

# Udacity Capstone Proposal

## CNN Project: Dog Breed Classifier

Chippy Mathew

June 2020

### Domain Background

In the last decade, machine learning (ML) became more popular thanks to very powerful computers that can handle to process a large number of data in a reasonable amount of time. Machine learning concept is introduced by Arthur Samuel in 1959 so it is not new, but today we can use lot of its potential. One more reason for this is that today there is a lot of digitized data that we need to successfully implement good ML models.

The Dog breed classifier is a well-known problem in ML. The problem is to identify a breed of dog if a dog image is given as input. If supplied an image of a human, we have to identify the resembling dog breed. The idea is to build a pipeline that can process real world user supplied images and identify an estimate of the canine's breed. This is a multi-class classification problem where we can use supervised machine learning to solve this problem. After completing this model, I am planning to build a web app where user can input an image and obtain prediction from this model. This project gives me an opportunity to build and deploy ML models, so I have chosen this as my capstone project.

### Problem Statement

The goal of the project is to build a machine learning model that can be used within web app to process real-world, user-supplied images. The algorithm has to perform two tasks:

***Dog face detector:*** Given an image of a dog, the algorithm will identify an estimate of the canine's breed.

***Human face detector:*** If supplied an image of a human, the code will identify the dog breed the human looks most like.

### Datasets and Inputs

For this project, the input format must be of image type, because we want to input an image and identify the breed of the dog. The dataset for this project is provided by Udacity. The dataset has pictures of dogs and humans.

***Dog images dataset:*** The dog image dataset has 8351 total images which are sorted into train (6,680 Images), test (836 Images) and valid (835 Images) directories. Each of this directory (train, test, valid) have 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds, some images are not full-sized. The data is not balanced because the number of images provided for each breed varies.

***Human images dataset:*** The human dataset contains 13233 total human images which are sorted by names of human (5750 folders). All images are of size 250x250. Images have different background and different angles. The data is not balanced because we have 1 image for some people and many images for some.

### **Solution Statement**

For performing this multiclass classification, we can use Convolutional Neural Network to solve the problem. A **Convolutional Neural Network (CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The solution involves three steps. First, to detect human images, we can use existing algorithm like OpenCV's implementation of Haar feature based cascade classifiers. Second, to detect dog-images we will use a pretrained VGG16 model. Finally, after the image is identified as dog/human, we can pass this image to an CNN which will process the image and predict the breed that matches the best out of 133 breeds.

### **Benchmark Model**

For our benchmark model, the CNN model created from scratch with an accuracy of at least 10%. This can confirm that the model is working because a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%. The CNN model created using transfer learning must have accuracy of 60% and above.

### **Evaluation Metrics**

The problem we are trying to solve is a classification problem. Because our data is an unbalanced, simple accuracy score is not very good here. Hence for this multi class classification, Multi class log loss will be used to evaluate the model. Log loss takes into the account of uncertainty of prediction based on how much it varies from actual label and this will help in evaluating the model.

### **Project Design**

Step 1: Import the necessary dataset and libraries, Pre-process the data and create train, test and validation dataset. Perform Image augmentation on training data.

Step 2: Detect human faces using OpenCV's implementation of Haar feature based cascade classifiers.

Step 3: Create dog detector using pretrained VGG16 model.

Step 4: Create a CNN to classify dog breeds from scratch, train, validate and test the model.

Step 5: Create a CNN to Classify Dog Breeds using Transfer Learning with resnet50 architecture. Train and test the model.

Step 6: Write an algorithm to combine Dog detector and human detector.

- If dog is detected in the image, return the predicted breed.
- If human is detected in the image, return the resembling dog breed.
- If neither is detected, provide output that indicates the error.

Our data is already divided into training, validation and test partitions so we can now use our train data to make a benchmark model using Convolutional Neural Networks. After creating a model we will test it with test data. When we get accuracy over 10% we will proceed on building a new model using transfer learning. With transfer learning, we can build our model with fewer data to give us a better result. We will use the same training data as before. We will then test our model with the same test data as before but now we expect our accuracy to be over 60%. Then we can try to experiment with different model parameters to get better results.

### **References**

1. Original repo for Project - GitHub:  
<https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification>
2. Resnet101:  
[https://pytorch.org/docs/stable/\\_modules/torchvision/models/resnet.html#resnet101](https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html#resnet101)
3. Imagenet training in Pytorch:  
<https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f6173/imagenet/main.py#L197-L198>
4. Pytorch Documentation: <https://pytorch.org/docs/master/>
5. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
6. [https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)
7. [http://wiki.fast.ai/index.php/Log\\_Loss](http://wiki.fast.ai/index.php/Log_Loss)
8. <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>