

Machine Learning Engineer Nanodegree

Capstone Project

Final Report

Pengseng (Isaac) Tan
5-10-2020

Project Overview

To design and implement a custom CNN-based Deep Neural Network (DNN) model to perform the task of classifying different breeds of dog based on an input image provided by user. The performance of this model will also be evaluated using the performance obtained from an existing CNN-based DNN model as a benchmark [Ref 1].

Domain Background

This project falls into the domain of Object Classification under Computer Vision. Since the advent of modern computers that are built from semiconductor devices, computer scientists have been attempting to apply mathematical algorithms to be executed automatically by computers to address a variety of tasks that seems trivial to human such as lesser complex problems such as line and edge detections in images to more complex problems such as either single or multiple object classifications or detections in images.

For performing complex tasks such as object classification or recognition from the perspective of using computers, a very important aspect is the selection of suitable features that will provide sufficient information to distinguish/discriminate one object class from another. As such, great emphasis has been placed on Feature Engineering and Extraction in both spatial and frequency domain such as Corner features, HAAR-like features, Histogram of Gradients (HOG), Sift Invariant Feature Transform (SIFT) to Speeded Up Robust Features (SURF) [Ref 2].

However, with the emergence of Convolutional Neural Networks (CNN) such as AlexNet [Ref 3] in 2012, the importance of feature engineering for obtaining good object classification results in imagery has decreased significantly over the years. This is due to the fact that a pretrained deep CNN (trained using tens of thousands or hundreds of thousands of images) will be able to extract much more robust and complex features that can provide object classification or recognition performance that are even superior to a standard human performance.

Project Statement

In this project, the objective is to demonstrate the capability to design and implement a CNN-based Machine Learning Classification pipeline for a multi-category classification model to provide a classification result based on the highest prediction probability for each user-supplied image. To be more specific, when provided with an image of a canine, this Machine Learning Classification pipeline will provide a prediction of the **canine's breed** such as an Australian Terrier or Beagle based on the breed with the highest probability computed from the output of the CNN model. However, if the user-supplied image belongs to that of a human, the Machine Learning Classification pipeline will then attempt to associate this human input image with the closest resembling dog breed. To provide a better understanding of the

above description, a simple step-by-step procedure of the Machine Learning Classification pipeline is provided as follows:

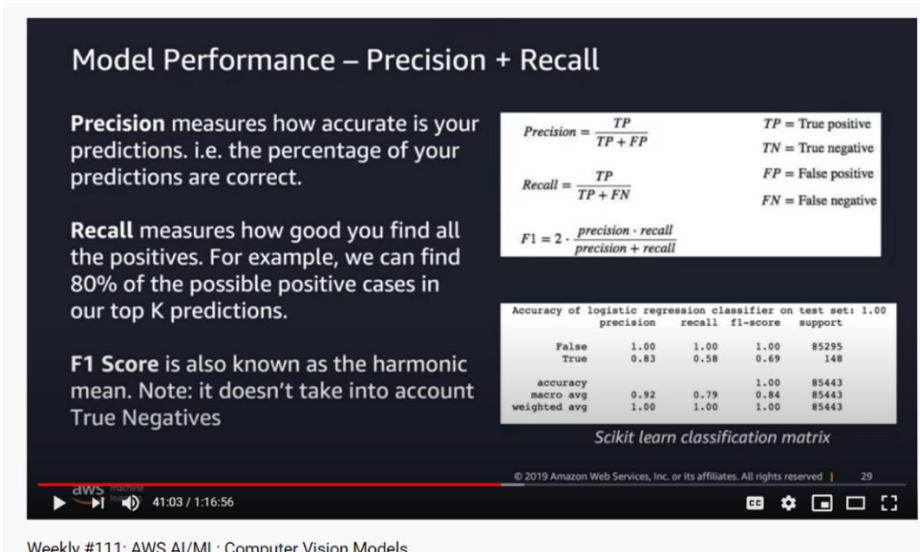
- 1) Invoke **M.L. Classification App/Function** by passing in an image as input parameter
- 2) Invoke **Human detector** on image. Also, Human detector is the provided OpenCV module (HAAR features) or another detector provided. If a human is detected, return resembling dog breed from **Classifier module**
- 3) Otherwise invoke **Dog detector** on image and Dog detector is implemented using pretrained VGG16 model. **If** a dog is detected, return predicted breed from **Classifier module**
- 4) if neither is detected in input image, provide output error message that indicates an input error

Evaluation Metrics

In a standard classification problem, the default evaluation metric that is used to quantify the performance of both the benchmark model and the solution model is usually the **Accuracy Score** in units of percentage in correctly predicting the correct breed of dog from the test dataset provided. For instance, if the test dataset contains a total of 150 dog images and 90 of these dog images are assigned with the correct dog breed, then the Accuracy Score will be $90/150 * 100\% = 60\%$.

However, in this project, as will be shown from the results of the exploratory data analysis for the Train dataset in the later sections of the report, it can be seen that it is not a balanced dataset as there are dog categories with more dogs than others. As such, instead of the Accuracy score, other metrics such as the **Precision Score P** and the **Recall Score R** will be chosen. Now, **P** is defined as the number of correct positive results divided by the number of all positive results returned by the classifier and **R** is defined as the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive) [Ref 4].

Figure 1: Mathematical Definition of **Precision** and **Recall** from AWS website



[illegible][illegible]

Finally, these datasets are available under the following references [Ref 6] and [Ref 7] for both dogs and humans.

Algorithms Discussion

In this project, two approaches are considered implementing the Machine Learning Classifier pipeline for Dog Breeds, namely traditional Machine Learning Algorithms that do not rely on Deep Neural Networks (DNN) or Convolutional Neural Network models which is a flavor of DNN.

When referring to traditional Machine Learning Algorithms, they can be less sophisticated algorithms such as Logistic Regression or Support Vector Machines to more sophisticated algorithms such as Naïve Bayes, Random Forest, Decision Trees or XGBoost. Now, although these algorithms can provide good performance for problems of varying complexity and for reasonable input data size, a feature selection process or feature engineering process will need to be applied to identify the features to be used for these algorithms. Now, this process is not for the inexperienced engineer or the faint-hearted especially if the input data classes are numerous as well as the data itself also being of high dimension, for example an RGB image will consist 3 dimensions of data whereas a multi-spectral imagery may have 10 – 30 dimensions in its data. Thus, much experience and skillsets are needed to identify and extract good features to be used for the traditional Machine Learning Algorithms.

With that consideration in mind, the solution to be adopted for the Machine Learning Classifier pipeline of Dog Breed Classification will be based on CNN models that will easily incorporate the complex feature selection process as part of its training process.

Benchmark Model

In this project, the benchmark model selected for comparison with the proposed solution will be the pretrained **VGG16** model [Ref 8] that has been trained, validated and tested using huge amounts of images from the ImageNet dataset for classifying 1000 categories of objects and humans. In order to train the VGG16 model to participate in the **ImageNet Large Scale Visual Recognition Challenge** [Ref 9], an amount of 1000 training images is allocated for training each of the 1000 categories for the classification task. Also, 50,000 validation images and 150,000 test images are also used as well during the training process of these DNN models.

Therefore, the value obtained from the benchmark **Top-1 Accuracy** performance of the VGG16 model on the ImageNet validation dataset will be a good objective comparison to the proposed solution for the Dog Breed Classification. Also, a list of all the benchmark values obtained from the various CNN models on the ImageNet validation dataset is shown on Figure 5 on the next page.

Figure 5: List of CNNs' performance on the ImageNet validation dataset [Ref 10]

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Depth refers to the topological depth of the network. This includes activation layers, batch normalization layers etc.

Alternatively, a second pretrained model such as the original **AlexNet** model can also be considered as the benchmark model to be used to quantify the performance of the proposed solution. This suggestion will show the progression of Artificial Intelligence capability of CNN-based models within the span of a decade from 2012 to 2020.

Data Preprocessing

For converting the pretrained VGG16 model into a dog detector, each input image data from train, validation and test datasets must go through a data transformation process that is listed as follows:

- Resize to 224 * 224 pixels
- Convert image data to Tensor
- Perform data normalization via subtraction of each pixel by a mean vector (due to RGB dimension) as well as division by a standard deviation vector

Also, all the above transformation steps are supported by importing a module named **transforms** in Pytorch. Following that, in order for the normalized tensor to be fed into the VGG16 model, an additional dimension containing the **batch size** parameter has to be inserted to be the first dimension of the input tensor as the last step of the transformation process.

Next, to build either a CNN network from scratch or to perform transfer learning, besides the above transformation steps, there are additional preprocessing steps that are required for the training and validation data. These preprocessing steps include two aspects and the first aspect involves data augmentation such as image data upsizing with a random factor before cropping to 224 * 224 pixels, random image rotation and random image horizontal flipping. The second aspect involves the shuffling of the order of the training data and validation data inputs for each new training epoch. Finally, the new transformation steps are supported by importing two modules named **datasets** and **Dataloader** in Pytorch

Implementation of Solution

For this project, after data exploration and preprocessing, the first task to develop and implement the Classifier component of the project is to build from scratch a CNN model to perform the Dog Breed Classification using the Pytorch Deep Learning framework. As such, a CNN model with 3 CNN layers with RELU activation functions, 3 MaxPooling layers, 2 Fully Connected layers and 2 Dropout layers (20% dropout) are created for this first task and two snapshots of this CNN model are as shown below:

Figure 6: List the layers used in CNN model built from Scratch

```
Net(  
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
  (conv2): Conv2d(32, 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)  
  (dropout1): Dropout(p=0.2, inplace=False)  
  (dropout2): Dropout(p=0.2, inplace=False)  
)
```


Figure 7: Summary of the CNN model built from Scratch

```
In [5]: summary(model_scratch, (3, 224, 224), batch_size=-1, device='cuda')
```

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 224, 224]	896
MaxPool2d-2	[-1, 32, 112, 112]	0
Conv2d-3	[-1, 64, 112, 112]	18,496
MaxPool2d-4	[-1, 64, 56, 56]	0
Conv2d-5	[-1, 128, 56, 56]	73,856
MaxPool2d-6	[-1, 128, 28, 28]	0
Dropout-7	[-1, 100352]	0
Linear-8	[-1, 512]	51,380,736
Dropout-9	[-1, 512]	0
Linear-10	[-1, 133]	68,229
Total params: 51,542,213		
Trainable params: 51,542,213		
Non-trainable params: 0		
Input size (MB): 0.57		
Forward/backward pass size (MB): 27.57		
Params size (MB): 196.62		
Estimated Total Size (MB): 224.76		

Initially, an attempt was made to use a 2-dimensional batch normalization layer after each convolutional layer so as to introduce stability to the weights of the neurons as well as to speed up the convergence of the model training. However, some issues came up with the validation results during the model training process, i.e. the validation loss increases instead of decreasing after each epoch (akin to overfitting phenomenon). Although tweaks are made such as reducing the value of the **momentum parameter** to **0.05** or setting the parameter **track_running_stats=False** in the batch normalization layer, it does not remove the issue of the validation results diverging after some epochs. Thus, the final design of the CNN model built from scratch does not include any Batch Normalization layers.

Having listed out the considerations made when designing the CNN model from Scratch to perform the Dog Breed Classification, this CNN model is then trained for 25 epochs. Next, the performance of the trained CNN when applied to the test dataset gives an accuracy score of **12%** (103/836 test images) as well as a test loss of **3.956805**. Thus, this result is better than a random guess that provide a right response roughly 1 in 133 times, corresponding to an accuracy score of less than 1%, as is stated under the Capstone Project description. However, when compared to the **benchmark model** VGG16 that has a **Top-1 Accuracy score** of **71.3%**, this CNN model built from scratch is definitely performing much worse than the benchmark performance.

Refinement of Solution

In the previous section, having observed that the performance obtained from the CNN model from scratch is not performing near to the benchmark performance, in this section, the method of Transfer Learning is applied to a pretrained CNN model so as to greatly reduce the time required for training and the size of the training dataset to achieve a good performance for the retrained model after transfer learning.

As a start, there are many choices that are available for choosing the pretrained model to go through the transfer learning process. Popular networks such as Resnet50 or Densenet121 seems to be good candidates as the size of these models are much smaller than VGG16. However, as the Dog detector as well as the benchmark model is based on VGG16 model, thus my choice for the pretrained model to go for transfer training is also VGG16 model.

Next, to implement transfer learning of the VGG16 model, it is necessary to replace the final classification layer that outputs 1000 output classes with a new classification layer that will only output 133 classes corresponding to the 133 categories of dog breeds. Thus, this modification of the classification layer is performed, and the modification of this layer is captured in the below snapshot along with the number of parameters that have to be retrained during the transfer learning.

Figure 8: Details of the modification to VGG16 classification layer

Linear-39	[-1, 256]	1,048,832
ReLU-40	[-1, 256]	0
Dropout-41	[-1, 256]	0
Linear-42	[-1, 133]	34,181
LogSoftmax-43	[-1, 133]	0
=====		
Total params: 135,343,557		
Trainable params: 1,083,013		
Non-trainable params: 134,260,544		

Input size (MB): 0.57		
Forward/backward pass size (MB): 218.78		
Params size (MB): 516.29		
Estimated Total Size (MB): 735.65		

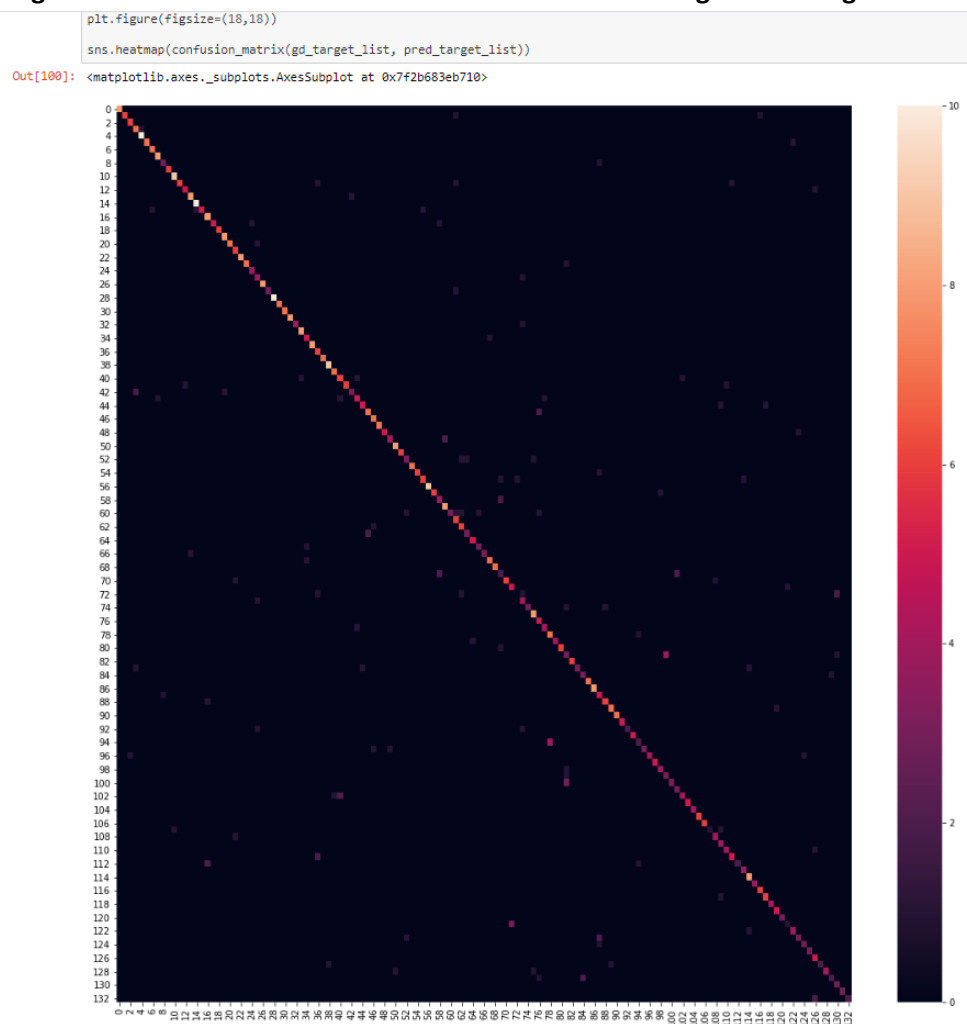
Once this modification is performed on the VGG16 model, the next step will be to perform the retraining of this modified model.

Model Evaluation and Validation

After the VGG16 model is modified for classifying the 133 categories of dog breeds, it then undergoes 25 epochs of transfer training using the same train and validation datasets as what was used for training the CNN model built from scratch. After this m-VGG16 model is trained, its performance is then evaluated using the same test dataset and it manages to produce an accuracy score of **83.0%** (700/836 test images) which is better than the benchmark value of 71.3% from original VGG16 model.

However, as described under the section on Evaluation Metrics, the use of accuracy score to evaluate the classification performance of the retrained model may not be the best approach due to the imbalance in the number of images for each dog breed category within the training and validation dataset. Thus, to further analyze the classification performance of the final model, a **Confusion Matrix** for all 133 categories as well as the computed values of both **Precision Score** and **Validation Score** for this model are provided as follows.

Figure 9: Confusion Matrix from Test dataset of 133 categories of Dog Breeds



In theory, a perfect Confusion Matrix will be identical to an Identity Matrix in which there are no non-zero off-diagonal values which means that no classes are misclassified. By observing Figure 9, it can be seen that the Confusion Matrix almost resembles an Identity Matrix with small numbers of off-diagonal cells with non-zero values. Thus, this indicates that the classification performance of the retrained VGG16 model is good. This observation is further supported by the computed **Precision score** and **Recall score** that have high values of **84.9%** and **82.7%** respectively. These values definitely exceed the benchmark performance of having Accuracy score greater than or equal to 71.3% set for the project.

Figure 10: Precision and Recall scores from Test dataset of 133 categories of Dog Breeds

```
In [101]: from sklearn import metrics
          metrics.precision_score(gd_target_list, pred_target_list, average='macro')

Out[101]: 0.8490606928576853

In [102]: metrics.recall_score(gd_target_list, pred_target_list, average='macro')

Out[102]: 0.8273093447905479
```

Thus, it seems that the Classification module is now ready to be integrated into the M.L. Classification App/Function that is the next and final task for the Capstone project.

As mentioned under the Project Statement section, the first part of the M.L. Classification App is to use a human detector to determine whether the input image is a human. For this project, a **custom human detector** built from a pretrained **Resnet model** built using **Caffe framework** that is also available from OpenCV is used. Also, its performance for the first 100 human and dog images from the test dataset is an Accuracy rate of 100.0 % for detecting human and 9.0 % error rate of mistaking dog for human which is better than the OpenCV **Haar features-based** face detector that has an Accuracy rate of 94.0 % for detecting human and 9.0 % error rate of mistaking dog for human. Next, the dog detector that is used for the App is a pretrained VGG16 network model that has an Accuracy rate of 99.0 % for predicting dog as well as 0% error rate of mistaking human for dog. With these performance numbers, the expectation is that there may be some dogs that are wrongly classified as humans or some dog breed classification errors.

Next, the implemented M.L. Classification App is first tested on 3 human images and 3 dog images from the test dataset. Following that, additional tests are then performed on 10 test images that are located under the “images” subdirectory under the project directory.

Table 1: Results from Test dataset











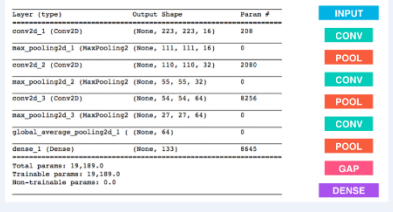
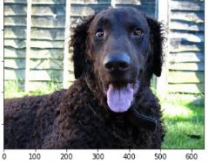
<p>Hello Human!</p>  <p>You look like a ... Bearded collie</p> <p>Correct human detection</p>	<p>Hello Human!</p>  <p>You look like a ... Wirehaired pointing griffon</p> <p>Correct human detection</p>	<p>Hello Human!</p>  <p>You look like a ... Brittany</p> <p>Correct human detection</p>
<p>Hello Dog!</p>  <p>My guess is that you are a Boxer</p> <p>Correct dog breed</p>	<p>Hello Dog!</p>  <p>My guess is that you are a Norwich terrier</p> <p>Correct dog breed</p>	<p>Hello Dog!</p>  <p>My guess is that you are a silky terrier</p> <p>Incorrect dog breed</p>

Table 2: Sample Results from “images” sub folder

<p>Hello Human!</p>  <p>You look like a ... Wirehaired pointing griffon</p> <p>Correct human detection</p>	<p>Hello Dog!</p>  <p>My guess is that you are a American water spaniel</p> <p>Correct dog breed</p>	<p>Hello Human!</p>  <p>You look like a ... Labrador retriever</p> <p>Incorrect human detection</p>
<p>Hello Dog!</p>  <p>My guess is that you are a Chesapeake bay retriever</p> <p>Incorrect dog breed</p>	<p>Oh No !! Not a human or a dog !!</p> 	<p>Hello Dog!</p>  <p>My guess is that you are a Curly-coated retriever</p> <p>Correct dog breed</p>

From the results as shown above in the two tables, it can be seen that there are some errors that are due to the human detector module as well as some errors that are due to the Dog Breed Classifier module.

Justification

From the results obtained from the retrained Classification module with accuracy score, precision score and recall score that exceeds the benchmark model performance value of 71.3%, it is concluded that this retrained Classification module has sufficiently addressed the requirements of the M.L. Classification App. Next, for the two detection modules, although there are some wrong detections in the human detector that detects dogs wrongly as human, however, it will not misclassify a human wrongly as non-human. As for the dog detector, it will always detect a dog correctly as a dog. Thus, the performance of the M.L. Classification App can be further improved by first executing the dog detector before executing the human detector in its pipeline.

Conclusion and Recommendations

In conclusion, the proposed solution for the M.L. Classification App has met the requirements of the Capstone project. However, improvements can definitely be obtained from the proposed solution such as using a human detection module with lesser error rate of detecting a non-human as a human. As for the retrained Dog Breed Classification module, some recommendations for improving its performance are as follows:

- Improving the custom human face detector such that the error rate of mistaking dogs or other animals as humans should be close to 0.5 % instead of 9.0% currently while the detection rate of human should stay retain at a value of ≥ 99.5 %. This can be achieved by using a different pretrained network model or using transfer learning
- Improving the performance of the proposed Dog Breeds Classifier (implemented using Transfer Learning) by using a larger dog dataset with balanced distribution of dog breeds as well as retaining data augmentation to this larger dataset.
- Use autotuning of the various hyperparameters of the final Dog Breeds Classifier such as learning rate, batch size, number of epochs so that the validation results will not be overfitted after just a few epochs

References

- [1] <https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification>
- [2] https://docs.duckietown.org/DT19/learning_materials/out/object_classification.html
- [3] <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>
- [4] https://en.wikipedia.org/wiki/F1_score
- [5] https://en.wikipedia.org/wiki/Confusion_matrix
- [6] <https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip>
- [7] <http://vis-www.cs.umass.edu/lfw/lfw.tgz>
- [8] <https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c>
- [9] <https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/>
- [10] <https://keras.io/api/applications/>
- [11] <https://review.udacity.com/#!/reviews/2230012>