**Capstone Project** **Maanav Shah**
**Machine Learning Engineer Nanodegree** **1st May, 2020**

# Convolutional Neural Networks (CNN) Project
# Dog Breed Identification

## Domain Background

The Dog breed classifier is a well-known problem in Machine Learning. The problem is to identify a breed of dog. The input can be a dog image or an image of a human, and we should be able to predict the breed of the dog or if that human was a dog, which breeds it would belong. 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 using Amazon SageMaker where users can input an image and obtain prediction from this model. This project gives me an opportunity to build and deploy Machine Learning models, so I have chosen this as my capstone project.

One another reason to build this project is to create an android application that will able to tell which breed does this dog belongs to, using the real-time camera.

In the last decade, ML becomes more popular thanks to very powerful computers that can handle to process lots of data in a reasonable amount of time. The machine learning concept was introduced by Arthur Samuel in 1959 so it is not new, but today we can use lots 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.

Dog breed identification problem is well known in the ML community. We can find it on Kaggle:
https://www.kaggle.com/c/dog-breed-identification/overview/description

Here is a link to an article where some similar academic work is done but on flowers:
https://www.researchgate.net/publication/320746968_Flower_species_recognition_system_using_convolution_neural_networks_and_transfer_learning

## Problem Statement

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

1. Dogface detector

   Given an image of a dog, the algorithm will identify an estimate of the canine's breed.

2. Human face detector

   Given an image of a human, the code will identify the resembling dog breed.

The bigger picture is that we take a picture with a phone and the app tells us what dog breed is on the picture. Additionally, we want that app to tell us to what dog breed a human is most look alike. We want to have an answer to a few questions here. The main question is What dog breed is on the picture? The second question is: How would you look if you were a dog? To answer these two questions we first have to answer the question: Is on the picture human or a dog?

This is a supervised learning problem and because we have our dog images divided into breed classes we will use classification predictive modeling more precisely multi-class predictive model.

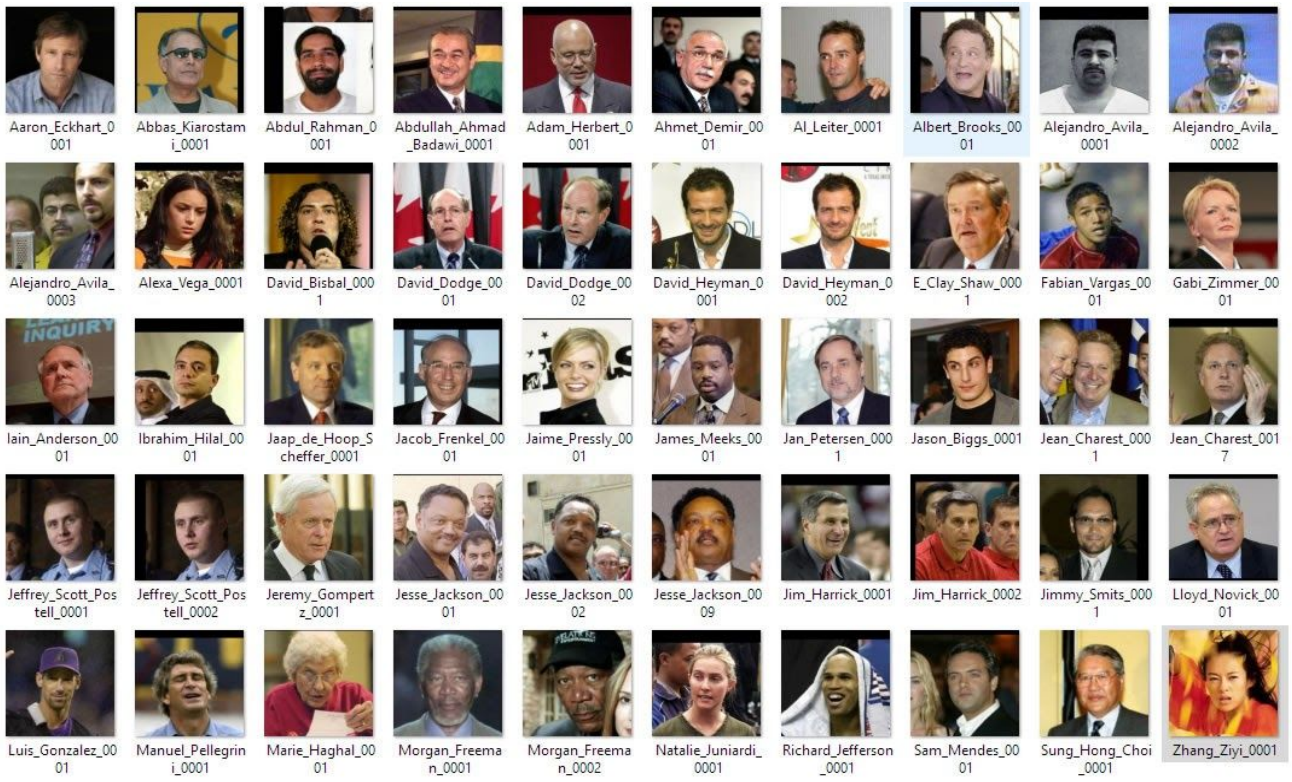
## Datasets and Input

To solve our problem our input data must be images because we want to the user takes an image of a dog (or human) with his phone, sent it to our server and we would return what dog breed is most likely in a picture (or to which dog breed is human most look like). All data for this project is provided by Udacity. We have pictures of dogs and pictures of humans.

All dog pictures are sorted in the training dataset (6,680 Images), testing dataset (836 Images), and validation dataset (835 Images) directory and all the images in these directories are sorted in breed directories. We have 133 folders (dog breeds) in every train, test, and validation set.

Human pictures are sorted by name of each human. We have 13,234 Files (Images), 5,749 directories (Humans). Our data is not balanced because we have one image of some people and several for others. The same is for dog images.

Dog images have different image sizes, different backgrounds, some dogs are in full sizes and some just ahead. Lightning is not the same. That is actually ok because we don't know how users' images will be, and we want that our model works on different types of images. Human images are all of the same size 250×250. Images are with different backgrounds, light, from different angles, sometimes with few faces on the image. Here are a few samples of our dog and human images:


We can see from the example above that some images have more than one human or a dog on the same image. Maybe it would be a good idea to remove those images. I will test this to see the results

## Solution Statement

We will use Convolutional Neural Networks (CNN) to make a model. CNN is a part of deep neural networks and is great for analyzing images. It would be great if we could mix CNN with XGBoost to find the best possible model but that idea needs more research because I don't know if this is doable. To find if the picture is human or not we will use the OpenCV model. And to find if the dog is on a picture we will use a pre-trained VGG16 model.

We will create our CNN model using transfer learning because we need a lot fewer images this way and we still can get great results.

## Benchmark Model

For our benchmark model, we will use the Convolutional Neural Networks (CNN) model created from scratch with an accuracy of more than 10%. This should be enough to confirm that our model is working because random guess would be 1 in 133 breeds which are less than 1% if we don't consider unbalanced data for our dog images.

## Evaluation Metrics

The problem we try to solve is a classification problem. Because our data is an unbalanced, simple accuracy score is not very good here. On Kaggle they use multi-class log loss metrics to evaluate models. I will also use this metric so I can compare it to results on Kaggle. I will also use F1 score testing because it considers precision and recall and it is easier for me to understand results.

We calculate F1 with the formula:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

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 know we expect our accuracy to be over 60%. Then we can try to experiment with different model parameters to get better results. We will use f1 score and log loss to evaluate our models.

## Project Design

We will divide the project design into the following steps:

1. Import the necessary dataset and libraries, pre-process the data, and create training, testing, and validation dataset. Perform Image augmentation on the training data.
2. Detect human faces using OpenCV's implementation of Haar feature-based cascade classifiers.
3. Create a dog face detector using the pre-trained VGG16 Model.
4. Create a CNN to classify dog breeds from scratch, train, validate, and test the model.
5. Create a CNN to Classify Dog Breeds using Transfer Learning with resnet101 architecture. Train and test the model.
6. Write an algorithm to combine the dog detector and human detector.
    - If a dog is detected in the image, return the predicted breed.
    - If a human is detected in the image, return the resembling dog breed.
    - If neither is detected, provide the output that indicates the error.

## References


1. GitHub Repository: http://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification
2. Restnet101 https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html#resnet101
3. ImageNet Training in PyTorch https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f
4. PyTorch Documentation https://pytorch.org/docs/master/
5. Convolution Neural Network (CNN) https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53