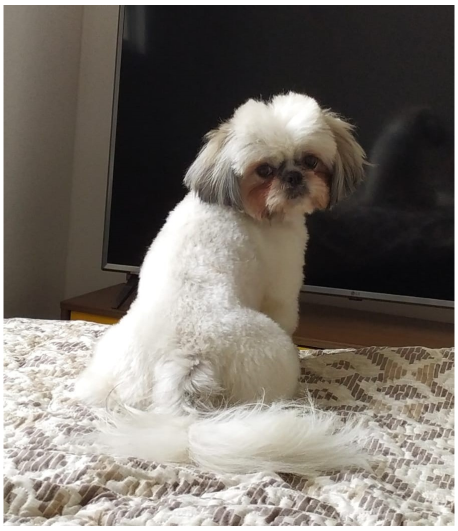
This project was very important for me to understand the different steps of creating a machine learning model. From data exploration, to preprocessing, training, and testing, all the steps were included in this project.

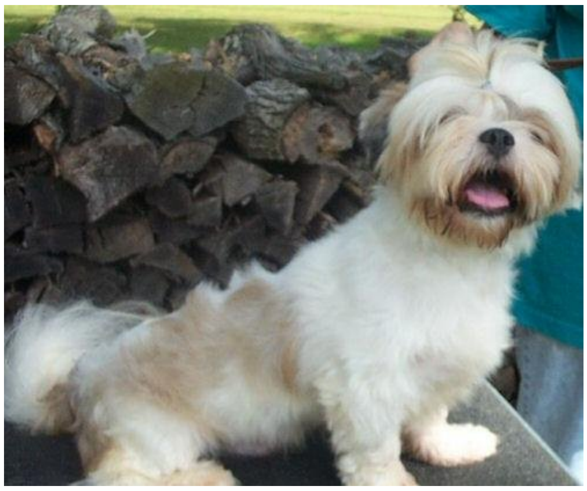
The most difficulty I had was making the configuration of the Convolutional Neural Network for the Python program to run correctly without any exceptions being raised.

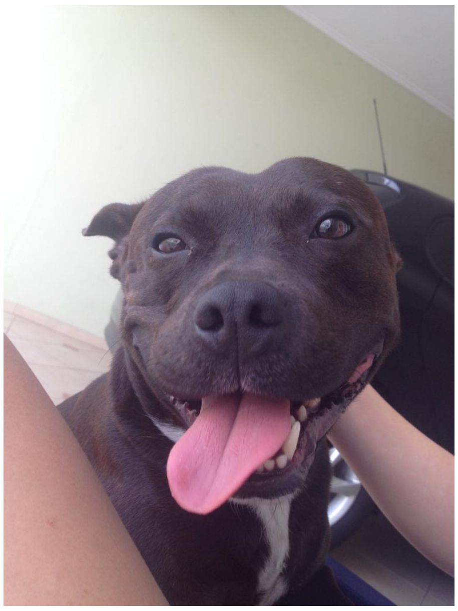
5.2 Improvement

Some possible points of improvement are:

- Add functionality to detect dog mutts;
- Improve human face detection, as there were human faces detected on some dog images;
- Change the algorithm to try and reach over 90% accuracy.


(a) Target: Shih tzu


(b) Prediction: Lhasa apso


(c) Target: Staffordshire bull terrier

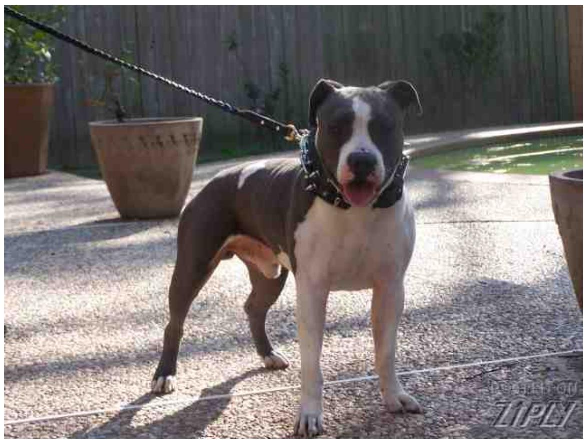

(d) Prediction: American staffordshire terrier

Figure 6: Comparison for wrongly classified dog breeds.