# Dog Breed Classification with PyTorch

A Project Proposal

## 1. Domain Background

Artificial Neural Networks have rapidly growing applications in the area of image-based classification problems. Moreover, recent advances in the ANN technology allows for the use/reuse of pre-trained Deep Artificial Neural Networks (e.g. AlexNet, Resnet, VGG, and others) that were trained to recognize thousands of different categories by sifting through millions of high resolution images. This re-utilization of pretrained Deep Neural Networks belongs to a field of Machine Learning called “Transfer Learning” where these pre-built models can be repurposed to be utilized in numerous applications of image recognition and classification.

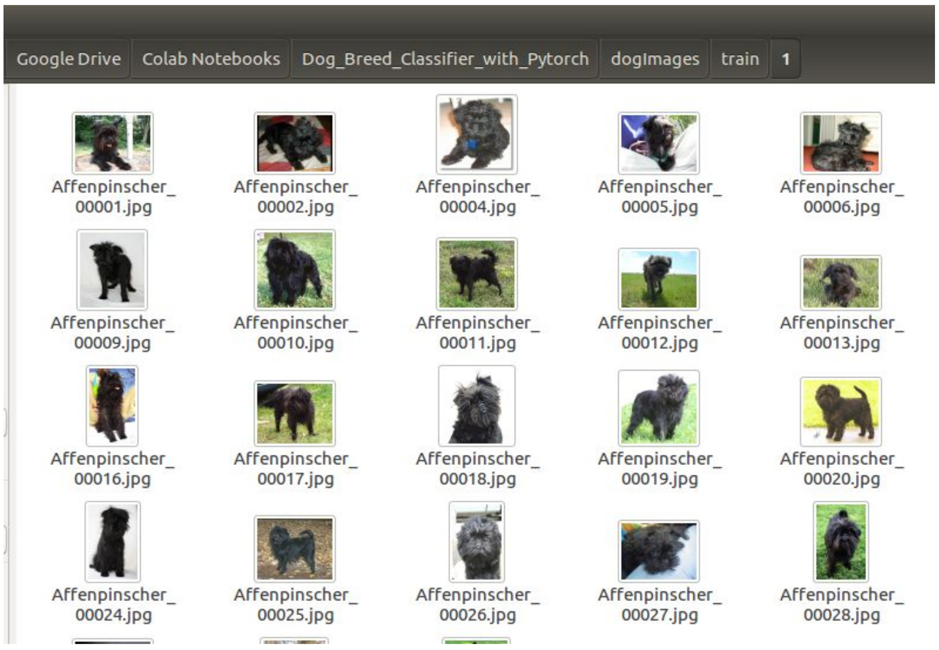
## 2. Problem Statement

Recognizing the breeds of dogs is certainly a challenging task. The estimated number of known dog breeds ranges between 200 and 350 different breeds, making the task of telling “which breed a dog belongs to” a very challenging task. This task gets more difficult when the breeds are visually similar and cannot be easily distinguished.

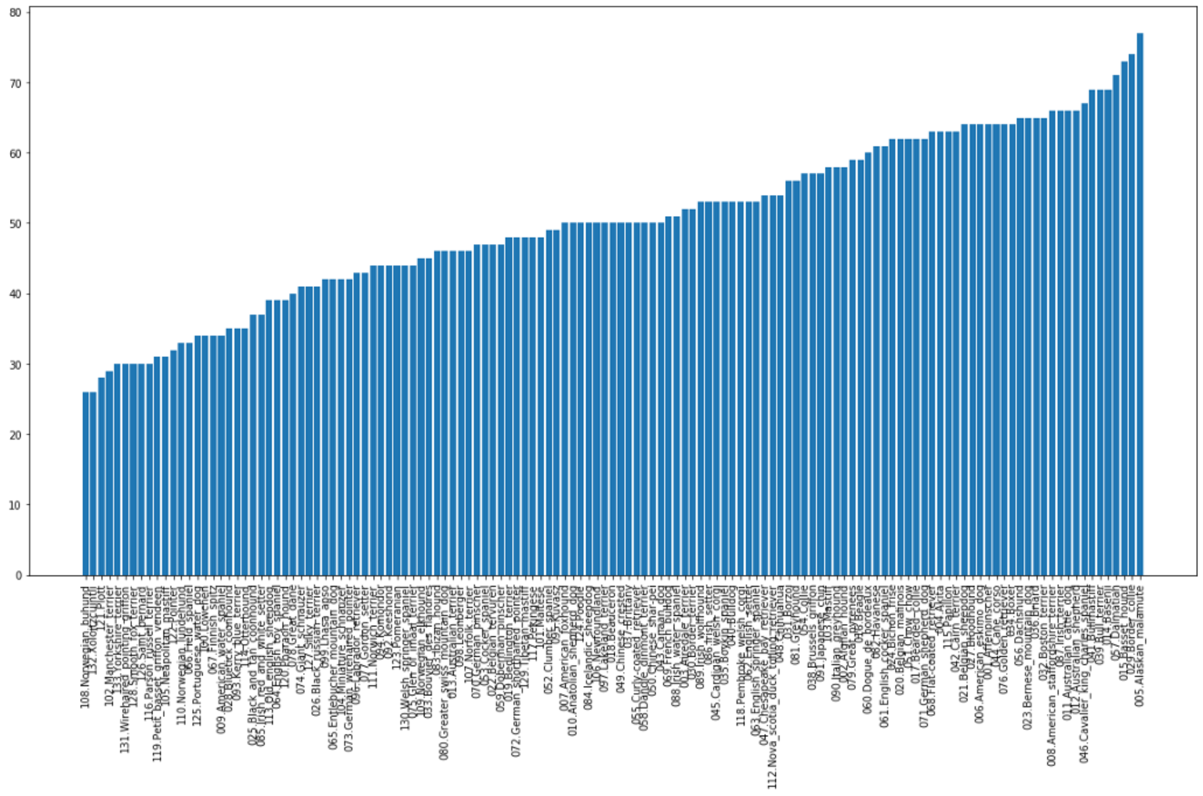
## 3. Datasets and Inputs

The dataset that will be utilized for this project is a collection of 8,753 dog images across 133 different breeds. The dataset is offered by Udacity and can be downloaded through the following link:
https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip

The comes in the form of images contained in numbered folders where each numbered folder is for a specific dog-breed. It’s important to note that the photos are not of the same size; hence, randomized cropping needs to be implemented as part of the data augmentation.

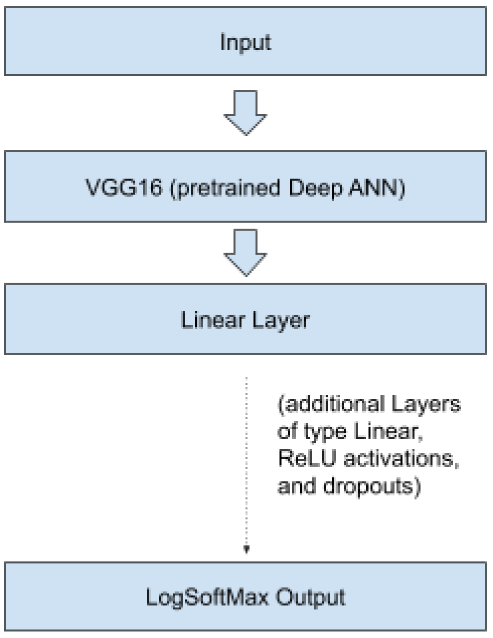


In terms of the balance of training data across the 133 classes (dog breeds), we can see that we have a considerable variance across the classes. However, the bulk of the classes remain in the range of 40-60 images which is close to an average of around 50.



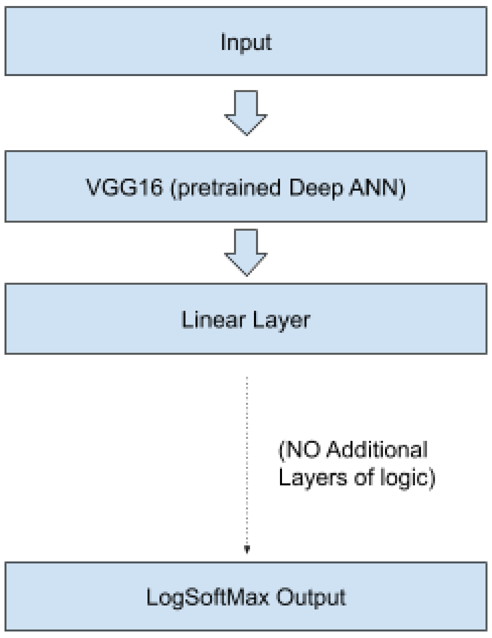
## 4. Solution Statement

The proposed solution for this dog breed classification problem is to utilize build a Deep Artificial Neural Network that utilizes one of the pre-trained Deep Neural Nets (VGG16) along with multiple additional layers in a single model that, upon training,can tell the dog breeds apart once given a photo of a dog.


Input
VGG16 (pretrained Deep ANN)
Linear Layer
(additional Layers of type Linear, ReLU activations, and dropouts)
LogSoftMax Output

## 5. Benchmark Model

Since the proposed solution is to build several additional layers on top of the VGG16 pre-trained model, the bench-mark model will be the VGG16 model with only a single Linear layer and a LogSoftmax output.



## 6. Evaluation Metrics

Due to the fact that we have a considerable variance across the 133 classes in our dataset, we will have to use the f-beta score for the evaluation of the performance of the model. Since there are no specific requirements for the balance of precision and recall of the model, **f1-score** will be the Evaluation Metric for the model.

Precision:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap h(x_i)|}{|h(x_i)|}$$

Recall:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap h(x_i)|}{|Y_i|}$$

F-beta score:

$$F_\beta = (1+\beta^2)\cdot\frac{\text{precision}\cdot\text{recall}}{(\beta^2\cdot\text{precision})+\text{recall}}$$

## 7. Project Design

The design for this project is a sequence of steps that follow the outline below:

1. Loading the data for the training, validation, and testing datasets
2. Data preprocessing
   a. performing data augmentation: this step will involve random resizing, cropping, rotating, and horizontal flipping for the input.
   b. performing data normalization
   This will involve pixel-wise RGB value standardization.
3. Building the benchmark model
4. Training the benchmark model
5. Testing the benchmark model
6. Building the target model
7. Training the target model
8. Assessing the target model Evaluation Metrics on the Validation Set
9. Saving the target model
10. Loading the saved model
11. Testing the target model
12. Assessing the target model Evaluation Metrics
13. Comparing the target model to the benchmark model
14. Finally, providing sample test outputs on the target model