MACHINE LEARNING ENGINEEER NANO DEGREE

CAPESTONE PROJECT: DOG BREED CLASSIFIER

February 26, 2021

1. DEFINITION

1.1 PROJECT OVERVIEW

This project deals with classifying dog breeds given images as input. Differentiating dog breeds is a challenging problem because there are low inter-breed and high intra-breed variation; to describe it more clearly, there are relatively few differences between breeds and relatively large differences within breeds, differing in size, shape, and color. Dogs are both the most morphologically and genetically diverse species on Earth. This problem is not only challenging but also the method used for solving this problem will apply to other fine-grained image classification problems [2]. For example, to identify breeds of cats and horses as well as species of birds and plants or even models of cars. Convolutional neural networks (CNNs) have been mostly used in applications such as object classification, scene recognition, and many other applications [1][3]. This capstone project aims to apply deep learning techniques for classifying dog breeds. So, I designed a CNN model that will classify dog breeds. And we know that deep learning models are data-hungry and we need huge data for training our neural networks. The required dataset is provided by Udacity for training the model which contains 8351 dog images in total with exactly 133 dog breeds.

1.2 PROBLEM STATEMENT

The project aims to design a CNN for identifying the breed of the dogs. At the end of this project, the code will accept any user-supplied image as input. If a dog is detected in the image, it will provide an estimate of the dog's breed. If a human is detected, it will provide an estimate of the dog breed that is most resembling. Besides, an accuracy of above 60% should be attained. One of the strategies for solving this problem is to design a neural network architecture from scratch and use transfer learning techniques on various pre-trained models, with the target to achieve accuracy better than the benchmark. The problem which we are dealing with is a multi-classification problem. As CNN provides better accuracy for the multi-classification model. I have designed a CNN model for identifying the breed of the dogs.

Also, I have designed a CNN using transfer learning techniques which helped in attaining high accuracy and it took less training time as I have used a pre-trained model for extracting features. The model architecture of both models will be discussed clearly in section 3.2.

1.3 EVALUATION METRICS

The main aim is to compare the performance of the benchmark model and the solution model. So, the performance of both models can be evaluated by accuracy score. Also, it is observed that the classes of the dataset are not balanced. To handle this imbalance, I will be using cross-entropy loss also known as log loss as an additional metric.

2. ANALYSIS

2.1 DATA EXPLORATION

There are two different sets of data to work with:

1. Dogs Dataset [4]: It contains 133 classes. Each class corresponds to a different dog breed. There are 8351 dog images in the dataset. The dataset is split into training, testing, and validation data. We have 6680 images in the training set, 836 images in the testing set, and 835 images in the validation set. Some of the sample images from the data are depicted below:

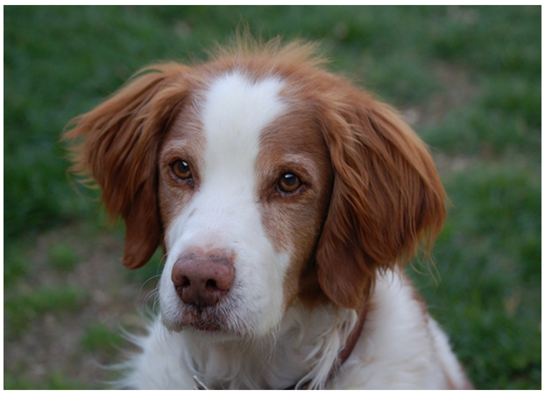


Figure 2: Brittany

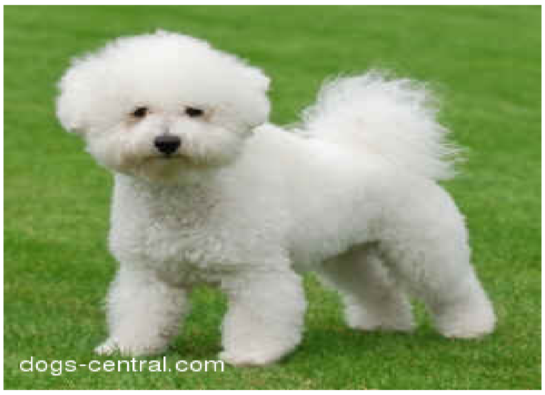


Figure 1: Bichon Frise

All the images are in .jpg format. The resolution of the dog images varies a lot, listing some examples here: 640 x 613, 487 x 600, 375 x 500. So, image resizing is necessary before feeding the data to the network. Because CNN structure presumes that the input would have a certain shape, like 224 x 224 x 3.

2. Human Dataset [5]: It contains 13233 human images. This dataset is used for testing the performance of the Human Face Detector. Some of the sample images from the human dataset are depicted below:

Figure 3: Suzanne Fox

Figure 4: Uzi Even

The datasets are provided in Udacity's workspace and the links to the dataset are provided in the reference section.

2.2 EXPLORATORY VISUALIZATION

Here we'll explore the dataset through some visualization. The dogs' dataset which we have contains 133 dog breeds. The class distribution of 50 dog breeds is depicted in figure 7. We can observe the class distribution is not uniform. The breed with the high-class distribution has 77 images and the breed with the low-class distribution has 26 images. This data imbalance could be a problem. To cope up with this problem data augmentation techniques can be used in the training process. Some sample images from the dog dataset are shown below. Their different sizes need to be considered before feeding the data into the network. This can be handled by resizing all the images to the same size during the data preprocessing.

Figure 5: Canaan Dog

Figure 6: Cavalier King Charles Spaniel

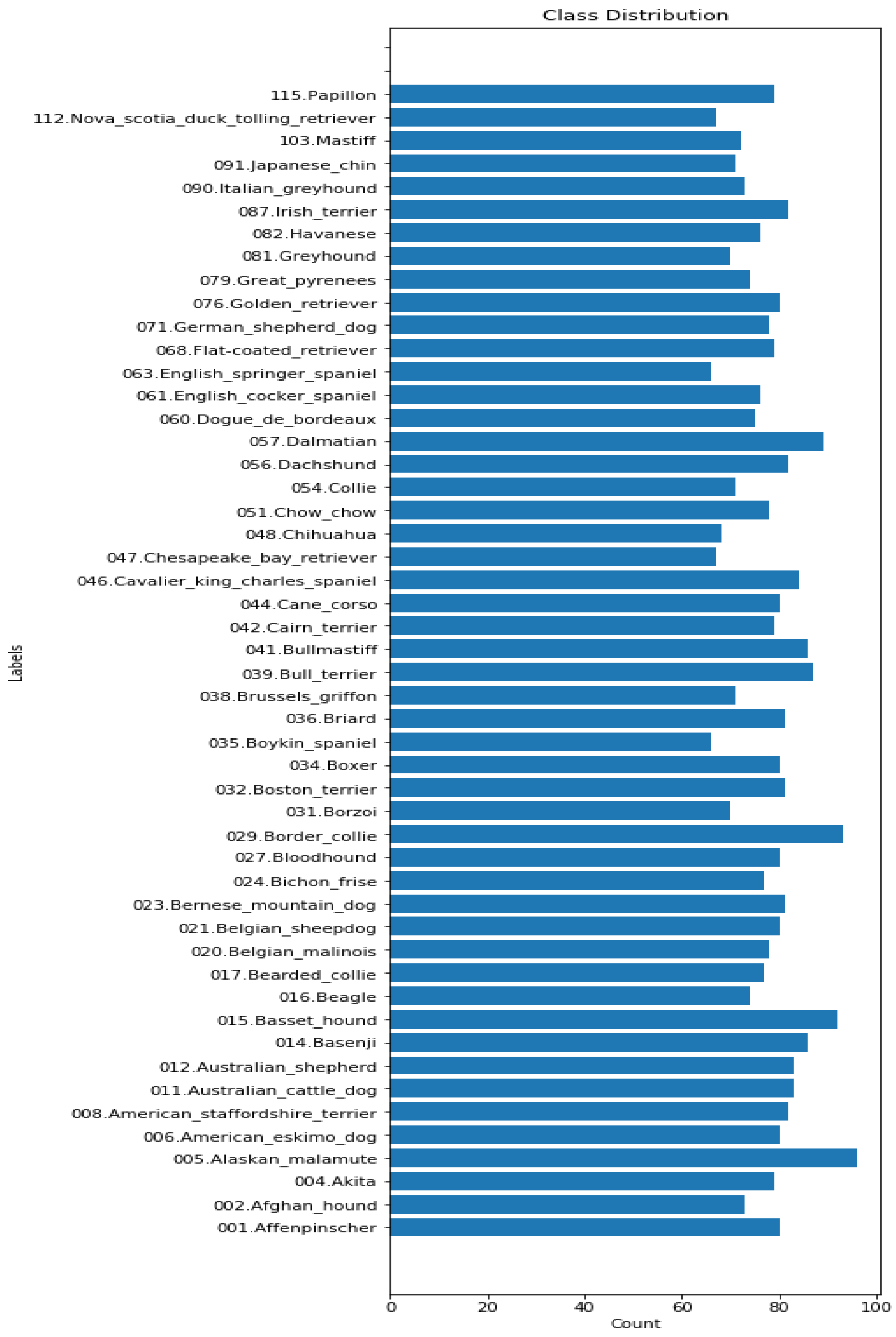


Figure 7: Class Distribution

2.3 ALGORITHMS AND TECHNIQUES

Following algorithms and techniques are used for the implementation of this project:

Human Detector: Pre-trained Haar feature-based cascade classifiers from OpenCV are used for recognizing the face of the humans in the images.

Dog Detector: To detect the dogs in the image pretrained ImageNet VGG16 model [6] is used.

Dog Breed Classifier from Scratch: As the project, requirements are to only achieve 10% or greater test accuracy. I have taken the simplest model from the VGG16 paper [6] and further simplified it.

Dog Breed Classifier using Transfer Learning: As the VGG16 network was also trained on dogs, among other things, it contains useful high-level feature information in the later convolutional layers. So, I have used the entire feature extractor part from the VGG16 pretrained model keeping the weights constant, to avoid overfitting when training on a small new dataset like the dataset we have. Then I replaced the classifier part with my dog breed classifier.

Testing the Application: I wrote an algorithm that accepts a file path of an image and first determines whether the image contains a human, dog, or neither. Then,

- If a dog is detected in the image, it returns the predicted breed.
- If a human is detected in the image, it returns the resembling dog breed.
- If neither is detected in the image, it returns an error.

2.4 BENCHMARK

The classes in the dataset are not balanced. It is very challenging to attain a high accuracy for this dataset. So, designing an efficient CNN model from scratch for classifying the dog breed will help in attaining good accuracy. For designing the CNN model, I have taken the simplest model from the VGG16 paper [6] and further simplified it. The architecture will be discussed in detail in section 3.2.

3. METHODOLOGY

3.1 DATA PREPROCESSING

As the VGG-16 model takes 224 x 224 images as input, I have resized the input tensor to 224 x 224 using transforms.resize(224) to rescale and interpolate the images. As the given dataset is not large, I have augmented the dataset through random horizontal flips using transforms.RandomHorizontalFlip() and random rotations by transforms.RandomRotation() with 30 degrees.

3.2 IMPLEMENTATION

In this section, I'll discuss the algorithms and techniques I have used in detail.

Human Face Detector: Here I used one of the pre-trained Haar feature-based cascade classifiers from OpenCV for recognizing the face of the humans in the images. I used the cascade classifier haarcascade_frontalface_alt.xml and implemented it in the face_detector(). This bought me good results. In 100 test images of humans, 98% were recognized as humans.

Dog Detector: To recognize the dogs in an image. I have used a pre-trained ImageNet VGG16 model. This pretrained model is only used for prediction and is not used for training. Firstly, I have resized all the images for the VGG16 model to 224x224 pixels as it is the input size of the pre-trained model. After that, the images are converted to a PyTorch tensor and are normalized. The parameters for normalization are the same as the parameters used when training the VGG16 model. Then I have passed this image to the prediction function for getting the index value. The VGG16 model [6] was initially trained on the ImageNet dataset. The dataset has 1000 classes. The indexes between 151 and 268 are dog categories. This means that if the VGG16 model returns an index in this range, a dog is detected in the image. The VGG16_predict() takes care of the prediction and the dog_detector() evaluates the index and returns true or false Boolean values accordingly.

Dog Breed Classifier from Scratch: As the input to the network is an RGB image cropped into 224x224 pixels, the input is in the shape 3x224x224. The desired number of outputs generated by the network is the same as the number of classes of the dataset. We have 133 classes in the dataset so the desired number of outputs generated by the network is equal to 133.

I have designed 3 convolutional layers with ReLU activation function and one max pooling layer (2,2), the first layer takes (3,224,224) as an input and converts it into a depth of 16 layers, I used a 3x3 filter with a stride length of 1 and padding of 1 to not get rid of edges.

- first convolutional layer (sees 32x32x3 image tensor).
- second convolutional layer (sees 16x16x16 tensor).
- third convolutional layer (sees 8x8x32 tensor).
- one pool layer (2,2) is used to reduce the size of the images to half.

Then flatten the image input using the view function to reshape the tensor.

Then, two fully connected Linear layers with ReLU activation function are added, and dropout layers for the hidden layers with a dropout of 25% to avoid the bias are added:

- first linear layer (64 x 28 x 28 -> 500).
- second linear layer (500 -> 133), as the number of output classes, is 133.

Dog Breed Classifier using Transfer Learning: Pre-trained modules are well trained to extract patterns and features very well.

I have chosen to load the VGG16 model for applying transfer learning techniques. After analyzing the structure of the model, I have frozen training for all "features" layers because if I changed it will take a long time to design and train the model. I changed the last linear layer output to be 133 classes to fit our classification output. Then I trained the classifier to get good results for our application.

Testing the application:

For testing, the application run_app() is created. Firstly, it is recognized whether a dog or a human is present in an image using the face_detector() and the dog_detector(). If both are not detected in an image, an error will be returned. The image is then passed to the predict_breed_transfer() which transforms the image, makes the predictions, and returns the dog breed.

3.3 REFINEMENT

The first solution was a baseline CNN designed from scratch with 3 convolutional layers. It was required to be better than a random prediction with 1/133 accuracy and achieve test accuracy greater than 10%. Passing those constraints means that the data pre-processing is reasonable for the dataset. After the data pre-processing, the VGG-16 model with batch normalization was chosen based on the comparison table of models pre-trained on ImageNet [6]. It is suitable because it has a pre-trained feature extractor which provided good results for the classification of ImageNet images including the classification of 133 dog breeds. After creating the Dog Breed Classifier using Transfer Learning, it achieved an accuracy of 85%. So, this model had better performance than the benchmark model.

4. RESULTS

4.1 MODEL EVALUATION AND VALIDATION

During training, both model's performance is evaluated based on the validation loss on the completion of each epoch. After training, the performance of the models is evaluated with the test loss. The human face detector was validated on the first 100 images of human and dog datasets: 98% human faces were detected in the human subset and 17% human faces were detected in the dog subset. The dog detector was also validated on the first 100 images of human and dog datasets: 100% dogs were detected in the dog subset and 0% dogs were detected in the human subset. The benchmark model was trained for 20 epochs, optimized by Stochastic Gradient Descent with the learning rate = 0.01 to reduce the Cross-entropy loss function. On the test set, the model achieved 11% of accuracy and a loss of 3.935. The solution model was trained for 10 epochs, optimized by Stochastic Gradient Descent with the learning rate = 0.01 to reduce the Cross-entropy loss function. On the test set, the model achieved an accuracy of 85% and a loss of 0.543. The accuracy achieved crosses the threshold of 60%, therefore the model's performance is considered pretty good, and the model is reliable.

4.2 JUSTIFICATION

The final results are found to be stronger than the benchmark result reported.

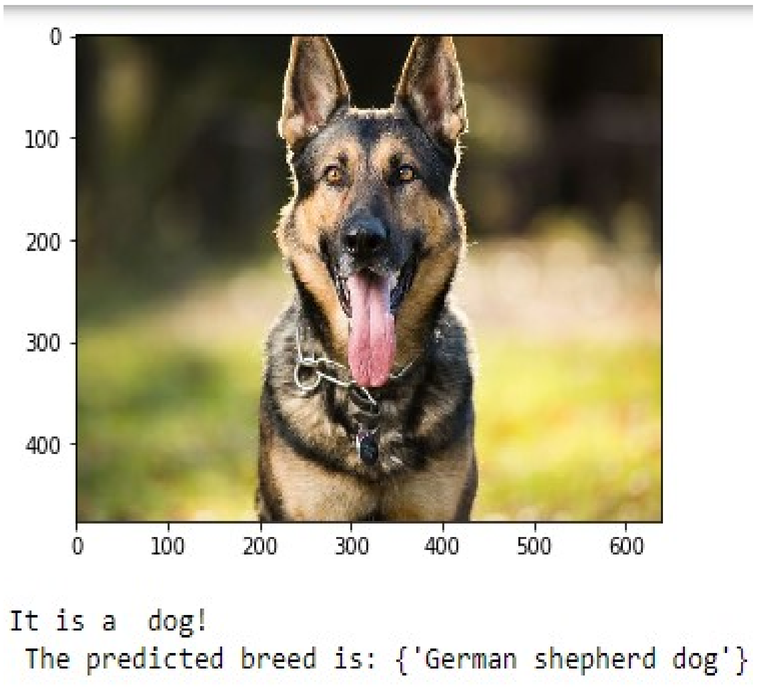
Benchmark model's accuracy: 11%

Solution model's accuracy: 85%

The solution model can classify the dog's breed accurately which can be analyzed qualitatively from the images in section 5.1.

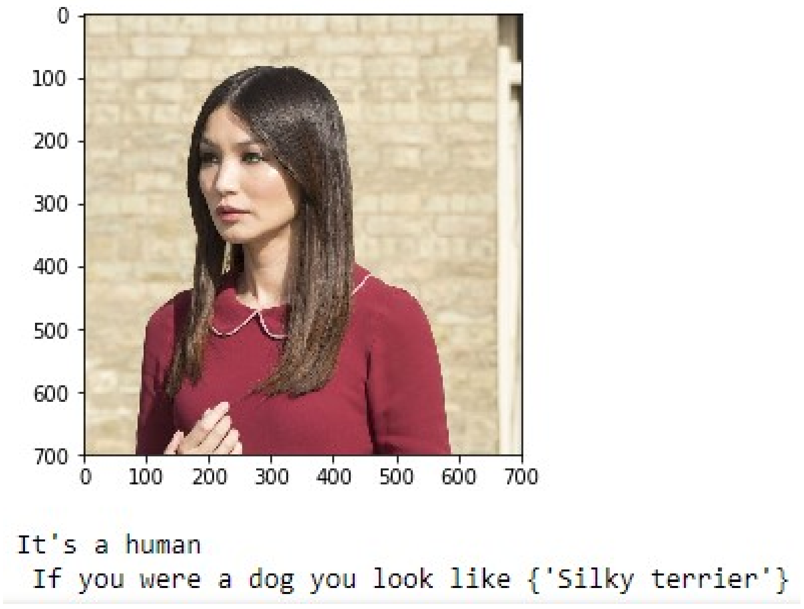
5. CONCLUSION

5.1 FREE-FORM VISUALIZATION



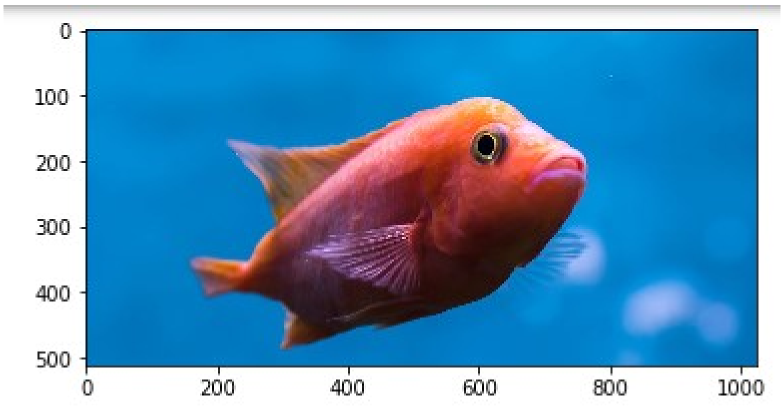
```
It is a  dog!
 The predicted breed is: {'German shepherd dog'}
```

Figure 8: Test Image 1



```
It's a human
 If you were a dog you look like {'Silky terrier'}
```

Figure 9: Test Image 2



```
Error: No dogs or humans are detected.
```

Figure 10: Test Sample 3

I have given the images of dogs, humans, and other things to test the application. So, the qualitative results are shown in figures 8, 9, and 10. Figure 8 is recognized as a dog and its breed German Shepherd Dog is predicted correctly. Figure 9 is recognized as a human and if the human presence in the image would be a dog it will look like a Silky Terrier is estimated. And finally, for Figure 10 the model has returned an error because it could not detect any dog or human in the image.

5.2 REFLECTION

The entire end-to-end problem solution can be summarized in 8 steps:

Step 1: Importing Dog and Human Datasets

Step 2: Data Preprocessing

Step 3: Detected humans using pretrained Haar feature-based cascade classifiers.

Step 4: Detected Dogs using the pre-trained ImageNet VGG16 model.

Step 5: Designed a CNN model to Classify Dog Breeds (from Scratch). Trained and tested the model performance.

Step 6: Designed a CNN model to Classify Dog Breeds (using Transfer Learning). Trained and tested the model performance.

Step 7: An application was designed and developed for showing the predictions for a given image.

Step 8: Tested the algorithm on sample images

It was a very good experience working on this project. To pick up the best one of all, it will be step 8. I was very curious to know how the model will perform and show the predictions

5.3 IMPROVEMENT

To my surprise, the model's performance was better than what I expected. With such a simple network I achieved an accuracy of 85%. Keeping the model's performance in view I would like to suggest the following improvements:

- According to Fédération Cynologique Internationale (FCI) [7], currently, there are 352 dog breeds. If we have the dataset with all these dog breeds, we can train our network with a large number of classes.
- Model fusion can improve the performance of the network further.
- Exploring more network architectures will improve the performance of the model.

6. REFERENCES

1. A. Krizhevsky, I. Sutskever, and G. Hinton. “Imagenet classification with deep convolutional neural networks”, Advances in neural information processing systems. 2012.
2. J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur. “Dog Breed Classification Using Part Localization”, Computer Vision–ECCV 2012. Springer Berlin Heidelberg, 2012. 172-185.
3. Hsu, David. “Using Convolutional Neural Networks to Classify Dog Breeds”, Stanford University.
4. The dog dataset.

   URL: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
5. The human (LFW) dataset.

   URL: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip
6. K. Simonyan, A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. ICLR 2015.
7. FCI.

   URL: http://www.fci.be/en/Presentation-of-our-organisation-4.html