# Write an algorithm for Dog Identification from images using Convolutional Neural Network

Kerim Kutluhan Tunalıoğlu

February 2023

## 1 Project Definition

The project is a part of my Udacity Nanodegree Program assignments.

### 1.1 Project Overview

This project is about how dog breeds could be identified from images using Convolutional Neural Networks. The Jupyter Notebook may be turned into a .py file in order to be used as the backend of an application. The algorithm works as follows:

- An image is fed to the algorithm by the user.
- The image first is checked against a sub-algorithm to detect whether the picture provided includes a human face.
- If a human face is not found, then another sub-algorithm is employed in order to determine whether a dog face is detected.
- If these two types of faces are detected (dog or human) then the algorithm presents its prediction of dog breed that resembles the most with the contents of the image fed.
- If there is not a match in terms of human or dog faces in the given image, then the algorithm prints out a statement noting that there is no faces detected and therefore no prediction is made.

### 1.2 Problem Definition

In order to identify dog breeds from images, 3 different CNN algorithms are either full or partially trained across the dataset provided by Udacity. If the image fed to the algorithm is recognized as a human or dog face, most resembling dog breed type is presented as the trained algorithm's prediction on what is shown in the image.

### 1.3 Metrics

Accuracy is chosen to be the sole metric in this assignment due to time limit because of events taking place at this moment.

## 2 Analysis of Data

### 2.1 Data Exploration

In order to check the performance of the dog and face detector sub-algorithms, 100 images are used out of 13233 images of human faces dataset provided by Udacity.

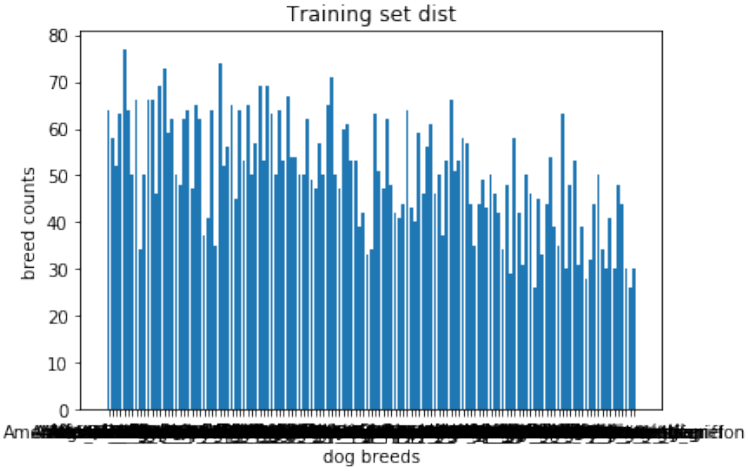
During the training, validating and testing of the CNN algorithms a total of 8351 dog images, provided by Udacity, comprising of 133 different dog breeds are employed. These images are divided as to be used as:

- 6680 images for training
- 835 images for validation
- 836 images for testing purposes.

### 2.2 Data Visualisation

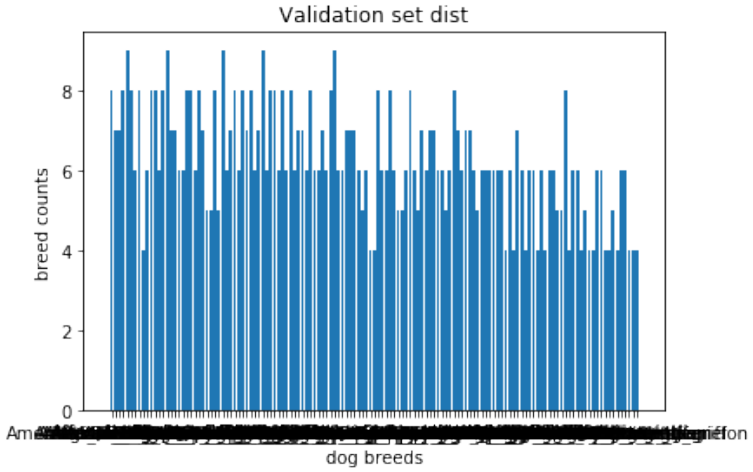
The training set data is visualised below:

Figure 1: Train Set



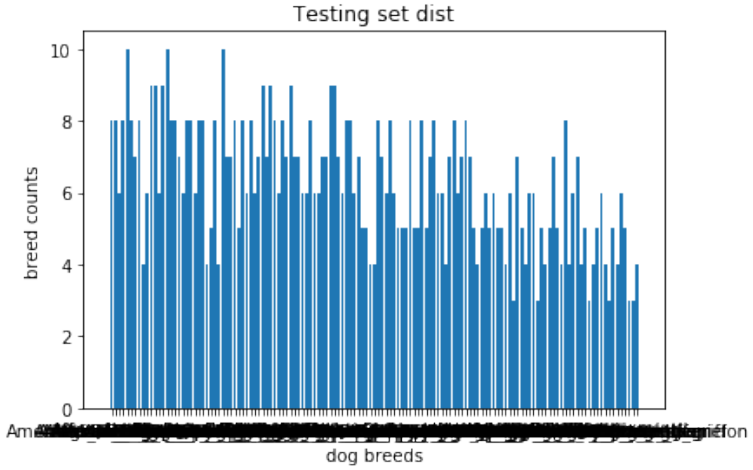

The validation set data is visualised below:

Figure 2: Validation Set




The test set data is visualised below:

Figure 3: Test Set




# 3 Methodology

## 3.1 Data Preprocessing

For human images, the face detector sub-algorithm uses OpenCV's python library's method CascadeClassifier in order to turn them into grayscale version

as is common practice in CNN processes.

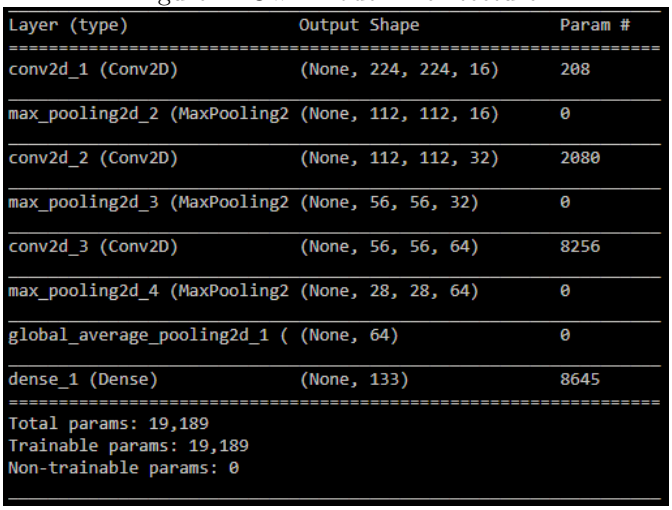
For dog images, the dog detector sub-algorithm employs libraries such as PIL, Keras along with ResNET-50's capabilities in order to turn images from 3D to 4D tensors and prepare for training the algorithm.

## 3.2 Implementation

There are a total of 3 CNN trained with data and are listed below:

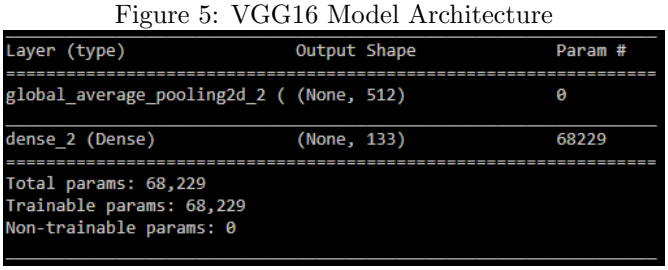
- Own algorithm with 3 deep layers. Accuracy is found to be around 2.9% after 7 epochs. The model architecture is shown below:

Figure 4: Own Model Architecture



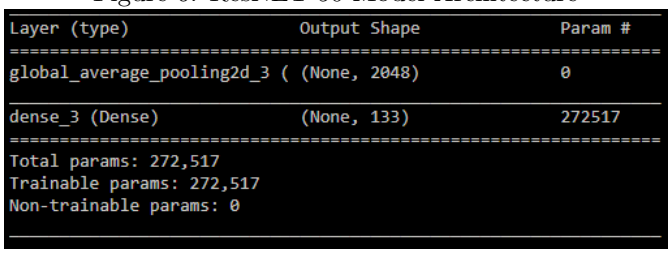

- VGG16 model with pre-set weights along with a final global average pooling 2 dimensional layer. Accuracy is found to be around 41.3%. The model architecture is shown below:

Figure 5: VGG16 Model Architecture



- ResNET-50 model with pre-set weights along with a final global average pooling 2 dimensional layer. Accuracy is found to be around 83.3%. The model architecture is shown below:

Figure 6: ResNET-50 Model Architecture




### 3.3 Refinement

As one can see, own algorithm provides poor results due to low number of layers and epochs trained. And therefore it would have taken a long while in order to attain acceptable accuracy ($> 60\%$ was chosen for this project). Considering these facts, pre-trained CNNs were found suitable for use since most of the necessary computation were already done and ready-to-go. To these ready-to-go CNN with weights attained, only 2 last layers were added, which are a global average pooling layer in 2D and a final dense layer, in order to form efficient transfer learning results. Final and most accurate model was acquired using transfer learning methodology with ResNET-50 bottleneck features.

## 4 Results

The sub-algorithm called face detector used for detection of human faces was tested across 100 dog images and 100 human images with accuracy results listed below:

- Human faces detected in 100 human face images provided in ratio: 100%, therefore it has 100% accuracy.
- Human faces detected in 100 dog face images provided in ratio: 11%, therefore it has 89% accuracy.

The sub-algorithm called dog detector used for detection of dog faces was tested across 100 dog images and 100 human images with accuracy results listed below:

- Dog faces detected in 100 human face images provided in ratio: 0%, therefore it has 100% accuracy.
- Dog faces detected in 100 dog face images provided in ratio: 100%, therefore it has 100% accuracy.

Own model architecture has reached around 2.8% accuracy in correctly identifying the breed type of the test set of the dog images provided.

VGG16 model architecture has reached around 41.3% accuracy in correctly identifying the breed type of the test set of the dog images provided.

ResNET-50 model architecture has reached around 83.3% accuracy in correctly identifying the breed type of the test set of the dog images provided.

Test set of the dog images provided contains at total 836 images.

A demonstration of the final algorithm (ResNET-50 used with transfer learning) can be shown below. All images are found on internet by myself, and are therefore not included in the training, validation and test sets:

Figure 7: Beagle

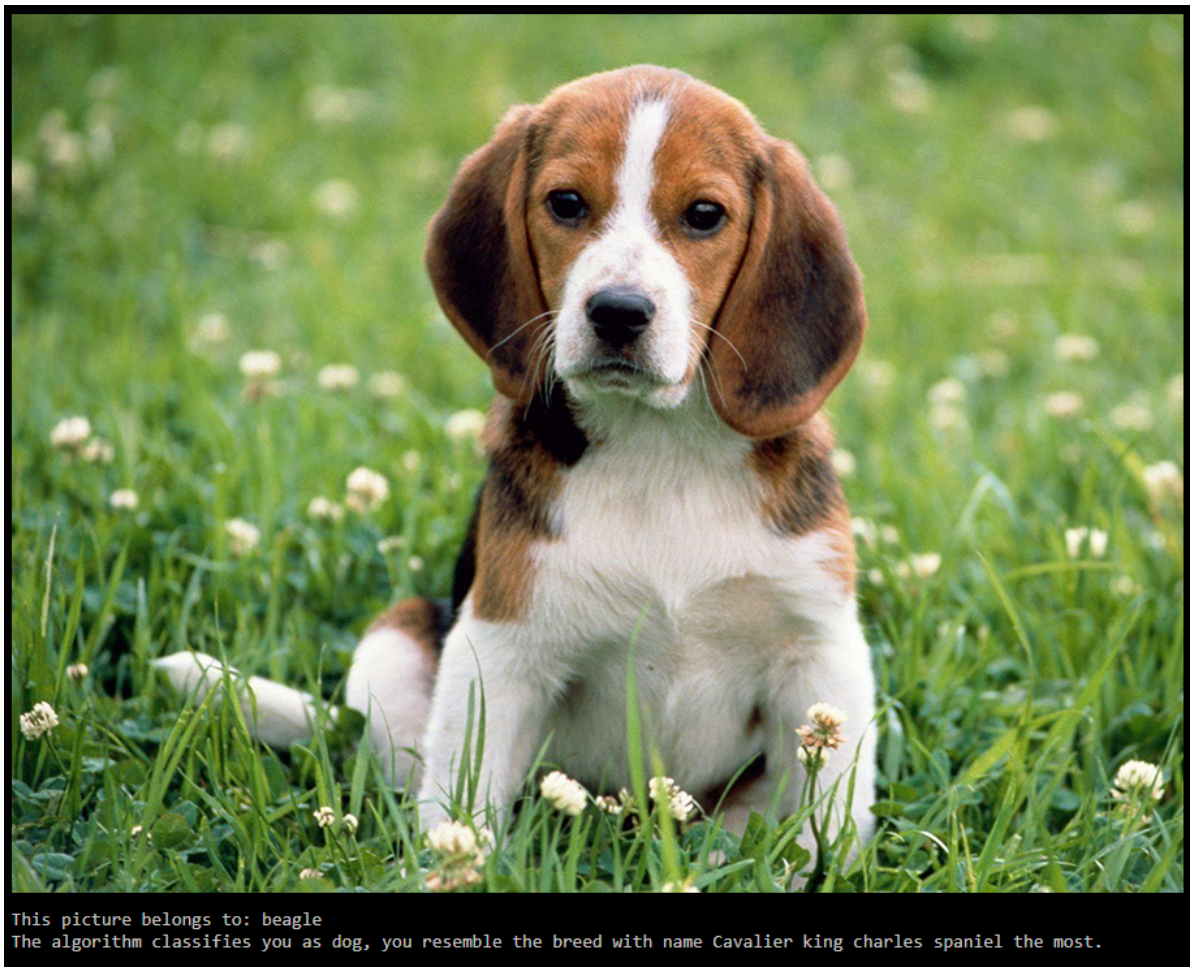


Figure 8: Stonhenge


Figure 9: Jack London

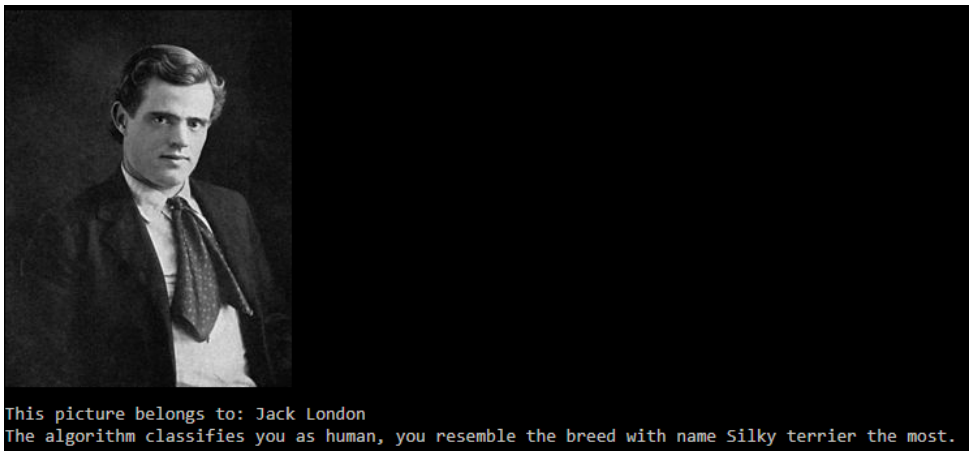


Figure 10: Sphinx

Figure 11: German Shepherd


Figure 12: Aldous Huxley

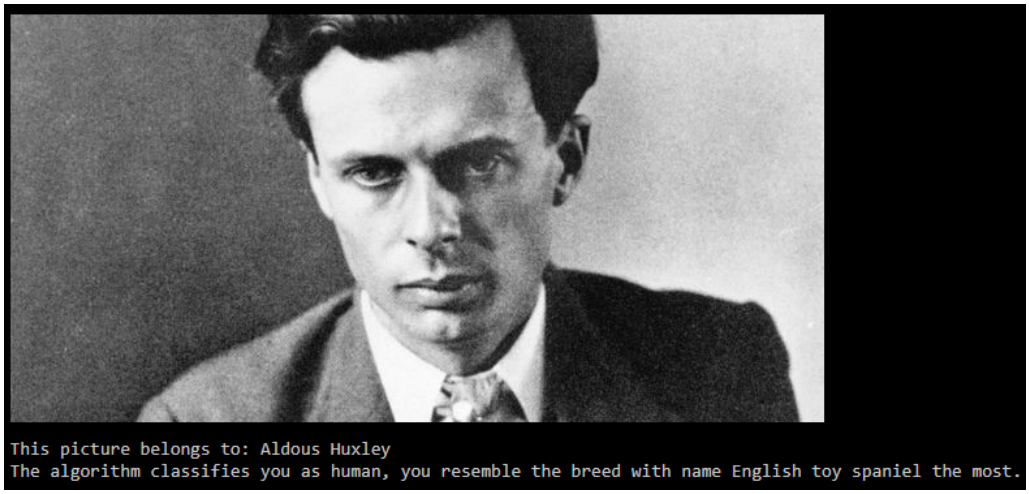


Figure 13: Wreck


# 5 Conclusion

As a summary, the project contains the following steps:

1. A sub-algorithm for detecting human faces are created.
2. A sub-algortithm for detecting dog faces are created.
3. Own CNN algorithm is created for dog breed identification, trained using the dataset provided by Udacity.
4. VGG16 CNN model is trained using transfer learning method for dog breed indentification, trained using the dataset provided by Udacity.
5. ResNET-50 CNN model is trained using transfer learning method for dog breed indentification, trained using the dataset provided by Udacity.

# 6 Improvements

Following possible improvements are found suitable for consideration:

- Own algorithm should be more complicated in terms of more layers in the deep area of the network. There should be more epochs in order to find the point in which overfitting starts.
- VGG16 and ResNET-50 models employed using transfer learning methods could be tried to cut shorter than it was implemented here. In this project, only last 2 layers are adapted for the dataset provided by Udacity.
- All models could be trained with a dataset created from the original dataset by data augmentation.

## For those of us who remain, Thank you for your time!