Udacity Machine Learning Nanodegree Capstone

Use Convolutional Neural Networks to Identify Dog Breeds

Hao Cui

1. Definition

1.1 Project Overview

Computer vision (CV for short) is one of the promising techniques in the machine learning world. The basic idea of the computer vision is to let the computer to see and understand the digital images as human beings do. These digital images will be processed and transformed into digitalized data that the computers could make sense of them by employing machine or deep learning algorithms to train the data. The computer vision techniques could be mostly used to address three types of problems: image classification, object detection, and neural style transfer. With the aid of the deep learning algorithms and exponentially increased digitalized images, the computers are now "smarter" enough to address most of the complex issues, such as classifying different images and detect an object with high accuracy. Given the enormous amount of images of real-world objects, there is a considerable opportunity to explore how computer vision could be applied in identifying and classifying the objects based on the images provided. Convolutional neural net(CNN) is one of the most frequently used algorithms in the computer vision realm to classify the images. There are several variants of the algorithm, depending on the CNN architecture. In this study, I will explore algorithms such as VGG16[1] and ResNet[2] to see how these algorithms could help to classify the images.

1.2 Problem statement

In this project, a machine learning method for image classification will be used by extracting the features from the trained images as the inputs for the model artifact for our classifier. Specifically, I will solve the problem of creating an image classifier to classify the breed of the

dog image. Then, I'll write a function to incorporate the human face detector and dog image with the image classifier to detect if the image is a dog or human or something else and provide the results of the dog breed given the CNN image classifier model I trained. Totally, there will be 133 breeds of dog as the label used for the training.

1.3 Evaluation Metrics

In this project, I'll use the accuracy as the metrics to evaluate the model (Total correct prediction/Total predictions). I didn't evaluate the models on metrics such as Confusion matrix, precision, recall, and f1-score. Because the problem focuses on identifying the images provided and classify the dog breeds based on ground truth labels. Other metrics are not necessarily used in this particular project since we just want to know how accurately our classifier could predict the label given the images provided.

2. Analysis

2.1 Datasets and inputs

There are two datasets originated from Kaggle Competition, including one for human images and another for dog images. These two datasets have already been uploaded into AWS S3. I used the !wget to download data and !unzip to unzip them into local folders. There are totally 13233 human images while 8351 total dog images. (See fig 1) Since these two datasets are pre-processed and cleaned, there will be no data wangling for these two data sets. The dog images

will be divided into three parts: training (for training model purpose), testing (for evaluate the model performance) and validation (for hyperparameter tuning).

```
# load filenames for human and dog images
human_files = np.array(glob("/data/lfw/*/*"))
dog_files = np.array(glob("/data/dog_images/*/*/*"))

# print number of images in each dataset
print('There are %d total human images.' % len(human_files))
print('There are %d total dog images.' % len(dog_files))
```

```
There are 13233 total human images.
There are 8351 total dog images.
```

Figur1: data samples

2.2 Data Exploration & Visualization

Given the objective of this project, I started with analyzing the dog images data sets, which are used for the training, validation, and testing. I noticed that the file name of the jpg image file has the same regular expression pattern (i.e., 103.Mastiff/Mastiff_06826.jpg). Thus, I extracted the dogs' names from these images based on the below code (see fig2) After getting all the dogs' names, I tried to figure out the data distribution of these dog breeds based on the images provided. After counting the number of each dog breed (see fig3), we identified that the dog breeds are not equally distributed for each type breed. Some types have more images for training and testing while some have less images. Also, we know there are 133 unique breeds and Alaskan Malamute has the most images(see fig4). To further investigate the dataset, I also listed the top 5 breeds got most images and the last 5 breeds got least images(see fig5).

```
# Extract the dogs' names
dogs=[]
for i in range(8350):
    dog_name=dog_files[i].split(".")[1].split("/")[0]
    dogs.append(dog_name)
dogs
```

Figure 2 extracting the dogs' names

```
# Count each breed
import pandas as pd
dogs=pd.DataFrame(dogs)
dogs.rename(columns = {0 :'Breeds'}, inplace = True)
dogs['Breeds'].value_counts()
```

Figure3 Counting the number of each breed

```
dogs.describe()
```

	Breeds
count	8350
unique	133
top	Alaskan_malamute
freq	96

Figure4 Summary of the dog images of data sets

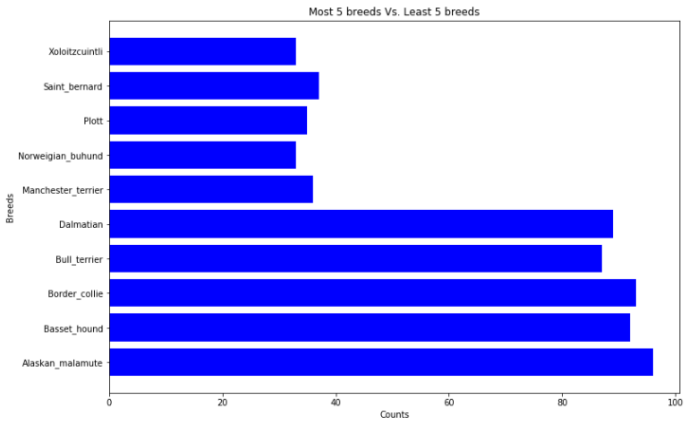

Figure5 the comparison between the 5 breeds with most images and least images

To further understand the datasets, I picked first 100 images of both dog images and human images to be tested in the human detector. As per expectation, most of the human images have been detected as the human images while very few dog images have been detected as the dog images. (See fig6). These dog images data also be used to see if the dog detector function created based on a pre-trained model works. The result is okay as around 80% of the dog images have been detected by this function. (See fig7)

```
]: human_files_short = human_files[:100]
   dog_files_short = dog_files[:100]

   #-#-# Do NOT modify the code above this line. #-#-#

   ## TODO: Test the performance of the face_detector algorithm
   ## on the images in human_files_short and dog_files_short.
   human_performance = np.mean([face_detector(img) for img in human_files_short])
   dog_performance = np.mean([face_detector(img) for img in dog_files_short])
   print("the percentage of human image detected: {:.2%} | the percentage of dog image mis-detected {:.2%}"
         .format(human_performance, dog_performance))
```

the percentage of human image detected: 98.00% | the percentage of dog image mis-detected 17.00%

Figure6 test results for human and dog images in human_detector

```
human_detected_as_dog = np.mean([dog_detector(img) for img in human_files_short])
dog_detected_as_dog = np.mean([dog_detector(img) for img in dog_files_short])

print('The percenatge of human file is detected as Dog : {:.2%}'.format(human_detected_as_dog))
print('The percentage of dog file is dectected as Dog : {:.2%}'.format(dog_detected_as_dog))
```

```
The percenatge of human file is detected as Dog : 0.00%
The percentage of dog file is dectected as Dog : 77.00%
```

Figure7 test results for human and dog images in dog_detector

2.3 Algorithms and Techniques

For human face dector, I used OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. This is a pre-trained face detectors, stored as XML files on github. In terms of dog_dectector, I built the function based on a pre-trained VGG-16 model, along with weights that 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. I used CNN to train the dog breed classifier with one CNN built from scratch as the benchmark of the model. Then I used a transfer learning based on a pretrained resnet50 model to build the breed image classifier with higher accuracy.

2.4 Benchmark

Before using a transfer learning model, I built a dog breed classifier with CNN architecture from scratch as the benchmark of the model. The accuracy of this model is comparatively low with 14% (see fig8)

Thus, I compared the model performance from a CNN dog classifier created from scratch and that from a pre-trained transfer model (ResNet). The performance of a pre-trained transfer model is way better than that from a CNN classifier created from scratch. Thus, in the final application, I used the Model from transfer learning to get the results of the classification of the breeds.

```
# call test function
test(loaders_scratch, model_scratch, criterion_scratch, use_cuda)
```

```
Test Loss: 3.694814


Test Accuracy: 14% (124/836)
```

Figure 8 model performance of the CNN model built from scratch

3. Methodology

3.1 Data Preprocessing

As mentioned earlier, these images are downloaded from a pre-processed source. Thus, the tidiness of these data are quite high. Apart from the extraction and analysis of the breeds name, I also preprocess the image data in order to improve the model performance. For example, I conducted image augumentation by horizonationly flip the image and rotate the image 10 degrees. Also, I resize the image for the training purposes. Besides, I split the training data and testing data. Finally, I set up the batch-size for the training processes.

3.2 Implementation

Totally, there will be six steps for this project:

- Step 1: Detect Humans
- Step 2: Detect Dogs
- Step 3: Create a CNN to Classify Dog Breeds (from Scratch)
- Step 4: Create a CNN to Classify Dog Breeds (using Transfer Learning)

- Step 5: Write your Algorithm
- Step 6: Test Your Algorithm

First, I will write a face detector function by using pre-trained face detectors from the OpenCV2- an open-source computer vision platform. Then, I will use the pre-trained VGG-16 Model to create a function for dog detector purposes. The VGG-16 Model, along with weights that 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. Then I will create a CNN to Classify Dog Breeds, although the performance may not be as accurate as other pre-trained transfer models. There are three convolution and max-pooling layers and two fully connected layer: FC1 and FC2. The final output is a (133,) array, which is in line with the 133 categories of dogs. (See fig9)

A particular Convolutional Neural Network (CNN) built-in image classifier framework will be chosen to train the data.
Following steps will be conducted during this step:

1. Define a neural network architecture
2. Define the training hyperparameter (epochs number, learning rate, number of layers, feature maps numbers, batch number etc.)
3. Define the Loss Function and performance metrics of the training model
4. Plot the Loss function and metrics results
5. Extract the optimal model obtained from the training process
6. Freeze the training model and terminate the training process

There are four main building blocks of the CNN

- Convolutional Layer:
  In this layer, the input (training samples) will be convoluted with filters (size: 28x28x3)
  For example, if our input is a 128x128x3 matrix (RGB image)- given the hyperparameter p: padding=0, and s: stride=1 and filters number is 10(detect 10 features)-then the output will be 101x101x10 matrix.
  There will be two types of filters used including: Low-Level filter, and High-Level filter. Low-Level filter will be used to detect edges (horizontal, curve, vertical)

High-Level filter will be used to detect complex edges (patterns and other characteristics from the fabric)

The convoluted output for one filter (feature detection) will be then fed into a rectified linear activation function or ReL for short (non-linearity function) along with a bias. A node or unit that implements this activation function is referred to as a rectified linear activation unit, or ReLU for short. The activated output will be used as the input for the next Layer.

- Max Pooling Layer

  The output from the convolution layer will be then downsized in the max pooling later. This operation will choose the maximum, or largest, value in each patch of each feature map. For example, if we have a 4x4 matrix input, then the output will be a 2x2 matrix

After the Model created from scratch, I then used transfer learning to build my model artifacts by using a pre-trained Model ResNet 50. Given the much better performance of the results from transfer learning. I choose the ResNet50 to deploy this model as the image classifier. Finally, I will create an application by incorporating the human and dog detector and deploying the Model from the transfer learning.

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (norm2d1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=100352, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=133, bias=True)
)
```

Figure9 CNN model summary

3.3 Refinement

As mentioned earlier, I compared the model performance from a CNN dog classifier created from scratch and that from a pre-trained transfer model(ResNet). The performance of a pre-trained transfer model is way better than that from a CNN classifier created from scratch. Thus,

in the final application, I used the Model from transfer learning to get the results of the classification of the breeds.

Using the first convolutional neural network model built from scratch, the goal was to beat random guesses and it achieved 14 % accuracy. (fig10) For the transfer learning, the performance is much better which has achieve 86% (fig11)

```
# call test function
test(loaders_scratch, model_scratch, criterion_scratch, use_cuda)
Test Loss: 3.694814

Test Accuracy: 14% (124/836)
```

Figure10

```
test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)
Test Loss: 0.507031

Test Accuracy: 86% (719/836)
```

Figure11

4. Results

4.1 Model Evaluation and Validation

- 98% of the first 100 images of human images could be detected by the human detector
- About 80 % of the first 100 images of dog images could be detected by the dog detector
- Using the first convolutional neural network model built from scratch, the goal was to beat random guesses and it achieved 14 % accuracy. (124 accurate/836 total)
- For the transfer learning (ResNet50), the performance is much better which has achieve 86% (719 accurate/836 total), which is much more higher.

4.2 Justification

98% and 80 % of the images could be detected by the human face and dog dector. The breed classifier built upon the pretrained ResNet 50 model could reach 86 % of accuracy based on our training and testing data. These should be okay to meet the objective of this project. However, there are some limitations which I will mention in the later section for these application I built.

We could improve the application by refining the model and mitigating the impact from the limitations.

5 Conclusion

5.1 Free-Form Visualization

This classifier has met the objective of this project, which is able to identify if the image is human being or dog. And this classifier could also classify the dog breeds though without 100% of the accuracy due to the limitations I mentioned in the below. The image will be used as input in the final application, which will be used to detect if the image is a human or dog and identify the dog breeds based on the trained classifier. (See fig12)

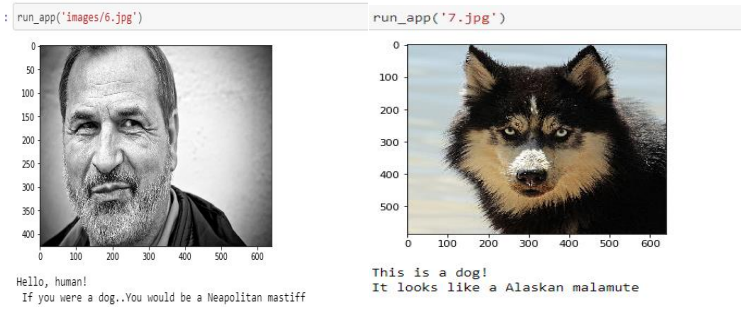

Figure 12

5.2 Reflection

This project makes me realize that with the aid of the deep learning algorithms and exponentially increased digitalized images, the computers are now "smarter" enough to address most of the complex issues, such as classifying different images and detect an object with high accuracy. As an PhD student in textile engineering, I realized that Computer vision (CV for short) is one of the promising techniques that could help us solve previously unsolvable problems. For example, Given the enormous amount of images of fashion products and fabric patterns, there is a considerable opportunity to explore how computer vision could be applied in the textile industry.

The algorithms and methodologies used in this project inspired me of investigating the possibility of creating a fabric texture image classifier by employing a deep learning algorithm.

5.3 Improvement and Limitations

There are four limitations of this project and three improvements could be done if we would like to improve the performance of the model.

1) We could involve more data samples(images) given we only have around 8000 images. It would be better to involve more data samples with more images for each breed of the dog.

2) Given the computing power I have, I only set epoch number for 20. We could change the hyperparameters such as epoch number (add more epoch number). Other parameters such as the number kernel (edge detectors) used in the mode could be added to improve the performance of the model. Also, I could've used the AWS SageMaker service to shorten the training time I used for this project.

3) I noticed that only 118 breeds are detected for dog detector; however, we have 133 breeds in the datasets. That is to say, our detector may not be accurately detecting at least several breeds in our training datasets given this limitation.

3) In this project, I only horizontally rotated 10 degrees of the training sample, we could add more image augmentation such as flip rotation etc. to take best advantage of the images we used to feed the model to be trained.

References:

[1] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).