Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #5

April 7th, 2019

Image Recognition with CNN

I. Abstract and Motivation

In recent years, convolutional neural network (CNN) has brought a revolution in the field of computer vision. Using CNN can greatly improve the accuracy of various computer vision tasks. In this assignment, we will try to use LeNet-5 which is a simple but groundbreaking neural network model. We will train a LeNet-5 network to train and classify images using MNIST data set which consists hand written digits from 0 to 9.

II. Approach and Procedures

LeNet-5

LeNet-5 is a simple groundbreaking CNN which contains all necessary parts for CNNs, convolutional layer, pooling layer and fully connected layer. Its network structure is given, so we do not need to adjust its structure.

MNIST Data Set

MNIST data set consists hand writing digit images from zero to nine. It has two parts, training set and test set. The training set has 60000 images and test set has 10000 images. All the image are gray scale and digits aligned to the center of images.

Platform and Implementation

My program uses Tensorflow and Keras. The LeNet-5 network is defined using Keras, and the MNIST data set is automatically loaded by using an API in Kears. Since MNIST image size is 28 * 28, and the LeNet-5 input image size is 32 * 32. So, images should be padded to 32 * 32 by using background padding.

Normally, training set should be further divided into training and validation set. The validation set is used to fine tuning the network structure. However, in this homework, the network structure is fixed, so validation set is not needed. Thus, all 60000 images are used in training. Training data is randomly shuffled before training.

Although, data augmentation can be used here to enlarge the training set. I do not do that, since images in the MNIST data set's data consistency is high, all images are already aligned.

So that, there are only two parameters. Number of epochs and batch size.

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #5
April 7th, 2019

Negative Images Improvement

Since all the images in training data are “positive” images, so low accuracy for “negative” images is foreseeable. Because, features in “positive” and “negative” images are numerical opposite sided. The potential solution is not difficult to think of. I doubled the training set by adding their “negative” images.

III. Experimental Results

Training results are different between trails with same parameters. Here I record the first trail of each combination of parameters. Since now all the data is required to get a test accuracy above 98%, so I designed some extreme settings to see the difference of results. I tried to find how each parameter affect the result and a best combination. However, due to this effect, it is hard.

Network Summary

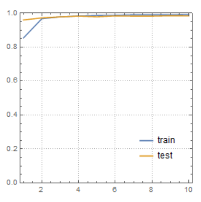
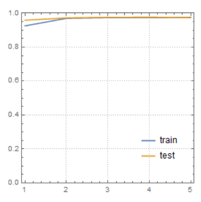
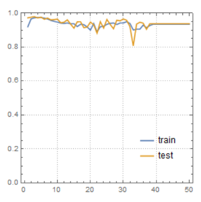
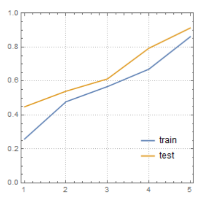
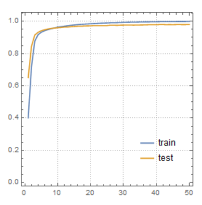
Generated by Keras

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 6)         156
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               48120
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_3 (Dense)              (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
```

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #5
April 7th, 2019

Train LeNet-5 on MNIST Dataset

Settings

	Setting 1	Setting 2	Setting 3	Setting 4	Setting 5
Batch size	128	4	4	4096	4096
# of epochs	10	5	50	5	50
Epoch training & test accuracy					
Final training accuracy	0.9922	0.9759	0.9351	0.8618	0.9992
Final test accuracy	0.9846	0.9738	0.9381	0.9132	0.9807
Is the best setting?	√				

Overall

	Mean	Variance
Final training accuracy	0.9528	0.0032
Final test accuracy	0.9581	0.00096888

Apply trained network to negative images

By using the best model setting above, its negative image accuracy is 0.1279, 0.1573, 0.1198, 0.1542, 0.1032 (5 trails, mean=0.1325, var=0.00053172).

After I enlarged the training data set by adding negative training images, its negative accuracy now become 0.9845, 0.9830, 0.9849, 0.9841, 0.9855 (5 trails, mean=0.9844, var=0. 00000088000) and positive accuracy become 0.9848, 0.9860, 0.9833, 0.9838, 0.9843 (same 5 trails, mean=0.9844, var=0.0000010730).

Zongjian Li
6503-3789-43
zongjian@usc.edu
EE 569 Homework #5
April 7th, 2019

IV. Discussion

[Problem (a)-1] Describe CNN components in your own words: 1) the fully connected layer, 2) the convolutional layer, 3) the max pooling layer, 4) the activation function, and 5) the softmax function. What are the functions of these components?

The fully connected layer:

It is a layer of connection that connecting inputs and outputs. Each ouput dimension is connected to all input dimensions, so it is fully connected. For each output dimension, all connected inputs are combined together by weights and biases, such as

$$O = \sum_i (w_i I_i + b_i)$$

where O is one of the outputs, and I is input, w is weight, b is bias. Then a given activation function may be used for this O as the final output.

The convolutional layer:

Convolutional layers extract image area features. Each layer has its image filter to extract local features from its inputs, and the weights of the filter can be automatically adjusted by using back propagation. The input can be multiple channels, so the convolutional layers may also have multiple filters for all input channels.

The max pooling layer:

Pooling layers summarize image area features to reduces the dimension of the data. This also increases the speed of calculation and avoids overfitting. They use a window scan through the outputs of a convolutional layer and give one output for each location. Usually the location of the windows does not overlap. The max pooling layer is one sort of pooling layers which picks the maximum value within its scanning window as the output at its current window location.

The activation function:

Since the combination of using weights and biases can only give linear results, and linear combination has its limitations to fit functions. To introduce some non-linear characteristic, an activation function which is non-linear is used for each layer. There are several kinds of activation functions. Some of them clip the output value between a small range to avoid errors in subsequent calculation accuracy. Some of them discard or reduce the impact of negative output values.

The softmax function:

Softmax function is used in the final step of the classification problem. It covers its input vector's classifications values into probabilities of classes by

$$\mathrm{O}_i = \frac{e^{I_i}}{\sum_j e^{I_j}}$$

where I is input, O_i is one of the outputs.

Then we can easily pick the class with largest probability or compare probabilities of classes.

[Problem (a)-2] What is the over-fitting issue in model learning? Explain any technique that has been used in CNN training to avoid the over-fitting.

Over-fitting in CNN means that the trained network over-matches the data in the training set and loses the generalization ability. This leads to a higher accuracy of the data in training set, but the accuracy of the test set is declined. As a result, the trained network has worse performance than those does not over-fitting. It tends to be over-fitting when the training set is too small while the number of training iteration is too large.

There are several ways to avoid over-fitting. The easiest and most intuitive way is to avoid to many iterations in the training process. Such as, stop the training when the accuracy does not change. This is called early stopping. Another way is data augmentation. It generates more training data from given training set, so the input training set become larger, and the data diversity also increased. Such as, for image training data set, some geometric modifications, adding distortion and noise can be performed to generate more training data. Other ways like regularization and dropout can also help to avoid over-fitting, and their principles are also different from the above two methods.

[Problem (a)-3] Why CNNs work much better than other traditional methods in many computer vision problems? You can use the image classification problem as an example to elaborate your points.

For image classification, to classify the object in an image, some filters that can match objects in classes are needed. For traditional methods, people design these filters. However, a person cannot design complicated filters, and the filter may not represent the underlying pattern of object very well. So, traditional methods have poor accuracy.

While, CNN is data driven. It learns these filters from labeled data automatically. So learnt filters can consist more parameters than people can understand and design. And these parameters are suitable for classifying training data since they represent the common patterns of the objects in classes.

Zongjian Li

6503-3789-43

zongjian@usc.edu

EE 569 Homework #5

April 7th, 2019

[Problem (a)-4] Explain the loss function and the classical backpropagation (BP) optimization procedure to train such a convolutional neural network.

Loss function is a sort of function to measure the difference between network output and expected output. Lower lost means better network performance to given data. So, we can try to minimize the loss to get better trained network. Backward propagation (BP) is one efficient way to find a path to adjust parameters to get lower loss. It utilizes gradient information to know which direction has lower loss from current state. Then, with this information, we can adjust parameters to that direction and then try to evaluate the network again and again. BP requires the network parameter space can be calculate gradient. Thus, for those networks that can get gradient information, BP is a efficient way to search good parameters among huge parameters space.

How batch size & number of epochs infect training

According to my observations, even I change these two parameters with in large range, the best test accuracy among multiply trails do not have significant difference. Most of reasonable parameter settings can give a test accuracy above 98% which is required by this homework. However, for some setting, results from trails seem more stable to stay above 98%. With a larger size of batch, training is faster while need more memory. For MNIST dataset, it is not large, so I even can feed all the training data together into the model. However, too large and too small batch size both has worse result than intermediate batch size. When the batch size is very small, the direction of gradient descent of each step is determined by few training data, since the diversity of a sequence of data, the direction of gradient descent is randomized, so the training does not give good result. Sometime, the gradient decent does not work. For very large batch size, there are too few training steps to slowly approach lower position. So, an intermediate batch size is good. For epochs, all the data in training set is fed to a network after one epoch. Too few epochs may lead to inefficient training, and too much epochs will lead to over-fitting. We can see that one of my setting, which I designed shows that the network over-fitting the training data set and get almost 100% accuracy, while the accuracy for test set get stable.

Why LeNet-5 in sub-problem b does not work for negative images, and why my solution works.

The digit image has very large object and background contrast. There pixel values have very large difference. For the training in sum-problem b, the network is trained to ignore while background which has high pixel values and generated filters that has strong response for those dark objects. All the training data follow this same pattern. So, among training process, this feature of filters is enlarged. Although, the negative image has visually similar digits, the pixel value distribution is different, so trained filters do not work this time.

After I add both positive and negative images into the training data set, and the data set is randomly shuffled. The network will not rely on those features before, and train some filters has both high response to positive and negative features. That is negative is a simple mathematical operation, which is easy for a neural network to learn. As a result, both positive and negative result have high accuracy.