A Convolutional Neural Network **without** AI Frameworks

Tianyu Guo
Sun Yat-sen University
guoty9@mail2.sysu.edu.cn

***Abstract*—Today, artificial intelligence(AI) has become an irresistible historical trend. Many AI-assisted learning frameworks have also emerged, such as Tensorflow, Keras, Pytorch, and Caffe. These tools greatly simplify the cost of AI learning and research. However, while they bring convenience, people's understanding of some underlying knowledge of artificial intelligence is very insufficient. This is the main reason why the barriers to learning AI are relatively low and many people are flocking to this industry. Therefore, this paper will build and train a convolutional neural network to recognize handwritten numbers without the aid of any machine learning library. After about 7 minutes of training, about 98% accuracy on the test set was achieved.**

I. Introduction

In recent years, the development of artificial intelligence in the field of science and technology is also obvious to all. From the controversy caused by the development of driverless cars to AlphaGo's victory [3] over the top Go player, artificial intelligence has attracted enough attention. Machine learning, a branch of artificial intelligence, has attracted wide attention, and deep learning, a branch of machine learning, has become a research hotspot in recent years.

Artificial intelligence has had its decline. But after a brief dip, there was a boom. So what exactly does AI rely on to turn things around? That's rapid iterations of hardware and software. Of course, the most important thing is its excellent problem solving ability. But imagine that without the computing power of current hardware and the iterative updates in software, there would be no AI today.

The software accelerator of AI is numerous learning frameworks, like TensorFlow [1], Pytorch [8], Keras [5], Caffe [4] and so on. They can help us build neural networks like building blocks, and can realize forward propagation, automatic differentiation and back propagation. In addition, they also do a lot of automatic processing of the details of neural network construction and training, such as what activation function should be applied, how to initialize the parameters, how to dynamically set the learning rate, what update algorithm is applied to the gradient of back propagation and so on. Although this brings great convenience to the study and research of neural network, it also makes people lack of understanding of the underlying knowledge of neural network.

This paper makes the following contributions:

- I use C++ language to build and train a convolutional neural network [1] to recognize handwritten digits without calling any machine learning library functions. About 98% accuracy was achieved in the test set (10,000 test images in MNIST) after a training round of about 7 minutes (60,000 training images in MNIST).
- The variation rule of data dimension in forward propagation is explained in detail.
- The method of calculating and updating the gradient of each layer in back propagation is introduced in detail.
- Finally, some other details of neural network training, such as parameter initialization and learning rate selection, are also introduced.

The rest of the paper is organized as follows: Section II introduces the background and history about AI and its accelerators. Section III elaborate the built Convolutional Neural Network. Section IV compares the convolutional neural network constructed in this paper with Pytorch.

II. Background

A. History of artificial intelligence

Artificial intelligence has been formally proposed since the 1950s and 1960s. In 1950, a senior student named Marvin Minsky and his classmate Dunn Edmond built the world's first neural network computer, and this computer also showed the beginning of computing. Coincidentally, in 1950, Alan Turing also came up with a remarkable idea called the Turing Test [2], in which he assumed that a computer would be an intelligent machine if it could converse with a human being without being identified as a machine. During the course of the year Alan had boldly predicted the feasibility of intelligent machines. In 1956, at a conference, the term artificial intelligence was formally introduced by computer expert John McCarthy. Later, this behavior was considered to be the official birth of artificial intelligence.

During this period of ten years, computers were widely used in mathematics and natural languages, mainly solving algebra and geometry problems. This situation has made many scholars see the machine to artificial intelligence development confidence. Even more, several scholars at the time believed that within twenty years machines would be able to do everything humans could. In the 1970s, artificial intelligence entered its first trough, because researchers' estimation of the difficulty of the project was wrong in the research on artificial intelligence, which not only led to the failure of the cooperation plan, but also cast a shadow on the development of

[1]The work was public at https://github.com/gty111/ConvNN

Fig. 1. GTX580

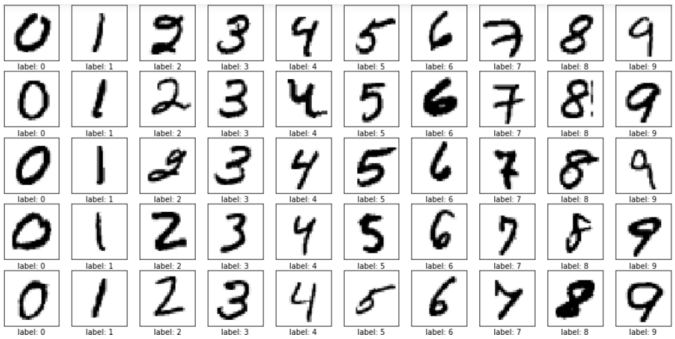

Fig. 2. Part of MNIST

artificial intelligence. At the same time, the public opinion is also slowly pressure, which is the loss of most research funds. At that time, artificial intelligence faced three technical dilemmas: First, the performance of the computer could not meet the requirements, which would cause many early programs could not be used in the field of artificial intelligence; Second, the problem is relatively complex, the early artificial intelligence is aimed at specific problems, because the specific problems are usually few and the complexity is very low, but as long as the difficulty of the problem increases, the program will be overwhelmed; Third, the amount of data was not enough. In those days, there were no large enough databases to support deep learning of programs, which would make it impossible for machines to read enough data to be intelligent.

In 2006, neural network expert Hinton proposed neural network deep learning algorithm [7], which greatly improved the capability of neural network and challenged the support vector machine. At the same time, it opened the wave of deep learning in academia and industry.

B. The rise of the GPU

Nvidia was founded in 1993. At first, GPUs were mainly used for graphics display adapters. However, with the development of artificial intelligence, Nvidia released the first computational graphics card, the GTX580 which AlexNet [6] is trained on. AlexNet is a great success in artifical intelligence. This success came from the efficient use of GPUs, ReLUs, a new regularization technique called dropout, and techniques to generate more training examples by deforming the existing ones. This success has brought about a revolution in computer vision; ConvNets are now the dominant approach for almost all recognition and detection tasks and approach human performance on some tasks.

The emergence of computable GPUs have led to a boom in machine learning. The booming development of machine learning feeds back into the GPU field. So machine learning and GPU are mutually reinforcing. And Nvidia's great strategic vision is doomed to its success. Today GPU is not only brilliant in the field of artificial intelligence, but also indispensable for people's entertainment life. It's even more important than CPU, and Nvidia's market cap proves it.

III. The Detail of Building and Training Convolutional Neural Network

I use MNIST (Fig.2) data set to be fed to the network. The overview of network's structure is shown in Fig.3. The meaning of symbol is shown in Tab.IV.

A. MNIST

MNIST data set comes from the American National Institute of Standards and Technology (NIST), which is a smaller version of NIST. The training set is composed of handwritten numbers from 250 different people. Fifty percent were high school students, 50 percent were from the Census Bureau staff, and the test set was a similar percentage of handwritten numerical data.

All the integers in the files are stored in the MSB first (high endian) format used by most non-Intel processors. Users of Intel processors and other low-endian machines must flip the bytes of the header.There are 4 files in MNIST which shown in Tab.I. Label file format is shown in Tab.II. The labels values are 0 to 9. Image file format is shown in Tab.III. Pixels are organized row-wise. Pixel values are 0 to 255. 0 means background (white), 255 means foreground (black).

TABLE I
MNIST Files

File name	description
train-images-idx3-ubyte	training set images
train-labels-idx1-ubyte	training set labels
t10k-images-idx3-ubyte	test set images
t10k-labels-idx1-ubyte	test set labels

TABLE II
TRAINING SET LABEL FILE (train-labels-idx1-ubyte)

offset	type	value	description
0000	32 bit integer	0x800000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

TABLE III
TRAINING SET IMAGE FILE (TRAIN-IMAGES-IDX3-UBYTE)

offset	type	value	description
0000	32 bit integer	0x800000803(2051)	magic number (MSB first)
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

TABLE IV
SYMBOL TABLE

NAME	description
fcw	weight of full connection layer
fcb	bias of full connection layer
convw	weight of convolution layer
convb	bias of convolution layer

B. Forward Propagation

1) $INPUT$*:* At first, the input of image will be normalized which means

$$INPUT_{(i,j,k)} = (INPUT_{(i,j,k)} - mean)/std \tag{1}$$

where $mean$ equals the mean of $INPUT$ and std equals the standard deviation of $INPUT$.

2) $INPUT \xrightarrow{conv1} A$*:* There are some default rules in convolution layer(with no padding).

$$\begin{aligned} IN.shape_{(0)} &= CONVW.shape_{(1)} \\ OUT.shape_{(0)} &= CONVW.shape_{(0)} \\ OUT.shape_{(1)} &= IN.shape_{(1)} - CONVW.shape_{(2)} + 1 \\ OUT.shape_{(2)} &= IN.shape_{(2)} - CONVW.shape_{(2)} + 1 \\ CONVB.shape_{(0)} &= CONVW.shape_{(0)} \end{aligned} \tag{2}$$

where $shape$ means the dimension of that tensor and index starts from 0.(Eg. in this paper IN.shape(0)=1 IN.shape(1)=28 IN.shape(2)=28)

$$A_{(i)} = convb1_{(i)} + \sum_j INPUT_{(j)} \otimes convw1_{(i,j)} \tag{3}$$

where $\otimes$ means operation of convolution. Mind that $A_{(i)}$ is a tensor whose shape is (26,26), $INPUT_{(j)}$ is a tensor whose shape is (28,28) and $convw1_{(i,j)}$ is a tensor whose shape is (3,3).

3) $A \xrightarrow{relu} B$*:*

$$B = RELU(A) \tag{4}$$

where RELU is a common function shown in Fig.4.

4) $B \xrightarrow{conv2} C$*:*

$$C_{(i)} = convb2_{(i)} + \sum_j B_{(j)} \otimes convw2_{(i,j)} \tag{5}$$

5) $C \xrightarrow{relu} D$*:*

$$D = RELU(C) \tag{6}$$

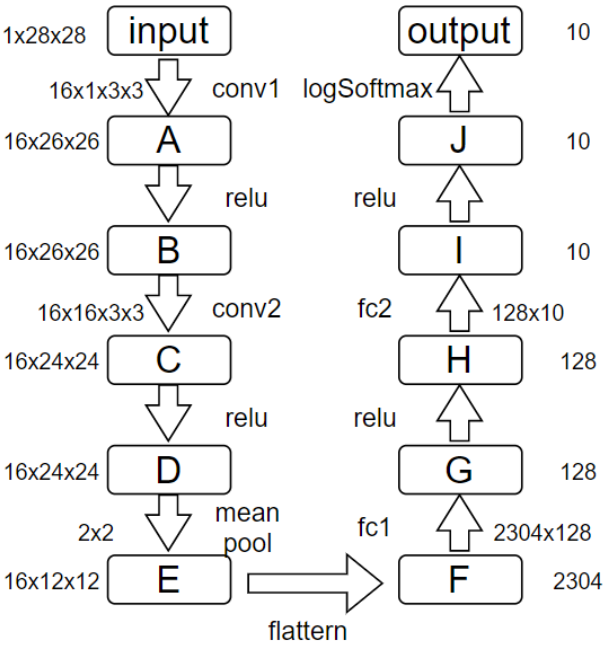

Fig. 3. Network Structure

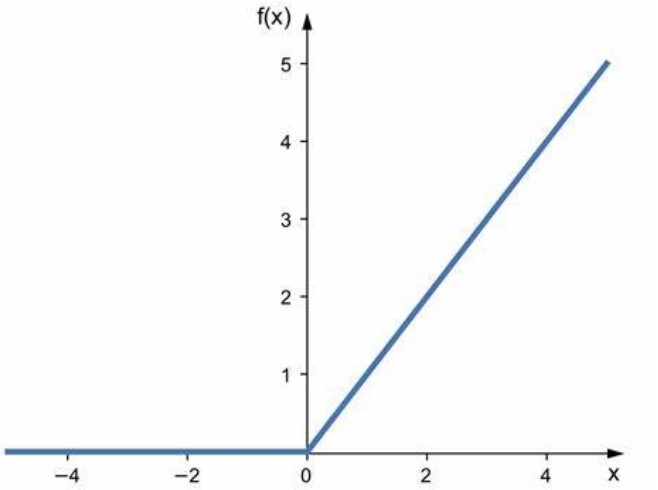

Fig. 4. RELU activation function

6) $D \xrightarrow{meanpool} E$*:*

$$E_{(i,j,k)} = (\sum_{m=j*2}^{j*2+1} \sum_{n=k*2}^{k*2+1} D_{(i,m,n)})/4 \tag{7}$$

7) $E \xrightarrow{flattern} F$*:*

$$F_{(i*12*12+j*12+k)} = E_{(i,j,k)} \tag{8}$$

8) $F \xrightarrow{fc1} G$*:*

$$G_{(i)} = fcb1_{(i)} + \sum_j F_{(j)} * fcw1_{(j,i)} \tag{9}$$

There are some default rules in full connection layer.

$$\begin{aligned} fcw.shape_{(1)} &= fcb.shape_{(0)} \\ IN.shape_{(0)} &= fcw.shape_{(0)} \\ OUT.shape_{(0)} &= fcw.shape_{(1)} \end{aligned} \tag{10}$$

9) $G \xrightarrow{relu} H$*:*

$$H = RELU(G) \tag{11}$$

10) $H \xrightarrow{fc2} I$*:*

$$I_{(i)} = fcb2_{(i)} + \sum_j H_{(j)} * fcw2_{(j,i)} \tag{12}$$

11) $I \xrightarrow{relu} J$*:*

$$J = RELU(I) \tag{13}$$

12) $J \xrightarrow{logSoftmax} OUTPUT$*:*

$$OUTPUT_{(i)} = log_e(\frac{e^{J_{(i)}}}{\sum_k e^{J_{(k)}}}) \tag{14}$$

where e means natural logarithm.

C. Backward propagation

1) ΔJ*:*

$$\Delta J_{(i)} = \begin{cases} 1 - \frac{e^{J_{(i)}}}{\sum_k e^{J_{(k)}}} & i = label \\ -\frac{e^{J_{(i)}}}{\sum_k e^{J_{(k)}}} & i \neq label \end{cases} \tag{15}$$

2) ΔI*:*

$$\Delta I_{(i)} = \begin{cases} \Delta J_{(i)} & I_{(i)} > 0 \\ 0 & I_{(i)} \leq 0 \end{cases} \tag{16}$$

3) $\Delta fcw2$*:*

$$\Delta fcw2_{(i,j)} = H_{(i)} * \Delta I_{(j)} \tag{17}$$

4) $\Delta fcb2$*:*

$$\Delta fcb2_{(i)} = \Delta I_{(i)} \tag{18}$$

5) ΔH*:*

$$\Delta H_i = \sum_j \Delta I_{(j)} * fcw2_{(i,j)} \tag{19}$$

6) ΔG*:*

$$\Delta G_{(i)} = \begin{cases} \Delta H_{(i)} & G_{(i)} > 0 \\ 0 & G_{(i)} \leq 0 \end{cases} \tag{20}$$

7) $\Delta fcw1$*:*

$$\Delta fcw1_{(i,j)} = F_{(i)} * \Delta G_{(j)} \tag{21}$$

8) $\Delta fcb1$*:*

$$\Delta fcb1_{(i)} = \Delta G_{(i)} \tag{22}$$

9) ΔF*:*

$$\Delta F_{(i)} = \sum_j \Delta G_{(j)} * fcw1_{(i,j)} \tag{23}$$

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

⇒

(2,2)	(2,1)	(2,0)
(1,2)	(1,1)	(1,0)
(0,2)	(0,1)	(0,0)

Fig. 5. ROT operation

origin ⇒

0	0	0	0	0
0	0	0	0	0
0	0	origin	0	0
0	0	0	0	0
0	0	0	0	0

Fig. 6. PAD operation

10) ΔE*:*

$$\Delta E_{(i,j,k)} = \Delta F_{(i*12*12+j*12+k)} \tag{24}$$

11) ΔD*:*

$$\Delta D_{(i,j,k)} = \Delta E_{(i,j/2,k/2)}/4 \tag{25}$$

12) ΔC*:*

$$\Delta C_{(i,j,k)} = \begin{cases} \Delta D_{(i,j,k)} & C_{(i,j,k)} > 0 \\ 0 & C_{(i,j,k)} \leq 0 \end{cases} \tag{26}$$

13) $\Delta convw2$*:*

$$\Delta convw2_{(i,j)} = B_{(j)} \otimes \Delta C_{(i)} \tag{27}$$

14) $\Delta convb2$*:*

$$\Delta convb2_{(i)} = \sum_j \sum_k \Delta C_{(i,j,k)} \tag{28}$$

15) ΔB*:*

$$\Delta B_{(i)} = \sum_j PAD(\Delta C_{(j)}) \otimes ROT(convw2_{(j,i)}) \tag{29}$$

where PAD operation is shown in Fig.6 and ROT operation is shown in Fig.5.

16) ΔA*:*

$$\Delta A_{(i,j,k)} = \begin{cases} \Delta B_{(i,j,k)} & A_{(i,j,k)} > 0 \\ 0 & A_{(i,j,k)} \leq 0 \end{cases} \tag{30}$$

17) $\Delta convw1$*:*

$$\Delta convw1_{(i,j)} = INPUT_{(j)} \otimes \Delta A_{(i)} \tag{31}$$

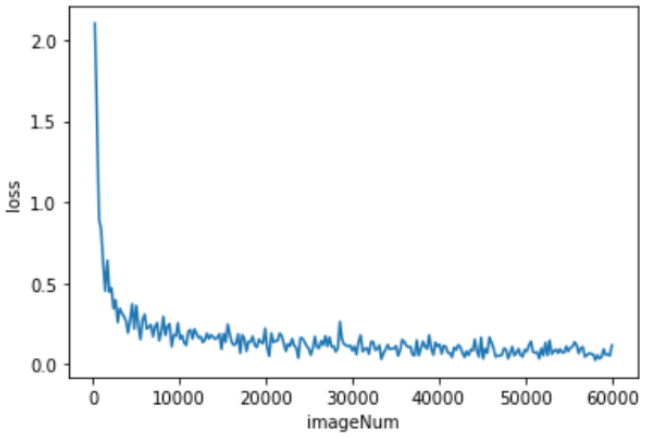

Fig. 7. Loss while training

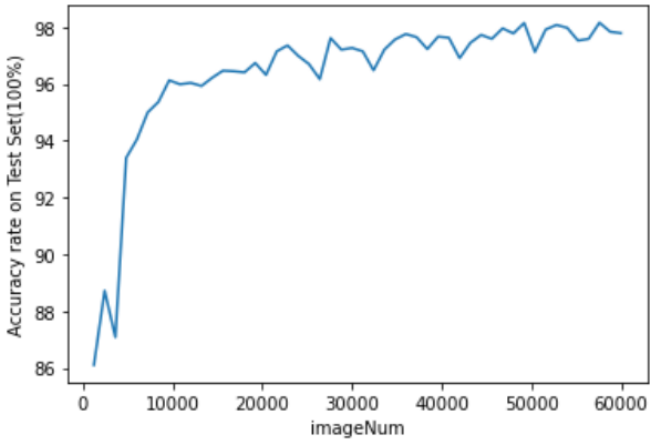

Fig. 8. Accuracy rate on test set while training

18) $\Delta convb1$:

$$\Delta convb1_{(i)} = \sum_{j}\sum_{k} A_{(i,j,k)} \tag{32}$$

D. Other details about ConvNN

The loss of built ConvNN is defined as follows

$$Loss = OUTPUT_{(label)} \tag{33}$$

Due to the $LogSoftmax$ operation, OUTPUT will all be negative. And the closer the loss is to zero, the better result is. The model's parameters are updated in a way as follows

$$P_{i+1} = P_i + \Delta P_i * lr \tag{34}$$

where lr means learning rate of the model. In this paper, learning rate is set to $5e^{-3}$. And the model is trained with fixed learning rate. The model parameters were initialized with a normal distribution with a mean of 0 and a standard deviation of 0.1.

E. Evaluation

From Fig.7 and Fig.8, we can see the convergence of the model is fast at the beginning, and then gradually becomes slower. So dynamically updating learning rate is future work. After 10000 images' training, the model can achieve 96% accuracy. This shows that the model does not need a lot of training data to achieve good results.

IV. Relative work

Pytorch gives an example to build convolution neural network on MNIST. The structure of that network is a little different from this paper which is the number of kernel of convolution. Tab.V gives a comparison with Pytorch. And the biggest difference is in training time. I guess Pytorch use openmp to accelerate the training time. This is left to the future work.

TABLE V
Comparison with Pytorch

metric	this paper	Pytorch
training time(one epoch)	about 7min	about 1min
accuracy rate on test set(after one epoch)	about 98%	about 98%

V. Conclusion

This paper introduces and achieves the construction of a convolutional neural network, the process of forward propagation and back propagation in detail, and finally realizes the accuracy rate similar to Pytorch.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[2] B. J. Copeland, "The turing test," *Minds and Machines*, vol. 10, no. 4, pp. 519–539.

[3] S. R. Granter, A. H. Beck, and J. Papke, David J., "AlphaGo, Deep Learning, and the Future of the Human Microscopist," *Archives of Pathology Laboratory Medicine*, vol. 141, no. 5, pp. 619–621, 05 2017.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 675–678.

[5] N. Ketkar, *Introduction to Keras*. Berkeley, CA: Apress, 2017, pp. 97–111. [Online]. Available: https://doi.org/10.1007/978-1-4842-2766-4_7

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, may 2017.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444.

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.