Udacity Machine Learning Nanodegree Capstone Proposal

Use Convolutional Neural Networks to Identify Dog Breeds

Hao Cui

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 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 devided 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

1.4 Solution statement

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 fig2) After the Model created from scratch, I will use transfer learning to build my model artifacts by using a pre-trained Model ResNet 50. 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)
)
```

Figure2 CNN model summary

1.5 Benchmark model

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.

1.6 Evaluation Metrics

In this project, I'll use the accuracy as the metrics to evaluate the model (Total correct prediction/Total predictions). Using the first convolutional neural network model built from scratch, the goal was to beat random guesses and it achieved 14 % accuracy. (fig3) For the transfer learning, the performance is much better which has achieve 86% (fig4)

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

```
Test Loss: 3.694814

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

Figure3

```
test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)
```

```
Test Loss: 0.507031

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

Figure4

1.7 Project design

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

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

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 fig6)

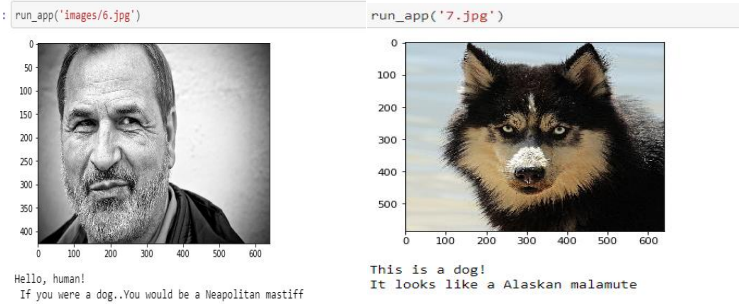

Fig6

1.7 Refinement and Limitations

1) could involve more data samples(images)

2) could change the hyperparameters such as epoch number(add more epoch number)

3) add more image augmentation such as flip rotation etc.

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).