Machine Learning Engineer Nanodegree Capstone Project Report
Dog Breed Classifier

Jasbir Singh
3-Mar-2021

1. Definition

1.1. Project overview

Dog breed recognition is a challenging problem considering the number of dog breeds and similarities between some of them. This becomes a complex computer vision task for humans as they need to remember all the dog breed features and distinguish between similar looking features. Normal human mind and memory will struggle to solve this problem and high computing power and memory is required to process this information. I believe this could be a potential question in every new dog owner's mind when they go out to buy a new dog. I am sure they would be looking forward to such a service where dog breed could be easily identified using their pictures. And, It will be a big savior when the dog breeds are so similar even expert get it wrong sometimes.

This specific problem domain relates to the image classification problem where many researchers have been working on. Image classification is one of the many specialized areas under computer vision where visual features within image are used to classify an image. The first work on modern convolutional neural networks (CNNs) occurred in the 1990s, inspired by the neocognitron. Yann LeCun et al., in their paper “Gradient-Based Learning Applied to Document Recognition” (now cited 17,588 times) demonstrated that a CNN model which aggregates simpler features into progressively more complicated features can be successfully used for handwritten character recognition. Subsequently, many researchers have taken this work forward and applied CNN models for variety of image recognition problems. Convolutional Neural Networks (CNN) model is a preferable choice here because of their proven track record and accuracy in image classification.

Solution to this problem would involve tying together a series of models for different tasks related to identifying humans, identifying dogs and identifying dogs bread resembling human face. Many researchers have contributed to this specific problem related to computer vision and machine learning. Some of published literature can be referred to develop an understanding of the aforementioned field [5] [6] [7] and apply that knowledge to develop a solution which bears closest resemblance to the specific image classification problem.

This project also involves creation of a CNN model to classify dog breeds. To train the model, I relied on Udacity's dataset which consists of more than 8,000 images of 133 dog breeds. Once trained, validated and tested, the model can be used to predict a dog breed for the a provided dog image.

1.2. Problem Statement

The aim of the project is to build a pipeline to process real-world user-supplied images. The algorithm is expected to identify dog's breed with certain level of accuracy for a given image and if a human image is passed to it, it will try to predict the closest resembling dog breed. If the algorithm is not able to predict human or dog then an error will be returned. Evaluate a model to detect dog breeds, humans and subsequently classify dog breeds that resembles humans. Following are the key problem characteristics, we need to find solution for:

- Identification of a dog or human in a provided image
- Identification of closest matching dog breed for a provide dog or human image
- Apply transfer learning to use features from large image datasets
- Accuracy of >=60% with test data

1.3. Metrics

New CNN model is compared against the benchmark model and multiple metrics were used to compare the performance. In addition to accuracy, other evaluation metrics such as precision, recall and f1 score are used. Additional metrics are primarily selected to address the data imbalance observed in the provided dataset.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Error rate = (FP + FN) / (TP + TN + FP + FN)

Precision = TP / (TP + FP)

Recall = TP / (TP + TN)

F1 Score = 2 * ((Precision*Recall) / (Precision + Recall))

TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative

2. Analysis

I) Data Exploration

Following datasets provided by Udacity are used for training, validation and testing purpose:

- The dog dataset[2] with 8351 dog images.
- The human dataset[3] with 13233 human images.

- ***Dogs Dataset***

The dogs dataset has 8,351 images which are sorted into train (6,680 images), test (836 images) and valid (835 images) directories. Each of the directories train, test and valid has 131 folders corresponding to different dog breeds. Each image in dog breed folder has different size, color, background, and orientation. Number of images in each dog breed folder is not same.

Here are few examples of dogs from dogs dataset:

Australian Cattle Dog

Borzoi

Bloodhound

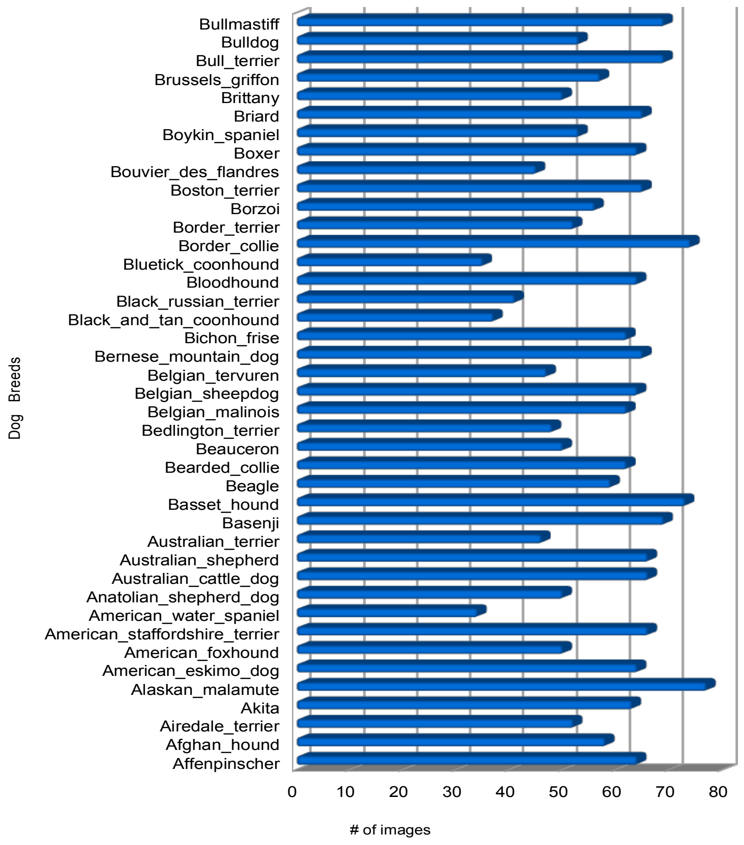
◦ ***Human Dataset***

The human dataset has 13,233 images which are sorted by person name and stored into respective 5,750 folders. 835 images) directories. All the image are of same size but with different background and angles. Each image in dog breed folder has different size, color, background, and orientation. Number of images in each person folder is not same.

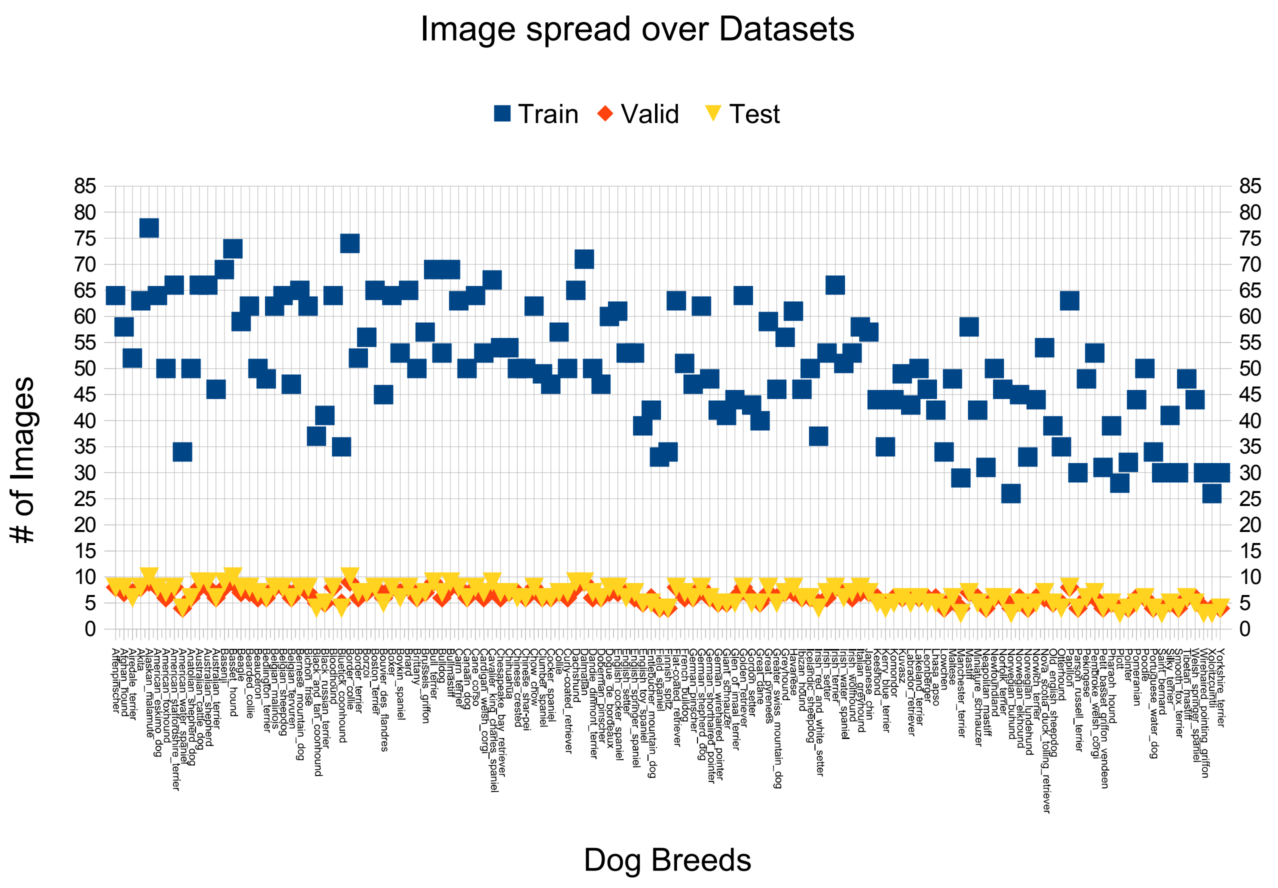
Here are few examples from human data set:

- **Data Visualization**

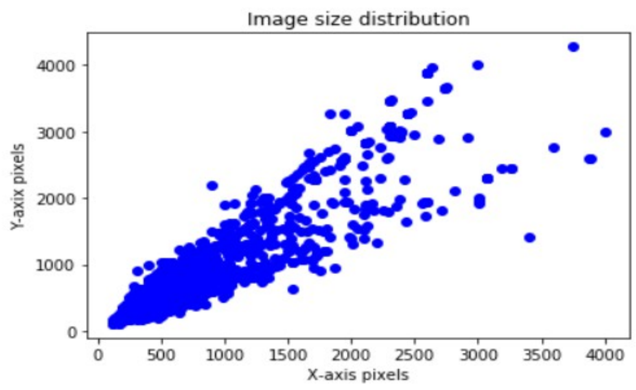
This section describes the data in visual forms and will help us understands the overall data composure. I have used graphs to showcase information contained within dataset. In the following graph, we can see image distribution among 41 dog breeds from A to B across train data set.



We have a good spread of number of images within train datasets and at the same time number of images various for each dog breed. Data augmentation is applied to address the data imbalance issue. In the following graph, we can see image distribution among train, valid and test data sets for 133 dog breeds:



In the following graph, we can see another data imbalance issue with varying image sizes in our dog dataset and data augmentation is applied to address it:



3. Methodology

- **Data Processing**

During our data analysis stage, I noticed data imbalances related to number of images and image sizes for various dog breeds. I have used torchvision's transforms module to meet the data processing requirements.

All the images are first resized into a 256x256 image and then further random cropped or center cropped to 224x224 for train and valid/test data sets respectively. On train dataset, horizontal flip and random rotation of 10 degree is also applied on each image for data augmentation.

Need to reduce the image size to 224x224 pixels is driven from ResNet50's requirements for an this image size as it is pre-trained with it. Finally, all the images are normalized with standard mean and standard deviation.

- **Implementation**

This section describes the algorithms and techniques used to implement the strategy listed in the project design.

  - **Detect Humans**

To detect human faces in the supplied image, I used OpenCV's implementation of Haar feature-based cascade classifiers. First, I used the cascade classifier with the file 'haarcascade_frontalface_alt.xml' in face_detector function and it was able to identify 98% of the humans faces from 100 human images and 17% human faces from 100 dog images. Then, I tried out the cascade classifier with the file 'haarcascade_frontalface_alt2.xml' in face_detector_ext function and this was able to identify 100% of the humans faces from 100 human images and 21% human faces from 100 dog images. Considering better performance with 'haarcascade_frontalface_alt2.xml', I used it for human face detection.

  - **Detect Dogs**

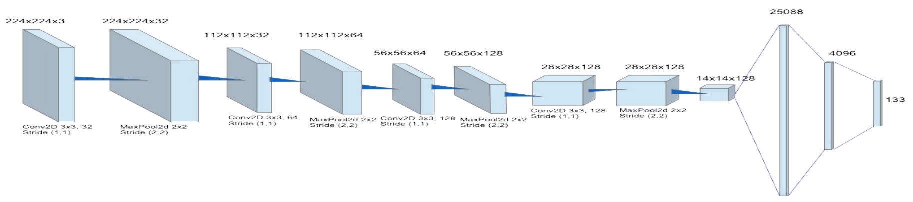
To detect dogs in the supplied image, I used the pre-trained VGG16 model using PyTorch. VGG-16 model, along with weights has been trained on ImageNet, a very large, very popular dataset used for image classification and other vision tasks. This model was not trained and used for prediction only. I transformed the images to 224x224 size, converted to PyTorch tensor and finally normalized using standard mean and standard deviation before input to VGG16 model. Given an image, this pre-trained VGG-16 model returns a prediction (from 1000 possible categories in ImageNet) and in order to validate if an image contains a dog, I checked the returned index to be between 151 and 268 (inclusive) in dog_detector function. To validate the VGG16 model performance, I also tried out another pre-trained model called Inception-v3. VGG16 performed better than Inception-v3 as it was able to predict 100% of the dogs from dog images and 0% from the human images whereas Inception-v3 was able to predict 100% of the dogs from dog images and incorrectly predicted a dog image from human images. I did not look into this further and used VGG16 instead for predicting dogs.

Note: The indexes between 151 and 268 are dog categories.

  - **CNN Model from Scratch**

I created a CNN model from scratch using PyTorch and trained, validated and tested this using the supplied dogs dataset. For CNN model from scratch, I followed standard CNN architecture having convolution layer,pooling layer, fully connected layers and drop out layers. I defined the forward behavior with four convolution layers, pooling layer after each convolution layer to reduce features by half and finally added two linear layers and dropout layers with probability of 25%.

High Level CNN Architecture is as follows:


224x224x3
224x224x32
112x112x32
112x112x64
56x56x64
56x56x128
28x28x128
28x28x128
14x14x128
25088
4096
133
Conv2D 3x3, 32 Stride (1,1)
MaxPool2d 2x2 Stride (2,2)
Conv2D 3x3, 64 Stride (1,1)
MaxPool2d 2x2 Stride (2,2)
Conv2D 3x3, 128 Stride (1,1)
MaxPool2d 2x2 Stride (2,2)
Conv2D 3x3, 128 Stride (1,1)
MaxPool2d 2x2 Stride (2,2)

Step by step forward pass for the model is as follows:

- First Convolutional Layer: 3 inputs, 32 outputs, 3x3 kernel
  - 224 x 224 x 3: 224x224 for image size, 3 for RGB
- Relu Activation function:
  - for non-linearity and better performance over tanh or sigmoid
- Pooling layer: 2x2 kernel
  - reduces the dimensionality of each map and retaining important information
- Second Convolutional Layer: 32 inputs, 64 outputs,3x3 kernel
- Relu Activation function
- Pooling layer: 2x2 kernel
- Third Convolutional Layer: 64 inputs, 128 outputs, 3x3 kernel
- Relu Activation function
- Pooling layer: 2x2 kernel
- Fourth Convolutional Layer: 128 inputs, 128 outputs, 3x3 kernel
- Relu Activation function
- Pooling layer: 2x2 kernel
- Flatten layer: 25088 length single vector
  - fully connected layer expects vector inputs
- Dropout layer: 25% probability
  - to prevent overfitting
- First Fully connected Linear Layer: 25088 inputs, 4096 outputs
- Relu Activation function
- Dropout layer: 25% probability
- Second Fully connected Linear Layer: 4096 inputs and 133 outputs (number of dog breeds)

- **CNN Model using Transfer Learning**

  I used transfer learning to create a CNN model that can identify dog breed from the supplied images. I selected Resnet50 which is one of the pre-trained models on ImageNet similar to other models such as VGG16 or Inception V3. ResNet50 is one of the models with lower Top-1 error rate. I have implemented the ResNet50 model using PyTorch as described her Transfer Learning using ResNet50 in PyTorch.

  I have frozen all the parameters from pre-trained model (required_grad=False) and added a dropout and fully connected layer at the end to fine tune the final classifier. I chose SGD over adam as SGD + momentum can converge better with longer training time compared to adam.

  Step by step process is as follows:

  - Load the pre-trained model
  - Freeze all the model parameters
  - Replace the last fully connected layer
  - Add a dropout layer to reduce overfitting
  - Add a new fully connected layer to classify 133 dog breeds

- **Testing the application**

  The run_app function ties together different pieces for human face detection, dog detection and dog breed classifier. Function first checks for presence of human or dog in the supplied image and if neither one exists, then an error is displayed. If a human or dog is detected in the supplied image then predict_breed_transfer transforms the image and predicts a dog breed out of 133 dog breeds in ImageNet. For a dog image, corresponding dog breed is predicted whereas for a human image, a closest resembling dog breed is predicted.

- **Refinement**

Even though, the CNN model from scratch met the expected accuracy of >=10% (14%) and with the application of transfer learning the accuracy can be further improved. I have selected ResNet50 for CNN model using transfer learning as it is one of the pre-trained models with lower Top-1 error rate. I have frozen all the parameters from pre-trained model and added a dropout (to avoid overfitting) and fully connected layer at the end to fine tune the final classifier and produce 133-dimensional output. With 10 epochs, I am able to achieve 74% accuracy which has met the >=60% accuracy expectation on test dataset.

4. Results

This section describes the evaluation and validation results for various models used to meet the project requirements.

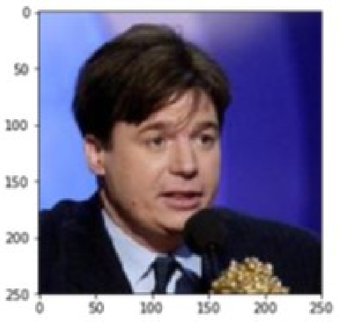
- **Model Evaluation and Validation**

  - **Human Face Detector**

    I tried OpenCV's implementation of Haar feature-based cascade classifiers with the file 'haarcascade_frontalface_alt.xml' and 'haarcascade_frontalface_alt2.xml' and both produced different results. With file 'haarcascade_frontalface_alt.xml', 98% of the human faces were detected in first 100 human images from human dataset and 17% of the human faces were detected in first 100 dog images from dogs dataset. Whereas with file 'haarcascade_frontalface_alt2.xml', 100% of the human faces were detected in first 100 human images from human dataset and 21% of the human faces were detected in first 100 dog images from dogs dataset. I selected the cascade classifier with the file 'haarcascade_frontalface_alt2.xml' considering its performance.

- **Dog Face Detector**

  I tried two pre-trained models VGG16 and Inception-v3 for dog face detection. Both the models performed at par with each other for 100 images from human and dog datasets. Both the models were able to identify 100% of the human faces in 100 human images. VGG16 was successful in not identifying any dog in 100 human images whereas Inception-v3 incorrectly identified one dog in 100 human images. Considering this discrepancy, I chose VGG16 over Inception-v3 even though the performance was comparable. Image with incorrect dog detection:



- **CNN from scratch**

  CNN model from scratch trained for 20 epochs produced an accuracy of 14% on test dataset with test loss of 3.74. Precision, Recall and F1 score for this model are as follows:

  - Test Precision score: 0.2170
  - Test Recall score: 0.1429
  - Test F1 score: 0.1262

  Accuracy and F1 score are close to each other and good values for model from scratch with expected accuracy.

- **CNN using transfer learning**

  ResNet50 CNN model using transfer learning trained for 10 epochs produced an accuracy of 74% on test dataset with test loss of 1.22. Precision, Recall and F1 score for this model are as follows:

  - Test Precision score: 0.7641
  - Test Recall score: 0.7221
  - Test F1 score: 0.7260

  Accuracy and F1 score are close to each other and good values for model with transfer learning.

- **Justification**

Both the CNN models, from scratch and using transfer learning performed better than expected (>=10% for scratch, >=60% using transfer learning) with test accuracy of 14% and 74% respectively.

CNN Model from scratch:

- A learning rate of 0.02 was used and worked well
- 20 epochs achieved the expected accuracy of >=10%
- Stochastic Gradient Descent (SGD) was chosen as it performs well with image classification

CNN Model using transfer learning:

- A learning rate of 0.001 was used and worked well
- 10 epochs achieved the expected accuracy of >=60%
- Stochastic Gradient Descent (SGD) was chosen as it performs well with image classification and worked well

5. Conclusion

- **Reflection**

Looking back at the project implementation, following high level steps describe the overall approach :

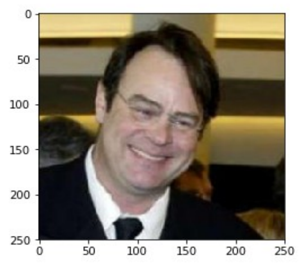
1. Import the required libraries
2. Analyze the available datasets
3. Evaluate the available models and Select model and Test the Human face detector function
4. Evaluate the available models and Select model and Test Dog detector function
5. Preprocess the images including data augmentation
6. Develop, train, validate and test a CNN model from scratch
7. Develop, train, validate and test a pre-trained CNN model using transfer learning
8. Evaluate models against the evaluation metrics
9. Document all the decisions and functions within notebook
10. Maintain all the code and report files in GitHub repository

Initially, I tried 100 epochs for CNN model from scratch but it could only finish 25 iterations after multiple attempts. Finally, I fixed epochs to 20 and was able to finish the training and achieve the required accuracy.

Both the models, from scratch and using transfer learning can be improved further to achieve greater test accuracies. With hyper-parameter tuning, data augmentation and additional epochs results can be further improved to achieve >=85% accuracies.

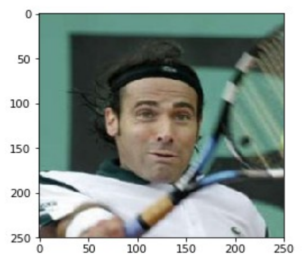
Sample output from the model predictions:

```
Human detected!
```



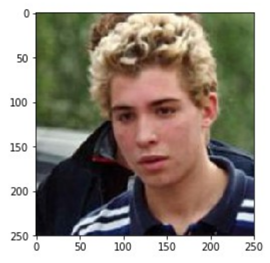
```
You look like a  Chihuahua
```

```
Human detected!
```



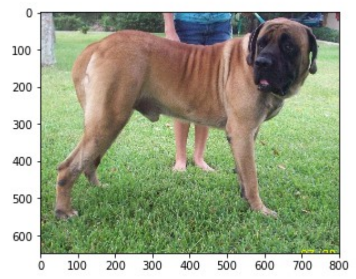
```
You look like a  Bull terrier
```

```
Human detected!
```



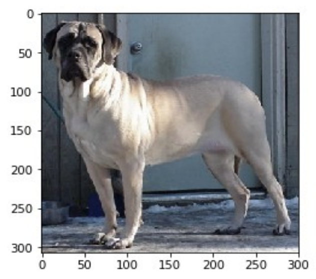
```
You look like a  American water spaniel
```

```
Dog detected!
```



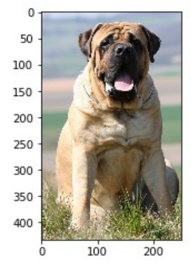
```
Detected breed is:  Bullmastiff
```

```
Dog detected!
```



```
Detected breed is:  Bullmastiff
```

```
Dog detected!
```



```
Detected breed is:  Bullmastiff
```

- **Improvements**
  - Different pre-trained model such as ResNet101 or 152 could be evaluated for better performance
  - Hyper-parameter tuning[8] (weights, learning rate, dropouts, batch size, etc.) can help in model performance improvement
  - Additional data with more dog breeds and/or more data augmentation can certainly help in improving model performance
  - Model can be modified to predict top 3 dog breeds rather than just one
  - Model can be made available to end users via web application

6. References

1. CNN Model: https://en.wikipedia.org/wiki/Convolutional_neural_network
2. Udacity Dogs Data set: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
3. Udacity Human Data set: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip
4. PyTorch Models - https://pytorch.org/docs/stable/torchvision/models.html
5. Modified Deep Neural Networks for Dog Breeds Identification by Aydin Ayanszadeh and Sahand Vahidnia: https://www.researchgate.net/publication/325384896_Modified_Deep_Neural_Networks_for_Dog_Breeds_Identification
6. Cats and Dogs: 2012 IEEE Conference on Computer Vision and Pattern Recognition: https://ieeexplore.ieee.org/document/6248092
7. Dog breed classification using part localization. Computer Vision: https://dl.acm.org/doi/10.1007/978-3-642-33718-5_13
8. Hyper-parameter turning: https://towardsdatascience.com/improving-the-performance-of-resnet50-graffiti-image-classifier-with-hyperparameter-tuning-in-keras-dbb59f43c6f7