

# UDACITY

## Data Science Capstone Project

### Use of Convolutional Neural Networks to Identify Dog Breeds

By

**Success Attoni**

In Fulfilment of the Award of a Data Science Nanodegree

January, 2016

---

#### Content

<b>1.</b>	<b>Project Overview.....</b>	<b>2</b>
1.1.	General	2
1.2.	Problem Statement	2
1.3.	Metrics	2
<b>2.</b>	<b>Analysis.....</b>	<b>2</b>
<b>3.</b>	<b>Methodology.....</b>	<b>3</b>
<b>4.</b>	<b>Results .....</b>	<b>8</b>
<b>5.</b>	<b>Conclusion .....</b>	<b>10</b>

## 1. Project Overview

### 1.1. General

Computer vision is one of the major advances in artificial intelligence technology. Today, this technology is at the heart of processes/activities in various industries such as the automobile, medicine, security, communication, etc. This project sets out to develop a Dog Breed Identification System using convolutional Neural Networks.

### 1.2. Problem Statement

The high-level task of the project is to develop a Dog Breed Identification System using convolutional neural networks. This system will take an image and predict whether the image contains a dog, a human, or neither of these. If the image contains a dog, the system will predict the breed of the dog, if it contains a human, it will predict the dog breed that the human resembles the most. If the image contains neither a dog or a human, the system reports that the image contains neither a dog or a human. The key subtask in the project include:

- Develop a human face detector: a routine that can detect if an image contains a human face
- Develop a dog detector: a routine that can detect if an image contains a dog
- Develop a dog breed classifier using convolutional neural networks: a machine learning model that can predict a dog's breed given an image of the dog
- Develop a function that uses the combines/uses the 3 products above to classify dog breeds.

### 1.3. Metrics

The prediction confidence of the various aspects of this system will be measured using "Accuracy". Where accuracy is defined as:

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

TP = True positives

TN = True Negatives

FP = False Positives

FN = False Negatives

In other words, accuracy is the ratio of correct predictions to the sample(data) size.

Accuracy was selected for this problem because the prediction classes are reasonably balanced. (see *Figure 2.1*).

## 2. Analysis

The dataset used in the analysis consists of 8,351 dog images across 133 dog breeds and 13,234 human (face) images. The dog images were used to train, test, and validate the classifier. The dog images are split into 3 sets as per the table 2.1 below:

Table 2.1 Image Count Per Dataset

Set	Training	Testing	Validation
Count of Images	6680	835	836

An analysis of the count of dog images per class in the training set is as shown in the visualization present in *Figure 2.1*

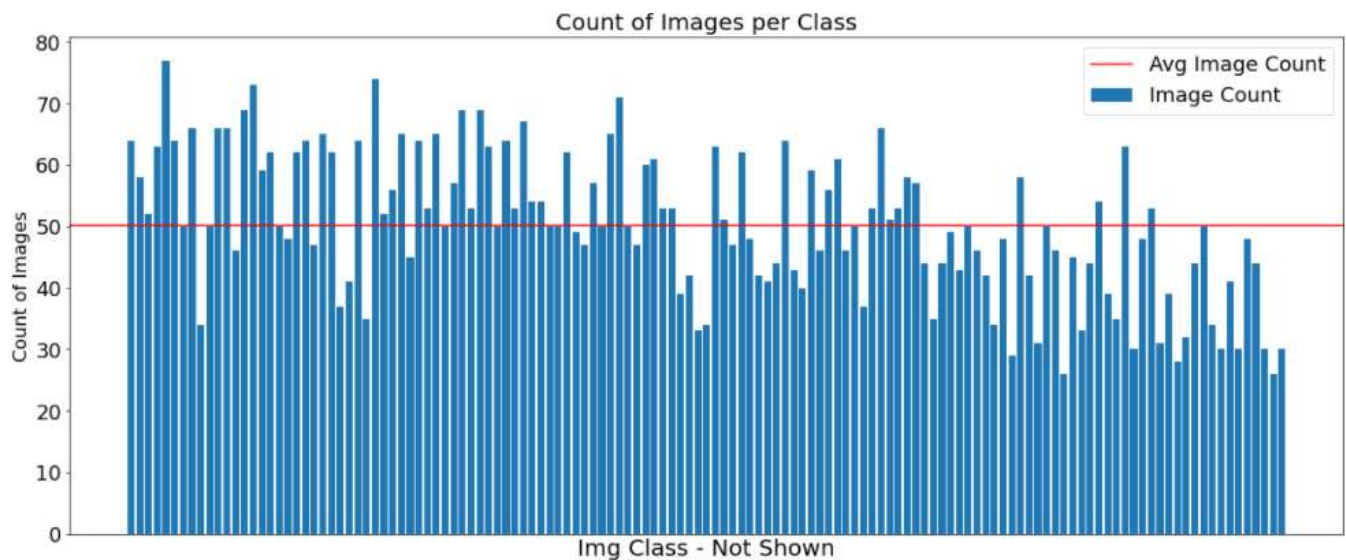


Figure 2.1 Image Count Per Class (Breed)

The human images were used to calibrate the performance of various pre-built human face detection algorithms (more details are presented in section 3)

### 3. Methodology

The steps used in building the project is as shown in the flow chart in *Figure 3.1*.

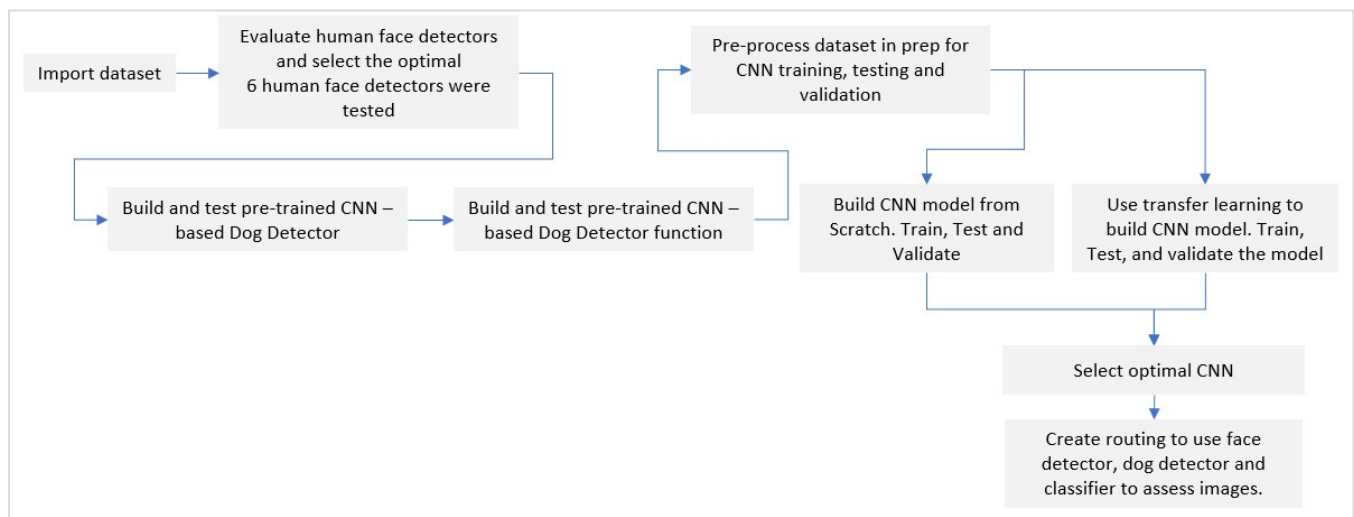


Figure 3.1 Project Flow Chart

### Data Processing:

Various transforms were applied to the images to prepare them as input into the algorithms. For the “OpenCV Haar feature-based cascade classifiers”<sup>1</sup> the images were resized and converted to BGR (Blue, Green, Blue), and subsequently greyscale before applying the algorithm to detect human faces. For the “OpenCV DNN (Deep Neural Network)”<sup>2</sup> Caffe and Tensorflow based face detectors – the images were resized to dimensions required by the respective algorithms.

All images fed to the pretrained CNNs were normalized in the same way; i.e. mini-batches of 3-channel RGB images of shape (3 x H x W), where H and W are expected to be at least 224. The images have to be loaded into a range of [0, 1] and then normalized using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]<sup>3</sup>. The training set for the CNNs were also augmented by applying transforms such as HorizontalFlip, and Rotation by 10 deg. *Figure 3.2* shows some of the sample dog images in the CNN training dataset.



*Figure 3.2 Sample dog images from Training dataset*

### Implementation:

This section presents the implementation of the algorithms for the human face detector, dog detector, dog breed classifier and the routine that uses these 3 products to run the system.

#### *Human Face Detector:*

Open-source human face detectors were adopted to identify images with human face. The algorithms were tested and the optimal detector selected based on performance (execution time and accuracy). The algorithms tested are listed below:

---

<sup>1</sup> OpenCV Cascade Classifier: [https://docs.opencv.org/3.4/db/d28/tutorial\\_cascade\\_classifier.html](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html)

<sup>2</sup> OpenCV: <https://datahacker.rs/face-detection-video-opencv/>

<sup>3</sup> PyTorch guide: [https://pytorch.org/hub/pytorch\\_vision\\_vgg/](https://pytorch.org/hub/pytorch_vision_vgg/)

Table 3.1 Image Count Per Dataset

S/N	Algorithm	Comment
1	Dlib_CNN	<a href="http://dlib.net/">http://dlib.net/</a> <a href="https://learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/">https://learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/</a>
2	Dlib_HOG	
3	OpenCV DNN in TF	<a href="https://datahacker.rs/face-detection-video-opencv/">https://datahacker.rs/face-detection-video-opencv/</a>
4	OpenCV DNN Caffe	
5	OpenCV face detector2	Based on haarcascade_frontalface_alt_tree.xml <a href="https://github.com/opencv/opencv/tree/master/data/haarcascades">https://github.com/opencv/opencv/tree/master/data/haarcascades</a>
6	OpenCV face detector	Based on haarcascade_frontalface_alt.xml <a href="https://github.com/opencv/opencv/tree/master/data/haarcascades">https://github.com/opencv/opencv/tree/master/data/haarcascades</a>

These algorithms were tested and compared before an optimal algorithm was selected. In the test, the algorithms were used to detect humans faces in 100 human face images and 100 dog images, and the percentage accuracy and elapsed time measured. The result of the test is presented in the *Figure 3.3* below.

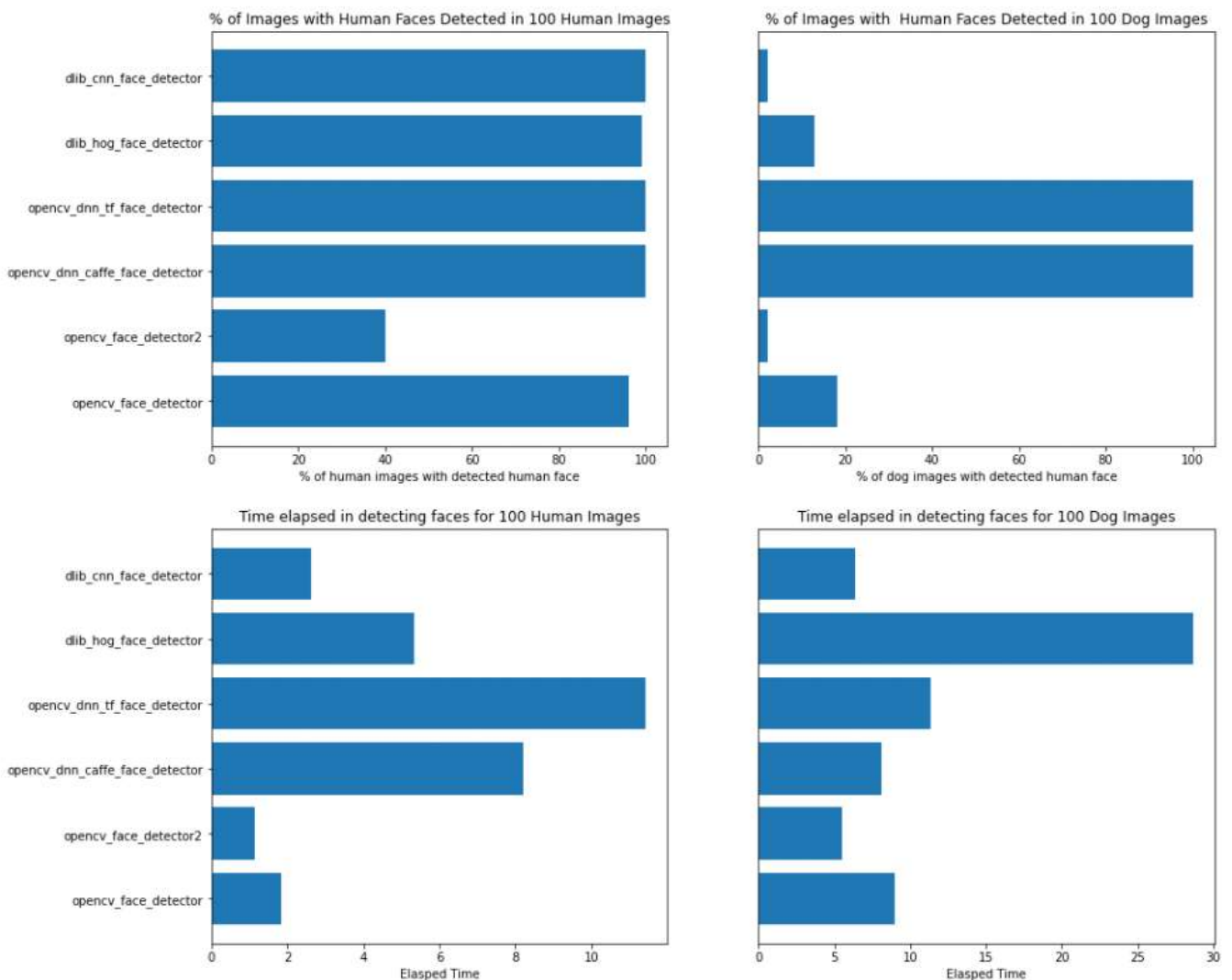


Figure 3.3 Comparison of Face Detector Algorithm Performance

From the figure, dlib CNN face detector was selected as the optimal algorithm for the project, because it has a high accuracy in detecting human face in the human images and not detecting human faces in dog images.

The dlib\_cnn algorithm was implemented in the dlib\_cnn\_face\_detector() function utilizing the “face recognition”<sup>4</sup> library. The function takes an image, identifies the human faces in the picture and return True if a human face is detected; False if otherwise.

#### *Dog Detector:*

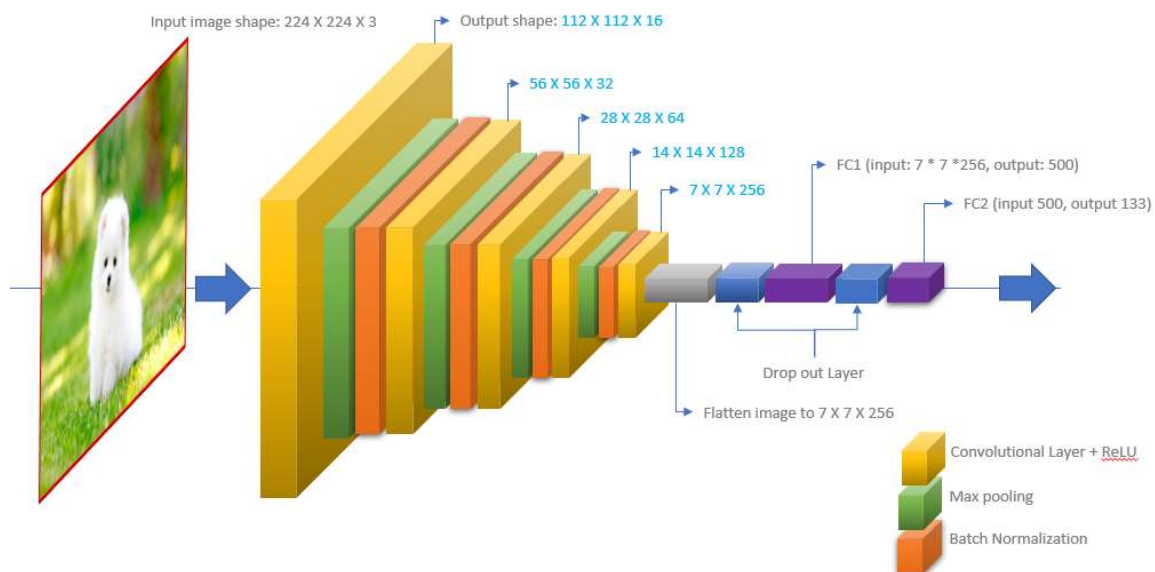
The dog face detector was built based on a pre-trained VGG-16 CNN architecture as implemented in pytorch. The model was pretrained on the ImageNet dataset with dog classes lying between an index of 151 to 268. The dog detector takes an image and returns True if a class index between 151 and 268 (inclusive) is predicted, otherwise it returns a false. The dog detector was tested and in 100 images of human faces it detected that 0% were dogs, in 100 images of dog faces it detected that 93% were dogs.

#### *Dog Classifier:*

Two dog classifiers were built, one from scratch and the other, using transfer learning. Both classifiers were built using pytorch.

#### CNN from Scratch.

After trying different architecture, constrained by runtime and memory limitations; the architecture selected for the CNN built from scratch is a shown in the schematic in *Figure 3.4* below.



*Figure 3.4 Dog Classifier Architecture for CNN built from Scratch.*

The architecture takes an input image of 224 x 224 (x 3 colour channels) and passes it through 5 convolutional layers.

The convolutional layers have a kernel of size 3 X 3 after each of the 1<sup>st</sup> to 4<sup>th</sup> convolutional layer a 2 X 2 max polling layer and batch-normalization (to standardize the input to the next layer) is applied. After the 5<sup>th</sup> layer the tensor is flattened and fed to a dropout layer then to a fully connected layer, subsequently followed by another drop out

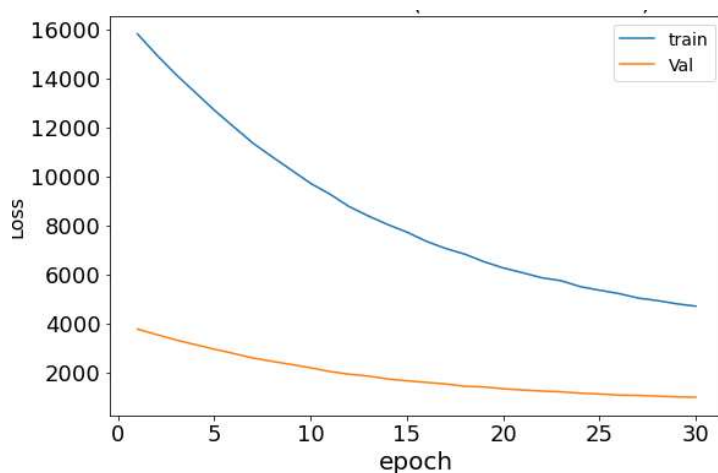
<sup>4</sup> Face-recognition library: <https://pypi.org/project/face-recognition/>

layer and fully connected layer, resulting in an output dimension of 133 (class probabilities for the number of dog breed classes in the training data set). The loss function was based on Cross Entropy Loss and the optimizer was based on Stochastic Gradient Descent (SGD). The learning rate was set to 0.001.

This architecture achieved an accuracy of 20% on the test dataset.

### CNN from Transfer Learning

The CNN built by application of transfer learning was based on the resnet50<sup>5</sup> (refer to the reference for details) architecture as implemented in pytorch. All layers were frozen to training, except the final layer which was edited to ensure that the output matches the dog breed class count of 133. The final layer was then trained on the training dataset. The loss function was based on Cross Entropy Loss and the optimizer was based on Stochastic Gradient Descent (SGD). The learning rate was set to 0.001. This model achieved an accuracy of 81% and was used as the classifier for the project. The training vs Validation curves are shown in *Figure 3.5*. The model was trained 30 epochs. The curves shows that the model can be trained for a bit longer to reduce the losses further, however due to time constraints the training was stopped at 30 epochs.



*Figure 3.5 Training Vs Validation Loss (Transfer Learning CNN)*

### *App routine*

This routine ties it all together; it utilizes the human detector, dog detector and dog classifier, thus presenting a user friendly way to run the workflow in jupyter notebook. The routine takes a list of image paths obtained by making selections from an “open file dialog” checks each image against the human detector, dog detector and the classifier (in series). The workflow for the routine is as shown in *Figure 3.6* below

### **Refinement:**

The to improve model predictability the images were augmented by applying transforms to the training data set as below:

```
train_data_transforms = transforms.Compose([transforms.Resize(256),
                                           transforms.CenterCrop(224),
                                           transforms.RandomHorizontalFlip(),
                                           transforms.RandomRotation(10),
                                           transforms.ToTensor(),
                                           transforms.Normalize(mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225])])
```

<sup>5</sup> Resnet50: [https://pytorch.org/hub/nvidia\\_deeplearningexamples\\_resnet50/](https://pytorch.org/hub/nvidia_deeplearningexamples_resnet50/)



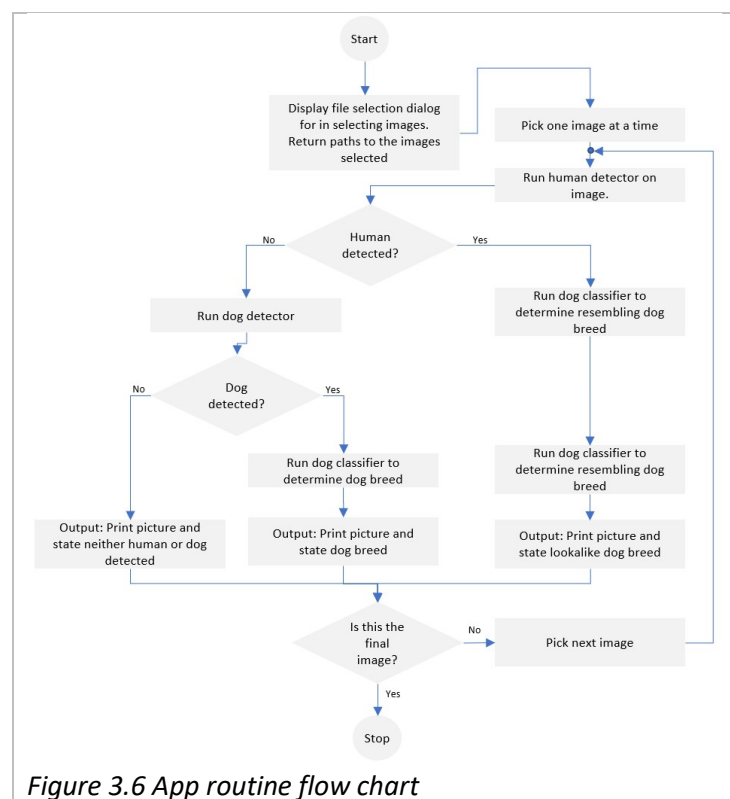


Figure 3.6 App routine flow chart

## 4. Results

Accuracy was the metric used in assessing the respective model performance as shown in *Table 4.1*

Table 4.1 Model Accuracy

Model	Optimal Human Face Detector	Dog Detector	Dog Classifier (transfer learning)
Accuracy (%)	99	93	81

Clearly the dog classifier had the lowest accuracy. Trials made with VGG-16, and inception CNN gives higher accuracy numbers. However, those models were not used due to GPU memory limitations. They were expensive to run and often resulted in system crashes. Thus, to enable me to complete this project and showcase the skills, I opted to go with resnet50.

Some of the predictions made are as shown in *Figure*

4.1, Figure 4.2, and Figure 4.3 below:
















Test Image	Likely a dog of breed: Afghan hound 	Likely a dog of breed: Border collie 	Likely a dog of breed: Chow chow 
Random Sample Image of Predicted Breed	 Afghan Hound Dog Breed Information ...	 Border Collie Dog Breed Information ...	 Chow Chow - Wikipedia

Figure 4.1 Prediction Set 1

Test Image	Likely a human, resembles a: Havanese dog breed 	Likely a human, resembles a: Havanese dog breed 	Likely a human, resembles a: Irish water spaniel dog breed 
Random Sample Image of Predicted Breed	 Havanese Dog Breed Information	 Havanese Breed: Characteristics, Care ...	 Irish Water Spaniel Dog Breed Information



**Figure 4.2 Prediction Set 2**

Test Image	<p>Likely a dog of breed: Belgian malinois</p> 	<p>Likely a dog of breed: Chow chow</p> 	<p>Likely a dog of breed: Poodle</p> 
Random Sample Image of Predicted Breed	 <p>Belgian Malinois Dog Breed Information ...</p>	 <p>the Chow Chow Is an Unusual Dog Bre...</p>	 <p>Poodle Calendar - Poodles Cale...</p>

**Figure 4.3 Prediction Set 3**

Test Image	<p>Neither dog or human</p> 	<p>Likely a dog of breed: Alaskan malamute</p> 	<p>Likely a dog of breed: Irish water spaniel</p> 
Random Sample Image of Predicted Breed	<p>Unable to detect human or dog</p>	 <p>Alaskan Malamute Dog Breed Information</p>	 <p>Irish Water Spaniel Dog Breed ...</p>

**Figure 4.4 Prediction Set 4**

From prediction set 1, the algorithm performs reasonably well in predicting the dog breeds of images with dogs. From prediction set 2, the resembling dog breeds predicted for the humans to my eyes doesn't necessarily look like the humans, but this is a subjective statement. Prediction set 3 shows that where there is more than one dog breed in the image, the algorithm seems to predict for the dominant dog in the image. Prediction set 4, shows that if a human and a dog are in the image the algorithm can predict the dog breed, but for the image tested it doesn't identify the human. Prediction set 4 also shows one image where the algorithm was unable to identify a dog or human. This can be attributed to the quality of the image.

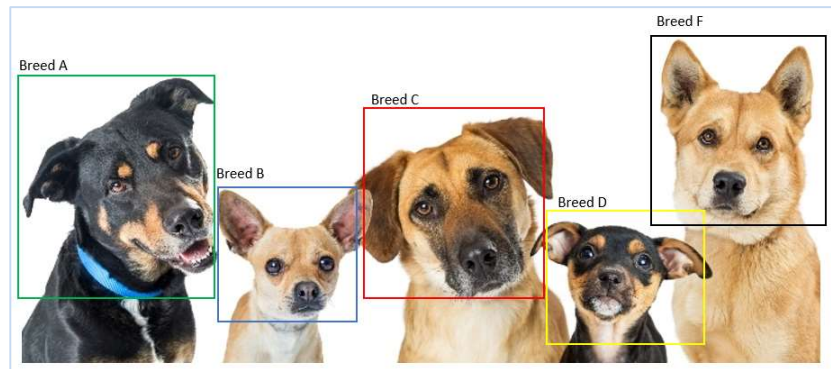
These tests are suggestive of the fact that in general, the algorithm makes very good predictions within the limitations of what was implemented in the algorithm and the CNNs used herein.

## 5. Conclusion

In this project a dog breed classifier was built using convolutional neural networks. The whole system was made up of 3 parts, a human detector, dog detector and the dog classifier. The optimal system was based on dlib CNN for human detector, a VGG16 based dog detector and a resnet50 based dog classifier. The classifier achieves an accuracy of 81% on the test dataset. The major challenge in the project was attempting to build a CNN from scratch. the ability to design the optimal architecture for this problem is no mean feat. The challenge is even more difficult with limitations in compute resources. This impacts how big the model can be, the training times. Training the big networks tested in building this project often resulted in my 6GB Nvidia GeForce GTX 1060 GPU running out of memory. Using transfer learning was a game changer as it simplifies the problem, but there was still compute challenges depending on the pretrained model selected. Resnet50 was selected because my GPU could handle it comfortably.

Possible improvements to the projects may include:

- In an image with multiple dog breeds, enable the classifier to predict the breed for each dog, instead of uncontrollably and randomly predicting the breed for one dog. See concept in *Figure 5.1*.



*Figure 5.1 Predicting breed of each dog in an image*

- Train the transfer learning CNN based classifier longer than 30 epochs and use the training vs validation loss curves to determine the optimal time to stop training.
- Add functionality to identify the colour of the dog.
- In an image containing a human(s) and dog(s), enable the routine to identify that the image contains human(s) and dog(s) and predict the dog breed for all the dogs in the image.
- Make the algorithm predict the location of the dog. Example: "This is a dog of breed X in a **garden**"

This algorithm can be developed further and deployed as a mobile app, enabling users to take a picture of a dog and quickly get a prediction of its breed or a web app where users can upload an image of a dog and get a prediction of its breed.