Udacity Machine Learning Engineer Nanodegree
Capstone Project Report

Project Title: Dog Breed Classifier

Guru Raj Akuthota

May 17, 2021

Project Overview

Image Classification is an age-old problem dating back to identifying numbers on post cards for automated pin code detection. With more and more complex data available in the world and a need to classify them for various purposes be it identifying a skin cancer (Like the one at: https://www.jmir.org/2018/10/e11936/) to facial recognition on traffic cameras (Like the one at: https://www.researchgate.net/publication/308387275_Face_recognition-based_real-time_system_for_surveillance), there is ever increasing growth in the field of computer vision, which heavily relies upon machine learning for cues for prediction and processing heavy image data on the fly.

The current project aims at exploring one such classification problem, "dog-breed" classification which is a multi-class classification problem. The intent of project is educational and also to incorporate the learnings as a part of Machine Learning Engineer Nanodegree on Udacity.

Problem Statement

The aim of the project is to create an ML pipeline using state of the art Convolutional Neural Network (CNN) Architecture which will be fed with dog images and human images to

1. If an input image is a dog, classify it as dog and output the breed to which it belongs.
2. If an input image is a human face, then output that it is a human and match the human face to a resembling dog breed
3. If it is neither dog nor human, output a corresponding error message

Data Exploration

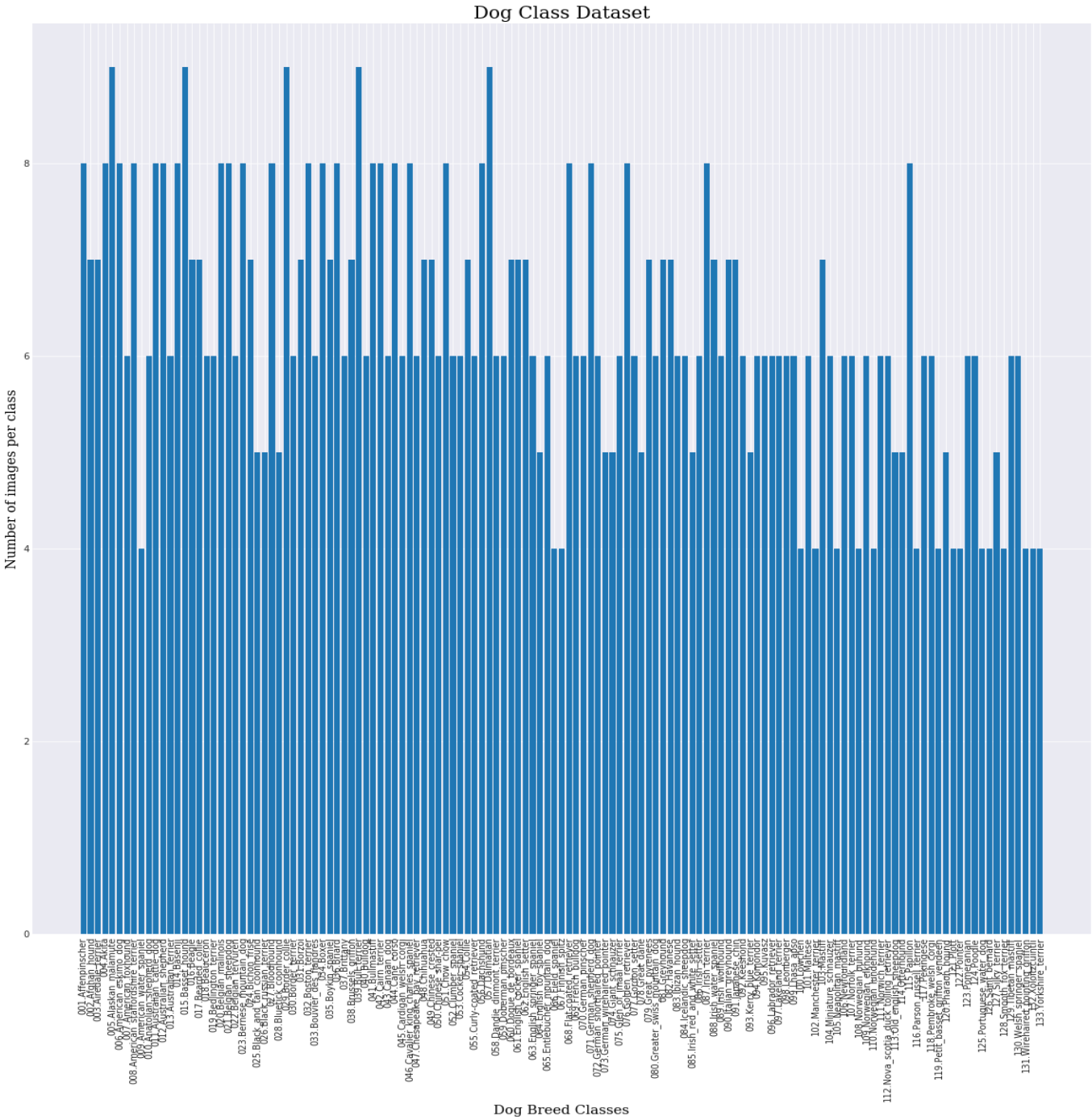
The project uses the dataset provided by Udacity under the project template project-dog-classification.

Datasets Analysis

Dog Breed Data:

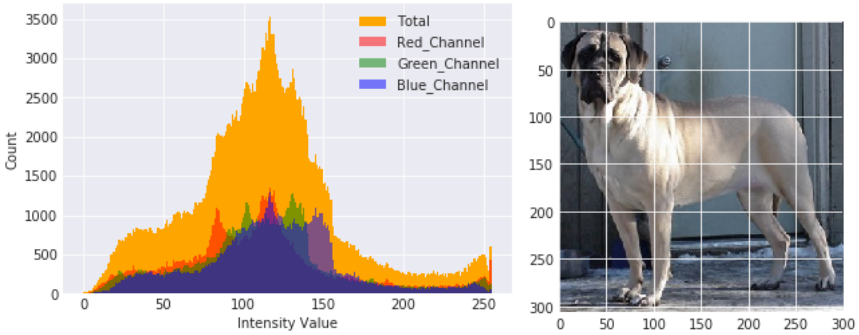
- Image Dataset Source: Udacity
- Size: 8351
    - Train : 6680
    - Test : 836
    - Validation : 835
- Number of classes (dog breeds): 133
- Dataset information:

- Current dataset is imbalanced for each breed (somewhere from 4 to 8)
- The image sizes are disproportionate.

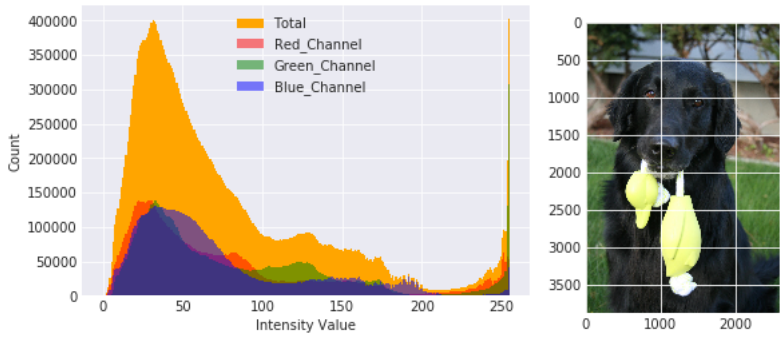


As we can see, 133 classes of dogs are present in dataset, each with a minimum for 4 pictures to a maximum of 9 pictures per folder.

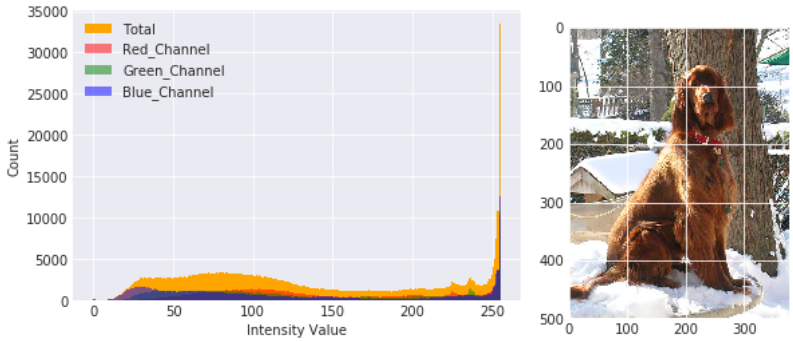
Another interesting analysis is around pixel intensities that show how the pixel intensities are spread across the image, which shows the need for augmenting data for the project so that we can better predict the dog breed based on different characteristics of a dog (nose, ear, legs etc). Also to note that images are not just dog images but also there are lot of background entities like walls, items etc that interfere with the entire image space which could be crucial for data reprocessing.



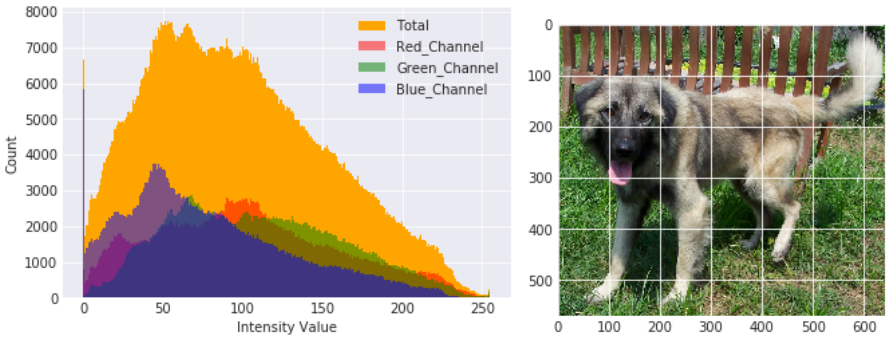
Mastiff



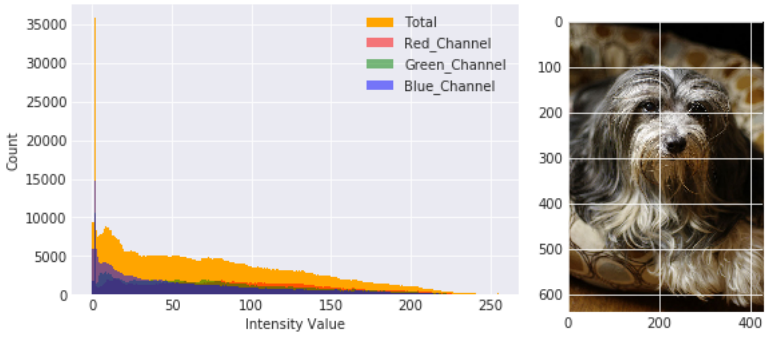
Flat-coated_retriever



Irish_setter



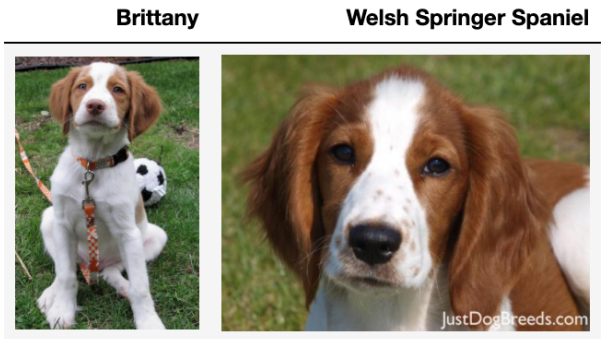
Norwegian_elkhound

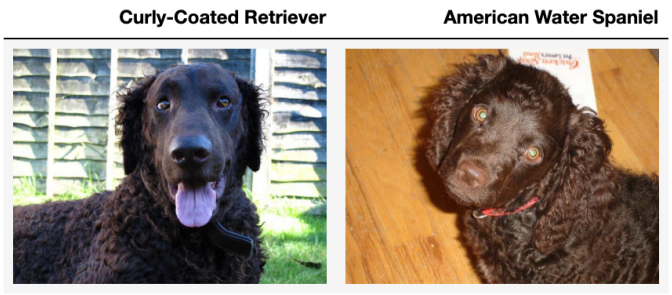


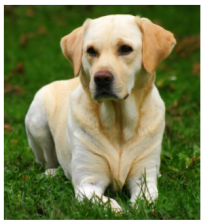
Lowchen

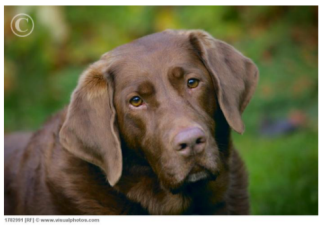
As we can see for some images the intensity curves are biased towards the extremes which could be a deciding factor when weights are computed in the model.

Sample Data from Dataset:







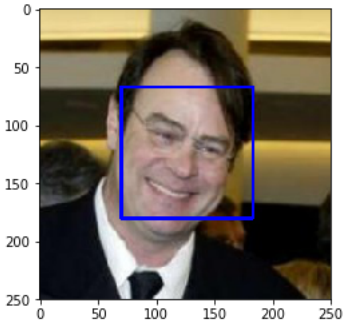


Different colored Labradors (Yellow, Chocolate, Black)

Human Face Data:

- Image Dataset Source: University of Massachusetts
- Size: 13233
- Number of classes (people): 5749
- All images are sorted and equally sized (250 x 250)

Sample Data from Dataset:



Evaluation Metrics

As this is a classification problem, the performance of the ML model will be evaluated on the standard and key decision metrics of:

- Accuracy
- Log Loss

While accuracy of a model talks about number of correct predictions over the total number of predictions, which is a definite requirement for a classification problem to go ahead and use the model for say production work loads.

Log loss is a probabilistic metric that takes into consideration the probabilities of prediction per class and averages it out across the probabilities of prediction classes, thus giving a robust metric to judge the performance of the model compared to ideal probability.

Methodology

Algorithms and Techniques

As per the design tenets, the project followed a top-down approach during implementation.

Project Design

1. Import Datasets and preprocess data
    a. Split Data in train, test and validation set
    b. Scale images to same size (224x224)
    c. Augment the training data
2. Create model for Human Face Detection
    a. Pre-process data
    b. Implement Harr Cascade Classifier
3. Create model for Dog Face Detection
    a. Use pretrained VGG16 Model for modelling dog face detection
4. Implement a CNN for dog breed classification
    a. Create a model from scratch
    b. Train, validate and test the model for different epochs and note the performance
5. Implement a CNN for dog breed classification using Transfer Learning
    a. Leverage ResNet Architecture using transfer learning
6. Algorithm to combine Dog and Human Face Detection and CNN for classification
    a. If dog is detected, return the dog breed
    b. If human is detected, return matching dog breed
    c. Else throw error
7. Testing and evaluating the performance of the algorithm

The following algorithms have been used in the project:

1. Harr Cascade Classifier, a pretrained face detection algorithm provided as a part of OpenCV library to identify and detect human faces.
2. VGG16 Model, a pretrained model based on ImageNet classification dataset is used to detect dogs from the dataset.
3. Scratch Implementation of CNN with the following architecture for classifying dog breed (and also outputting suitable dog breed for human pictures)

4. Resnet 152, a pretrained pytorch model along with fully connected layer as used in sc ratch implementation for achieving better accuracy for prediction. Note: I've also tested a Resnet 101 model, but according to the paper: https://arxiv.org/pdf/1512.03385.pdf, Resnet 152 outperforms Resnet 101 in accuracy. The compar ative findings will be shared in **Results** subsection.

Data Preprocessing

Data preprocessing is the most crucial step for any Machine learning project. The need being to be able to normalize the data, clean the data and to add additional augmentation steps to prevent overfitting of the models.

The following were the preprocessing steps applied to input images before feeding them into training the classifiers

- The images have been cropped/resized to 224 x 224 for ease of implementation with pre-trained models.
- Images have been normalized using standard values for mean and standard deviation (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) as per pytorch documentation.
- Furthermore, images have been augmented with horizontal flips and random rotations to make the model robust for predictions
- The processed data was then converted to tensors (as pytorch is the model used) and loaded into data loaders. Also, the data was batched into a size of 32 and split as train, test and validation data.

Note: For ease of human face detection, the image was converted to grayscale. Apart from that no other transformation was required as all images were for same size and the model was pretrained.

Benchmarking

Before proceeding any further with the implementation, the following benchmark was established (as per the Udacity Project Template) for this project:

- The Scratch implementation of CNN model should have an accuracy of at least **10%**
- The transfer learning implementation of CNN model should have an accuracy of at least **60%**

Scratch CNN Classifier Model

The scratch implementation of the model has a following structure:

```
Net(
  (conv1): Conv2d(3, 36, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(36, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=100352, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=133, bias=True)
  (dropout): Dropout(p=0.3)
)
```

The following describes the model:

- It is a CNN model with three convolutional layers of kernel size 3, stride 1 and padding 1. It takes a 224 x 224 image as input.
- The next part of the model has a max pooling layer to reduce input size to 2.
- Following the max pooling layer two fully connected layers with relu activation function. The final layer has 133 output nodes to classify the dog breeds into their respective classes.
- The model uses a dropout of 0.3 to avoid overfitting
- The model uses a Cross Entropy Loss as loss criteria during training
- The model uses a Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.05
- The model was trained for 20 Epochs

Transfer Learning CNN Model

The downside of the Scratch implementation of CNN model is it achieved an accuracy of 14% which is quite less compared to the state-of-the-art classification models. Hence in order to boost the quality of the project, transfer learning has been used in order to leverage a pretrained model on ImageNet Dataset.

For exploratory purposes I initially used the ResNet 101 Architecture and later used the latest ResNet 152 Architecture for the project purpose.

On top of the model, in order to facilitate it to our prediction requirements, the fully connected layer was added (2048 input nodes, 133 output nodes with same SDG optimizer, CrossEntropyLoss criteria and 0.05 learning rate. Also, dropout of 0.3 was used to keep implementation consistent with scratch CNN.) The model was trained for 10 Epochs (lesser number of epochs than scratch model because on trial and test I've realized there is significant improvement in training and validation loss with lesser epochs and after a certain 12 to 15 epoch range, the improvement didn't go any further up.)

Results

Detection:

Human Face detection:

As observed, the Harr Cascade detector identifies humans with a 98% accuracy and dogs as humans with 17% from dog dataset.

```
Percentage of Humans detected in human dataset:  0.98
Percentage Dogs detected as humans in dog dataset:  0.17
```

Dog Detection:

The VGG 16 Model performs exceptionally well with all dogs detected as dogs with 100% accuracy.

```
Percentage of Humans detected as dogs in human dataset:  0.01
Percentage of Dogs detected as dogs in dog dataset:  1.0
```

Classification:

Scratch CNN:

The scratch CNN meets the benchmark criteria with an accuracy of 14%

```
Test Loss: 3.660247
Test Accuracy: 14% (118/836)
```

CNN with Transfer Learning:

Two transfer models have been evaluated: ResNet101 and ResNet 152. As per the paper: https://arxiv.org/pdf/1512.03385.pdf ResNet 152 outperforms ResNet101 in accuracy as can be seen with test loss of `0.498608` for ResNet 152 compared to `0.499880 for` ResNet 101 after 10 epochs each. But surprisingly the model accuracy for Resnet 101 is marginally better (84%) than Resnet 152 (83%), the reason for which might be the nature of the test dataset.

Anyways, both models exceed the benchmark criteria which is quite a good achievement.

Resnet 101:

```
Test Loss: 0.499880
Test Accuracy: 84% (703/836)
```

Resnet 152:

```
Test Loss: 0.498608
Test Accuracy: 83% (699/836)
```

All in all, the project meets the expectation in terms of benchmark for both scratch (14% accuracy) and transfer learning models (83% for Resnet 152 and 84% for Resnet 101) models respectively.

Improvements for future

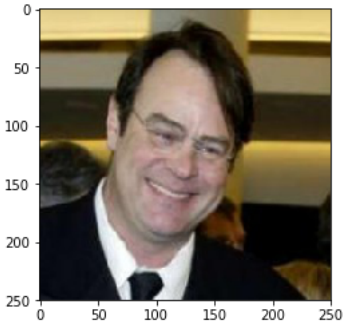
Some of the suggestions based on the learnings have been the following:

- Human accuracy detection can be improved with the use of other state of the art models
- Classification accuracy can be improved by using hyper parameter tuning and trying out other models than CNN (or an ensemble for better predictions).
- Could've used a better optimizer to reduce training time to try out different approaches.
- I could've tried to use k-fold validation and other similar techniques to overcome the issue of lesser data and do data augmentation a bit more aggressively.

Sample Outputs Produced by the model

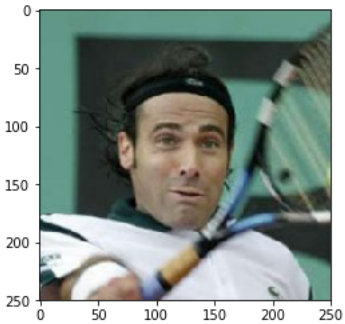
It's a human

This human resembeles the breed: Chihuahua



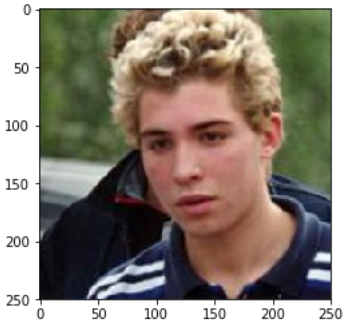
It's a human

This human resembeles the breed: American staffordshire terrier



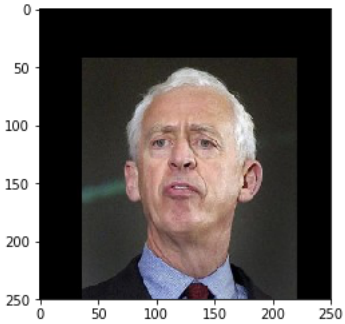
It's a human

This human resembeles the breed: American staffordshire terrier



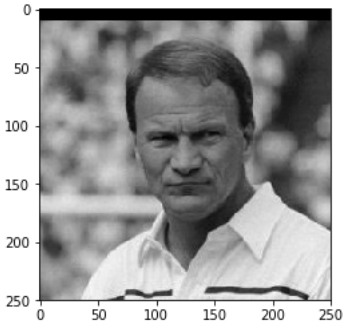
It's a human

This human resembeles the breed: American staffordshire terrier



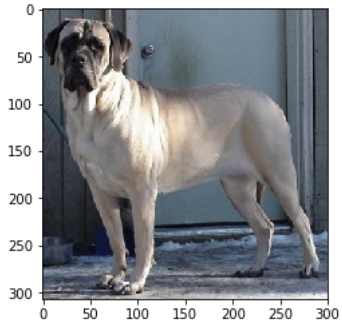
It's a human

This human resembeles the breed: Cane corso



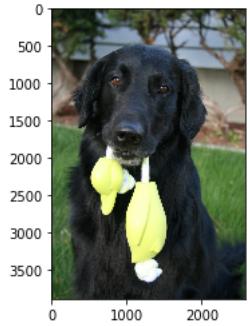
It's a dog!

To be precise, its a dog of breed: Mastiff



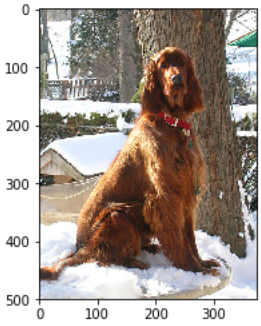
It's a dog!

To be precise, its a dog of breed: Flat-coated retriever



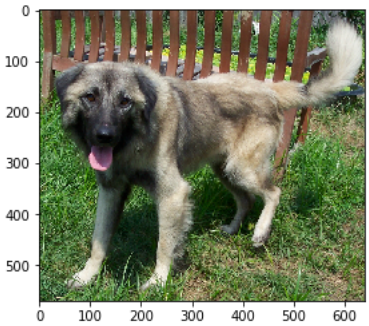
It's a dog!

To be precise, its a dog of breed: Irish setter



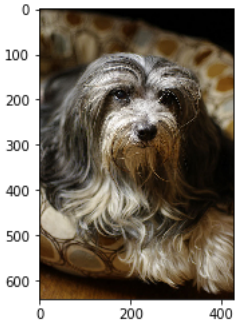
It's a dog!

To be precise, its a dog of breed: Norwegian elkhound



It's a dog!

To be precise, its a dog of breed: Havanese



Learnings

The project was a good learning experience and I got to learn the following:

- Performing data analysis and exploring datasets for machine learning use cases
- Learnt useful data pre-processing techniques that are critical in preparing data before training and validation
- Being able to design and execute a machine learning project from end to end and see-through different challenges encountered in the way
- Learning pytorch and OpenCV as the part of project which are important libraries
- Exploring use of transfer learning, a powerful and time saving technique that helps in building up on pre-trained models instead of reinventing the wheel.
- Being able to create a multi class classifier on my own and have fun along the way.

References

- Udacity Dog Breed Classification Project Template: https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification
- Udacity Project Evaluation Guidelines: https://review.udacity.com/#!/rubrics/2259/view
- ResNet50L https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33?gi=bbecc705c57a
- ResNet Paper: https://arxiv.org/pdf/1512.03385.pdf
- Harr Cascade Classifier: https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html
- LFW Dataset: http://vis-www.cs.umass.edu/lfw/
- CNN: https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1
- Transfer Learning: https://machinelearningmastery.com/transfer-learning-for-deep-learning/