Report on Project 2

Zhunxuan Wang, 13300180086
School of Mathematical Sciences

November 13, 2017

1 Convolutional Neural Network

1.1 Model Description

We decided to apply the classic convolutional neural structure [1] (shown below) on `MNIST` dataset

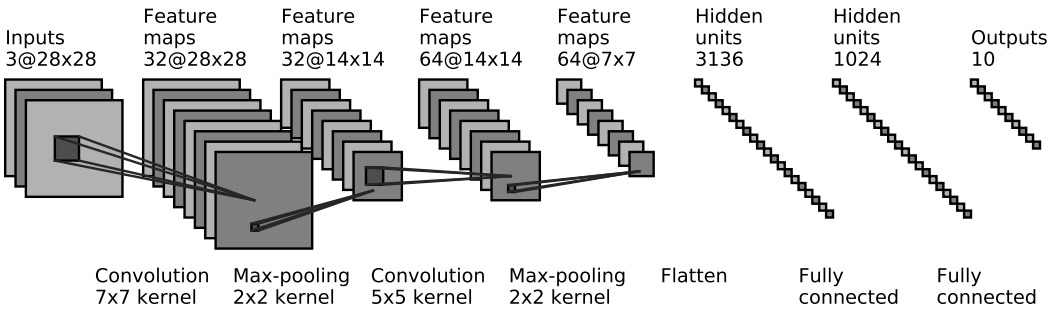

Figure 1: The CNN Structure on `MNIST`

where the input matrix is the input image (size: 28×28), and the output vector is the prediction 10-D vector. There are two convolution-pooling layers (32 and 64 neurons respectively) at the beginning of the structure. After that, the 7×7 feature maps are flattened to vectors. And then it comes a fully connected layer (from $7 \times 7 \times 64$ to 1024) with dropout. The last layer is another fully connected layer (from 1024 to 10) which provides the prediction vector.

The convolutional and the max-pooling layers will be discussed in detail in next subsections. Obtaining the prediction vector, we take the cross-entropy as the loss function

$$L\left(\mathbf{Y}, \hat{\mathbf{Y}}\right) = -\frac{1}{N} \sum_{n \in N} \sum_{i=1}^{10} Y_{n,i} \log \hat{Y}_{n,i}.$$

Applying the backpropagation [3] on the loss function by `AdamOptimizer` in `tensorflow`, an iteration of the training process is performed.

1.2 Convolutional Layer

The convolutional layer is the core building block of a convolutional network that does most of the computational heavy lifting.

First, we introduce the 2-D convolution

$$\mathbf{B} = \mathbf{A} * \mathbf{K}$$

where $\mathbf{A}$ is the original matrix, $\mathbf{K}$ is the kernel (point spread function) and $\mathbf{B}$ is the convolution result where

$$B_{i,j} = \sum_{u=1}^{m} \sum_{v=1}^{n} K_{u,v} \cdot A_{i+m-u-c_1+1, j+n-v-c_2+1}$$

where m and n are the size of the kernel $\mathbf{K}$, c_1 and c_2 are the defined center coordinates of the kernel (generally the center coordinates are the center point of the kernel, and thus m and n are odd numbers). For keeping the size of $\mathbf{B}$ the same as $\mathbf{A}$, we usually pad zeros around $\mathbf{A}$ before convolution.

For a convolutional layer (m inputs and n outputs), we have multiple ($m \times n$) kernel matrices, and the j-th output will be

$$\mathbf{O}_j = \sum_{i=1}^{m} \mathbf{I}_i * \mathbf{K}_{i,j}$$

where $\mathbf{I}_i$ is the i-th input and $\mathbf{K}_{i,j}$ is the kernel adopted from the i-th input to the j-th output.

1.3 Max-Pooling Layer

The pooling layer operates independently on every depth slice of the input and resizes it spatially, using the `MAX` operation in this model. The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding $\frac{3}{4}$ of the activations [2].

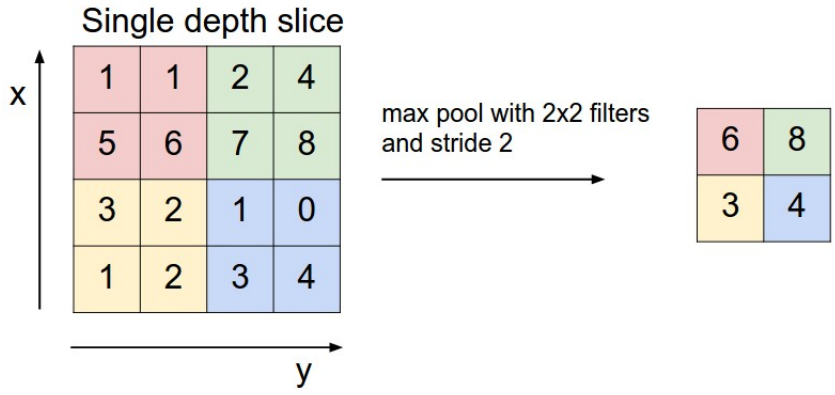

Figure 2: An Example of Max-Pooling

In the example showed above, the size of the filter is 2×2 and the stride is 2, and mathematically, the output matrix $\mathbf{O}$ of max-pooling layer is subject to

$$O_{i,j} = \max_{k,l \in \{1,2\}} (\mathbf{I}_{i,j})_{k,l}$$

where $\mathbf{I}_{i,j}$ is the submatrix (a 2×2 block in the example) at coordinates (i, j)

1.4 Implementation by `tensorflow`

We take the softmax cross entropy as the loss function, setting the max epochs to 2000, the learning rate to 10^{-4}, and the batch size to 100, using the adam optimizer, we have the running result as the picture shows below

```
0.048                    0.968
0.815                    0.967
0.885                    0.974
0.912                    0.973
0.93                     0.973
0.938                    0.974
0.956                    0.976
0.951                    0.979
0.956                    0.979
0.953                    0.981
```

Figure 3: The Screenshots of Running Result

And we have the accuracy and loss plot against epochs of training and testing process

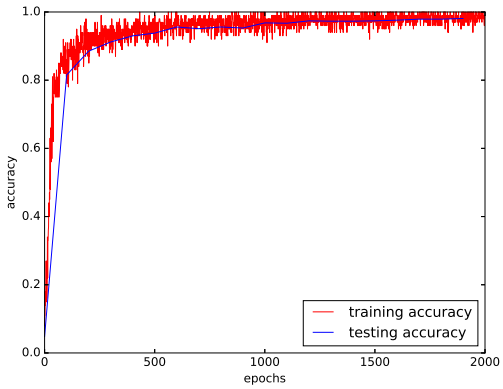

Figure 4: The Accuracy Plot against Epochs

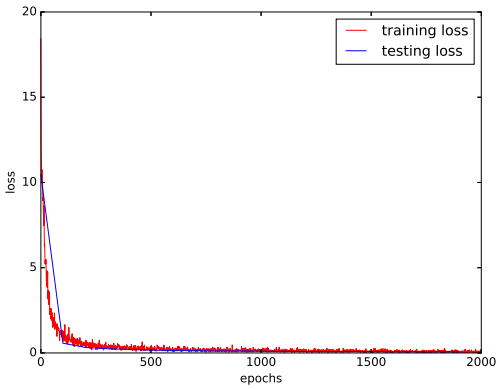

Figure 5: The Loss Plot against Epochs

As we observe from the running result picture and the plots, the convergence speed is extremely fast when the number epochs is less than around 100. This model shows a steady convergence state as the number of epochs reaches around 1200. The final testing accuracy is 0.981, of which the cross entropy loss is 0.061292458.

As we infer from the result above, this model's capability is much stronger than the fully connected model that the accuracy of CNN is approximately 8 percentage higher than FC. And it reached the convergent state without overfitting.

2 Conclusion

The classic CNN model showed a great performance on `MNIST`, which considerably improved the accuracy compared to the fully connected network we experimented on before. It showed a fast convergent trend (accuracy around 0.98) with the training processes pass, and it was not overfitting.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Jawad Nagi, Frederick Ducatelle, Gianni A Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber, and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pages 342–347. IEEE, 2011.

[3] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.