Report for Artificial Neural Network HW2

Jiayi Weng
Department of Computer Science
Tsinghua University
wengjy16@mails.tsinghua.edu.cn

1 Network Architecture and Hyperparameter Setting

I have designed two network architectures and tuned the hyperparameters. Details of them are listed below. The two architecture are similar with each other. There are both two "Conv2D + ReLU + AvgPool2D" modules and a final "Linear" layer. The loss function is designed as softmax cross-entropy loss.

1.1 Original CNN

It is the default setting provided by the original code. Detailed network architecture can be found in Table 1.

Table 1: Origin CNN, the size is presented as "Channel $\times$ Height $\times$ Width".

Layer	Type	Input Size	Output Size	Kernel	Pad	Init σ
1	Conv2D	$1 \times 28 \times 28$	$4 \times 28 \times 28$	3×3	1	1
2	ReLU	$4 \times 28 \times 28$	$4 \times 28 \times 28$			
3	AvgPool2D	$4 \times 28 \times 28$	$4 \times 14 \times 14$	2×2	0	
4	Conv2D	$4 \times 14 \times 14$	$4 \times 14 \times 14$	3×3	1	1
5	ReLU	$4 \times 14 \times 14$	$4 \times 14 \times 14$			
6	AvgPool2D	$4 \times 14 \times 14$	$4 \times 7 \times 7$	2×2	0	
7	Linear	196	10			0.1

The initial learning rate is set to 0.01. For better convergence, it will become a half each 10 epochs. The batch size is 100, and training epoch is equal to 40. As for SGD, the momentum is 0.9 with zero weight decay.

1.2 Designed CNN

I have tried several parameters and find this configuration best. Detailed network architecture can be found in Table 2.

Other hyperparameters are exactly the same as "Original CNN", as mentioned above.

2 Experiment

2.1 Experiment Setting

All of these experiments are conducted on a Linux server. The CPU infomation is "Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz".

Table 2: Designed CNN, the size is presented as "Channel $\times$ Height $\times$ Width".

Layer	Type	Input Size	Output Size	Kernel	Pad	Init σ
1	Conv2D	$1 \times 28 \times 28$	$12 \times 28 \times 28$	3×3	1	1
2	ReLU	$12 \times 28 \times 28$	$12 \times 28 \times 28$			
3	AvgPool2D	$12 \times 28 \times 28$	$12 \times 14 \times 14$	2×2	0	
4	Conv2D	$12 \times 14 \times 14$	$10 \times 14 \times 14$	3×3	1	1
5	ReLU	$10 \times 14 \times 14$	$10 \times 14 \times 14$			
6	AvgPool2D	$10 \times 14 \times 14$	$10 \times 7 \times 7$	2×2	0	
7	Linear	490	10			0.1

2.2 Loss

The loss in training procedure is illustrated in Figure 1, and the numeric result of training and testing loss is in Table 3.

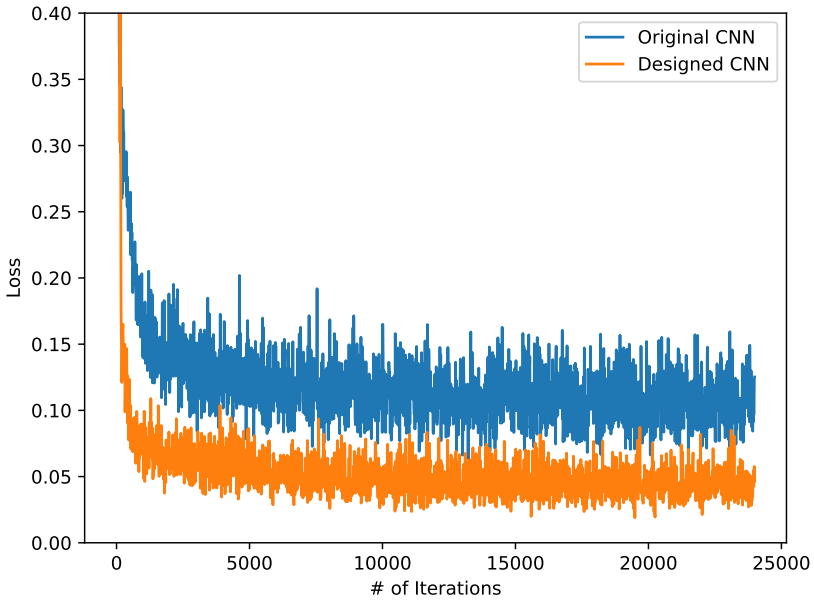


Figure 1: Training loss of different network.

Table 3: Experiment Result

Metric	Best MLP	Original CNN	Designed CNN
Train Loss	0.01364	0.086528	0.031458
Train Acc.	**99.61%**	97.45%	99.00%
Train Time	**164.739s**	1637.48s	9776.924s
Test Loss	0.02306	0.08680	0.05082
Test Acc.	98.13%	97.31%	**98.38%**
#Param	203530	**2158**	6120

2.3 Accuracy

The accuracy in training procedure is illustrated in Figure 2, and the numeric result of training and testing accuracy is in Table 3.

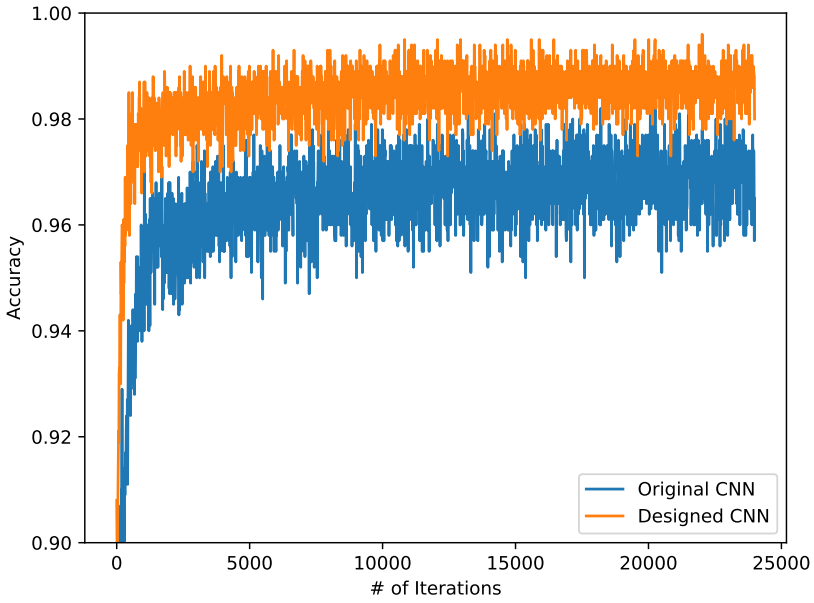
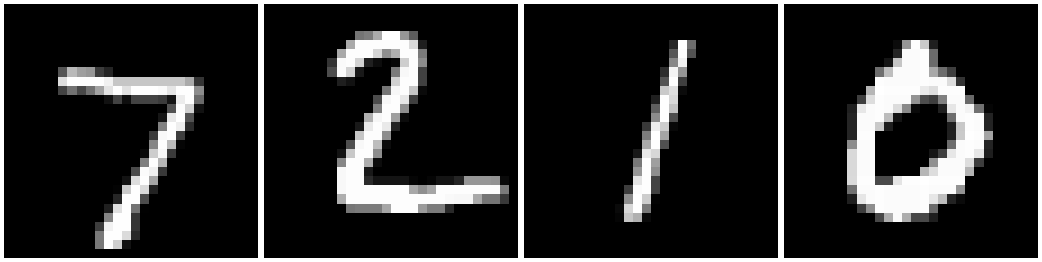


Figure 2: Training accuracy of different network.

2.4 Visualization

I have selected the first 4 sample in MNIST, as shown in Figure 3.



(a) (b) (c) (d)

Figure 3: The first 4 sample in MNIST.

For the first CNN architecture, its first layer has 4 feature maps, as shown in Figure 4. Result of these 4 data passing through first convolution layer can be found at Figure 5.

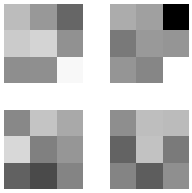
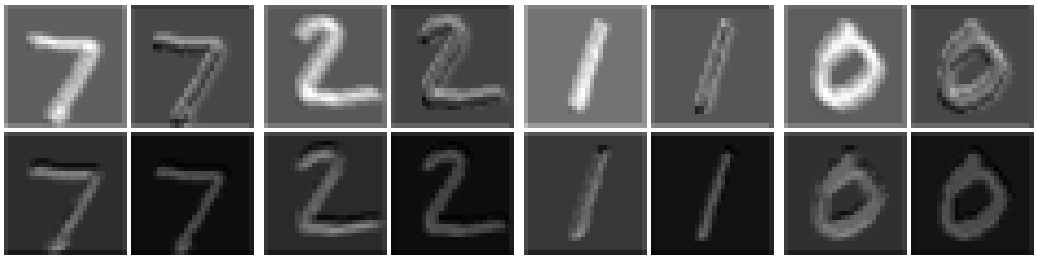


Figure 4: Visualization of the first layer's feature map in Orignal CNN.

For the second CNN architecture, its first layer has 12 feature maps, as shown in Figure 6. Result of these 4 data passing through first convolution layer can be found at Figure 7.



(a) (b) (c) (d)

Figure 5: Visualization of the first layer's output in Orignal CNN.

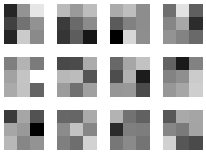
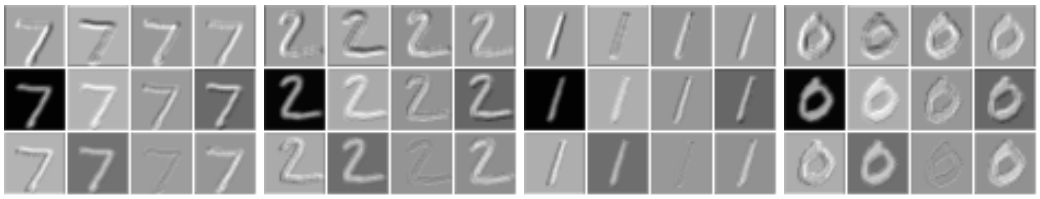


Figure 6: Visualization of the first layer's feature map in Designed CNN.



(a) (b) (c) (d)

Figure 7: Visualization of the first layer's output in Designed CNN.

2.5 Analytics

Several conclusions can be drawn according to Table 3.

First, the training time of CNN is much longer than MLP, due to the 2D-convolution operation. I use "im2col" to accelerate the training procedure, but, to achieve a satisfied result, it needs more time to train the network. Also, from the information in WeChat group, I can confirm my implementation is faster than most of my classmates'.

For the convergence, Figure 1 and 2 indicate that after 10000 iterations the CNN almost reaches a local minimum. At about 2000 iterations, the network begins to slow down its fast converge speed. But MLP seems better. It converges faster in the first 1000 iterations.

As for the accuracy, MLP can get the highest score on the trainig time, which is 99.61%. On the test set, it also performs very well, the 98.13% classification accuracy. On the contrary, the Original CNN cannot obtain better result. The best result it achieved is 97.45% on the training set and 97.31% on the test set. After adjusting the architecture of network, we can see a huge improvement of CNN. The Designed CNN has the best score 98.38% on test set.

The only drawback of MLP is the number of parameter in the network. You know, CNN utilizes the spatial information, so the model size is much smaller than MLP. However, the representation ability of MLP is stronger than CNN. From my perspective, CNN is a subset of MLP.