# UDACITY Machine Learning Engineer Nanodegree

Capstone Project: Dog Breed Classifier using CNN

Chee Chung Lee

September 5th, 2020

## Definition

### Project Overview

The project was part of Udacity’s Machine Learning nanodegree and is popular also in deep learning nanodegree.

The aim of this project is to create an algorithm that could be part of a web application that is able to identify dog breeds if given an image as input. If the image contains a human face, then the algorithm will return the breed of dog that most resembles this person. If the image contains a dog, then the algorithm will return the breed of the dog. If the image contains neither, it will return an error to try again.

Our concern is to identify dog breeds from any image regardless if it contains human or dog. The data set imported from Udacity provided in the workspace consists of 8351 dog images in 133 different categories. The project has been successfully executed in Udacity workspace (GPU-enabled) with PyTorch and several convolution neural network (CNN) models. Besides predicting dog breeds with this technology, this machine learning algorithm can be applied to identifying other objects, animals, plants, behaviours and expression of people, diagnostics of equipment or even cancer detection. Modern CNN used in computer vision is founded by Yann LeCun which he proposed early form of back propagation learning algorithm in 1987.

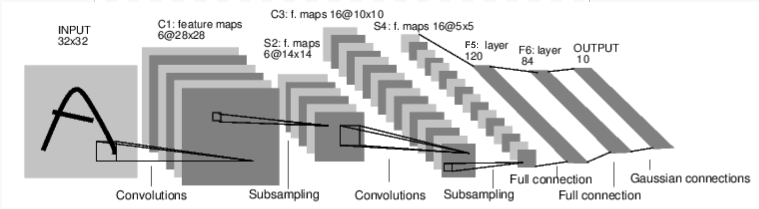


Figure 1: How convolution neural network takes input and produce output

### Problem Statement

The aim of this project is to create an algorithm that is able to identify a dog breed if given an image as input. For this problem to be solved, the data requires to be processed and classified with a high accuracy of above 70%. The algorithm has 3 main tasks to be performed and solved:

1. If the images supplied to the application has a human, then the application will return the breed of dog that most resembles the person in the image.

2. If the images supplied to the application has a dog, then the application will return the breed of dog.

3. If the image supplied to the application is neither a human or a dog, the application will return an error.

For us to solve these 3 tasks, Convolutional Neural Network with Transfer learning to classify the dog breeds is used. The benefits of Transfer Learning is that it can speed up the time it takes to develop and train a model by reusing the modules of already developed models. The feature extraction part of the model is reused whereas the classification part is re-trained. Since the feature extraction process is the most complex modelling challenge, reusing it allows you to train a new model with less computational resources and training time.

First a Convolution Neural Network from scratch (without using Transfer Learning) is built and the accuracy obtained was nearly 6% which is reasonable because of simple architecture implementation. By using transfer learning approach the accuracy increased to ~80%. This is because image recognition requires more complex feature detection and VGG16 possess these and able to achieve extremely high accuracy.

### Metrics

In order for us to solve the problems for the project, performance metrics need to be present to verify that the models we use are working well and the models we produce will give us a correct prediction. The first performance metrics we will use is accuracy. Accuracy is defined with the calculation as the following:

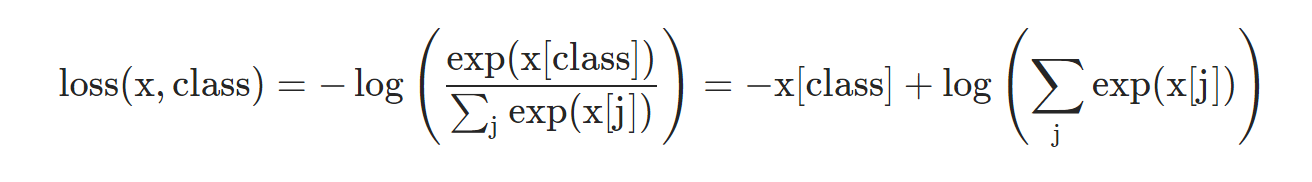
Accuracy=Number of correctly classified items/ All classified items

Accuracy is used when evaluating the performance of the model prediction for dog detector, human detector and the testing phase for CNN dog breed classification model. The human and dog detectors were tested on 100 images of each.

The human and dog detectors were tested on 100 images of each. The human face detector relied on OpenCV’s Haar Feature-based Cascade Classifiers and was trained on 13233 human faces which has been imported from sklearn. Whereas the dog detector was built on a pre-trained VGG16 model with ImageNet of 1000 classes. Its purpose was to confirm if an image was a dog or not. The accuracy of these two detectors were evaluated and their prediction were highly accurate at 96% for human detector and 96% for dog detector.

The other performance metrics we look at is the log loss which is used during the training and validation phase when building and developing the CNN dog breed classification model. Log Loss of the training and validation set can be evaluated after each epoch cycles to check if the model is improving and not overfitting. We do this because the problem we try to solve is a classification problem and the data set provided is unbalanced with uneven number of samples in different classes.

This log loss takes into account the uncertainty of the prediction based on how much it varies between actual label and the prediction. Thus, when working with log loss, the sample image would have probabilities to match with all the different classes. Log loss is defined using the PyTorch nn.CrossentropyLoss with the calculation of the following:



Loss is calculated based on the summation of all the difference between the probability of the classes by the prediction and the hot labels of one 1 and others being 0 through the cross entropy function. Training loss and validation loss is evaluated to be at a reducing trend after each epoch cycle.

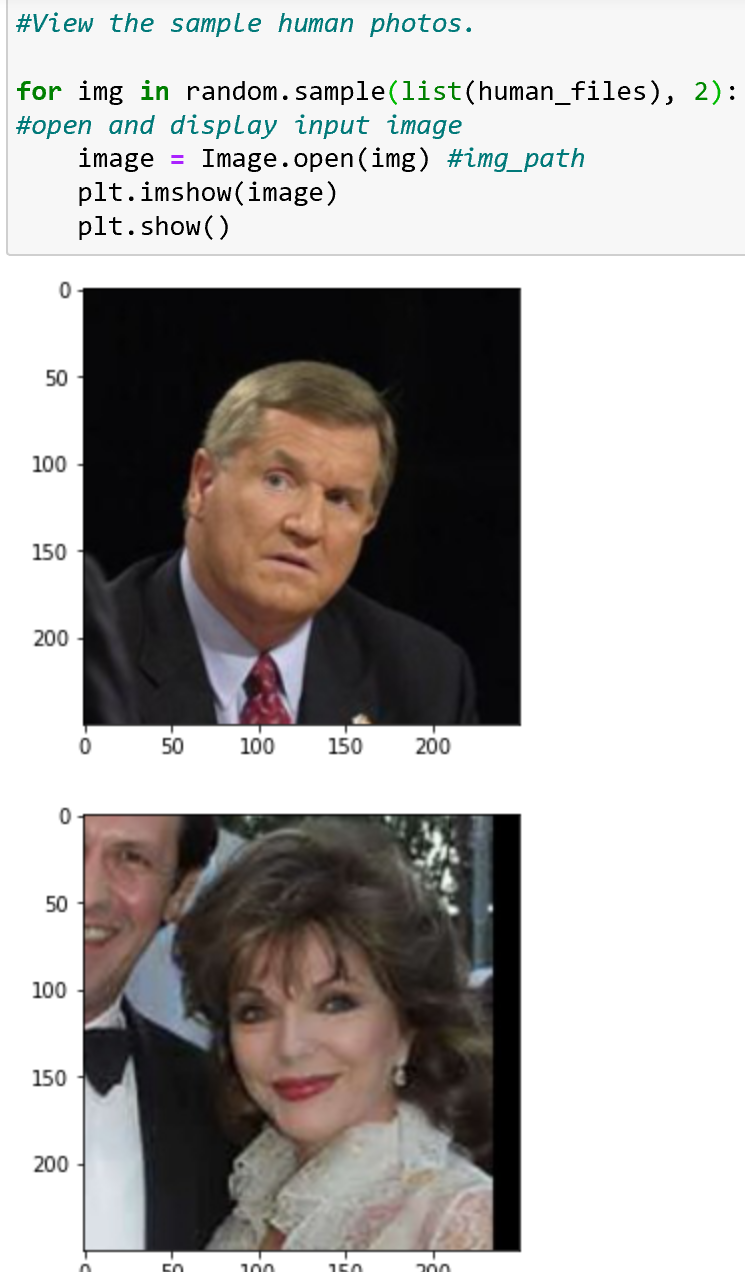
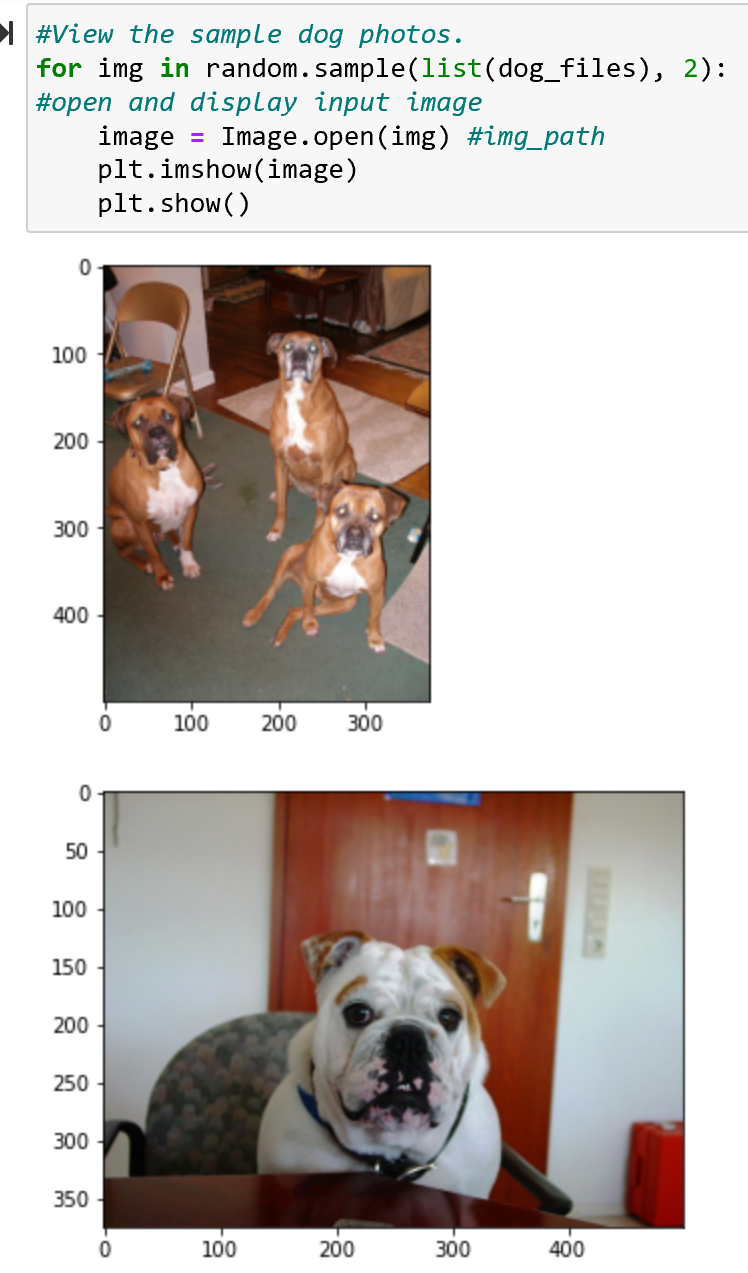
## Analysis

### Data Exploration

Two datasets were downloaded i.e. Dog Dataset and Human Dataset are used which are provided by Udacity. Dog dataset contains images of 133 classes of dogs in 133 folders each. All the images are not of the same dimensions, orientation of object of the dog or person are not the same. Number of dog or people in the image are sometimes more than 1. The links and insights to both the datasets are given below:

Human Dataset:

There are in total 13233 human images. Example of the images of the human dataset can be viewed in figure below:

Dog Dataset:

There are 133 total dog categories.

There are 8351 total dog images.

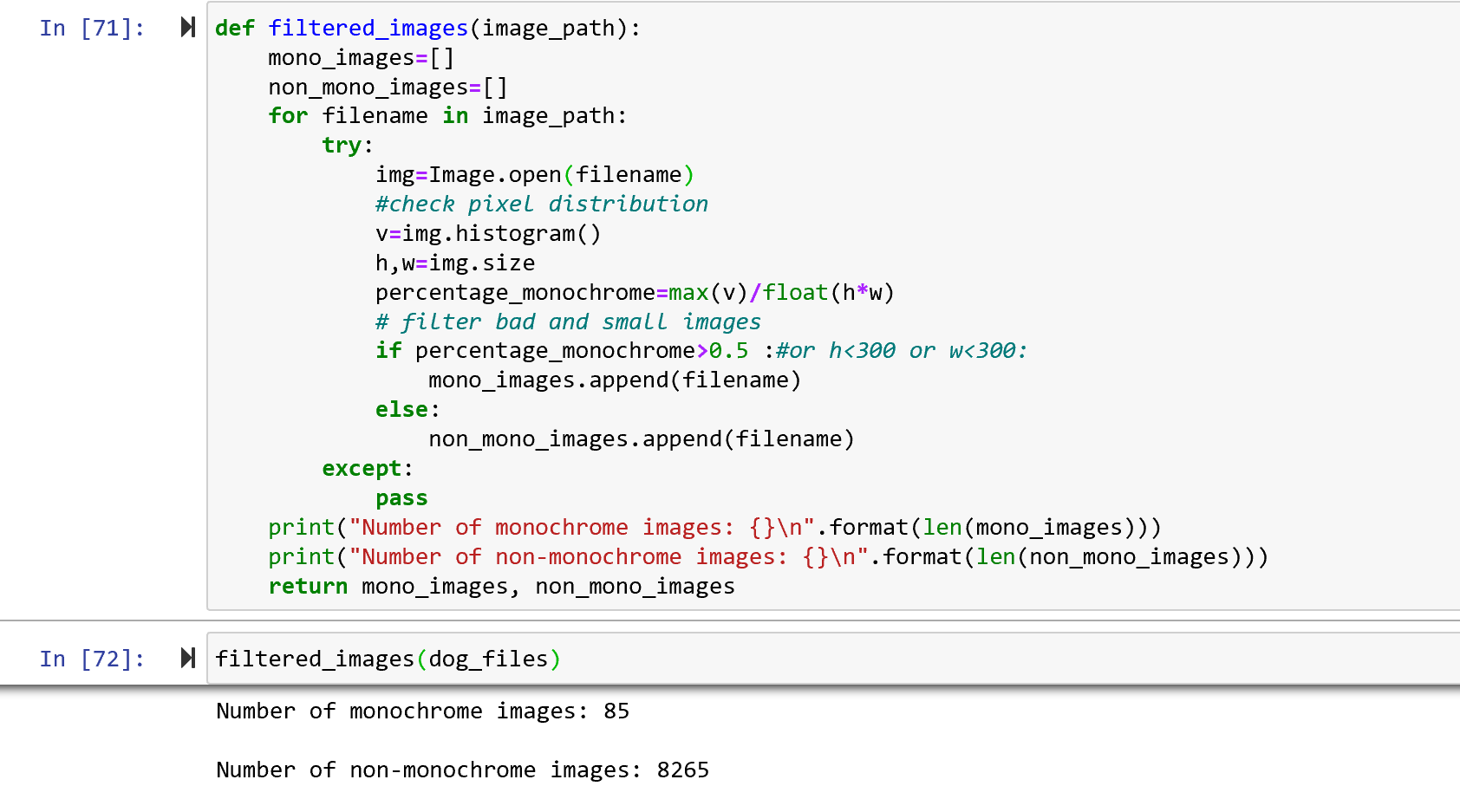
There are 6680 training dog images.

There are 835 validation dog images.

There are 836 test dog images.

Based on the sample above it can be seen that images for dog images are not of the same size. The number of dogs in the image can be more than 1 as well.

As for human images, they shape are the same. They are consistently the same for every image. They can have more than one human face in the image as well. There is no blurred image as running monochrome filter, it can be seen that the number of images which has a monochrome>0.5 is only 85 which is ~1% of the dog images has 50% of the image being a single colour. I would say the images are high quality and ideal for training.



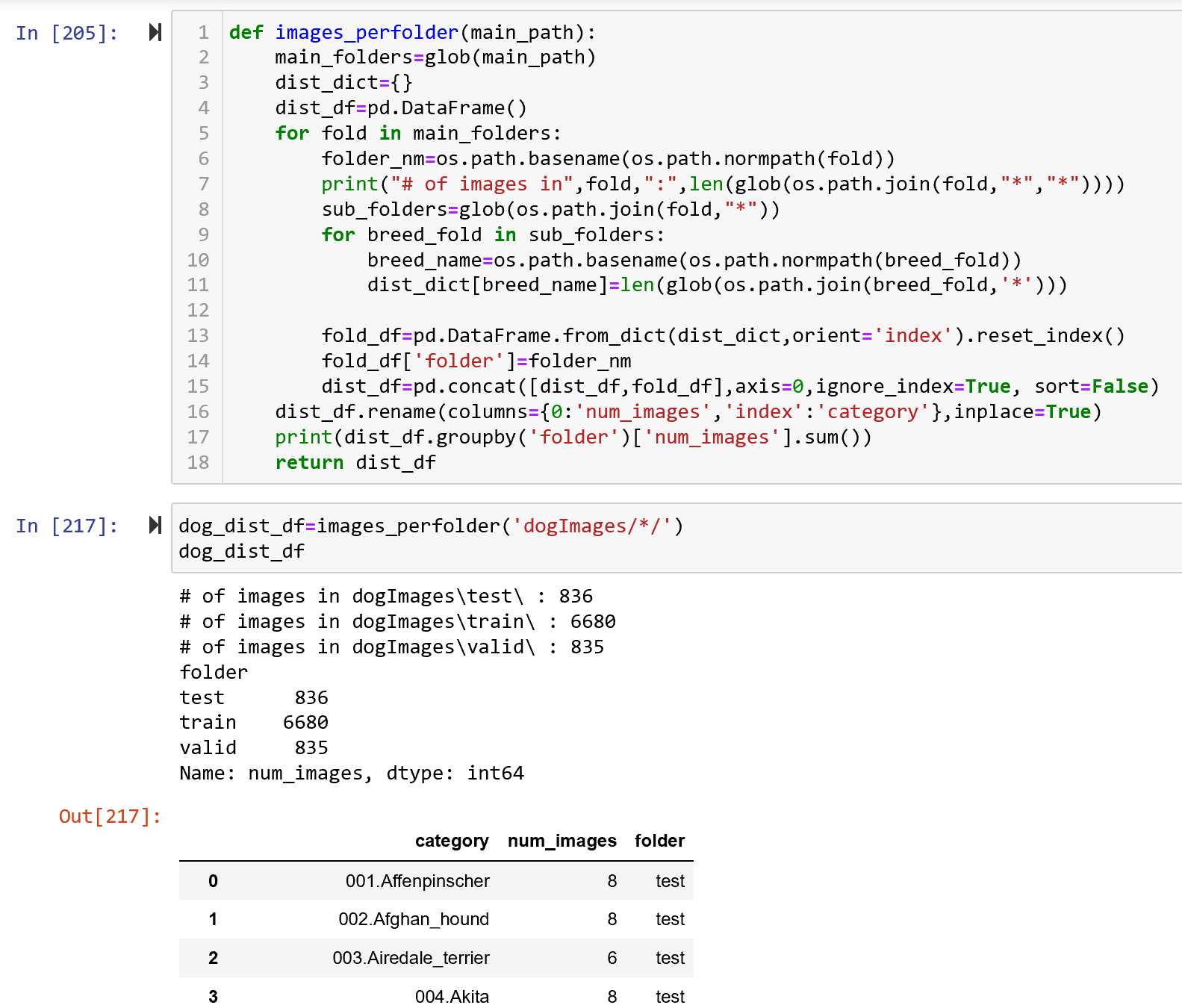
### Exploratory Visualization

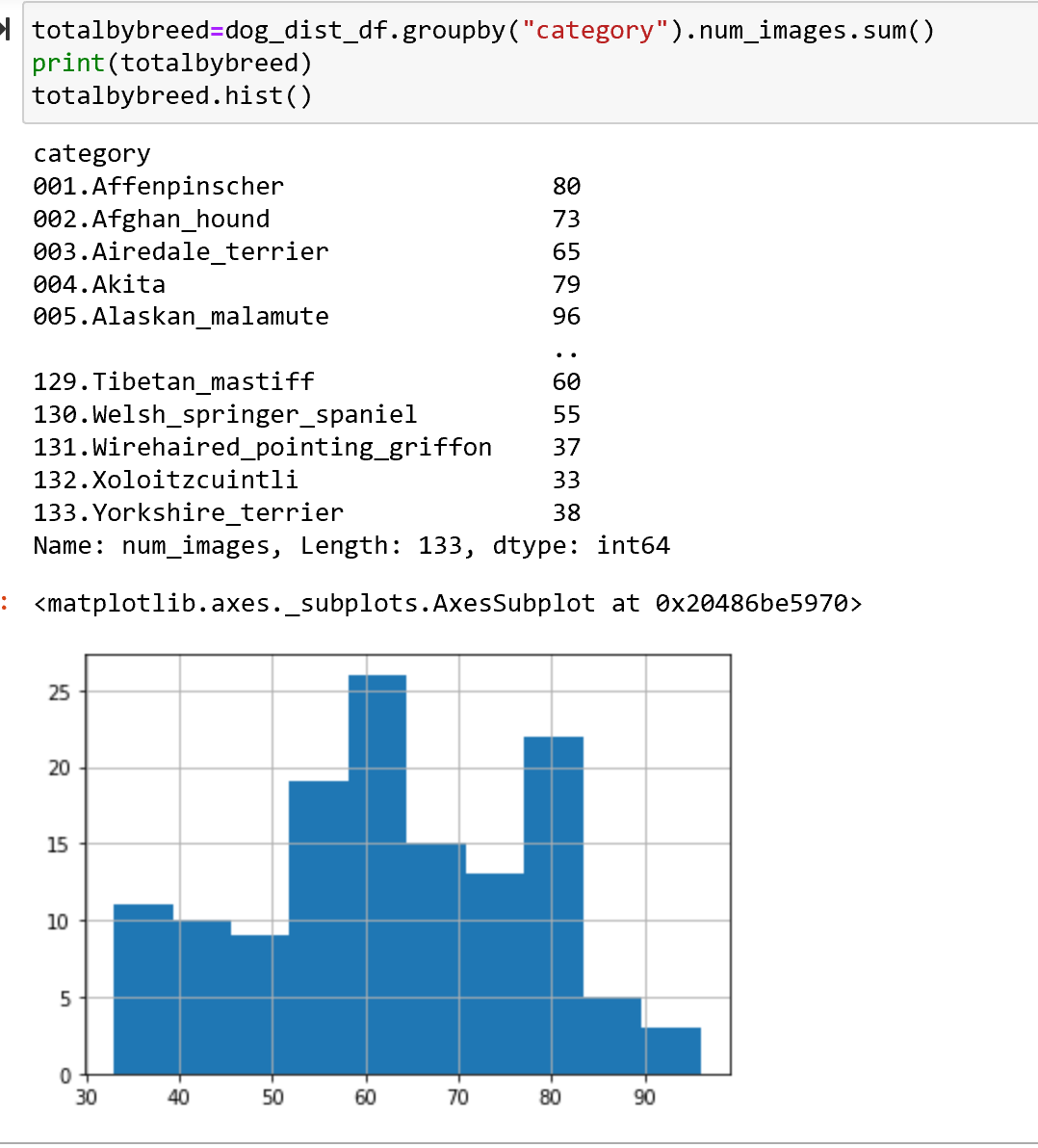
Dog images dataset have 8351 images which are separated in 3 file directories which are train (6680 images), test (836 images) and valid (835 images). Each of these 3 file directories have 133 folders which corresponds to the 133 different classes of dog breeds.

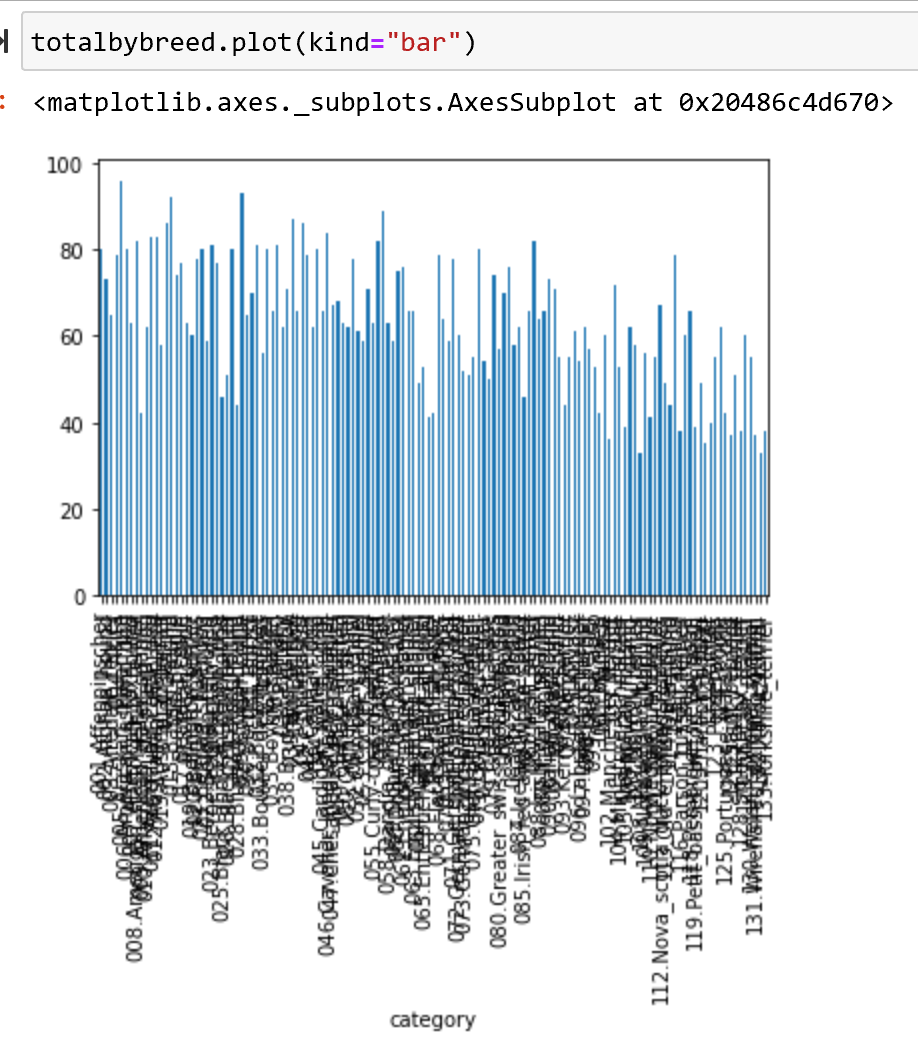
Human images dataset have 5,750 folders sorted by the name of the human and contain 13,233 total images. Our data is not balanced because we have one image of some people and several for others. (the difference is from 1 to 9 images in most cases) The same is for dog images as seen in histogram and plot of frequency of folder per dog breed. The dog images per breed varies from 33 to 96 images unevenly. Somehow the low index numbering of dog breed has higher number of images compared to higher index numbered dog breeds.

Dog images have different image sizes, different colour, different backgrounds, different angle and posture and orientation of the dog, some dogs are in full sizes and some just a head. The variation is good so that we can train the model to identify a breed regardless of these factors and minimise overfitting.

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. Human classification of the 5,749 folders was not performed, thus not a huge concern as long as a human can be identified in this project.







### Algorithms and Techniques

The use of a filter to a 2D array of input that produce an output is a convolution. The repeated use of the same filter which is smaller than the 2D array of inputs by moving the filter across the whole 2D array produces a map of outputs called feature map. This map indicates the locations and strength of a detected feature in the 2D array input, such as an image. What this means is the result is highly specific features that can be detected anywhere on input images.

Multiple use of convolution to create multiple feature map can be performed in parallel to create a convolution layer. These convolutional layers are the major building blocks used in convolutional neural network. Each output of each convolution of the layer will be linked to be an input for all the convolution for the next convolutional layer. The links between inputs and output for multiple convolution in multiple layers is a Deep Learning algorithm called the convolution neural network.

The convolution neural network learn and improve its prediction by automatically adjusting the filter bias and weight of the filters in parallel through the use of training dataset which has a set input and actual outcome. The neural network knows how much to learn and improve through back propogation of the loss between the actual and predicted outcome. With this loss, the neural network adjusts the weights and biases accordingly as it learns what type of features to extract through every training cycle.

Our input data are a bunch of dog and human images and our output required is a multiclass dog breed classification. Images are highly complex as it possess a lot of parameters and features to determine its classification. Besides producing feature matrix, CNN is able to carryout dimensionality reduction which suits the huge number of parameters in an image which needs to be reduced.

Thus, to solve this problem, we need to perform multiclass classification utilising CNN.

For a CNN to work affectively there are numerous parameter/technique in CNN architecture to improve the learning of the algorithm. They are:

* Dropouts. Based on dropout ratio it will randomly deactivate some of the connection between convolution layers for each feed to reduce overfitting of the algorithm.
* Learning rates is altered to ensure the best prediction can be obtained within the number of epochs
* Epochs are just the number of times the data is used to train the algorithm. Overtraining a model with too high a Epoch number can result in model being overfitting and validation loss will increase instead of reduce.
* Batch size are also critical. Smaller batch size allows the model to be trained more times and faster compared to higher batch size. However the downside of batch size being too small is the gradient estimate for the training of the model will be less accurate.
* Optimizer is used for back propagation. The ones tested are Adam and Stochastic gradient descent. Adam has a gradual reduction in learning rate to ensure the minimum loss can be achieved through learning. Stochastic gradient descent learns faster, but it has higher chance it does not achieve the best model prediction accuracy as the minimum loss is not achieved.
* Transfer learning is a technique by using pretrained algorithms which are highly accurate for CNN. It is incorporated by altering the output layers of the pretrained model.

### Benchmark

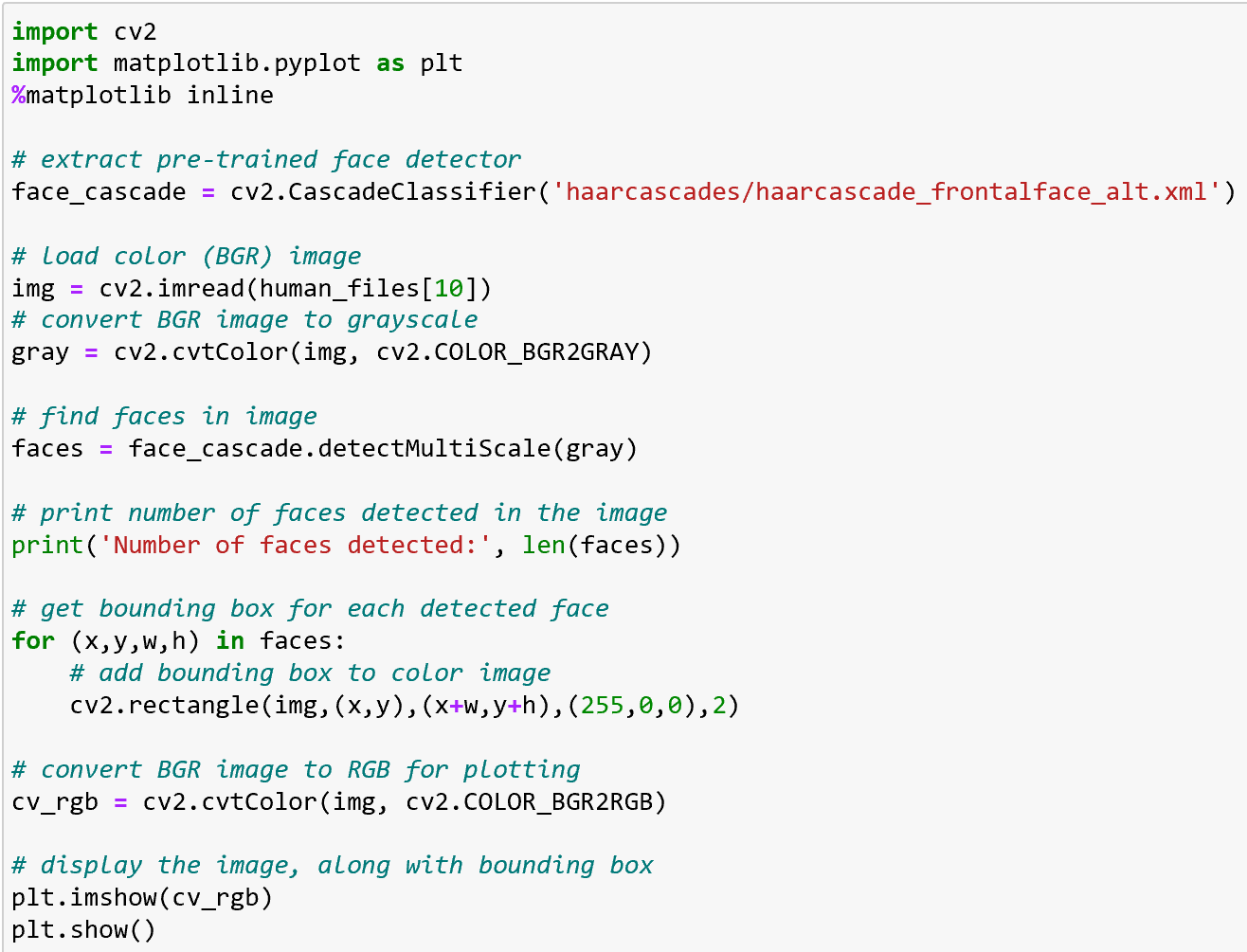
Benchmark model will be the CNN built from scratch which had approximately 10% accuracy rate. 10% is accuracy signifies that the model built is better than a random guess because if it was a guess it would have been 1 out of 133 (breeds).

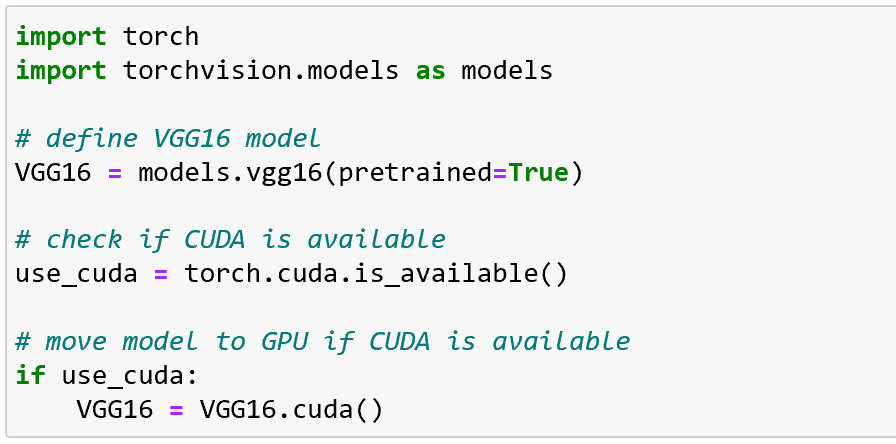
The CNN model created using transfer learning must have an accuracy level of 60% or above to show that the model is working well and trained well for the problem at hand.

## Methodology

### Data Preprocessing

Before any analysis can be performed, data pre-processing needs to take place to ensure the data can be fed correctly to the model in the correct format.





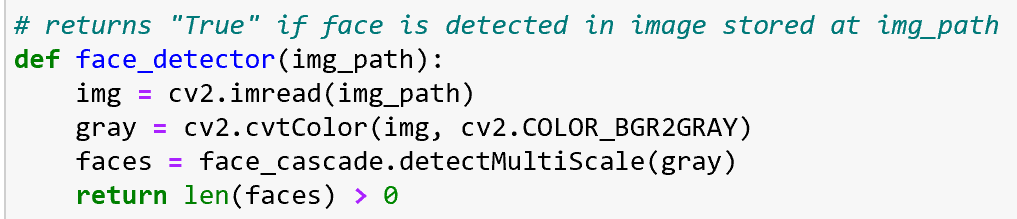
### Implementation

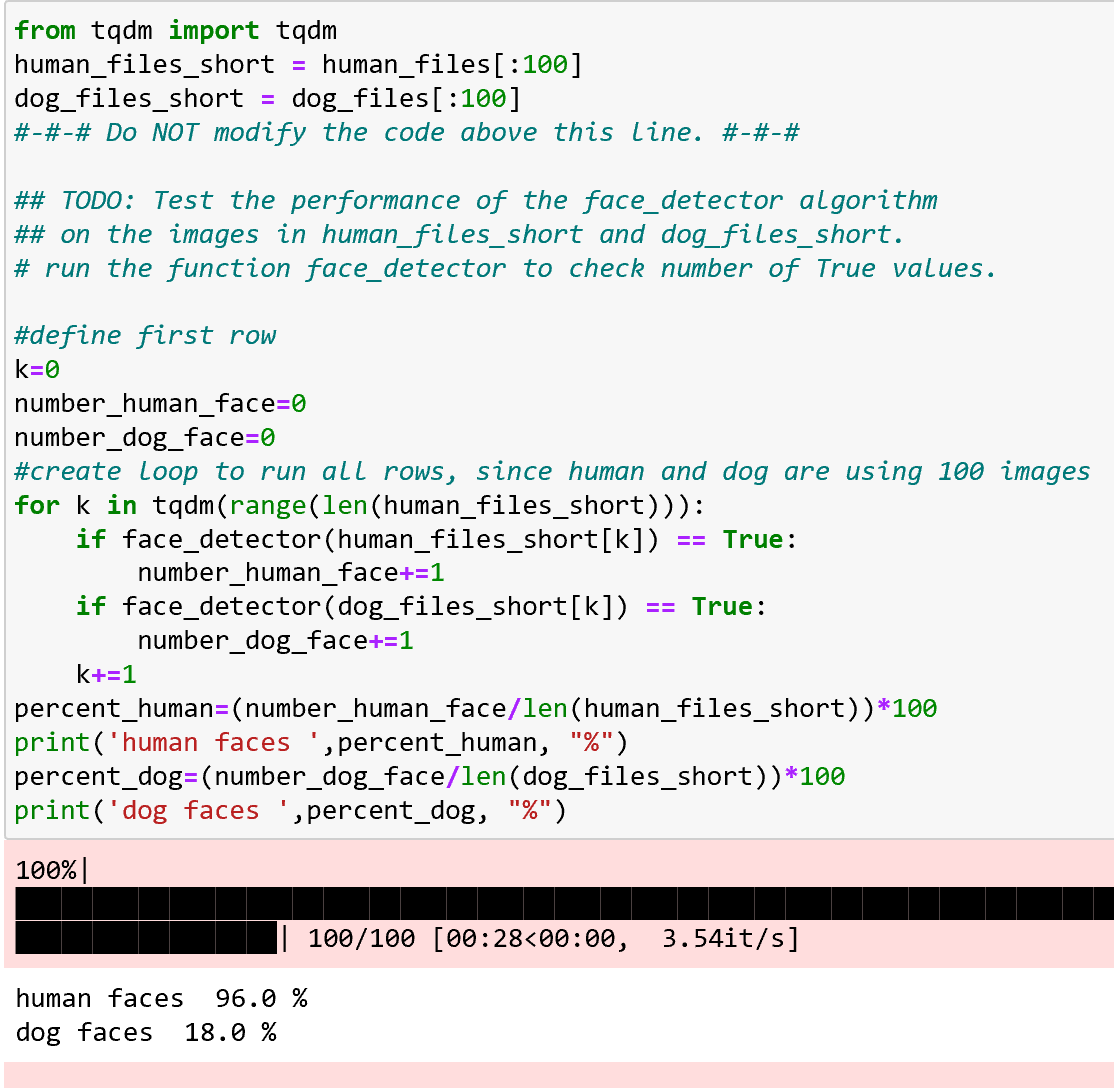
To solve this problem, we would deploy a multiclass classification utilising convolution neural network. A convolutional neural network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance through learnable weights and biases to various patterns/objects in the image. The distinct patterns in the images are able to be differentiated from each other using the algorithm.

In this section, the workflow for approaching a solution given the problem is as follow.

Step 1: Detect Humans

Explore different face detectors from OpenCV to be used. By creating a face detector we would be able to tell if a human is present in the image. To do this we will implement Haar feature-based cascade classifiers. The workflow required to detect faces are:



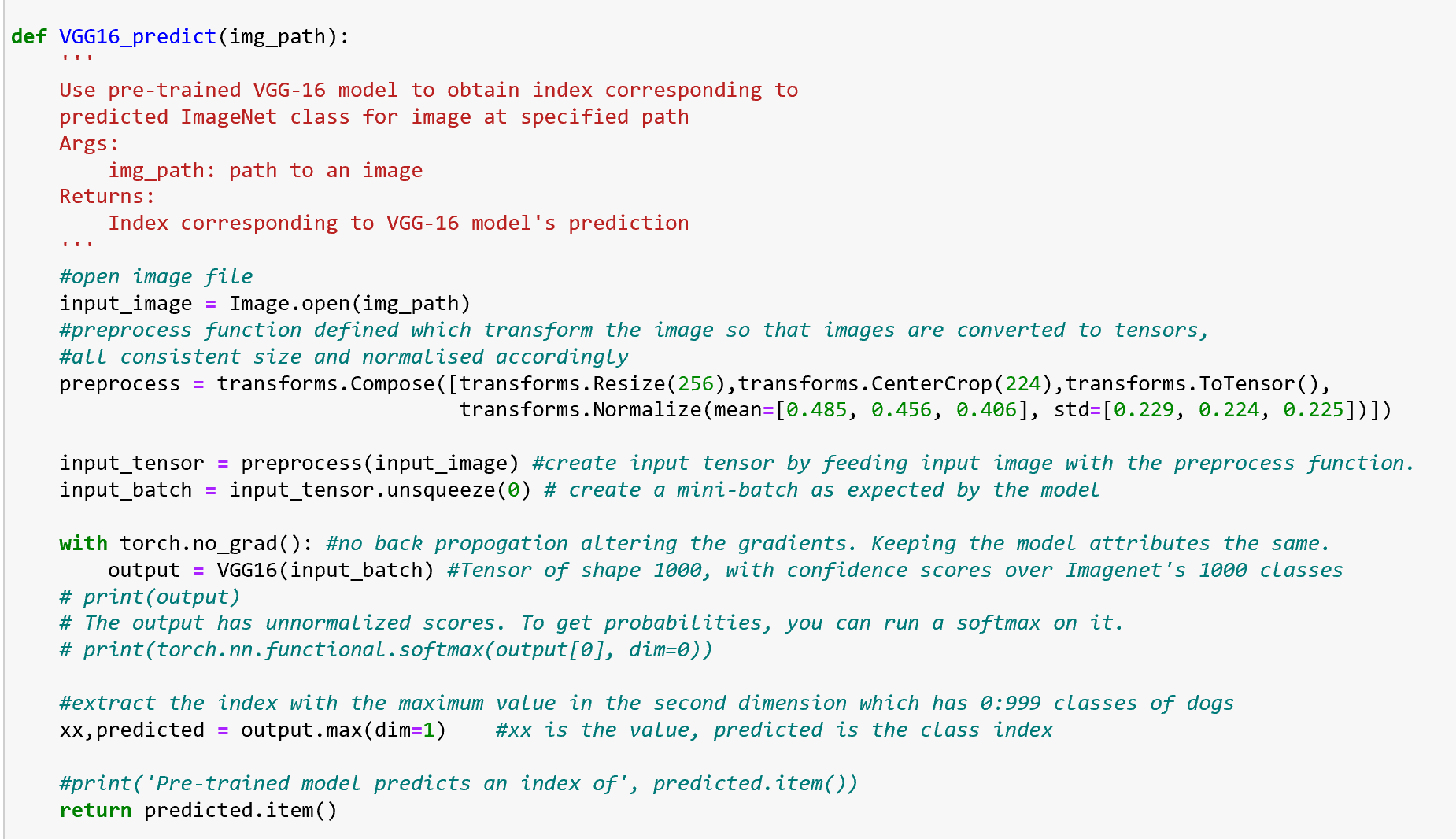
-initialize pre-trained face detector

* load image of humans imported
* convert image to grayscale
* test the human detector by loading human and dog images
* verify that the human detector can detect human faces accurately and cant detect dog in the image.

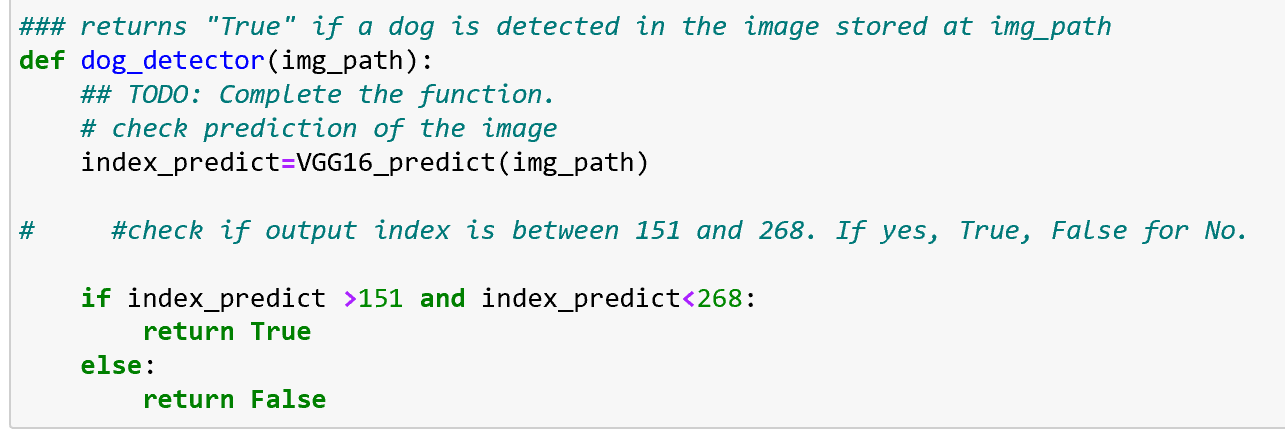
Step 2: Detect Dogs

We will use the pre-trained model VGG16 to detect dogs in images. This model has been pretrained for 1000 different classes through ImageNet.

* Define VGG16 model



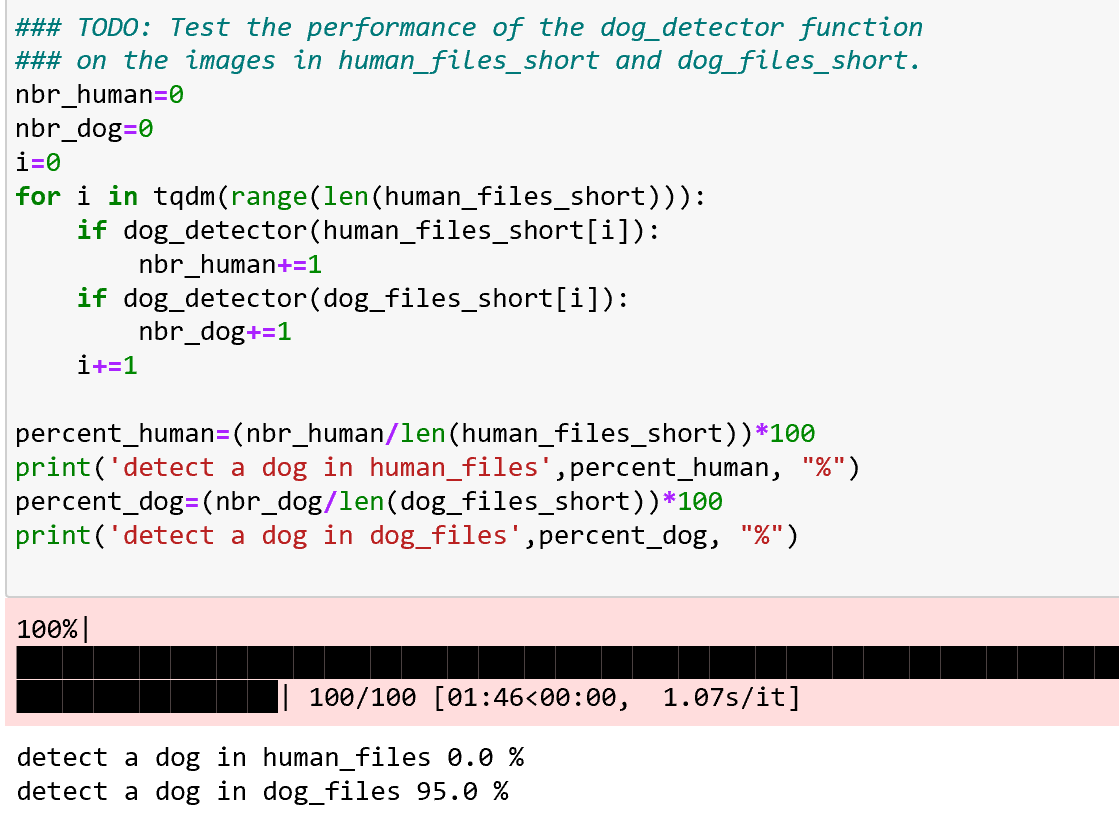
– Use GPU for better performance if available



– load and pre-process the image

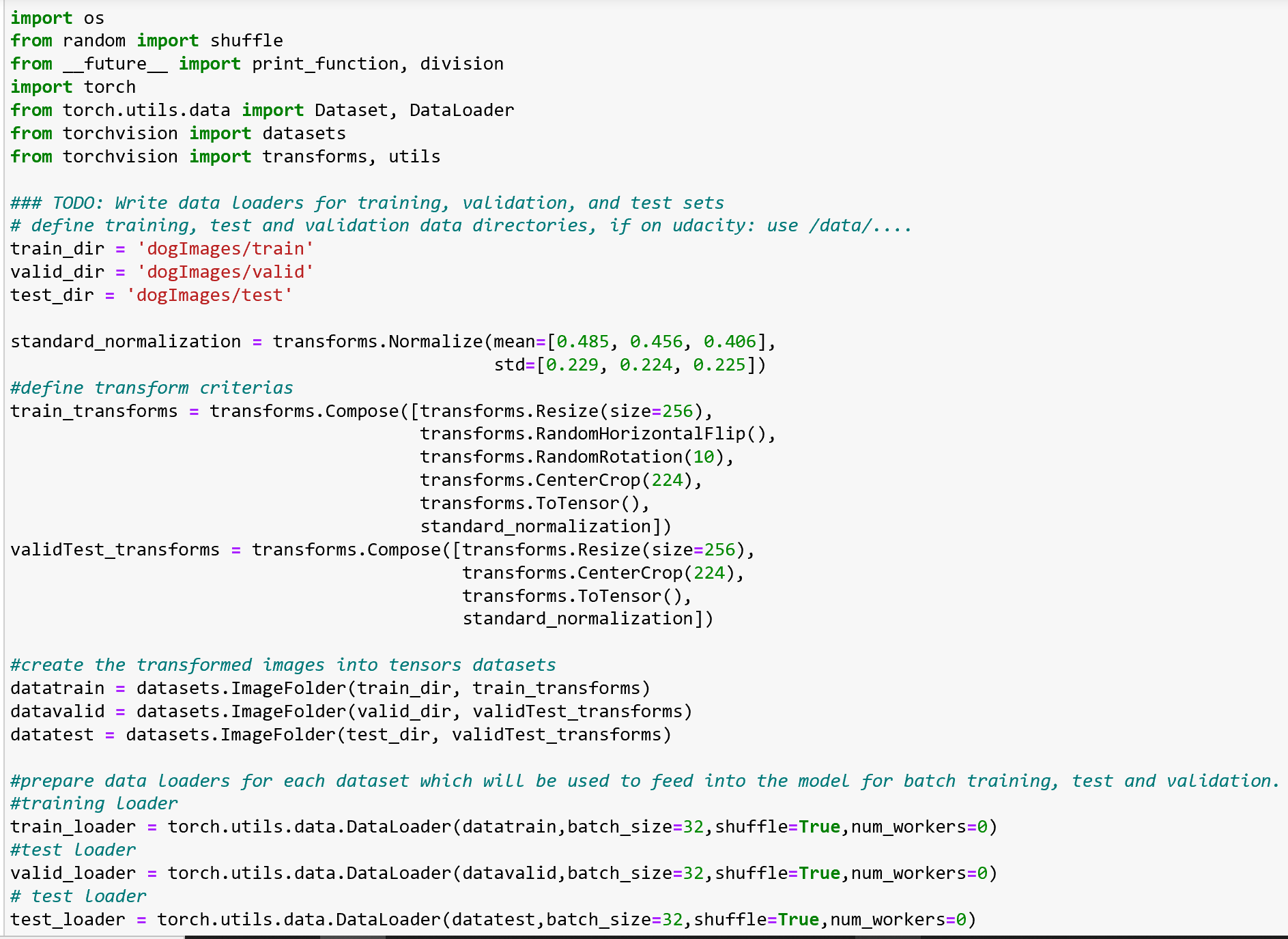
– send an image to the VGG16 model

– verify that the dog detector can detect dog in images accurately and cannot detect dog in the human images.

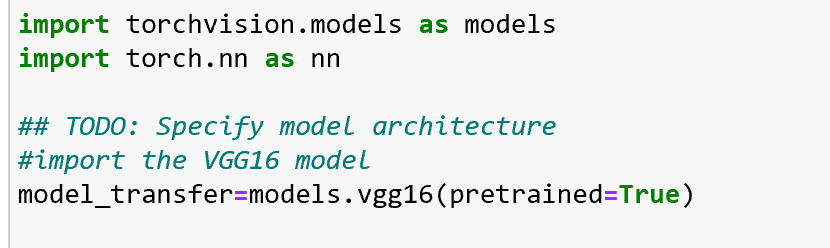
- Model return true if the index is 151 to 268 as the classes between 151 and 268 is for dogs. 

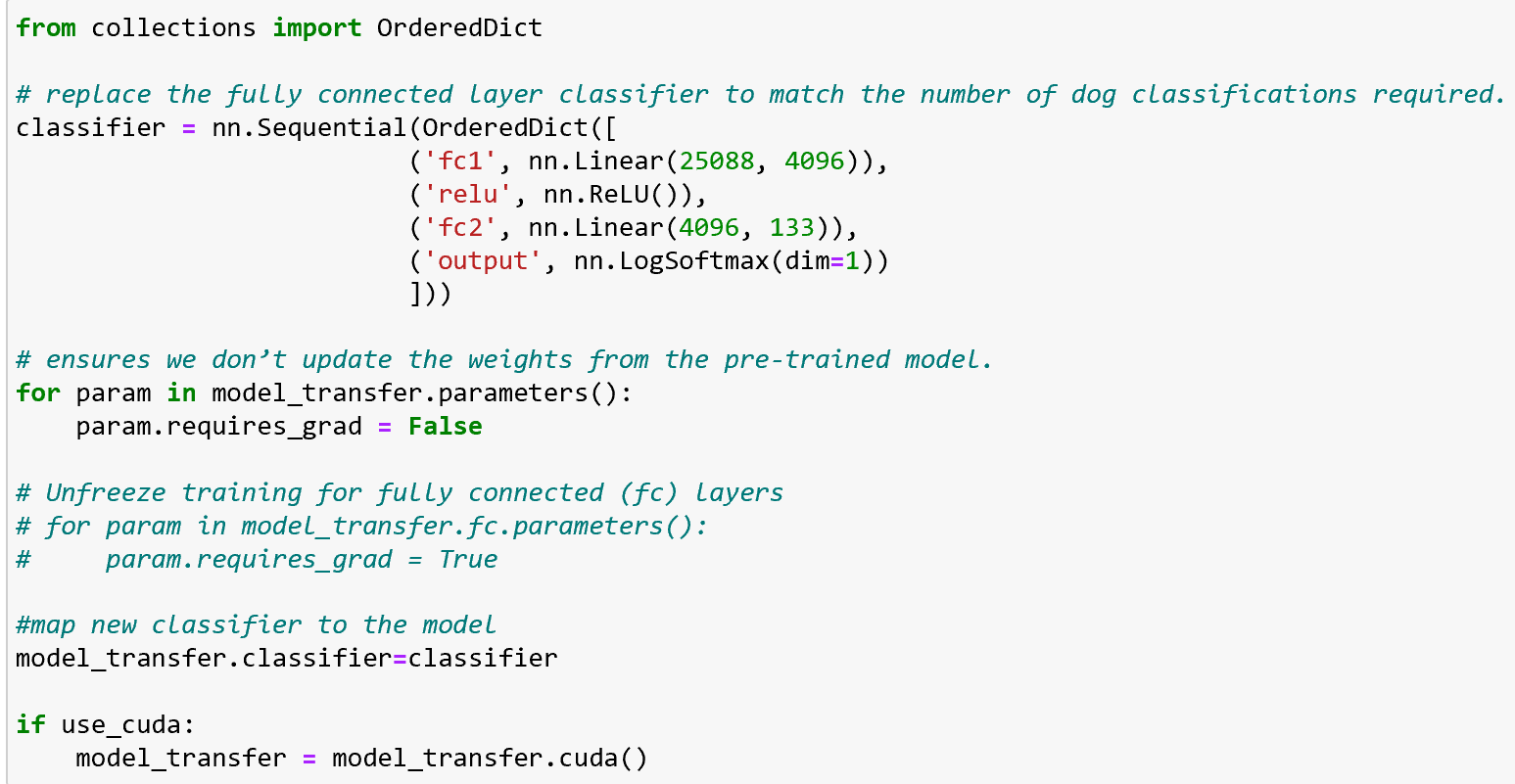
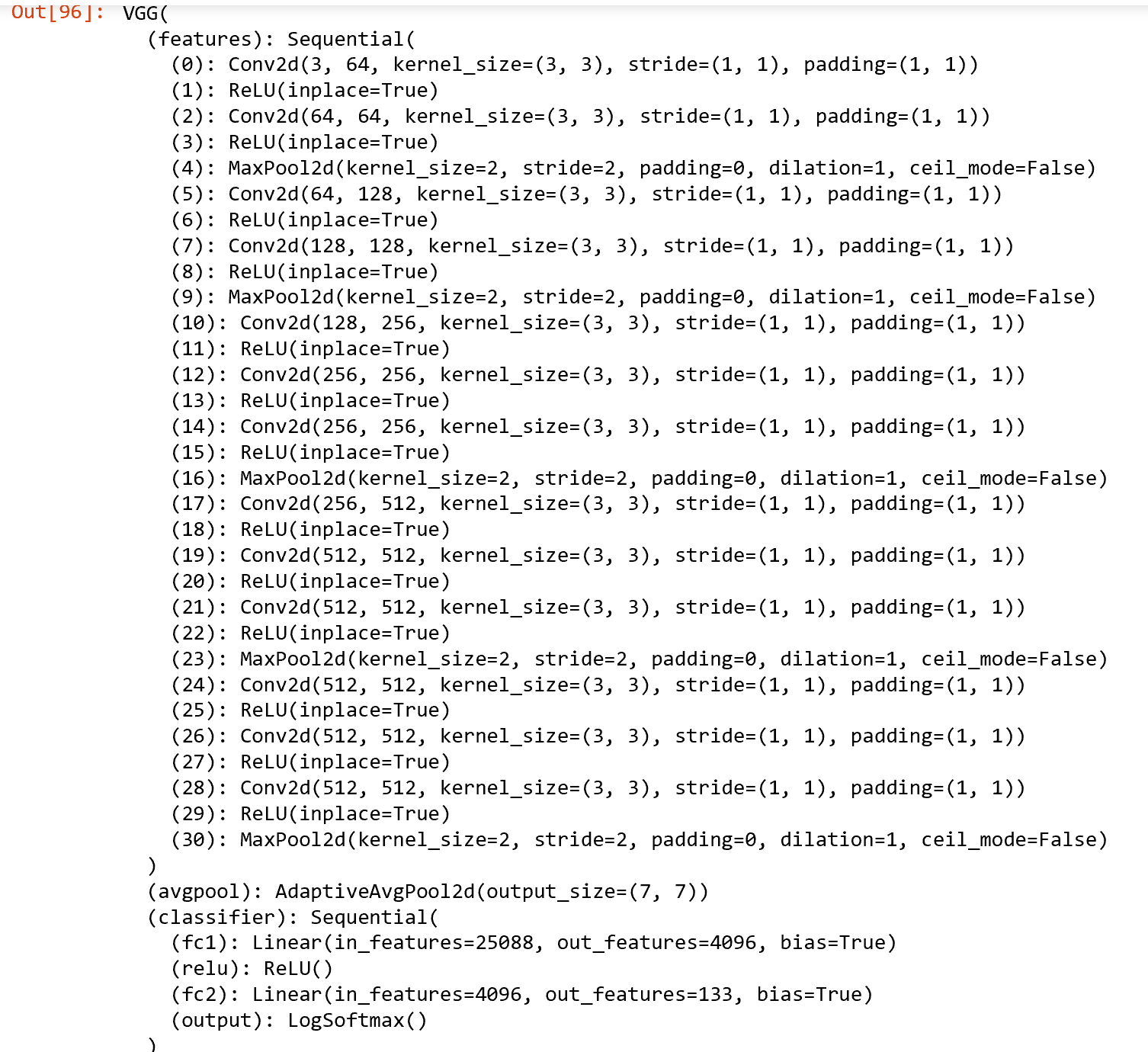
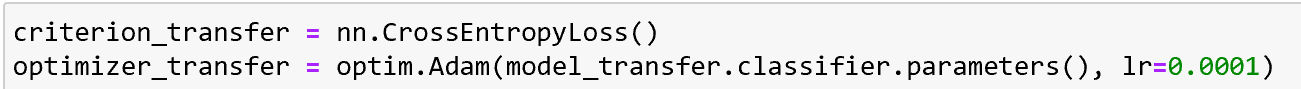
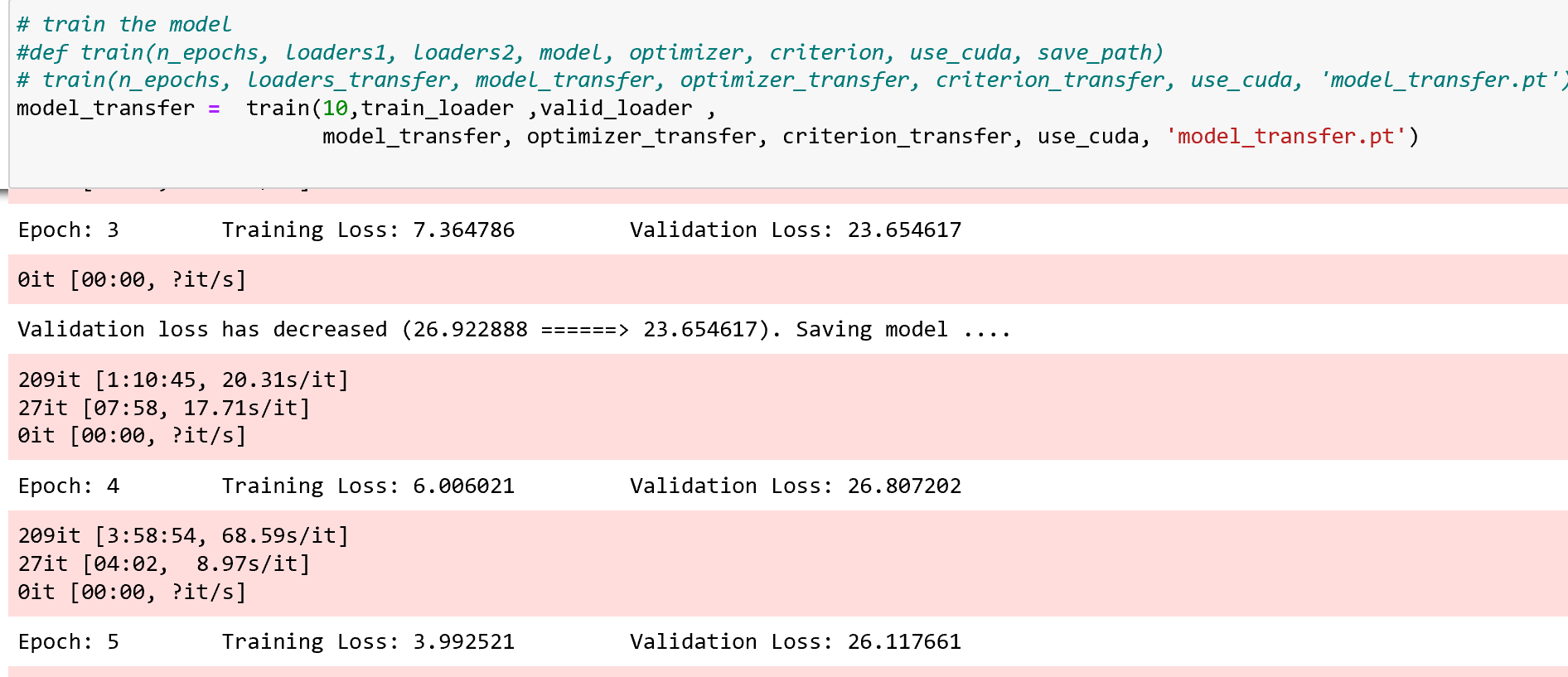
Step 3: Create a benchmark CNN model to classify dog breeds (from scratch)

The dog data is already divided into training, validation and test data. The data in images needs to be converted to batch data in the form of tensors for the model. Training and validation is used concurrently to verify that the validation loss of the model created is actually reducing and improving and not just training loss getting reduced. Once the model is sufficiently trained through a few epochs, it's accuracy is being evaluated using the test data. If it doesnt hit 10% accuracy, the model architecture and training parameters can be fine tuned. Once this is achieved, it can considered as the benchmark model.



Step 5: Create a CNN model using transfer learning using Resnet or VGG to clasify dog breeds. With transfer learning, we can maintain the model architect and just integrate it with the output layer classification we require. We train, validate and test the model just as in step 4. This time we expect the accuracy to be over 60%.



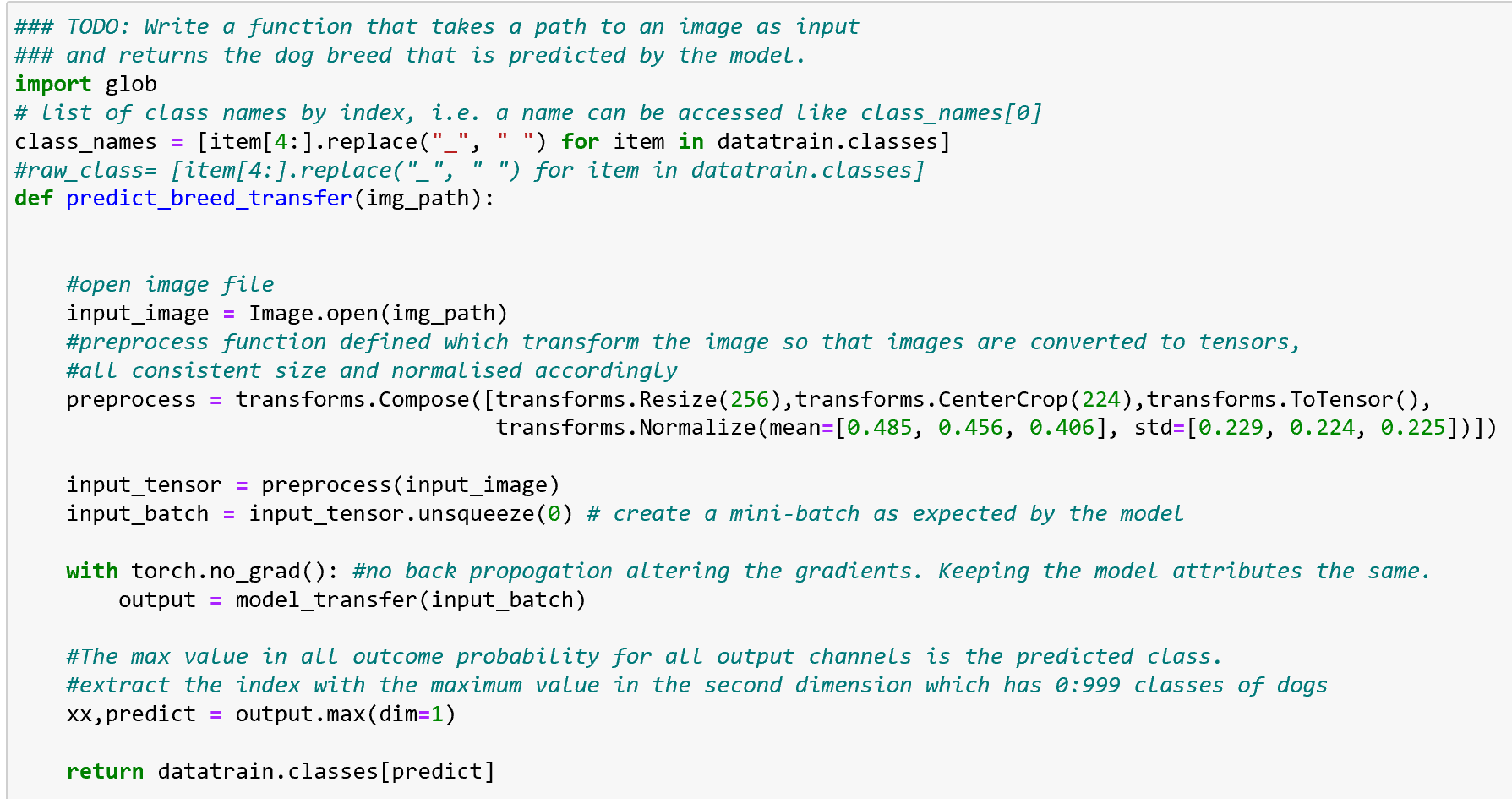
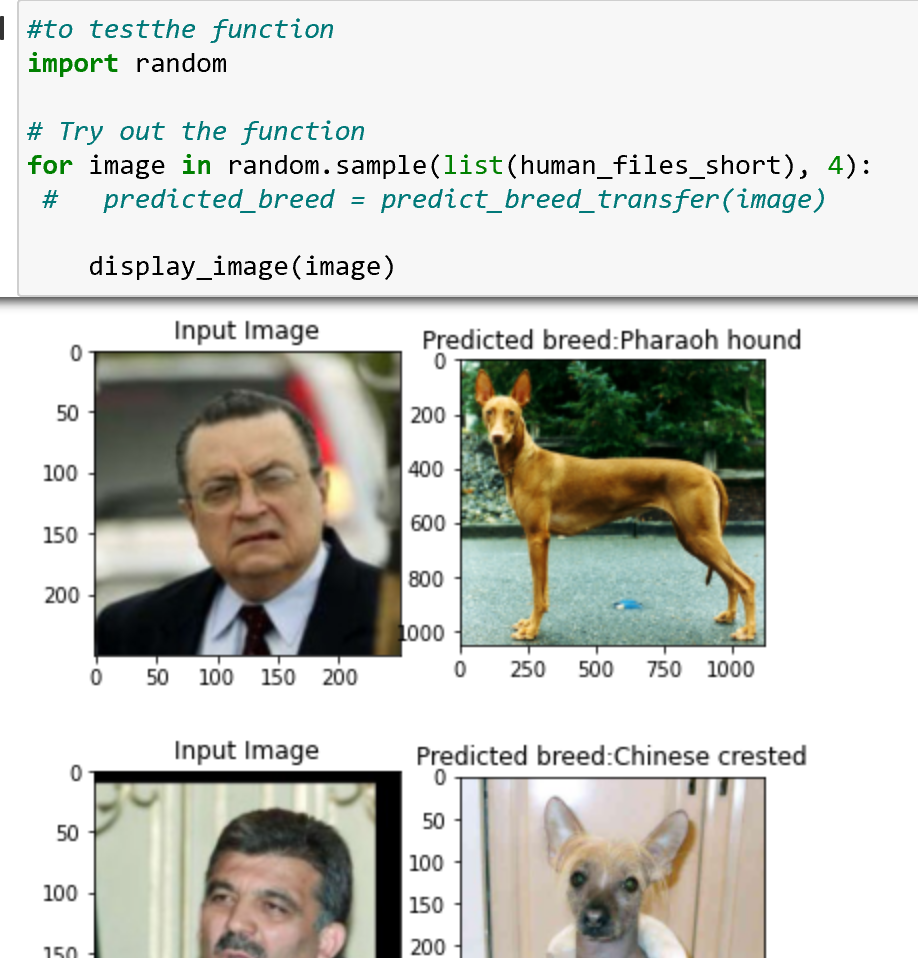
Step 6: Create and test the algorithm.

Create the algorithm by combining the human detector, dog detector and the transfer learning dog breed classifier model. The response the algorithm should produce are the folowing:

- If the image input contain a human, confirm that human is present and what dog breed the human resembles.

- If the image input contains a dog, confirm that a dog is present and what dog breed is the dog.

- If the input image contains neither, provide output which indicates error.

### Refinement

The process of improving upon the algorithms and techniques used is clearly documented. Both the initial and final solutions are reported, along with intermediate solutions, if necessary.

The benchmark model which was built from scratch was fine-tuned with various parameters such as the following:

* Learning rate variation between 0.01 to 0.003
* Number of convolution layers
* Number of input and output each layers
* Number of linear output.
* Varied the dropout layers.
* Varied the type between SGD or Adam.

## Results

### Model Evaluation and Validation

The final model’s qualities—such as parameters—are evaluated in detail. Some type of analysis is used to validate the robustness of the model’s solution.

To evaluate and validate the model, log loss for training and validation were evaluated. The model is trained as much as possible with the given parameters. The optimum model is generated based on the minimum validation log loss is achieved.

Once that has been performed, model accuracy is checked by running test data on it.

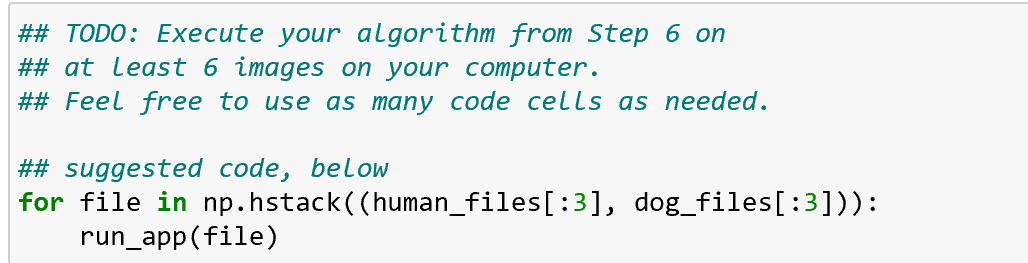
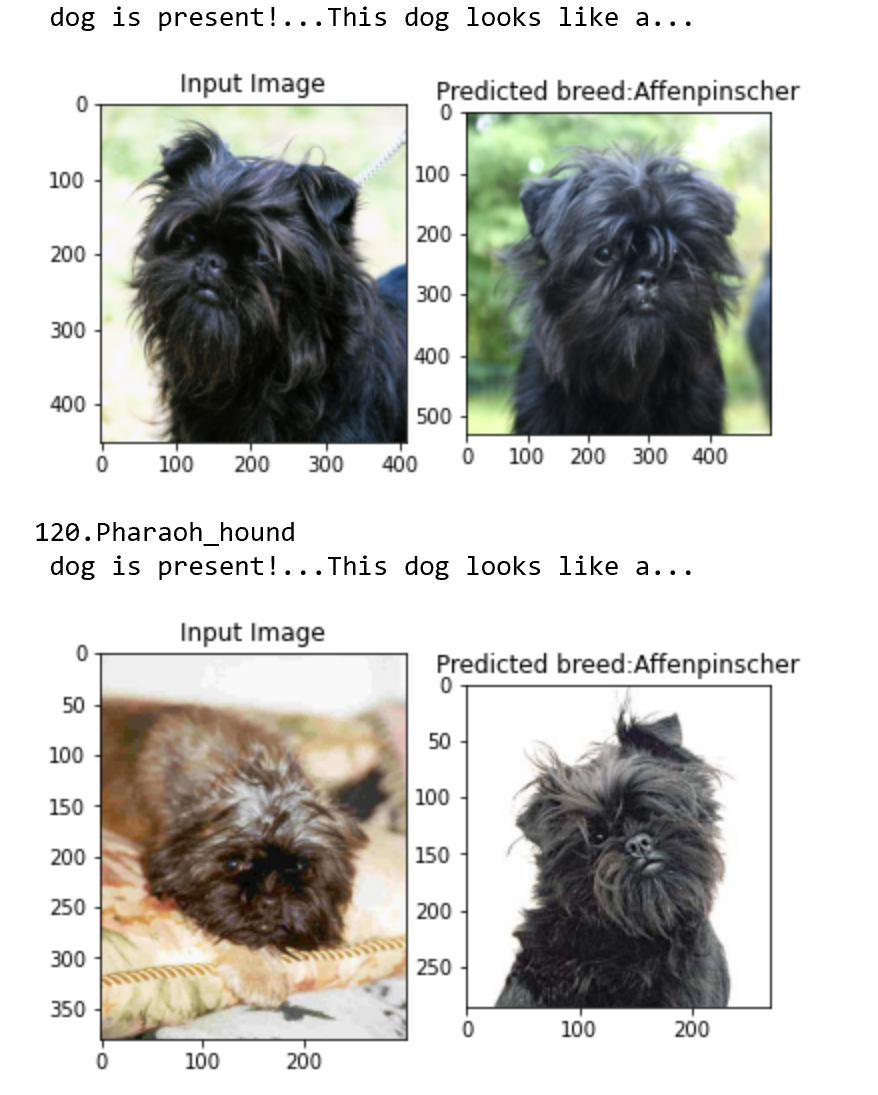
Through the training, validation and testing of the model, numerous parameters are varied and results are as below:

In your Model Evaluation and Validation section, please be sure to provide some discussion about the final parameters or characteristics of the model. How do these align with the characteristics of the dataset? Why would these be a robust solution to the problem?

* Dropouts were evaluated when developing the benchmark model. No dropout changes was made on the pretrained model of VGG16.
* Learning rates between 0.005 and 0.05. It was seen low learning rate takes many Epochs and a lot of time to achieve the lowest validation loss for the model. Final learning rate chosen was chosen to be xxx as it is able to achieve minimum validation loss around 12 epochs.
* Epoch chosen is based on the validation loss for the model. High epoch will overtrain the model to be overfitting causing accuracy of model to reduce in the test set. Final epoch will be a few epochs more than minimum epochs to achieve minimum validation loss at 15 epochs.
* Batch size chosen were 32. are also critical. Smaller batch size allows the model to be trained more times and faster compared to higher batch size. However the downside of batch size being too small is the gradient estimate for the training of the model will be less accurate.
* Optimizer used are Adam and Stochastic gradient descent. Adam able to generate model accuracy of xxx. Whereas using the SGD optimizer, the model accuracy was only able to achieve xx .
* Transfer learning was used on pretrained model of VGG16. Accuracy was xxx with Adam.

Final model's accuracy is at 78%. By attempting to provide personal pictures of dogs and humans as input, the model is able to predict the dog breed accurately. Thus, I believe the model is robust enough to predict similar data not in this dataset.

### Justification

Bench mark model only produced accuracy of 9% whereas the transfer learning model developed is able to be accurate up to 78%. The problem is solved as it can predict dog breed class with a high accuracy above 60%. It is also able to predict dog breed on data images not from the dataset input from personal computer.

Another approach that you could take would be to demonstrate that your tuned model is robust  
would be to perform a k-fold cross validation. In this case, you'd document how the model  
performs across each individual validation fold. If the validation performance is stable and  
doesn't fluctuate much, then you can argue that the model is robust against small perturbations  
in the training data.

References

<https://en.wikipedia.org/wiki/Yann_LeCun>