PROJECT DESCRIPTION :

Given an image of a dog, this algorithm will identify an estimate of the canine’s breed. If supplied an image of a human, the code will identify the resembling dog breed.

OBJECTIVE:

DATASET UNDERSTANDING:

train\_files, valid\_files, test\_files - numpy arrays containing file paths to images

train\_targets, valid\_targets, test\_targets - numpy arrays containing onehot-encoded classification labels

dog\_names - list of string-valued dog breed names for translating labels

There are 133 total dog categories.

There are 8351 total dog images.

There are 6680 training dog images.

There are 835 validation dog images.

There are 836 test dog images.

There are 13233 total human images.

The images are in the size of 224x224 jpg format.

**Data Understanding**

* loading the data-set
* understanding the inter-class variation between the dog
* identify the evaluation model

**Feature Engineering and Exploratory Data Analysis**

In the feature engineering and exploratory data analysis the number of human images in dog files and number of dog images in human file are predicted.

MODEL ARCHITECTURES:

VGG-16 bottleneck features

[ResNet-50](https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogResnet50Data.npz) bottleneck features

**Modelling and Evaluation**

* defining pipelines that combine: standardization of the numerical features, feature assembly, and a selected binary classifier (logistic regression, random forest classifier or gradient boosting classifier)
* splitting the data-set into training and test set
* pipeline training and tuning using grid-search with cross validation on the training data
* analyzing model performance in cross validation (using AUC metric) and extracting feature importances

STEPS TO EXECUTE:

[Step 0](https://viewooizm3ck72.udacity-student-workspaces.com/notebooks/dog-project/dog_app.ipynb#step0): Import Datasets

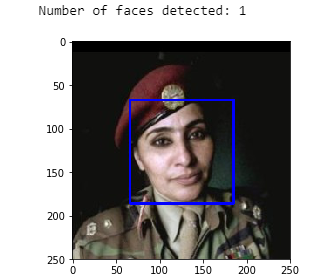
train\_files, valid\_files, test\_files - numpy arrays containing file paths to images

train\_targets, valid\_targets, test\_targets - numpy arrays containing onehot-encoded classification labels

dog\_names - list of string-valued dog breed names for translating labels

[Step 1](https://viewooizm3ck72.udacity-student-workspaces.com/notebooks/dog-project/dog_app.ipynb#step1): Detect Humans

The project includes OpenCV's implementation of [Haar feature-based cascade classifiers](http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html" \t "_blank) to detect human faces in images. OpenCV provides many pre-trained face detectors.



[Step 2](https://viewooizm3ck72.udacity-student-workspaces.com/notebooks/dog-project/dog_app.ipynb#step2): Detect Dogs

This step has pre-trained [ResNet-50](http://ethereon.github.io/netscope/#/gist/db945b393d40bfa26006) model to detect dogs in images.The first line of code downloads the ResNet-50 model, along with weights that have been trained on [ImageNet](http://www.image-net.org/" \t "_blank), a very large, very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of [1000 categories](https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a).

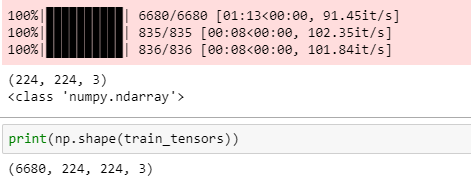


[Step 3](https://viewooizm3ck72.udacity-student-workspaces.com/notebooks/dog-project/dog_app.ipynb#step3): Create a CNN to Classify Dog Breeds (from Scratch)

 In this step, you will create a CNN that classifies dog breeds.

**Pre-process the Data**

The images are rescaled by dividing every pixel in every image by 255.



### (IMPLEMENTATION) Model Architecture

### 

### 

### 

[Step 4](https://viewooizm3ck72.udacity-student-workspaces.com/notebooks/dog-project/dog_app.ipynb#step4): Use a CNN to Classify Dog Breeds (using Transfer Learning)

[Step 5](https://viewooizm3ck72.udacity-student-workspaces.com/notebooks/dog-project/dog_app.ipynb#step5): Create a CNN to Classify Dog Breeds (using Transfer Learning)

 CNN must attain at least 60% accuracy on the test set.

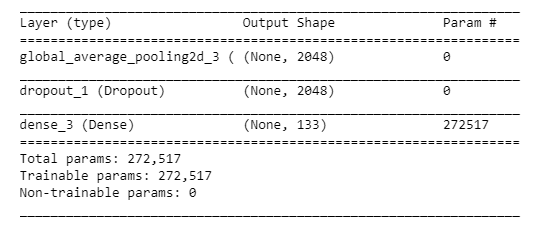
In Step 4, the code contains transfer learning to create a CNN using VGG-16 bottleneck features. This includes bottleneck features from a different pre-trained model. To make things easier,pre-computed the features for all of the networks that are currently available in Keras:

* [VGG-19](https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogVGG19Data.npz) bottleneck features
* [ResNet-50](https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogResnet50Data.npz) bottleneck features
* [Inception](https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogInceptionV3Data.npz) bottleneck features
* [Xception](https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogXceptionData.npz) bottleneck features

The files are encoded as such:

Dog{network}Data.npz

where {network}, in the above filename, can be one of VGG19, Resnet50, InceptionV3, or Xception..



[Step 6](https://viewooizm3ck72.udacity-student-workspaces.com/notebooks/dog-project/dog_app.ipynb#step6): Write My Algorithm

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 in the image, provide output that indicates an error.

[Step 7](https://viewooizm3ck72.udacity-student-workspaces.com/notebooks/dog-project/dog_app.ipynb#step7): Test My Algorithm

1.I created a folder in the notebook and uploaded the images in it.

2.Next I created a list in the program.

3. And I used my algorithm to identify dog and human being.

