Report of minipj 2

Luyang Huang

Basically, this project mainly focus on comparing different neural network algorithm. We first define a binary classification problem and then test three kind of neural network architecture.

I.Problem Definition

We want to classify two kinds of car which are sports car and SUV. However, getting enough training data is very hard. So we start crawling images on bing, a searching engine. We search for keyword “SUV” and “sports car” and get pictures from search results. Hence, we don’t need to label pictures. The code is in crawl4data.py. The crawl is mainly based on selenium module.

We finally get 402 sports car images and 375 SUV pictures which are enough for training.

II. Model

In this work, we build three neural networks common in Computer Vision, which are VGG13, Basic -CNN and Pure-FullConnection.

VGG13[i]

VGG13 is a CNN-based architecture with 6 Convolutional Layers with each one followed with a max pooling layer and the activation function with RELU. Here we use pre-trained weight of these convolutional layers to see if pre-trained layer can extract features better. Then we add three full connection layer with first two have activation function of RELU and last one with log softmax. The loss function is negative log likelihood. In the training procedure, we set Convolutional layers as non-trainable.

The parameters of layers are shown in Fig1.

Basic-CNN

We here define basic CNN as a simple version of VGG with only 2 Convolutional layers and 2 linear transformation layers. This model has smaller number of parameters. Because our dataset is a rather small one, training on VGG13 is not efficient and is hard for parameters to get optimized. So we define basic-CNN to test the performance of CNN.

The parameters of layers are shown in Fig2.

Pure-FullConnection

Finally, we decide to use only full connection layer which means we see raw images as features. The model parameters are shown in Fig3.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 224, 224]           1,792
              ReLU-2         [-1, 64, 224, 224]               0
         MaxPool2d-3         [-1, 64, 112, 112]               0
            Conv2d-4        [-1, 128, 112, 112]          73,856
              ReLU-5        [-1, 128, 112, 112]               0
         MaxPool2d-6          [-1, 128, 56, 56]               0
            Conv2d-7          [-1, 256, 56, 56]         295,168
              ReLU-8          [-1, 256, 56, 56]               0
            Conv2d-9          [-1, 256, 56, 56]         590,080
             ReLU-10          [-1, 256, 56, 56]               0
        MaxPool2d-11          [-1, 256, 28, 28]               0
           Conv2d-12          [-1, 512, 28, 28]       1,180,160
             ReLU-13          [-1, 512, 28, 28]               0
           Conv2d-14          [-1, 512, 28, 28]       2,359,808
             ReLU-15          [-1, 512, 28, 28]               0
        MaxPool2d-16          [-1, 512, 14, 14]               0
           Conv2d-17          [-1, 512, 14, 14]       2,359,808
             ReLU-18          [-1, 512, 14, 14]               0
           Conv2d-19          [-1, 512, 14, 14]       2,359,808
             ReLU-20          [-1, 512, 14, 14]               0
        MaxPool2d-21            [-1, 512, 7, 7]               0
           Linear-22                 [-1, 1024]      25,691,136
             ReLU-23                 [-1, 1024]               0
          Dropout-24                 [-1, 1024]               0
           Linear-25                  [-1, 256]         262,400
             ReLU-26                  [-1, 256]               0
          Dropout-27                  [-1, 256]               0
           Linear-28                    [-1, 2]             514
================================================================
```

Fig1. Architecture of VGG13

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 10, 215, 215]           3,010
            Conv2d-2           [-1, 10, 98, 98]          10,010
         Dropout2d-3           [-1, 10, 98, 98]               0
            Linear-4                   [-1, 32]         768,352
            Linear-5                    [-1, 2]              66
================================================================
```

Fig2. Architecture of basic-CNN

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 512]      77,070,848
            Linear-2                   [-1, 32]          16,416
            Linear-3                    [-1, 2]              66
================================================================
```

Fig3. Architecture of Pure-FullConnection

III. Metric and Experiment

We define accuracy as our metric:

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

We first resize all pictures to 224*224*3 and set them as raw input. Then we split dataset to 3 parts.

Training part contains 550 images. Test part contains 137 images. Development set contains 90 images.

The distribution of labels on each part is shown below.

Set name/Label name	Sports Car	SUV
train	204	193
test	66	62
development	47	40

Table1. Distribution of data set

Our experiment contains two parts. First we set dropout as 0.2 and learning rate as 0.001 and train for 20 epochs. After loss can not become smaller, we set dropout as 0.5(larger) and learning rate as 0.0001(smaller) and then fine tune for another 20 epochs.

We choose negative loss likelihood as loss function and choose Adam as optimizer.

IV. Results

Name/Method	Training after 20 epochs	Fine-Tune
VGG13		**0.93**
Basic-CNN	0.65	0.85
Pure-FullConnection	0.50	0.52

Table2. results of different models

Clearly, after finetuning step, the performance of model raises significantly. We get best score on pre-trained model, because pre-trained model is trained on images(similar input) and the data is much larger than our dataset. The pre-trained data is much better!

Second, CNN performs better than taking raw data as direct features. This is what CNN counts for, fast and efficient selecting important feature and choose most important features(max pooling).

Third, the full-connection performs badly on classificaiton.(Like almost random guess!) This may due to:

1. The features of car and SUV are not so clear before selecting by CNN.

2. Full-connection layer has too many parameters and it is hard for model to converge if it has too many parameters and is with few training samples!

Reference

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