MACHINE LEARNING ENGINEER

NANODEGREE

CAPSTONE PROJECT REPORT

DOG BREED CLASSFIER

Jiawei Sun

1. Definition

1.1 Project Overview

With the rise of deep learning techniques, many problems in computer vision can have improved solutions with higher accuracy. Dog image classification is a classical problem and is defined as predicting the dog breed for a user-supplied image. Each breed of dog may have its own special traits, which help define the particular breed. Developing a dog breed classifier enables in-experienced dog owners to identify their dog's breed, so that they can better understand dog behavior and be aware of the potential genetic problems.

In this project, a convolutional neural network (CNN) is built to classify dog breeds. The framework of a convolutional neural network includes an input layer, an output layer and several hidden layers. The hidden layers of a CNN consist of convolutional layers, pooling layers, fully connected layers and normalization layers. The model in this project is trained on a dataset provided by Udacity, which includes 8351 dog images across 133 breeds and 13233 human images.

1.2 Problem Statement

The goal of the project is to develop a deep learning algorithm that can be used to distinguish dog breed for a given image. For a user-supplied image, the classifier should first identify whether the face in the image is a dog or not. If it's an image of a dog, the algorithm will identify an estimate of the canine's breed. If a human is detected in the image, the algorithm should identify a dog breed that the human resembles. If neither a dog or a human face is detected, an error message is returned as the output. With the use of transfer learning, the classifier should perform an accuracy higher than 60%, which is set by Udacity.

1.3 Metrics

We use "accuracy" to compare the performance of my dog classifying model with that of the benchmark model. Accuracy is defined as the number of items correctly classified divided by the number of all classified items. In the equation, TP, FN, FP and TN represent the number of true positives, false negatives, false positives and true negatives, respectively.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

We use Cross Entropy Loss as the loss function to optimize the machine learning algorithm. The loss is calculated on training and validation datasets, and it measures the model performance in these two data sets. Cross-entropy is a better measure than MSE for this project, because the decision boundary in a classification task is large, while MSE doesn't punish misclassifications enough.

There's a data imbalance in the dataset, which will be explained in the analysis section. For this reason, I will use F1 score as an additional performance measure.

2. Analysis

2.1 Dog Images

Udacity provides 8351 dog images. Among them, 6680 images (80%) are used for training, 835 images (10%) for validation and 836 images (10%) for testing. Training set contains 133 unique dog breeds. On average there are 50 images in each class. Within a dog breed class, the maximum image count is 77 and the minimum count is 26. Figure 1 shows a couple samples of images in dog dataset. Figure 2 illustrates the imbalanced distribution of images across dog breeds in the training dataset, in order words, the number of images in each dog class is different. Test and valid datasets also show such an imbalanced dog class distribution.

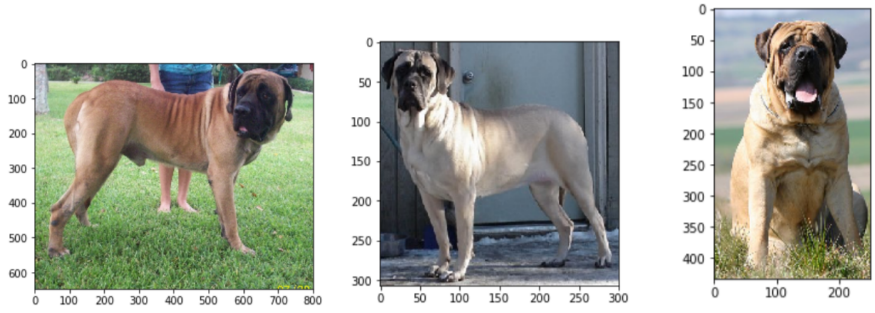


Figure 1: samples of images in dog dataset

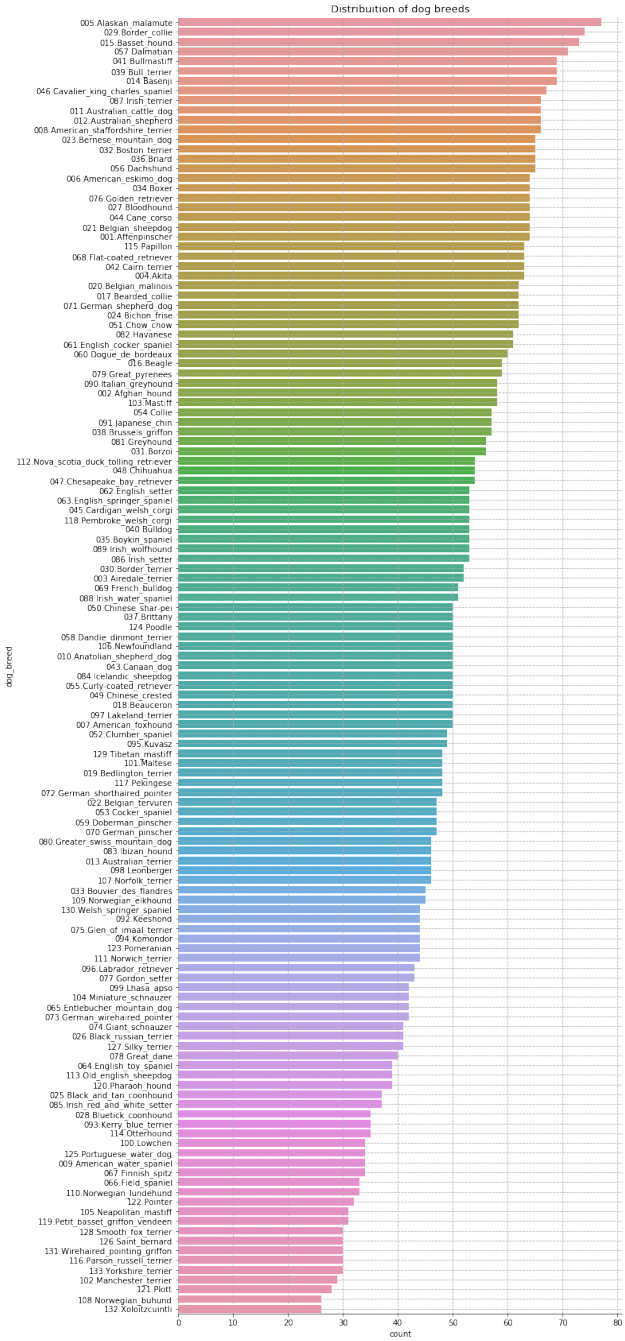


Figure 2: distribution of dog breeds

2.2 Human Images

There are 13,233 human images in total. Sample images in human dataset have been shown in Figure 3. Human set contains 5749 distinct human names. The average number of images in each class is 2.3 (the maximum image count within a class is 530 and the minimum count is 1). According to the histogram in Figure 4, majority of people have less than 9 images, and the

distribution of people with different image count is right-skewed. For example, George W Bush is an outlier and has 530 images.

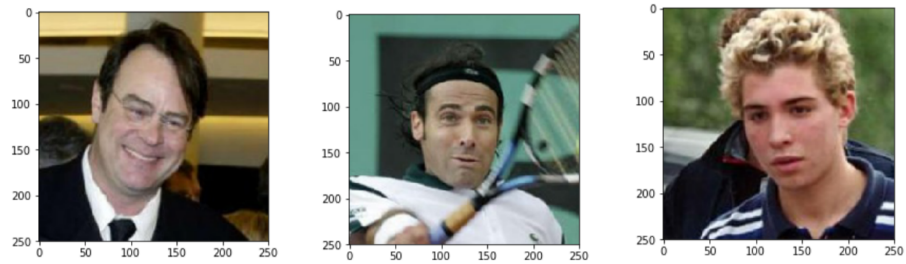


Figure 3: samples of images in human dataset

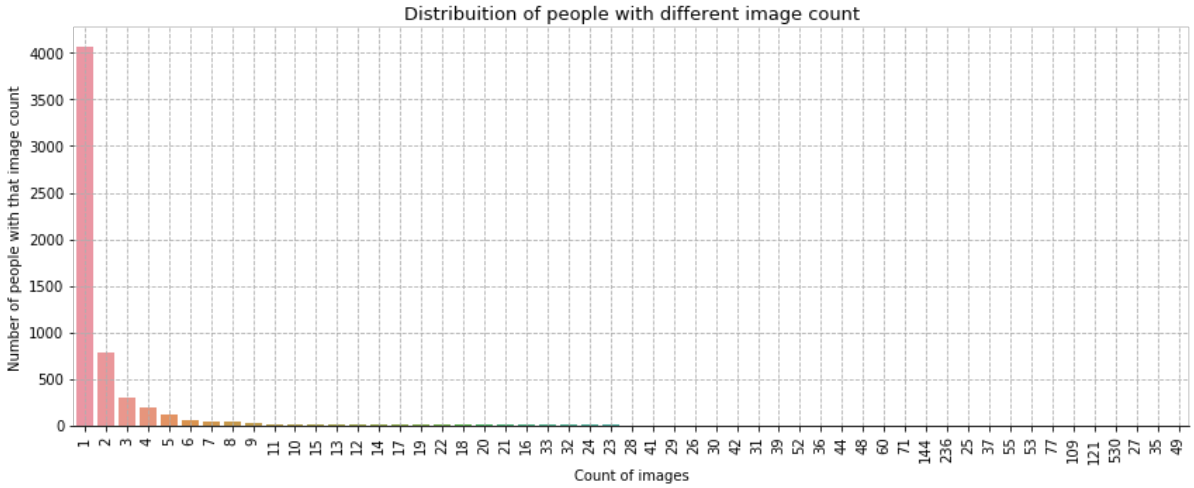


Figure 4: distribution of people with different image count

2.3 Algorithms and Techniques

Convolutional Neural Network is applied to solve this multiclass classification problem. The solution is divided into three sections, detecting human images, detecting dog images, and predicting dog breeds.

- To detect human faces, we utilize OpenCV's implementation of Haar feature-based cascade classifiers. OpenCV is a popular library for computer vision, which uses machine learning algorithms to search for faces within a picture [1].
- To detect dogs, we use a pretrained VGG16 model as well as the pre-trained weights of the model. VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman, and achieves a top-5 test accuracy of 92.7% in ImageNet [2].
- The goal of the project is to predict the dog breed based on a user-supplied image. In the prediction step, we use the model ResNet 50. We select Cross Entropy Loss as the loss function, because Cross-entropy is a better measure than MDE for classification. We use SGD as the optimizer because of the high cost of running back propagation over the full training set.

2.4 Benchmark

A random guess across 133 classes of dogs provides an expected accuracy of less than 1%. The CNN model created from scratch is required to have an accuracy of at least 10%. The pre-trained model with the usage of transfer learning must have at least an accuracy of 60%, which indicates that more than half of the dogs should be classified correctly.

3. Methodology

3.1 Data Preprocessing

We have one set of dog images and one set of human images as the input to the face detection and dog breed prediction models. When we use a pre-trained classifier of OpenCV to detect human faces, we don't preprocess the images for training purpose, but we want to use the images to test the performance in identifying human faces. For this reason, before using the pre-trained face detector, we need to convert the images to grayscale. When we develop the dog face detector and dog breed classifier, we need to preprocess the images before inputting them to train the models. I applied Random Resized Crop to stretch or shrink the image to 224x244 pixels, because this is the default input size for model VGG 16. I used Random Resized Crop & Random Horizontal Flip on training data for the purpose of image augmentations and resizing jobs. Image augmentation will give randomness to the dataset so that it prevents overfitting. In the train, validation and test datasets, I followed the dataset preparation approach used for VGG16, normalizing the tensor with a mean of (0.485, 0.456, 0.406) and a standard deviation of (0.229, 0.224, 0.225).

3.2 Implementation

In this section, I will explain in details about algorithm and techniques that I have used in solving dog breed classification problem. The implementation can be divided into four sections, human face detector, dog detector, dog breed classifier from scratch, and dog breed classifier with transfer learning.

3.2.1 Human face detector

We use OpenCV's implementation of Haar feature-based cascade classifier to detect human faces. We download and extract one of the pre-trained face detectors provided by OpenCV.

Then, the detectMultiScale function digests the grayscale image as an input parameter, executes the classifier and returns with a NumPy array of detected faces. After that, we define a function called face_detector to return True if the number of humans detected is greater than 0, and False otherwise.

3.2.2 Dog detector

We use a pre-trained VGG-16 model to detect dogs in images. First, we download the VGG-16 model, along with weights that have been trained on ImageNet, a large and popular dataset used for image classification tasks. Then, we create a function that accepts an image path and returns the index corresponding to the ImageNet class predicted by the pre-trained model. The prediction is derived from 1000 possible categories in ImageNet, so the predicted output is an integer ranging from 0 to 999. According to the dictionary, if the predicted index is between 151 and 268, inclusive, this image is predicted to contain a dog.

3.2.3 Dog breed classifier from scratch

In this step, we create a CNN that classifies dog breeds from scratch and attain a test accuracy of at least 10%. Before implementing the model architecture, we need to load image data for training, validation and test and specify normalization and transforms. The architecture contains 3 convolutional and pooling layers, and 1 max pooling layer.

Steps:

1) The first convolutional layer receives an input image with a 3x244x244 tensor. The input image passes through a convolutional layer with a filter_size=2, stride_size=2, padding=0 and produces a 112x112x16 tensor (16 filters). Pooling is applied with a kernal_size=2 and stride=2 to produce a 56x56x16 tensor.

2) The second convolutional layer receives a 56x56x16 tensor, this passes through a convolutional layer with a filter_size=2, stride_size=2, padding=0 and produces a 28x28x32 tensor (32 filters). Pooling is a applied with a kernal_size=2 and stride=2 to produce a 14x14x32 tensor.

3) The third convolutional layer receives a 14x14x32 tensor, this passes through a convolutional layer with a filter_size=2 and padding=1 and produces a 14x14x64 tensor (64 filters).Pooling is a applied with a kernal_size=2 and stride=1 to produce a 7x7x64 tensor.

4) Flatten image input to fit the 12877 first fully connected layer.

5) Add a dropout layer to increase the generalization ability of the architecture and to prevent overfitting.

6) Add 1st hidden layer, a fully connected (linear) layer with a 128 7 7 input tensor and a 500 output tensor.

7) Add another dropout layer.

8) Add 2nd hidden layer, the fully connected layer with a 500 input tensor and a 113 output tensor. In this step, the number of outputs is reduced to the number of classes (113 classes of dog breeds).

3.2.4 Dog breed classifier with transfer learning

We use transfer learning to create a CNN that attains greatly enhanced accuracy. Resnet50 is a pre-trained Deep learning model. To find all the unknown parameters would require lots of data (in millions). It is very difficult to get such large labelled dataset. we use pre-trained models as a starting point for our training process, instead of training the own model from scratch. VGG16, ResNet50 and SE-ResNet50 all give very high accuracy, precision and recall values. Of these, ResNet50 achieves the highest precision and recall and also takes much lesser epochs to train [3]. I want to freeze the feature part of the model and only train the classifier part, so I set the requires_grad falg to false for the architecture, and then I added a new classifying layer. After that I updated the requires_grad flag to true for the classifier and cab therefore be trained.

3.3 Refinement

The CNN model created from scratch reaches an accuracy of 13%. Although it's higher than the accuracy of 10% required by Udacity, the model performance can still be significantly improved by applying transfer learning. The CNN model with transfer learning performs an accuracy of 72% with 20 epochs, which also meets our benchmark expectations.

4. Results

4.1 Human Detector

The human face detector function was developed by using OpenCV's implementation of Haar feature based cascade classifier. The model detects 98 human faces from first 100 human images and detects 17 human faces from first 100 dog images.

4.2 Dog Detector

The dog detector function was created with a pre-trained VGG-16 model. The model detects 0 dog face from first 100 human images and detects 94 dog faces from first 100 dog images.

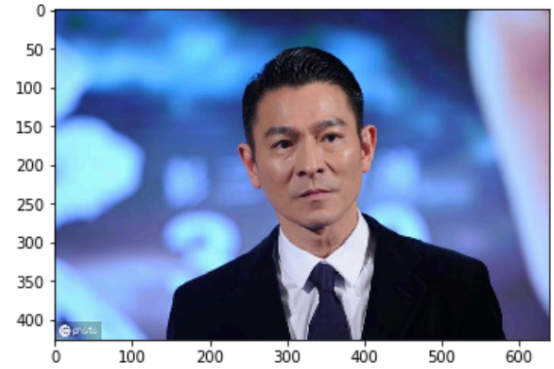
4.3 CNN with Transfer Learning

The CNN model was created with the transfer learning from ResNet50 architecture, and trained for 20 epochs. The accuracy on test image sets is 72%, which means it correctly predicts the dog breeds of 603 images out of 836 images. Due to the imbalance in the distribution of dog breeds in the image datasets, we also evaluate the model performance based on F1 score. It achieves F1 score of 0.68, which is much higher than the F1 score of 0.11 in the CNN model built from scratch.

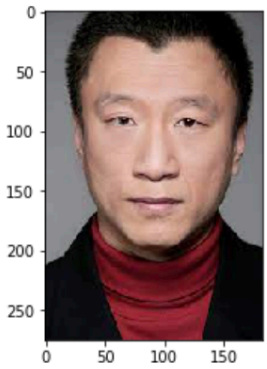
5. Conclusions

5.1 Free-Form Visualization

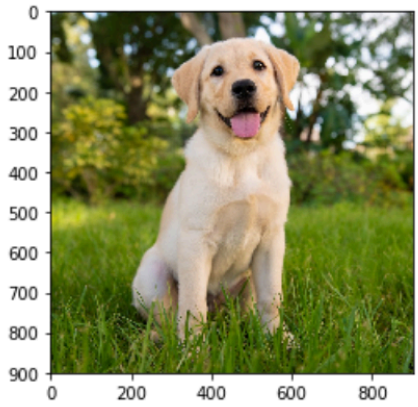
In this section, I would like to show some sample outputs of our final model.



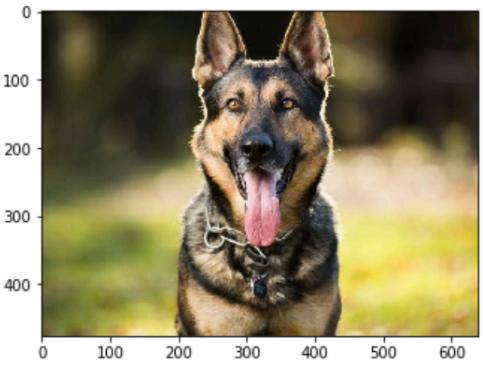

```
Hello, human!
If you were a dog..You may look like a Basenji
```



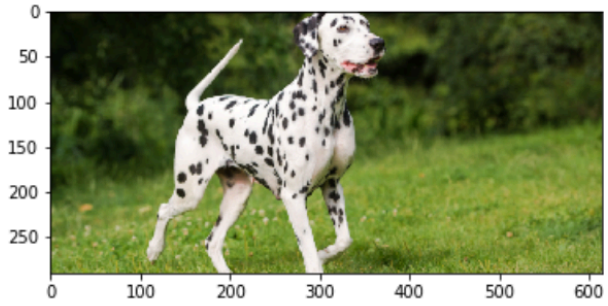
```
Hello, human!
If you were a dog..You may look like a Chinese crested
```



```
Dogs Detected!
It looks like a Golden retriever
```



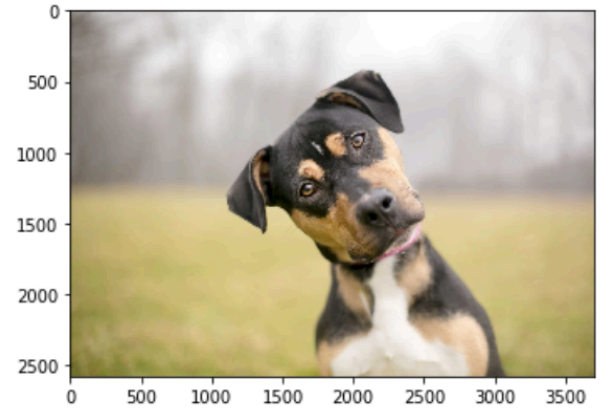
```
Dogs Detected!
It looks like a German shepherd dog
```



```
Dogs Detected!
It looks like a Dalmatian
```

5.2 Improvement

It's possible that the model is not able to tell a dog breed when there is a dog face in the image. For example, the image below returns with an error.



```
Error with predictions.
```

To further improve the model performance, there are two facets that need to be considered. The dog dataset only contains 133 dog breeds, and this number could be increased to a greater value to cover all the dog breeds. Furthermore, trying more architectures for feature extraction or performing more image augmentation can help us avoid overfitting and improve the accuracy.

Reference

[1] https://realpython.com/face-recognition-with-python/

[2] https://medium.com/@shikharsrivastava_14544/face-recognition-using-transfer-learning-with-vgg16-3caeca4a916e

[3] https://link.springer.com/article/10.1007/s42979-020-0114-9