Final Report: Use Convolutional Neural Networks to Identify Dog Breeds

Udacity Capstone Project - Machine Learning Engineer Nanodegree

July 2019

1 Background

The total U.S. Pets industry has been growing steadily over the past decades (Fig. 1). In social media, it is commonly seen that pets owner sharing images, videos and texts about their pets. Pets are part of the numerous family members in the States. Many people are keen to learn every possible details about their pets to better take care of them and to better understanding the animals. One of the commonly seen questions that we can see from online users' posts or comments are: "Does anyone know the breed of my dog?", "Your dog looks so cute. What's his breed?", etc. Being able to automatically identify the dog will be a useful tool for pet owners and animal enthusiasts. In the meantime, people are also curious if they look like any specific breed of dog. Being able to provide a probability of animal resemblance will be a fun thing for people to play with in an app, web game, and so on.

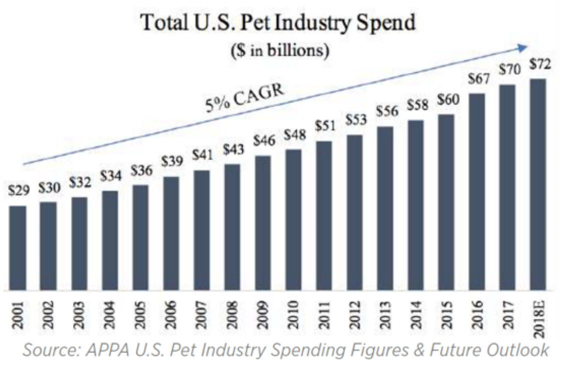

Figure 1: Plot of Total U.S. Pet Industry Spend 2001-2018e

2 Problem statement

Ideally, for animal classification, the customers upload an image and expect a classification result to be returned. Convolutional Neural Networks (CNN) are commonly used for image related machine learning problem. Here, we will work with dog breed classification problem specifically and will work on building a CNN model for this problem. Using CNN model to aid computer vision is a very interesting and promising direction and is also a trending technique in many classifications problem. For our specific problem here, dog breed classification, we want to build a pipeline that will eventually help process user-supplied images. Given an image of a dog, the algorithm will identify the breed of the dog. If there exists a human in the image, the code will return the resembling dog breed.

3 Data exploration

The datasets we will use are provided by Udacity with 133 dog breeds and is similar to Stanford Dogs Dataset with 120 dog breeds. There are 13233 total human images, which can help us train a model to identify the human image. And there are 8351 total dog images for dog classification. For human face images, they all have the same dimension: 250×250. For dog images, they have variable dimension: roughly around 400×400, which will be resized to 256×256. The distribution for the training set for each class is plotted below. As we can visualize clearly, the training dataset is unbalanced; the fewest has about 30 images while the largest has about 80 images.

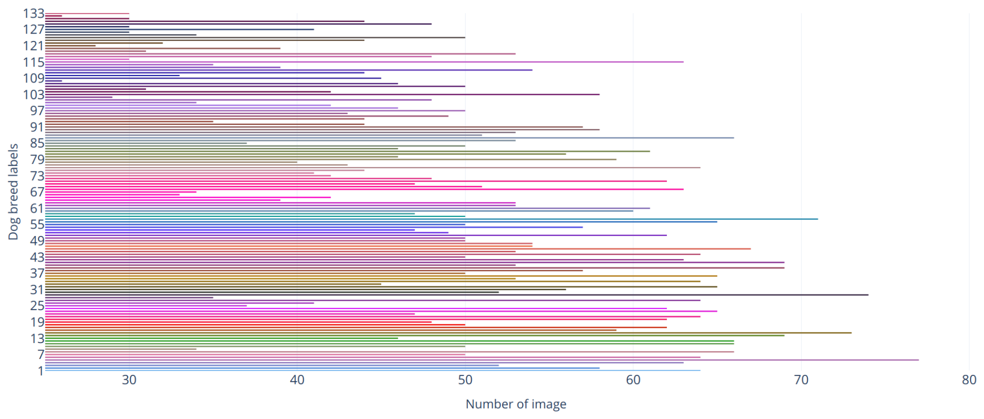

Figure 2: Distribution of training dataset

We also included two sample images from the dataset showing the images with dog only and with humans in Figure 3. Notice that classifying dog breed is not an easy task because even for humans, it is difficult to distinguish between a Brittany and a Welsh Springer Spaniel. Some breeds, for example, labradors, even have different colors.

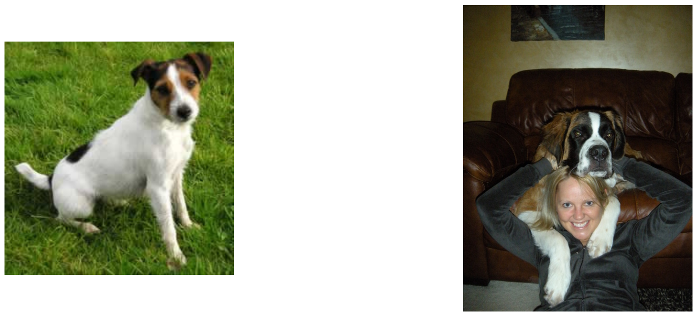

Figure 3: Images with dog and human

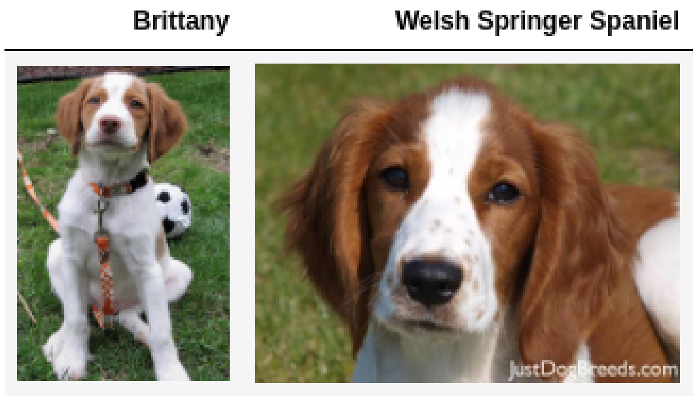

Figure 4: A Brittany and A Welsh Springer Spaniel

4 Implementation

We split our project into several steps below. We will first perform human face detection and dog detection to extract human and dog region of interest (ROI) within a large image. Then we will train a CNN model to classify the dog breed. Finally, we will explore the transfer learning model as an alternative approach for the same problem.

4.1 Human face detection

We use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. OpenCV provides many pre-trained face detectors, stored as XML files on github. We have downloaded one of these detectors and stored it in the haarcascades directory. For the first 100 human images and the first 100 dog images, we have found 98% of the human images have human faces and 17% of dog images that detect human faces.

4.2 Dog detection

For dog detection, we use the pre-trained VGG-16 model. It has 13 convolution + ReLu layers and 5 max pooling layers and 3 fully connected + ReLU layers (Fig. 4). The weights have been trained on ImageNet, 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. The output of the VGG-16 always be an integer between 0 and 999, inclusive. While looking at the dictionary, we notice that the categories corresponding to dogs appear in dictionary keys 151-268, inclusive, to include all categories. Thus, in order to check to see if an image is predicted to contain a dog by the pre-trained VGG-16 model, we need only check if the pre-trained model predicts an index between 151 and 268 (inclusive). To quick test the performance of the VGG-16 for the dog detection. We test the first 100 human images and the first 100 dog images, we have found 0.02% of the human images have dog and 100% of dog images that detect dog.

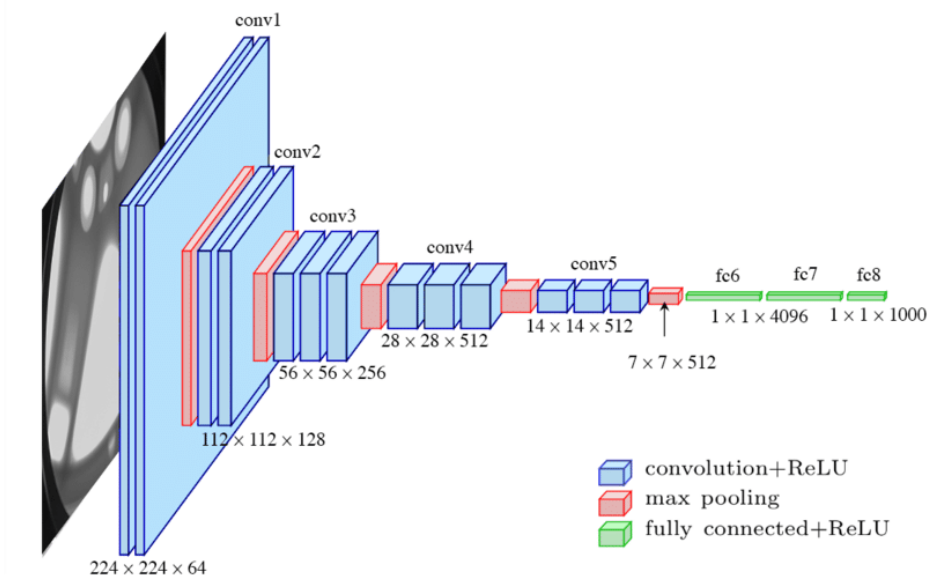

Figure 5: Architecture of VGG16

4.3 CNN model from scratch for dog breed classification

Now that we have functions for detecting humans and dogs in images, we need a way to predict breed from images. In this step, we will create a CNN that classifies dog breeds. We will create a CNN model for the dog breed classification. I will use several convolutional layers, each followed by a Max Pooling2D to reduce complexity of the stacked model. We will convert each feature map in the Max Pooling Layer into a scalar. The model will be based from scratch and will output the dog breed. We use Cross-Entropy as loss function and Adam optimizer. Since the training, testing, validation images have already been pre-split into folders. We will use these images directly from the folders to do validation. The architecture we constructed is displayed below.

```
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))
(conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(conv5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(dropout): Dropout(p=0.2)
(conv_bn1): BatchNorm2d(224, eps=3, momentum=0.1, affine=True, track_running_stats=True)
(conv_bn2):
BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(conv_bn3):
BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(conv_bn4):
BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(conv_bn5):
BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(conv_bn6):
BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(fc1): Linear(in_features=12544, out_features=512, bias=True)
(fc2): Linear(in_features=512, out_features=133, bias=True)
)
```

Note that:

- Five convolution layers are used all with the colvolution of size = 3, stride = 1 and padding = 1.
- Five convolutional layers are defined with 16/32/64/128/256 filters respectively, each followed by a max pooling layer.
- Also two connected linear layer at the end are used.
- Relu activations are used after each layers except the last one.
- Max pooling layers of 2*2 are applied.

- Batch normalized are applied after each max pooling.
- Dropout is applied with the probability of 0.25.

4.4 Create a CNN to Classify Dog Breeds (using Transfer Learning)

Instead of training a deep neural network from scratch, which would require a significant amount of data, power and time, it's often convenient to use a pre-trained model and just finetune its performance to simplify and speed up the process. Here, we transfer another commonly used CNN model in computer vision: Rsenet50 to improve feature engineering and transfer learning for the dog breed classification. We can then compare the performance of these two models. During the training stage, we used image rotation, horizontal flip and center crop etc. techniques to help augment the data.

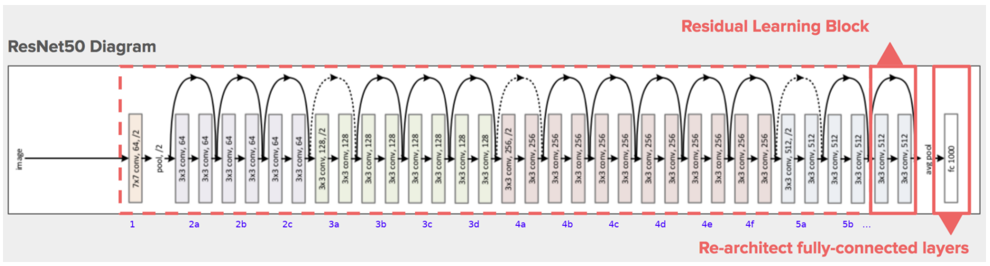

Figure 6: Architecture of ResNet50

5 Results

5.1 Benchmark Model Results

The CNN model we customized can be a benchmark model. During the training process, we saved the model only when the validation loss decreases. For the saved model, we calculated that the accuracy of training data was 72% (4816/6680) and training loss was 1.207671;the accuracy of validation data was 23% (196/835) and the validation loss was 3.324067; the accuracy of test data was 22% (184/836) and test loss was 3.229028.

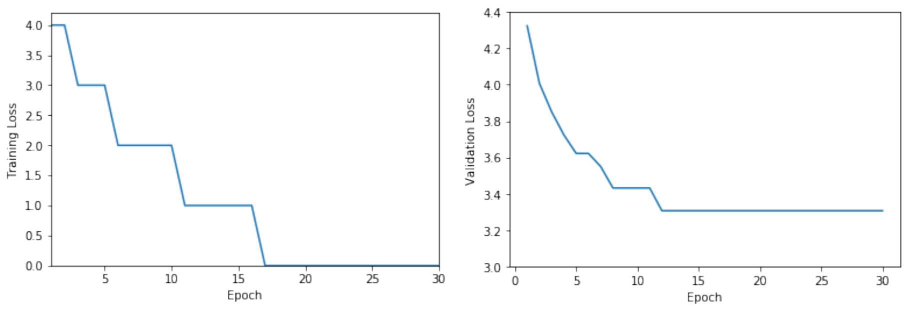

Figure 7: Loss of validation and training data over each epoch

The confusion matrix is shown below. Showing the

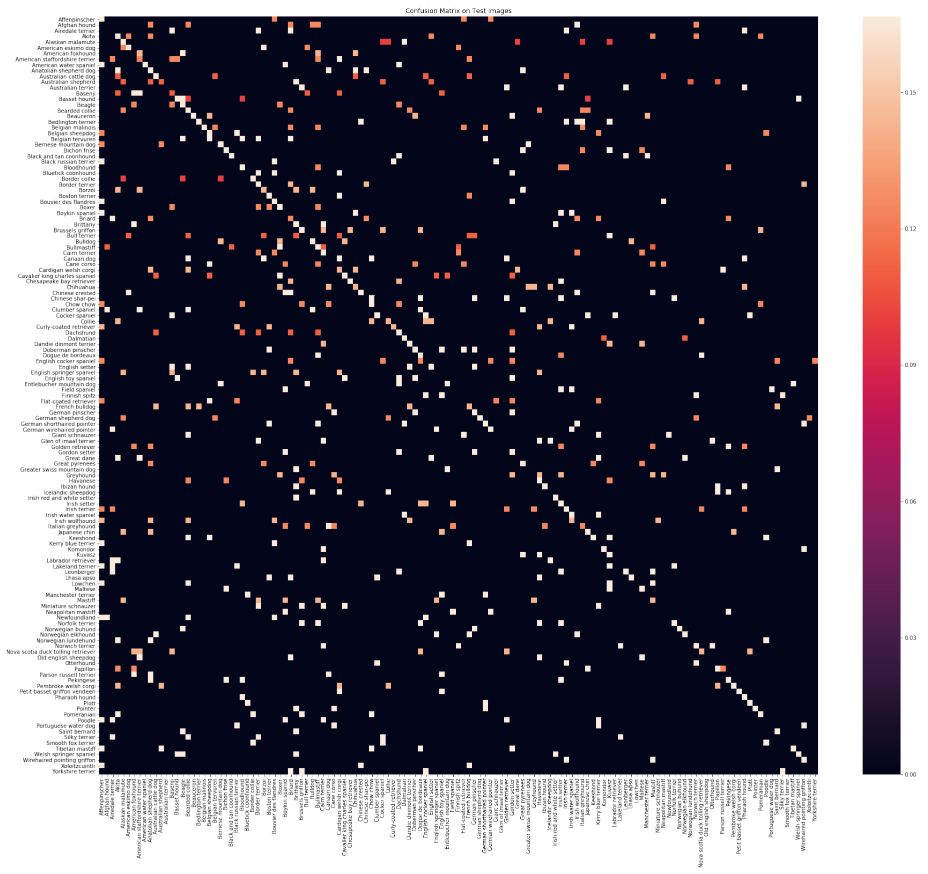

Figure 8: Confusion matrix for CNN model

From the F1 score plot of all tested data, we found that Dachshund has the highest F1 score, indicating the model works best with this breed.

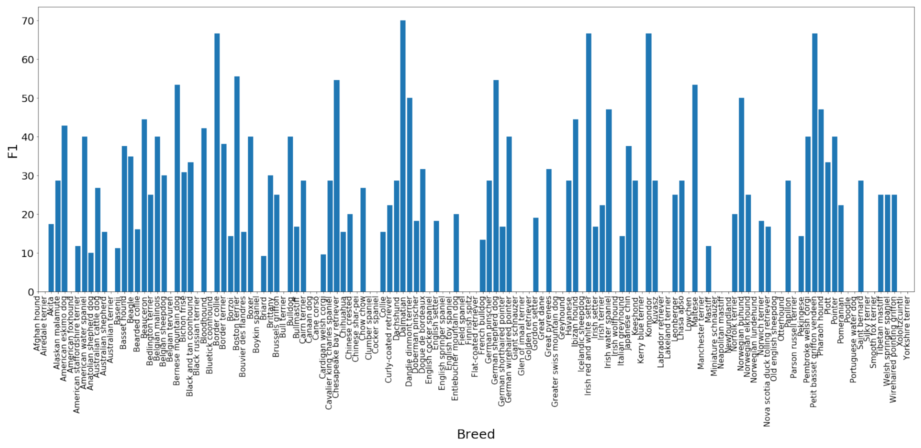

Figure 9: F1 score of all test classes

5.2 Transfer learning Model Results

During the training process, we saved the model only when the validation loss decreases. For the saved model, we calculated that the accuracy of training data was 88% (5894/6680) and training loss was 0.432195;the accuracy of validation data was 83% (696/835) and the validation loss was 0.590911; the accuracy of test data was 79% (662/836) and test loss was 0.641934. Below we show the loss during each epoch. We found that for transfer learning model, model has better fit as both training and validation loss decreases to a similiar level.

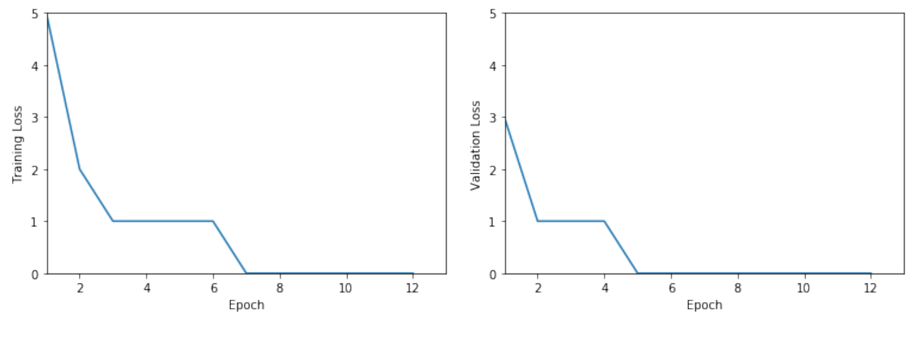

Figure 10: Loss of validation and training data over each epoch

Here, we also plotted the heat map showing the confusion matrix.

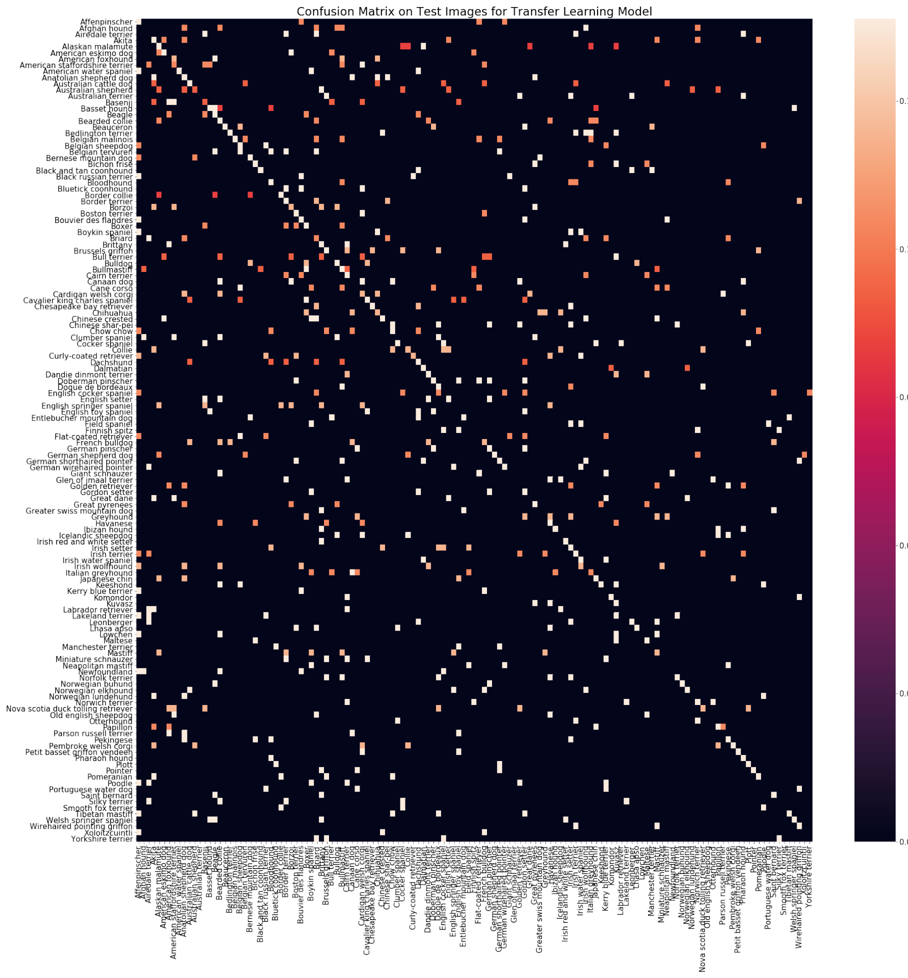

Figure 11: Confusion matrix for ResNet50 Trasnfer Learning model

We also show some results examples for the algorithm we developed that can classify dogs and humans into certain dog breed.

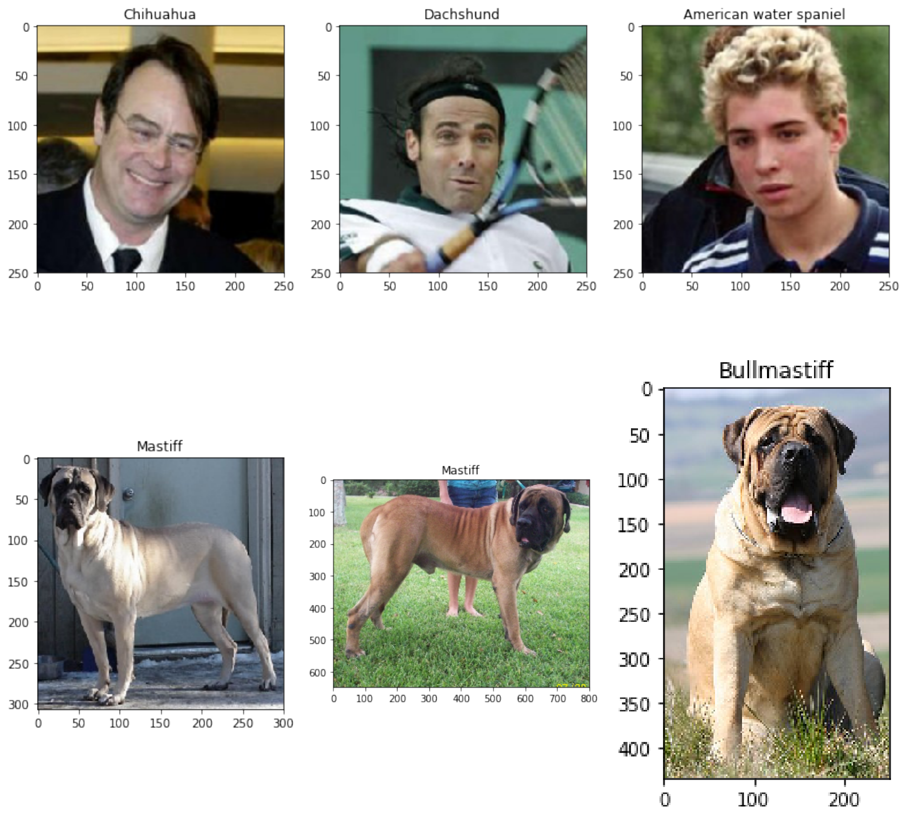

Figure 12: Results of example images with estimated dog breed on the top

From the F1 score plot of all tested data, we found that a lot more breeds of dog have the highest F1 score, indicating the transfer learning model is more robust for different dog breeds.

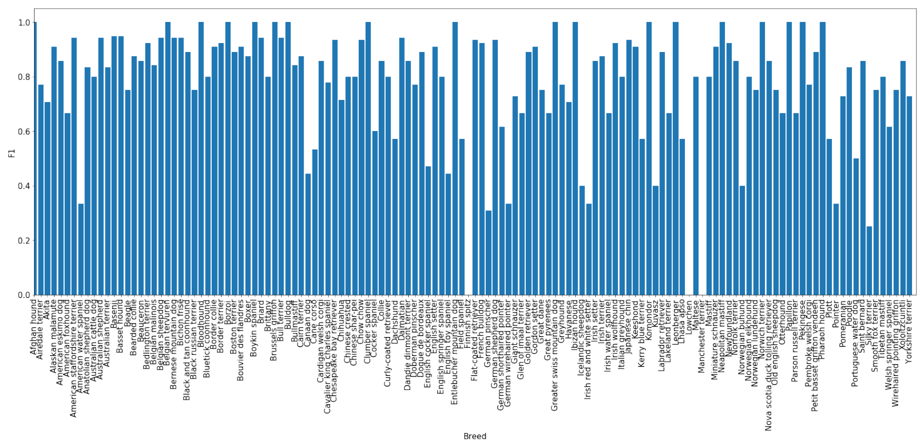

Figure 13: F1 score of all test classes

6 Conclusion

For this project, we applied OpenCV's implementation of Haar feature-based cascade classifiers to detecting human faces and VGG16 to dog detection. We constructed a customized CNN model to train the dog images for dog breed classification and achieved an accuracy of 23%. Finally, We applied Rsenet50 and transfer learning to compare the performance of the customized CNN model and the transfer learning results and the accuracy was increased to 82%. This means for the classification problem here, more complex deep CNN can outperform the simpler CNN. Future improve will involve more training dataset of different breeds of dog images to overcome overfitting; this include increasing the number of dog images collected for training and also augmenting the existed with more complex processing methods. For the training process, we can also induce auto hyper-parameter tuning for tuning weight initializings, learning rates, drop-outs, and batch$_{s}izes.Wemayalsoconsiderensemblesofmodels.$

Reference

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).