# Capstone Proposal: CNN, Dog Breed Classifier

## Domain background:

Image classification is always an important topic in computer vision. The accuracy of the image classification itself is an essential metric to evaluate how good a neural network architecture is. Also, be able to do the image classification is an important part of artificial intelligence and can be applied to a lot of useful things. Also, there are a lot of useful pre-trained models in this area such as VGG16 which could be very useful for transferring learning. Thus, I choose the dog breed classifier as my capstone project.

**How is dog breed classification useful**: to classify the correct dog breed is extremely useful and challenging. Even a human has problems distinguishing between two very similar dog breeds. Thus, if we can train a model to find the difference between all kinds of dogs, it would make categorizing dogs much easier and save a lot of human times.

**History relevant to the project**: I will briefly discuss a few famous image classification problems.

- Mnist dataset, this is a dataset that contains images of handwritten digits. The goal is to correctly classify the number between 0 to 9. This is a simple dataset that are often used to create a simple CNN project. People can use this dataset to learn fundamentals of CNN.
- Dog vs Cat project: https://www.kaggle.com/c/dogs-vs-cats. This is a project that aim to classify whether an image is dog or cat. This project is similar to the dog breed classifier where both of the projects are dealing with animal images. The dog breed classifier is much harder as we have 133 labels. But this could be a good start point of my project.

## Problem Statement:

Complete a classifier that if given an image of a dog, should be able to output its breed.

**Possible solutions**:

- Train a neural network from scratch where the output is a vector of num of classes. And the dog breed is the label that has the largest value.
- Apply transfer learning on some pre-trained models. This could improve training efficiency.

## Dataset and inputs:

The dataset that I used for this project is the provided data set in the Jupiter notebook that consists of 13k human images and 8k dog images. The human images are used for the first part of the project which is using an OpenCV function to detect human faces. The dog images are the core part of the project. The data for dog images are organized as follows. There are train, valid, and test folders inside the dog_images folder that will be used for training, validation, and testing. For each folder there are 133 subfolders and each subfolder represent a different breed to the dog and it contains all images of dogs that belong to this breed. These breeds are the target of the dog breed classification. I will use the training images as the input to train my dog breed classifier and test the result against the test set.

Dog: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip

Human: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip

## Solution Statement

Here is the solution that I proposed for my model.

**Input transformation:**

- For the training images, my code will first resize the image to h, w = 80, 80 and then use the random crop to adjust it to 64 * 64 which is the input size for my neural network; for the test images, my code directly resizes the image to 64 * 64. The reason for such size is 64 is enough to encode different dog classes which won't be too large to cause the inefficiency in training.
- I used random crop and random horizontal flip to augment the dataset.

**Model Architecture:**

- The input size for my CNN is (3, 64, 64). The first step is to use convolutional layers to reduce the image size while also increasing the number of channels. Here I use four convolution layers conv0-3 to transform the input into (24, 7, 7). Reason: Use convolution to find the correlation within images at different locations.
- Now I get the flatten input of 24 * 7 * 7. And start to construct the fully connected layer. Here I have three fully connected layers fc1-3. And fc3 is the output layer that has 133 classes. Reason: Use fully connected layers to achieve the final classification.
- I also applied dropout in fully connected layers to prevent overfitting. Reason: prevent overfitting.

After completing the training, I should be able to test my model against the test data to see how well it performs.

## Benchmark model:

- This project requires to have an accuracy of about 10% which is 10 times better than guessing (less than 1%).
- I can also compare my architecture against successful image classification models such as VGG16 (need to perform transfer learning)
- I can also use some toy models that do well against the mnist dataset to see how those architectures work against my problem.

## Evaluation metric:

- As a classification problem, the nature metric is the accuracy. I will use this metric in my project.
- At the same time, cross entropy loss is very important for the back propagation of the neural network as well as evaluation of the validation data and test data.

## Project design:

- Import data sets that I mentioned before.

- Use the OpenCV function to detect human faces in the human data. Also try to apply the same function as a binary classifier for human/not human.
- Use VGG16 pre-trained models to detect dogs based on the predicted labels (the label within a range can be classified as dogs)
- Build a model from scratch to classify the dog breeds. The CNN design is mentioned above.
- Use transfer learning to create that complete the same task as before. I plan to use VGG16 and modify the output layer to suit my use case
- Finally, build a function that detects human and dog for an image. If it a human image, then output the resembling dog breed. If it is a dog image, output the breed type. If neither is true, output not detected.