The improvement of the neural network structure to the handwritten digit recognition

Fei Yang

Department of Computer Science, University of Alberta
Email: fei5@ualberta.ca

1 Introduction

Convolutional neural network (CNN) is a representative neural network in deep learning, made up of neurons that have learnable weights and biases [1]. Fig.1 shows an example of CNN. CNN can effectively reduce the dimensionality of a large amount of data into a small amount of data and retain the important image features, in order to learn grid features (such as picture pixels) and has a stable effect, so it is widely used in the field of computer vision. In Lab5, we implemented gradient descent on the MNIST dataset for recognizing handwritten digits, which can be seen as a linear fully connected layer. The accuracy we got is around 85% and the images plotted using the vector W are blurring and some outputs look like several numbers overlapped. The reason why the accuracy is not ideal is that it is difficult to retain the original features of the image in the statistical method. In this project, We designed a basic CNN and used the MINST dataset as the experimental dataset to train. The accuracy on the validate dataset reached 96%, and the accuracy on test dataset reached 97.02%, achieving a relatively better recognition effect than Lab5. The GitHub link for the code of this project is: `https://github.com/feiooo/The-improvement-of-the-neural-network-structure-to-the-handwritten-digit-recognition`

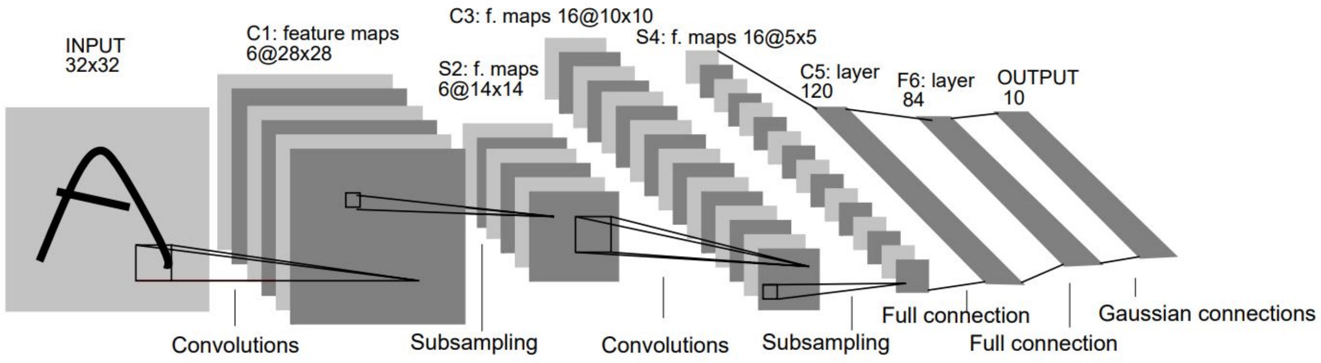

Figure 1: an example of Convolutional Neural Network(LeNet-5) for digits recognition [2]

2 Methods

2.1 Basic Theory of Convolutional neural network (CNN)

A typical CNN mainly consists of three parts, input layer, hidden layer and output layer. Hidden layer including three types of layers, convolutional layer, pooling layer, and fully connected layer. The convolutional layer is responsible for extracting local features in the image. The calculate processing example is shown in Fig.2. The pooling layer is used to greatly reduce the magnitude of the parameters (dimensionality reduction). We will use Maxpooling in our network. The calculate processing example is shown in Fig.3. The fully connected layer is used to output the desired results.

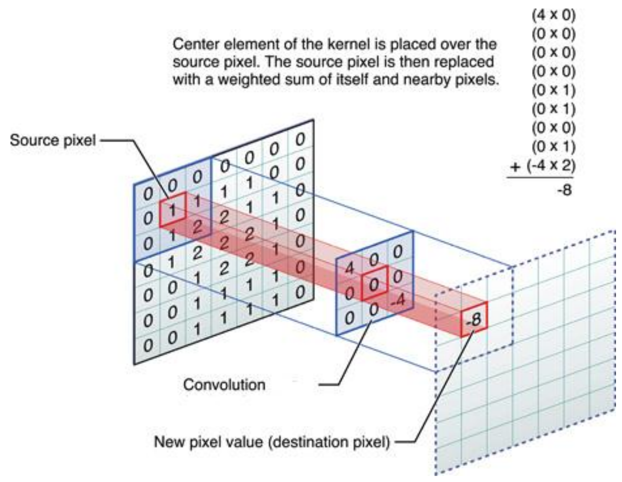
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution
New pixel value (destination pixel)

Figure 2: Convolutional layer calculation [3].

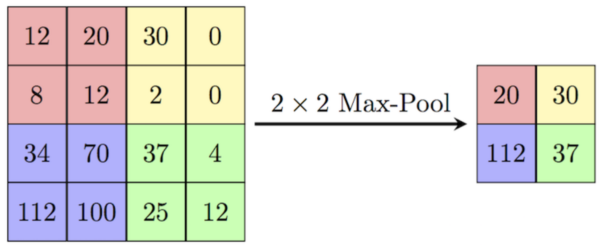

Figure 3: Maxpooling calculation [4].

2.2 Design a CNN

The network structure we designed is shown in Fig.4.

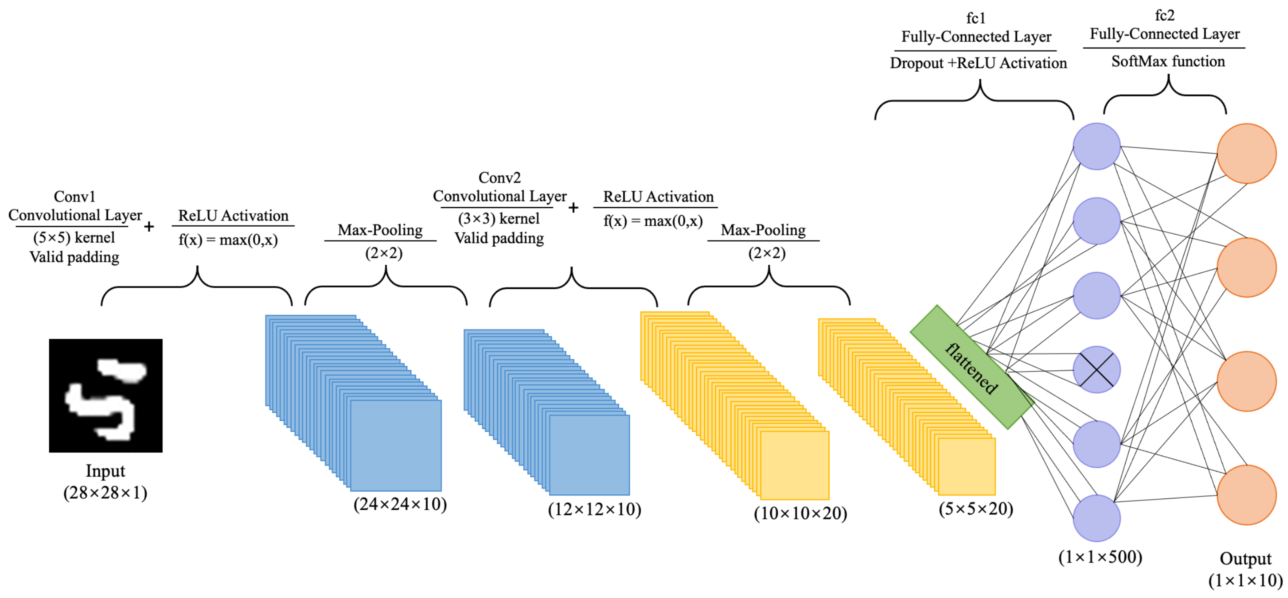

Figure 4: The architecture of our neural network

Since the MNIST dataset is a grayscale image dataset, the size of the input image is (28×28×1). The input image first passes through a convolutional layer with a kernel size of (5×5). The output channel of the first layer is set to 10, which means the output's size is (24×24×10)1. A ReLU activation function(Fig.5) is used to make output nonlinear. The next layer is a maxpooling layer with a (2×2) filter. The output size of the second layer is (12×12×10). Then enter another convolutional layer. Its kernel's size is (3×3) and the output channel is 20. After throwing the output into a ReLU activation function and passes through another maxpooling layer with a (2×2) filter, there is a necessary preprocessing before entering the fully connected layer. The view() function is used to "reshape" the output of the previous layer into one dimension (ie. flatten(5×5×20) to (1×1×500)). The next layer is a fully connected layer. The input size is (1×1×500) and the output is (1×1×500). We use a dropout fraction of 0.2 after this fully connected layer, which means in each training iteration, in each epoch, there are 20% nodes stop updating weights. This can make the model more general, so it will not rely too much on some local features, avoiding the model overfitting [5]. After a ReLU activation function, the image data will enter the last layer. The last layer is a fully connected layer. Since there are 10 types of handwritten numbers (0∼9), the output size should be 10, which means the output is (1×1×10). At the end of the network, use the Softmax function(Fig.6) to map the output to the interval [0,1], and the cumulative sum of the probability of the predicted result should be 1.

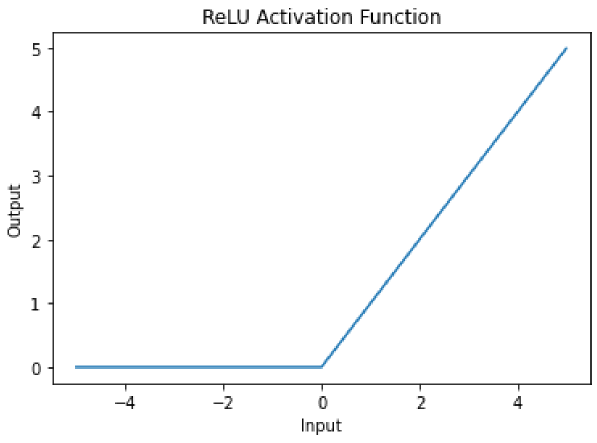

Figure 5: ReLU activation function [6]

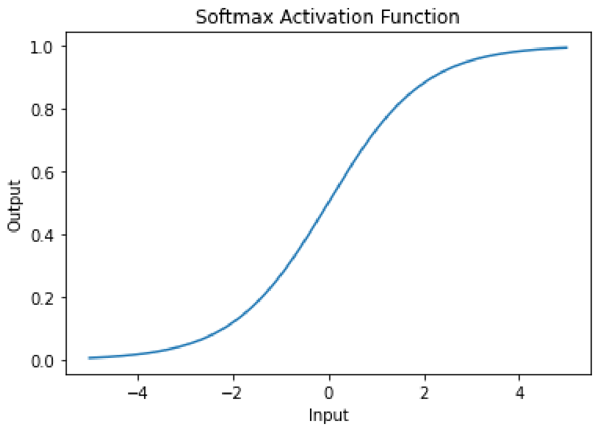

Figure 6: SoftMax function [7]

2.3 Train, Validate and Test

After the CNN innitializationwe create a CNN network and pass the prepocessed training dataloader into the network for training. Compared the predicted labels with the real target labels and count the correct number to calculate the accuracy. As for the optimizer and loss function, Stochastic Gradient Descent (SGD) [8] is used as optimizer. zero_grad() function is used to zero the gradient, and step() function is used to update the network parameters after a backpropagation. Cross-entropy loss [9] is used to calculate the loss. This experiment designed 40 epochs, batch size is 128, learning rate is 0.014, momentum is 0.9876, and weight decay is 0.01. These parameters are based on the experience gained in the CMPUT328 class. During the training process, we validate the model every 5 epochs, to be able to find problems with models or parameters in time. After the training and validating, use the test dataloader to test the trained model. By comparing the predicted label with the real label, the number of correct predictions can be counted to calculate the recognition accuracy. The specific results can be seen in the Evaluation section.

3 Evaluation

The loss change during train processing is shown in the Fig.7. The loss change and accuracy change during validate processing are shown in the Fig.8 and Fig.9. It can be seen from the figure that loss has been decreasing, and it is stable at about 0.0120 by the fifth epoch. In the tenth epoch, the accuracy has reached about 95%, which shows that the training efficiency is high.

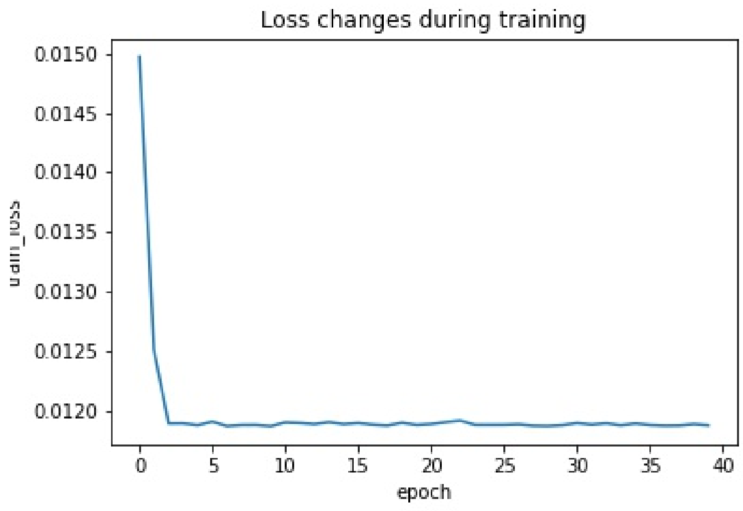

Figure 7: Loss changes during training

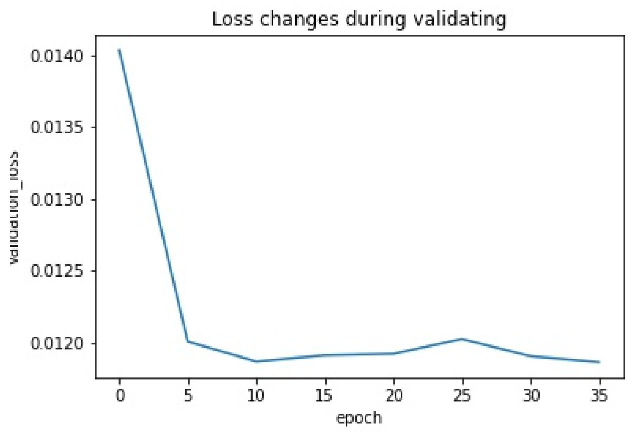

Figure 8: Loss changes during validating

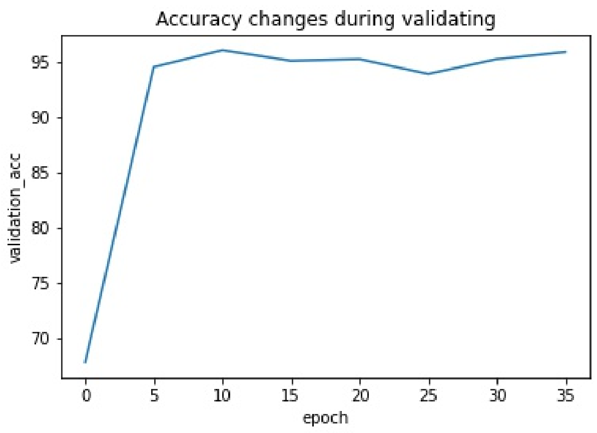

Figure 9: Accuracy changes during validating

The comparison results of our method and lab5's method are shown in Table 1. Some test sample results are shown in Fig.10. All of the 8 test sample images are predicted correctly, which is more accurate than the result of the statistical method (Gradient Descent) in the lab5. Compared with the statistical method, the neural network (CNN) method retains a large number of important information features in the images, and the accuracy of recognition is a qualitative leap.

Method	Correct/Total	Accuracy	Total Time Spent
Statistical method (Gradient Descent)	8574/10000	85.74%	0.0279s
Neural Network (CNN)	9702/10000	97.02%	1.6770s

Table 1: Results of Different Methods on Test Dataset

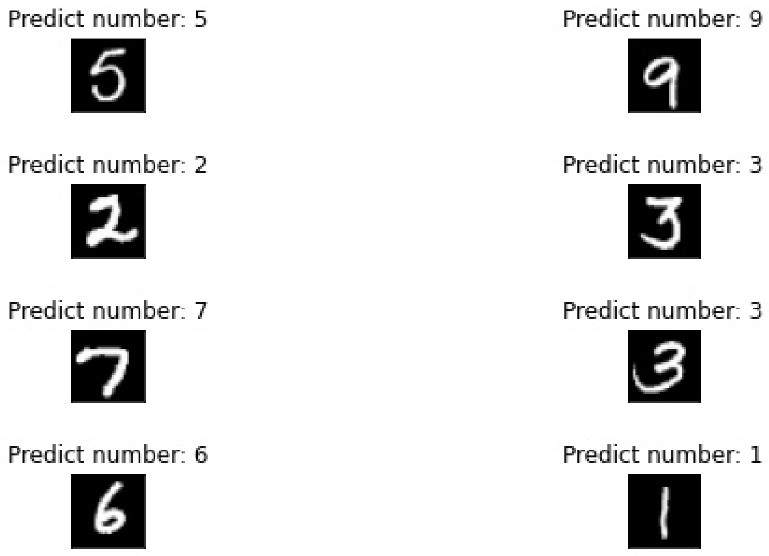

Figure 10: Test sample results

References

[1] CS231n Convolutional Neural Networks for Visual Recognition: https://cs231n.github.io/convolutional-networks/

[2] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[3] 6 basic things to know about Convolution: https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411

[4] Max-pooling / Pooling: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

[5] Batch Normalization and Dropout in Neural Networks with Pytorch: https://towardsdatascience.com/batch-normalization-and-dropout-in-neural-networks-explained-with-pytorch-47d7a8459bcd

[6] ReLU function: https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec

[7] What is the Softmax Function? https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

[8] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747(2016).

[9] Loss Functions (Cross-Entropy Loss): https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html