Exercise 1

MNIST handwritten digit regconition using back-propagation and convolutional neural network

Student: Khuong G.T Diep

Advisor: Prof. Yong-Guk Kim

October 14, 2022

1 Introduction

The main point of this report is to compare the performance between the 7-layer convolutional neural network and 3-layer back-propagation network. For this project, the MNIST database of handwritten digits, which has a training set of 60,000 examples and a test set of 10,000 examples, is used for this problem. The digits have been size-normalized and centered in a fixed-size image[1].

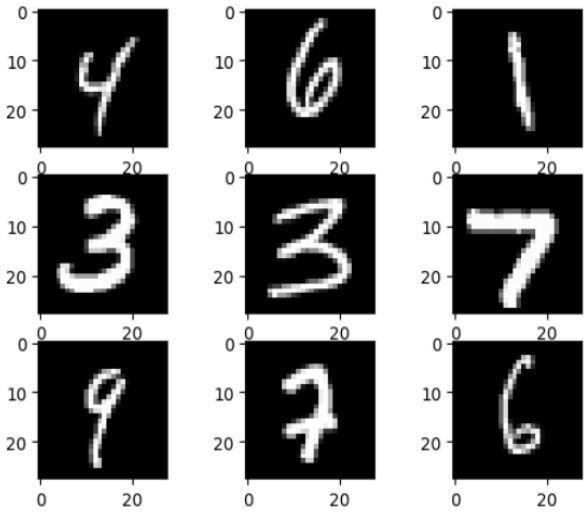

Figure 1: MNIST handwritten digit dataset

2 Methods

2.1 Backpropagation

Back-propagation is a process that repeatedly updates the network's weight values in an attempt to minimize the difference between the network's actual output vector and the desired output vector [2]. Backpropagation is a popular approach in machine learning for training feedforward neural networks.

2.2 Convolutional neural network

A convolutional neural network (CNN) is a type of artificial neural network used in deep learning (ANN). It is similar to conventional ANNs in that it is made up of neurons that self-optimise through learning. Each neuron will continue to receive input and conduct an operation (such as a scalar product followed by a non-linear function), which is the foundation of many ANNs. The entire network will still express a single perceptive score function from the input raw picture vectors to the final output of the class score (the weight). The main significant difference between CNNs and normal ANNs is that CNNs are mostly utilized in visual pattern recognition [3].

2.3 Batch normalization

Batch normalization is a technique used between neural network layers that continuously normalizes the output from the previous layer before transferring it to the next layer. BN is used to stabilize the neural network, maintain data distribution, increase accuracy, and may be used to most models to improve the convergence rate of a neural network [4].

2.4 Activation function

In artificial neural networks, activation functions are utilized specifically to convert input signals into output signals that are then sent as input to the following layer in the stack. A neural network with no activation functions operates as a linear regression model with limited performance and power. A neural network sometimes has to do with more difficult tasks than only learn and compute a linear function, such as modeling complex types of data like images, videos, speech, etc. For this reason, large, high-dimensional, nonlinear datasets with a model that has several hidden layers and a complex structure should use activation functions [5].

The ReLU and Softmax are the two activation function which are used for my progam.

1. The ReLU known as rectified liner unit is a non-linear activation function which is widely applied in neural networks. ReLU is more effective than other functions because just a certain number of the neurons are stimulated at once rather than all of them simultaneously.

$$f(x) = max(0, x) \tag{1}$$

2. The Softmax function is a sigmoid function combination. A sigmoid function returns values in the range 0 to 1, which can be interpreted as probability of data points in a specific class. Unlike sigmoid functions, which are utilized for binary classification, the softmax function can be used for multiclass classification tasks.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for j} = 1, ..., \text{K} \tag{2}$$

3 Experiment

Some of the techniques used in the my program are referenced from the MNIST-Classify project of Lihao [6]. For further references, my program can be found on github by the following link: https://github.com/KhuongDiep911/MNIST-Digit-Recognition

3.1 Back-propagation network

This approach employs a back-propagation network (BN) with three fully connected layers. The input images are flattened before being fed into the first layer. In addition, each fully connected layer includes the batch normalization technique and the ReLU activation function. All parameters are then passed into the softmax activation function which converts a vector of numbers into a vector of probabilities.

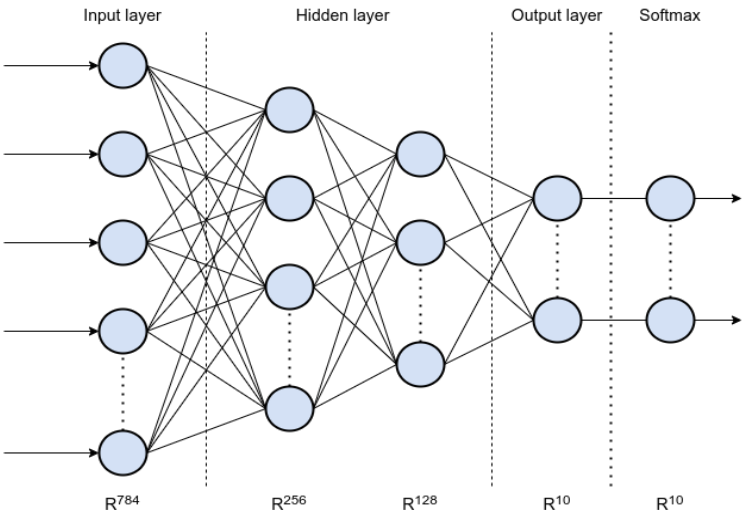

Figure 2: 3-layer back-propagation architecture

	Layer	Output Shape
Fully connected 1	Linear	[1, 256]
	BatchNorm1d	[1, 256]
	ReLU	[1, 256]
Fully connected 2	Linear	[1, 128]
	BatchNorm1d	[1, 128]
	ReLU	[1, 128]
Fully connected 3	Linear	[1, 10]

Table 1: Back-propagation network and output shape of each layers

3.2 Convolutional neural network

The architecture of the model consists of five convolution layers and two fully connected layers. First of all, the input images are fed into the convolution layer. After passing through each convolution operation, the image is abstracted to a feature map, also known as an activation map. In addition, each convolutional layer includes the batch normalization technique, the Max-pooling operation, and the ReLU activation function. Secondly, two fully connected layers are designed, which have the same structure as the back-propagation network described in Section 3.1. Finally, the softmax activation function is applied to convert a vector of numbers into a vector of probabilities.

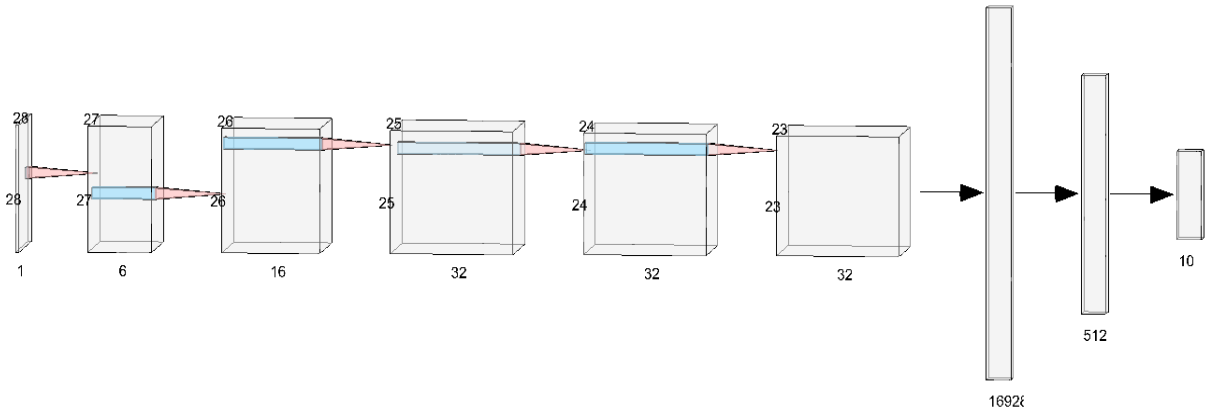

Figure 3: The 7-layer convolutional architecture

	Layer	Output Shape
Convolution 1	Conv2d	[6, 28, 28]
	BatchNorm2d	[6, 28, 28]
	ReLU	[6, 28, 28]
	MaxPool2d	[6, 27, 27]
Convolution 2	Conv2d	[16, 27, 27]
	BatchNorm2d	[16, 27, 27]
	ReLU	[16, 27, 27]
	MaxPool2d	[16, 26, 26]
Convolution 3	Conv2d	[32, 26, 26]
	BatchNorm2d	[32, 26, 26]
	ReLU	[32, 26, 26]
	MaxPool2d	[32, 25, 25]
Convolution 4	Conv2d	[32, 25, 25]
	BatchNorm2d	[32, 25, 25]
	ReLU	[32, 25, 25]
	MaxPool2d	[32, 24, 24]
Convolution 5	Conv2d	[32, 24, 24]
	BatchNorm2d	[32, 24, 24]
	ReLU	[32, 24, 24]
	MaxPool2d	[32, 23, 23]
Fully connected 1	Linear	[1, 512]
	BatchNorm1d	[1, 512]
	ReLU	[1, 512]
Fully connected 2	Linear	[1, 10]

Table 2: Convolutional neural network and output shape of each layers

4 Results and conclusion

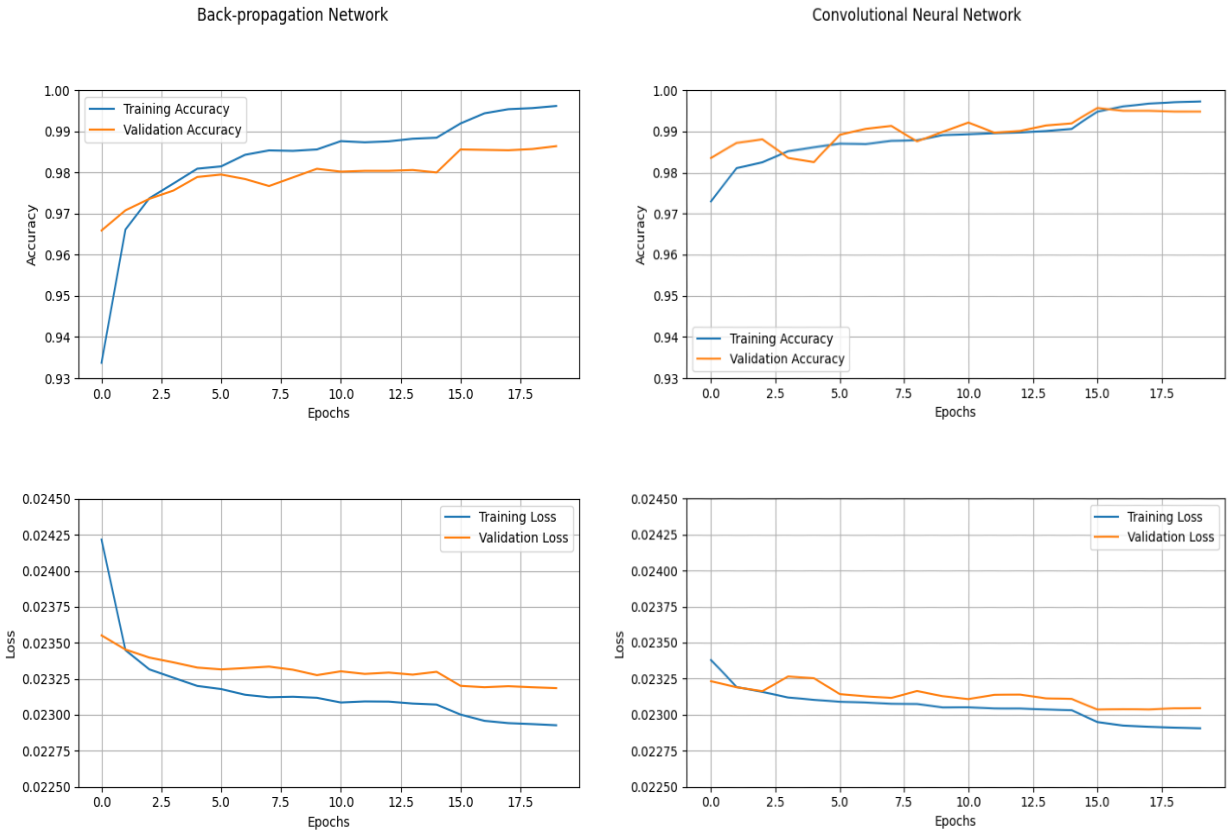

Figure 4: Accuracy and loss of the two models

As seen in Fig. 4, the CNN's initial accuracy and loss values appear to be better than the BN's values. During the training process, the BN model's gap between training and validation for both accuracy and loss values grows dramatically, but the CNN model's gap is tiny. When the training procedure is completed, the CNN model has higher accuracy and lower loss for both training and validation than the BN model.

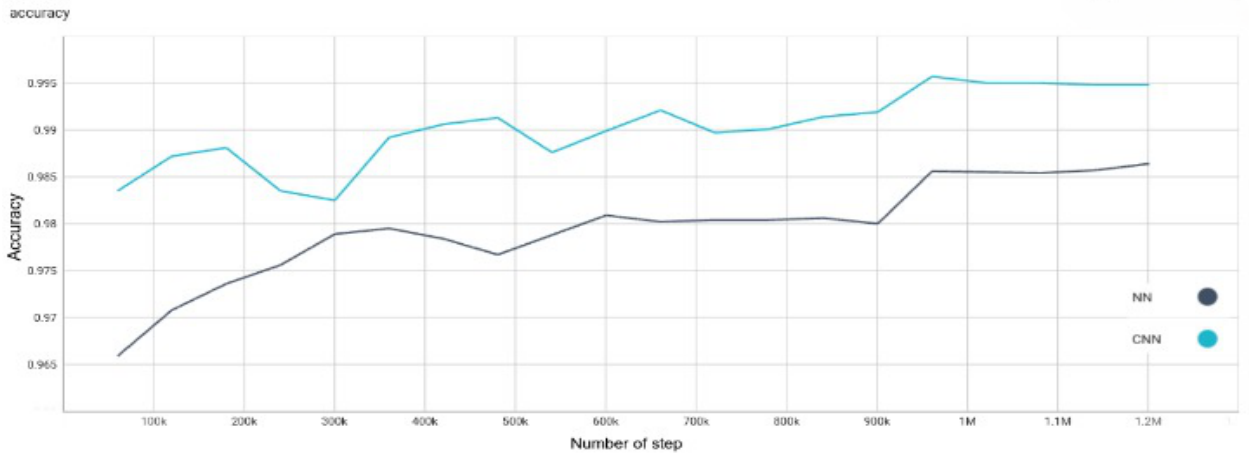

Figure 5: A comparison of the accuracy during training of the two methods

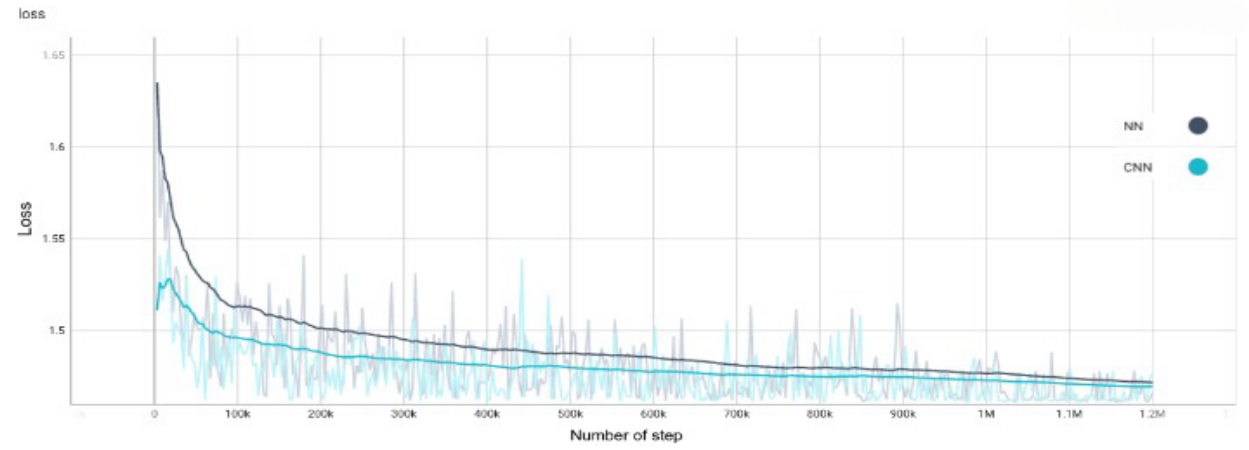

Figure 6: A comparison of the loss during training of the two methods

Method	Result
NN	0.986
CNN	0.996

Table 3: Results of the two models on the test dataset

Fig. 5 and 6 indicate that the CNN outperforms the BN during the training phase, with higher accuracy and lower loss. The results on the test dataset, moreover, revealed that the CNN model with an accuracy of 0.996 beats the BN model with an accuracy of 0.986 for the handwritten digit recognition task.

References

[1] Y. LeCun. "The mnist database of handwritten digits." (), [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[3] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: 1511.08458 [cs.NE].

[4] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/36072923bfc3cf47745d704feb489480-Paper.pdf.

[5] S. S. Siddharth Sharma and A. Athaiya, "Activation functions in neural networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 4, pp. 310–316, 2020.

[6] Lihao, *Mnist-classify*, 2021. [Online]. Available: https://github.com/lh9171338/MNIST-Classify.