# Dog Breed Classifier using Convolutional Neural Network

**Reza Jalayer**

**January 2021**

## 1. Definition

### 1.1 Project Overview

Image classification has a wide range of real-world applications and is used in different fields including automotive industry, security industry, health care industry, gaming and social media platforms. It also plays an important role in remote sensing images and is used for various applications such as environmental change, surveillance, geographic mapping, disaster control [1]. Correct classification has vital importance, especially in medicine for example. Therefore, improved methods are needed in this field.

Deep learning [2] is a powerful machine learning technique for solving a wide range of computer applications including image classification. It is composed of multiple processing layers that can learn more powerful feature representations of data with multiple levels of abstraction. In the deep learning technique, several numbers of models are available such as convolutional neural network (CNN).

Deep convolutional neural networks [3] provide better results than existing methods in the literature due to advantages such as processing by extracting hidden features, allowing parallel processing and real time operation. The concept of convolutions in the context of neural networks begins with the idea of layers consisting of neurons with a local receptive field, i.e., neurons which connect to a limited region of the input data and not the whole [4].

Nearly every year since 2012 has given us big breakthroughs in developing deep learning models for the task of image classification. Due to its large scale and challenging data, the ImageNet (image-net.org) challenge has been the main benchmark for measuring progress. ImageNet is a very large, very popular dataset used for image classification and other vision tasks. Some of the major architectures that made that progress possible are:

AlexNet: The 2012 paper from the university of Toronto became one of the most influential papers in the field after achieving a nearly 50% reduction in the error rate in the ImageNet challenge [5].

VGGNet: The 2014 Oxford University paper extended the idea of using a deep networking with many convolutions and ReLUs (Rectified Linear Units). A deep network with lots of small 3x3 convolutions and non-linearities [6].

## 1.2 Problem Statement

In this project I will go through building a machine learning model for dog breed classification. The problem starts by first distinguishing an image as a human or a dog image, and if it is detected as a dog image the dog breed will be predicted. If a human image is detected, then a “resembling” dog breed will be the shown output.

## 1.3 Solution Statement

As mentioned in section 1.1, CNN is a good technique for such image classification problem. The CNN training/validation input will be the RGB dog images. In this project two CNN models will be used: CNN from scratch (that can serve as a benchmark) and CNN using transfer learning from one of the state-of-the-art models mentioned in domain background section (VGG-16)

## 1.4 Datasets and inputs

The dataset used for this project includes 13233 human images as well as 8351 dog images. The dog images are grouped into 133 different breeds (classes). The dog images will be used for train/test the dog breed classifier model.

## 1.5 Evaluation Metrics

Cross Entropy loss (log loss for multi class data) will be calculated during model training as well as model testing on test data.

$$CE_{loss} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} t_{ij} \log{(p_{ij})}$$

Where $N$ and $C$ are number of samples and classes, $t_{ij}$ is only 1 if sample $i$ belongs to class $j$, *and* $p_{ij}$ is the predicted probability.

Also the accuracy will be evaluated on test data by simply calculating number of “correct predictions” divided by “total predictions”. However, since this is an unbalanced dataset (unbalanced dog images classes, see section 2.1 ), so this accuracy may not be a good indicator because classes with fewer sample would be affected by classes with more data samples.

$$accuracy = \frac{number\ of\ correct\ (dog\ breed)\ predictions}{number\ of\ total\ predictions}$$

## 2. Analysis

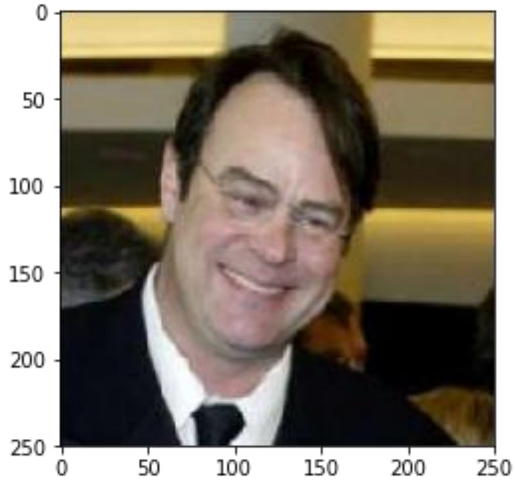
### 2.1 Data Exploration and Visualization

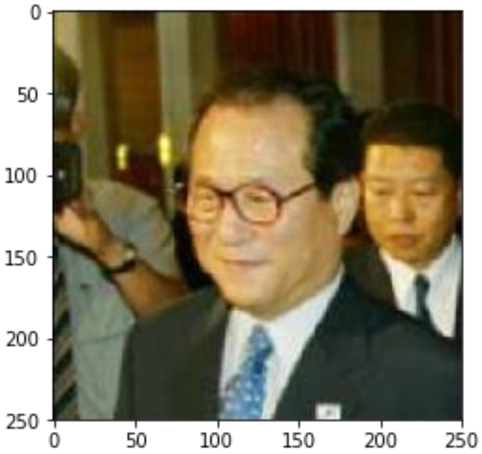
The dataset used for this project includes 13233 human images as well as 8351 dog images. The dog images are grouped into 133 different breeds (classes). The dog images will be used for train/test the dog breed classifier model. The images are BGR color images. and the average size of a dog image is 529by567 pixels. More information on dog images summarized below, showing a wide range of dog image size.

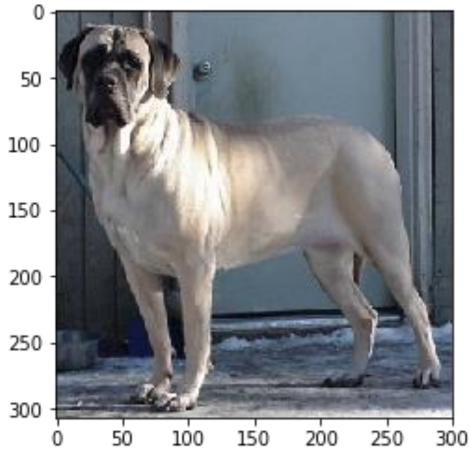
| | ***dog image height*** | ***dog image width*** |
|---|---|---|
| count | 8351.000000 | 8351.000000 |
| mean | 529.044905 | 567.032571 |
| std | 333.197594 | 389.006647 |
| min | 113.000000 | 105.000000 |
| 25% | 360.000000 | 375.000000 |
| 50% | 467.000000 | 500.000000 |
| 75% | 600.000000 | 640.000000 |
| max | 4003.000000 | 4278.000000 |

On the other hand all human images are exactly 250by250 pixels.

The human/dog images will be used to detect a human or dog, and then the dog images will be used to train a dog breed classifier. Each human image contains at least one human (face) and each dog image contains only one dog.







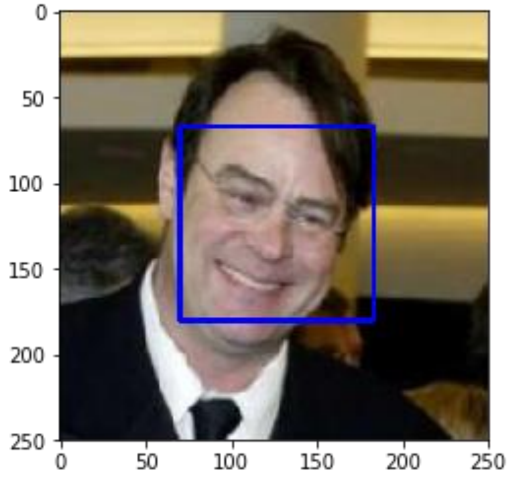
Dog images data is an unbalanced dataset, number of samples in each of the 133 classes shown below and it is clear that the dataset is unbalanced (i.e. different number of samples per class)

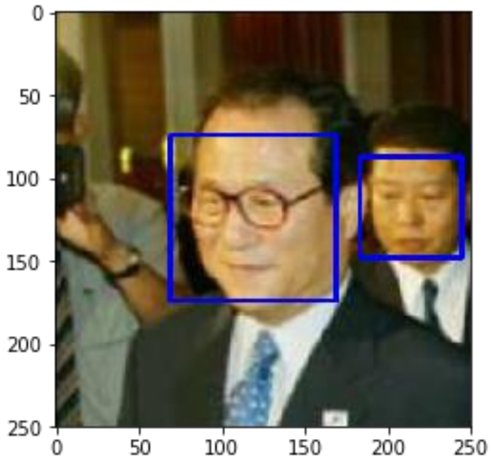
```
array([96, 93, 92, 89, 87, 86, 86, 84, 83, 83, 82, 82, 82, 81, 81, 81, 80,
       80, 80, 80, 80, 80, 80, 79, 79, 79, 79, 78, 78, 78, 77, 77, 76, 76,
       75, 74, 74, 73, 73, 72, 71, 71, 71, 70, 70, 68, 67, 67, 66, 66, 66,
       66, 66, 66, 66, 66, 65, 65, 64, 64, 63, 63, 63, 63, 63, 62, 62, 62,
       62, 62, 62, 62, 62, 61, 61, 60, 60, 60, 60, 60, 59, 59, 59, 59, 58,
       58, 58, 57, 57, 56, 56, 55, 55, 55, 55, 55, 55, 54, 54, 53, 53, 53,
       52, 51, 51, 51, 50, 49, 49, 49, 46, 46, 44, 44, 44, 42, 42, 42, 42,
       41, 41, 40, 39, 39, 38, 38, 38, 37, 37, 36, 35, 33, 33])
```

## 2.2 Algorithms and Techniques

Human face detector:

For this section, we use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images [7]. OpenCV provides many pre-trained face detectors, stored as XML files on github. We have downloaded and stored one of these detectors. The output of algorithm is the number of faces detected. Following two sample images show how the algorithm detected human faces: one face in first image, and two faces in second image.





Dog Detector:

For this section, we use a pre-trained model to detect dogs in images. The pretrained model is VGG-16, along with weights that have been trained on ImageNet. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories, out of which 118 are dog categories. The implemented algorithm returns the index of the predicted class (an integer 0-999), if the index is between 151-268, then a dog is detected.

Dog Breed classifier:

Assuming a dog or human is detected in an image, the dog breed classifier will predict the dog breed (or resembling dog breed in case of a human!). As mentioned in project overview, CNNs are really effective for image classification as the concept of dimensionality reduction suits the huge number of parameters in an image.

At first a CNN model will be developed from scratch consists of 3 convolutional/maxpool layers followed by 3 fully connected layers. This model will be trained and tested and can serve as a benchmark model. This model can be called “CNN_scrach”

After that a pretrained VGG-16 CNN will be used as a transfer learning. VGG16 is one of the best performing architecture for large scale image recognition [6].

For such transfer learning only the fully connected layers needs to be trained. The model will be tested, and the improved accuracy will be measured. This model can be called “CNN_transfer”

## 2.3 Benchmark

A simple benchmark model that will be used is multi-class classification logistic regression (using sk-learn package) just to show the basic feasibility of the dog breed classification using the provided input data. We do not expect a good accuracy for this benchmark model since this may not be a good approach for such complicated image analysis. This model can be called “LR_model”.

The next model that can also act as a benchmark will be the CNN_scratch mentioned in previous section. We expect a higher result accuracy compared to the LR_model. The expected accuracy for LR_model is less than 10% and for CNN_scrach is over 10%, and of course a lower cross entropy loss for CNN_scrach compared to LR_model.

The CNN_transfer model is expected to have a much higher accuracy of at least 60%, and a much lower cross entropy loss.

## 3. Methodology

### 3.1 Data Preprocessing

For the purpose of model train/test, the 8351 dog images are split into 3 groups: 80% train data, 10% validation data and 10% test data.

For CNN models, the selected input tensor size is (224,224,3). This means all the dog images are resized to 224×224 pixels. From section 2.1, we see that the dog images have a wide range of pixels (heigh × width), with an average of 529×567. On the other hand, selecting a larger tensor would incur expensive computation or memory problem. I found 224 a conventional input tensor size for similar image analysis problems. Also, the input tensor is normalized to prevent the so-called vanishing (and exploding) gradients [8].

Other transforms used (on train data only) are random 10-degree rotation and random scaling (0.9-1.1), this is because similar breeds on different pictures can have different scales due to closeness to camera or camera zoom. Also 10-degree rotation is to cover random dog position in the picture.

For LR_model training, the input data preprocessing only includes image resizing (224,224) and normalization.

### 3.2 Implementation

Human Face detector and dog detector explained in section 2.2.

The LR_model is implemented using the SK-learn logistic regression [9], simply as:

LR=LogisticRegression(C=1,solver='lbfgs',multi_class='multinomial',max_iter=10).fit(Xtrain,Y_train)

CNN models:

CNN_scratch structure is shown below: it consists of 3 convolutional layers with 16,32,64 filter depth. The kernel size is always 3, with a stride and padding of 1. Maxpool filter with kernel size of 2 is applied at the end of each convolution layer. After that there would be 3 fully connected layers with hidden nodes of 200 in each layer. The input layer to fc1 has a size of (224/8 × 224/8 × 64): note that 64 is the filter depth of conv3 layer, and 224/8 is coming from 3 maxpool layers. The dropout function is applied at the end of fc1 and fc2 layers to prevent overfitting.

```
Net(
(conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mod
e=False)
(fc1): Linear(in_features=50176, out_features=200, bias=True)
(fc2): Linear(in_features=200, out_features=200, bias=True)
(fc3): Linear(in_features=200, out_features=133, bias=True)
(dropout): Dropout(p=0.5)
)
```

CNN_transfer has a big structure based on VGG-16. In such transfer learning we will use the VGG-16 model as is, except training the fc layers to have desired number of classes, so we freeze all the parameters of all convolution layers, and the training will be done only for 3 fc layers. As mentioned original VGG-16 has 1000 output classes but our problem only needs 133 dog breed classes.

The structure shown below, please note that the final fc layer has 133 outputs and not 1000.

```
VGG(
(features): Sequential(
(0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(1): ReLU(inplace)
(2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(3): ReLU(inplace)
(4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=F
alse)
(5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(6): ReLU(inplace)
(7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(8): ReLU(inplace)
(9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=F
alse)
(10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(11): ReLU(inplace)
(12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(13): ReLU(inplace)
(14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(15): ReLU(inplace)
(16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=
False)
(17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(18): ReLU(inplace)
(19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(20): ReLU(inplace)
(21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(22): ReLU(inplace)
(23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=
False)
(24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(25): ReLU(inplace)
(26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(27): ReLU(inplace)
(28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(29): ReLU(inplace)
(30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=
False)
)
(classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace)
    (2): Dropout(p=0.5)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace)
    (5): Dropout(p=0.5)
    (6): Linear(in_features=4096, out_features=133, bias=True)
  )
)
```

### 3.3 Refinement

If the model network or number of parameters is too big then we may encounter overfitting problem. That is what I observed in the CNN_scratch training model that during the training iterations the training loss was decreasing but not the validation loss. I ended up increasing the dropout to 0.5 (started from 0.2) and also tried to reduce the number of hidden nodes in fc layers: started from 500 and got to 200. I also tried to add some regularization but did not see much difference so I removed it.

For CNN_transfer, I did not face any problem.

## 4. Results

### 4.1 Model Evaluation and Validation

CNN_scratch model training ran for 100 iterations (epochs), and the model with lowest validation loss is saved as the best model (not necessarily the 100th iteration one).

CNN_transfer is only ran for 10 iterations and gave a very good result, that is the advantage of using a transfer learning. The following table summarizes the cross-entropy loss as well as accuracy of all three models. For loss and accuracy definitions see section 1.5.

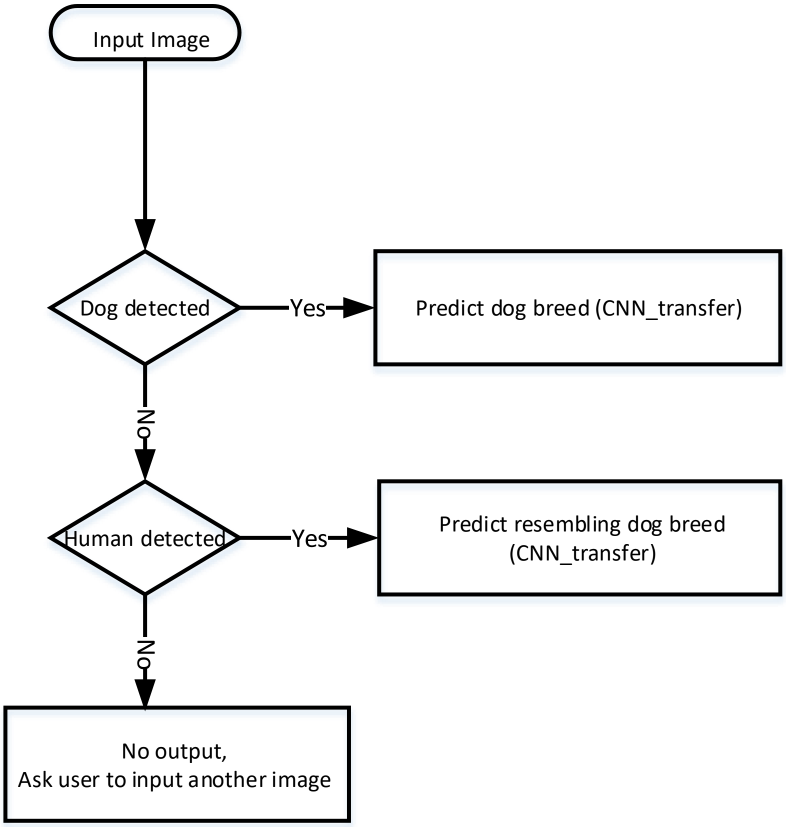
| | Cross Entropy loss | | | Accuracy |
|---|---|---|---|---|
| Model Name | Training Loss | Validation Loss | Test Loss | |
| LR_model | | | 5.56 | 5% |
| CNN_scratch | 3.34 | 3.78 | 3.76 | 11% |
| CNN_transfer | 0.73 | 0.67 | 0.69 | 79% |

### 4.2 Justification

It is clear that the performance of the CNN_transfer model is highly improved compared to other models and provides a good overall accuracy. The next section will show some sample prediction results.

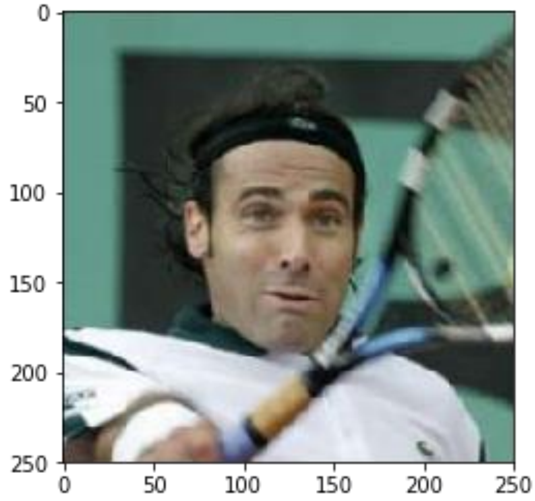
## 5. Final Application

The final application is implemented by the following steps:

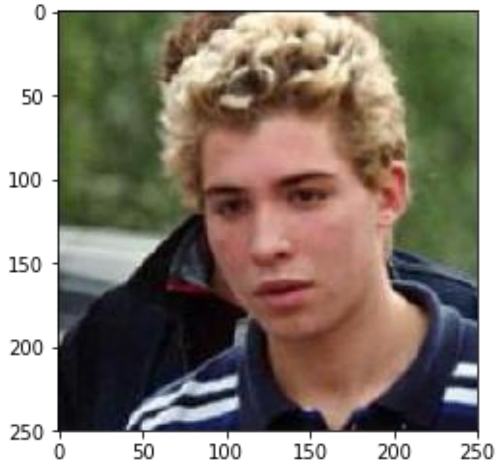

Input Image
Dog detected
Yes
Predict dog breed (CNN_transfer)
No
Human detected
Yes
Predict resembling dog breed (CNN_transfer)
No
No output,
Ask user to input another image

Few outputs of the logic shown below:

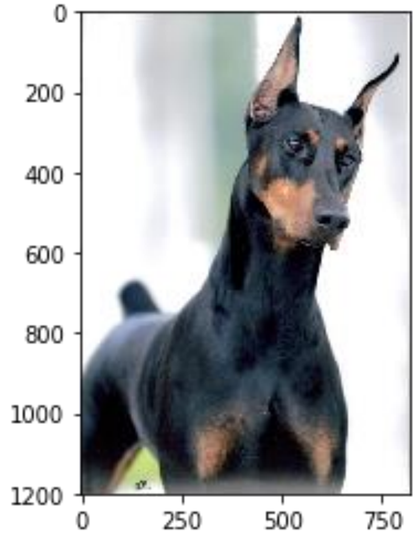
```
Hello Human!
 You look like a Brittany
```



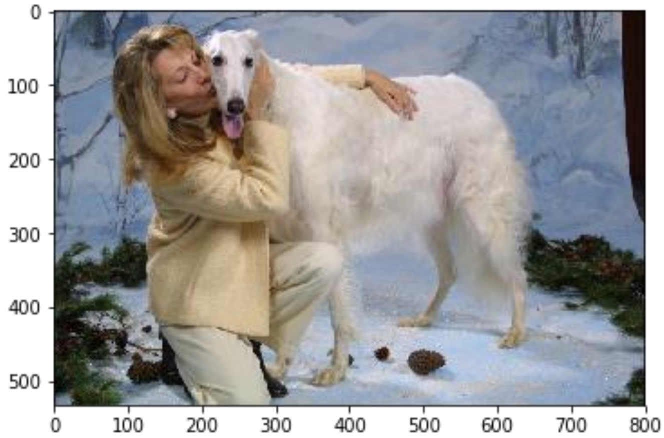
```
Hello Human!
 You look like a Afghan hound
```



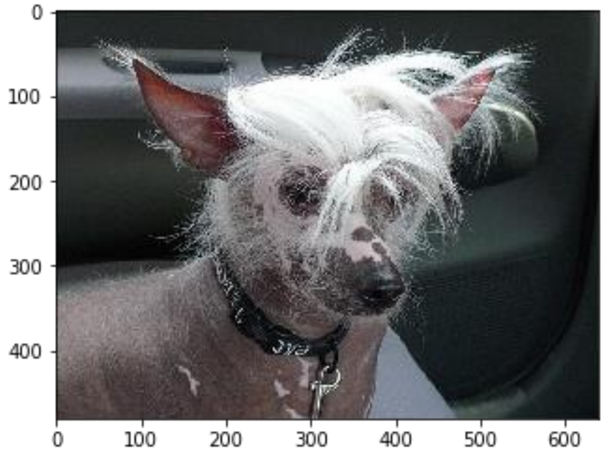
```
Hello Dog!
 Your predicted breed is Doberman pinscher
```



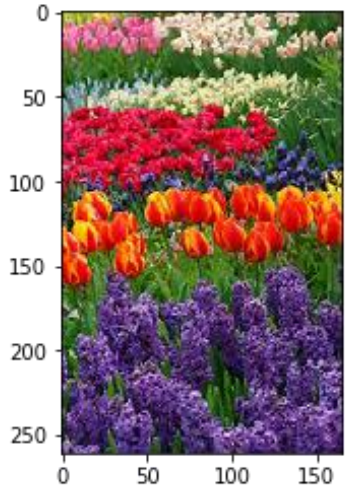
```
Hello Dog!
 Your predicted breed is Borzoi
```



```
Hello Dog!
 Your predicted breed is Chinese crested
```



```
error: input file contains no dog or human, please try again
```



## References

[1] P. Deepan, L.R. Sudha, The Cognitive Approach in Cloud Computing and Internet of Things Technologies for Surveillance Tracking Systems, 2020

[2] Yann LeCun, Yoshua Bengio and Geoffrey Hinton, Deep Learning, Nature, VOL 521, MAY 2015

[3] Saad Albawi, Tareq Abed Mohammed, Saad Al-Zawi, Understanding of a convolutional neural network, International Conference on Engineering and Technology (ICET), Aug 2017

[4] ralhad Gavali ME, J. Saira Banu PhD, Deep Learning and Parallel Computing Environment for Bioengineering Systems, 2019

[5] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012

[6] Karen Simonyan and Andrew Zisserman, VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, In Proc. ICLR 2015

[7] OpenCV: Cascade Classifier: https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

[8] How to normalize features in TensorFlow | by Chris Rawles | Towards Data Science: https://towardsdatascience.com/how-to-normalize-features-in-tensorflow-5b7b0e3a4177#:~:text=Normalizing%20inputs%20to%20nodes%20in%20a%20network%20helps,you%20may%20just%20want%20to%20normalize%20your%20inputs.

[9] sklearn.linear_model.LogisticRegression — scikit-learn 0.24.1 documentation (scikit-learn.org): https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html