Achieving Handwritten Number Recognition through Deep Learning Strategies based on Conventional Neuron Networks

Kailiang Liu *

Abstract

Recognizing handwritten numbers is a common function of modern electronic devices that aids us with great convenience in aspects such as typing and vehicle identification. Aiming to simulate this function, this paper trains an algorithm that is based on Convolutional Neuron Network structure, a prevailing method for picture classification in Deep Learning, and identifies pictures of handwritten numbers by converging them to matrices. Later trials proves that such a model achieves a reliably high accuracy of over 98% out of the dataset for test use, and is thus capable of accomplishing normal number classification.

1 Introduction

Numbers are important concepts in both daily life and the field of computer science. Humans have been using numbers as tools far before the emergence of the first civilizations, and till today the functions of numbers have covered almost all aspect of life, including counting, trading, grading, using as ID proof, etc. The field of computer science, on the other hand, is entirely relying processing statistics, which, essentially speaking, are strings consists of zeros and ones. Nevertheless, though both being numbers, numbers written down cannot be directly processed by machines as statistics. To converge written numbers into electronic statistics, for decades people would have to manually input the written numbers into the computer, which costs a great amount of time and efforts. Fortunately, with the development of machine learning, handwriting classification technology has emerged, and today it has become a basic function for any smart device: numbers written with hands on smartphones almost instantly turns to be texts in electronic documents, and cameras can now easily extract the plate numbers of cars and make a digital list to keep track on. Such a technology significantly enhances the efficiency of information exchange, and this paper is proposing a possible that make this breakthrough obtainable.

*Advised by: Guillermo Goldsztein

2 Methods

Handwritten numbers identification, from the aspect of Machine Learning, can be attributed to the genre of classification problems, making neuron network a suitable model type for data procession here. As inputs of this problem begins with raw pictures, we introduce Convolutional Neuron Network to achieve better feature extraction. The following are explanations of computer science concepts and causal relationships involved in the problem-solving process above.

3 Classification problems

Machine Learning addresses to a branch of computer science that use statistics to train models capable of making predictions and instructing decisions [NAS+20]. General Machine Learning problems can be divided into three sections, among which the classifications problems model on discrete dataset illustrated with labels, a part in each line of data that has been given special realistic meaning. For identifying handwritten numbers, each picture is processed as a matrix, in which the elements representing pixels are undoubtedly discrete. As the digits used only ranges from zero to nine, there are exactly ten possible labels for any picture containing a written number. These two features of number identification perfectly match the description of common classification problems, and it is thus that solution methods to common classification problems can also be used in our problem.

4 Conventional Neuron Network

As a common models for classification problems, a conventional neuron network consists of three parts: the input layer, the hidden layer(s), and the output layer [MCZ+21]. Designed in resemblance to the cognition process of human brain, each layers contains numerous "neurons" named nodes, and nodes in neighboring layers all connect with each other.

Each connection between nodes in neighboring layers represents a linear relationship containing two parameters and b preset by the model builder, through which the value at any node can be expressed as

$$o_k = \phi(\sum_{j=1}^{m} \omega_{jk} x_{jp} + b_{jp}) \tag{1}$$

where m presents the number of nodes in the previous layer, $_{jk}$ marks the weight parameter connecting the j-th node in the previous layer and the k-th node in current layer, o_{jp} suggests the value of the j-th node in the previous layer, and o_k represents the value of the k-th node in current layer.

Notice that each value in any nodes not belonging to the input layer can be seen as the output of its connected nodes in the previous layer, so literally every value in the entire neuron network can be obtained given the initial input

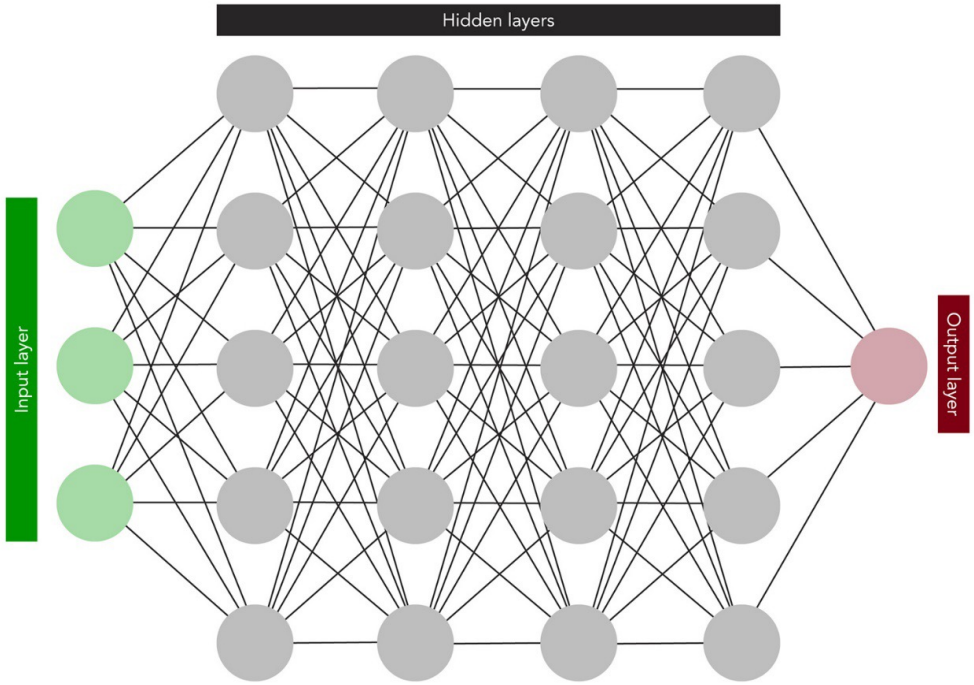


Figure 1: A basic graph indication of the structure of a neuron network

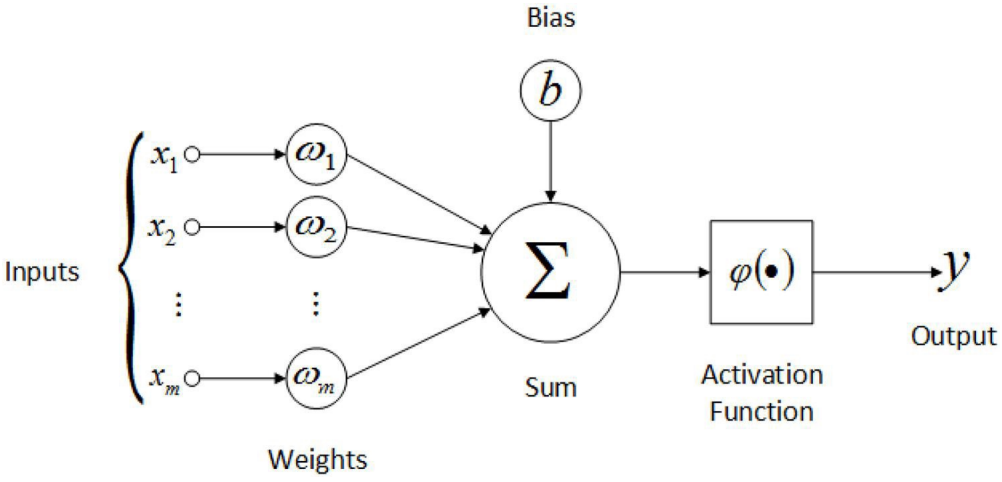


Figure 2: A graph indication of the calculation of output within a conventional neuron network

layer. After adding up the weighted inputs, the sum needs to first go through

a nonlinear activation function to finally obtain the value of output. This is mainly because that a linear relationship lacks of the flexibility for fitting random features. A commonly used activation function is the sigmoid function, whose expression is given as follows:

$$\phi(x) = \frac{1}{1+e^{-x}} \tag{2}$$

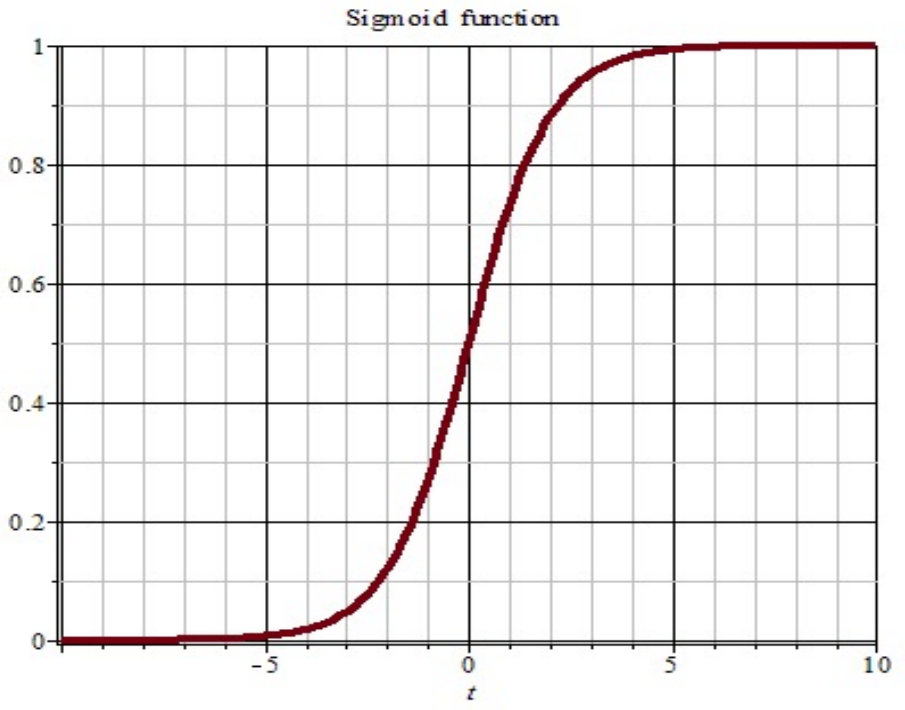


Figure 3: A part of the graph for the sigmoid function

Despite the excellent differentiability of the sigmoid function, its range of output also perfectly matches with the range of possibility, making it an ideal activation function for classification problems. Still, when $|\sum_{j=1}^{m}(\omega_{jk}x_{jp}+b_{jp})|$ appears to be larger, the sigmoid function is less sensitive to change in its input.

As is mentioned before, all weight parameters ω and bias parameters b are initially set solely relying on the instinct of the modeler. Whether these parameters actually fit the prediction model depends on the error values at corresponding output points, which is obtained through loss function involving the final output y and the given label $y^{'}$ of statistics used. Taking the sum of absolute values of differences or the standard deviation as error would damage the differentiability of the loss function. As a result, the expression for loss function at a certain point resembles to that of calculating variance

$$E(k) = (y_k - y_k^{'})^2 (3)$$

in which y_k represent the value of the k-th node in the output layer.

To successfully build a fitting model, the major task for the neuron network is to adjust the values of ω_k till the values of $E(k)$ become minimized. This goal is achieved through the process named back propagation, with Gradient Descend being its core method.

When attempting to get downhill, one descends most efficiently by following the steepest road down [WMG+17]. Gradient Descend use the principles in finding the minimized value of $E(i)$: it takes the derivative of $E(k)$ to ω_k at current point, and then change ω_i sightly to proceeds to the negative direction of the derivative on the graph of $E(k)$, which is commonly multidimensional, then repeat the above process over and over until the value of derivative approach zero.

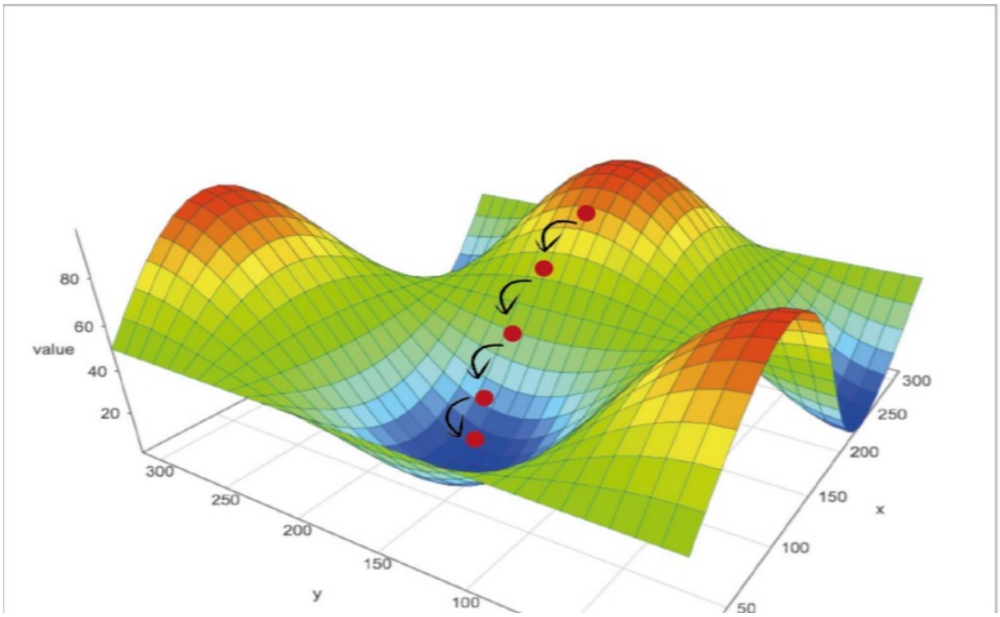


Figure 4: A picture illustrating how Gradient Descend reach the minimum of a 3D function

Taken the sigmoid function as activation function and $E(k) = (y_k - y_k^{'})^2$ as loss function at an output point, the change in the weight parameter would be

$$\Delta\omega_{jk} = \alpha \cdot E(k) \cdot y_k(1 - y_k) \cdot o_{jp} \quad (4)$$

With α represents the learning rate, a preset constant that control the magnitude of $\Delta\omega_{jk}$ to prevent it from oscillation or remaining at a local minimum on the graph of $E(k)$.

In addition, despite $E(k) = (y_k - y_k{}')^2$, another expression for the loss function at a certain node can be express as

$$E(jp) = \sum_{k=1}^{n} \omega_{jk} E(k) \tag{5}$$

where $E(jp)$ represents the theoretical error at the j-th node in the previous layer. Basically, this new expression considers the final error as the accumulation of errors in previous layers, and distributes the ultimate value of loss function at the output layer to the entire neuron network through weight parameters. It is therefore true that all nodes in the neuron network have a reachable theoretical error, meaning that the equation for Gradient Descend above can be expanded to a form suitable for any nodes within the network, which is

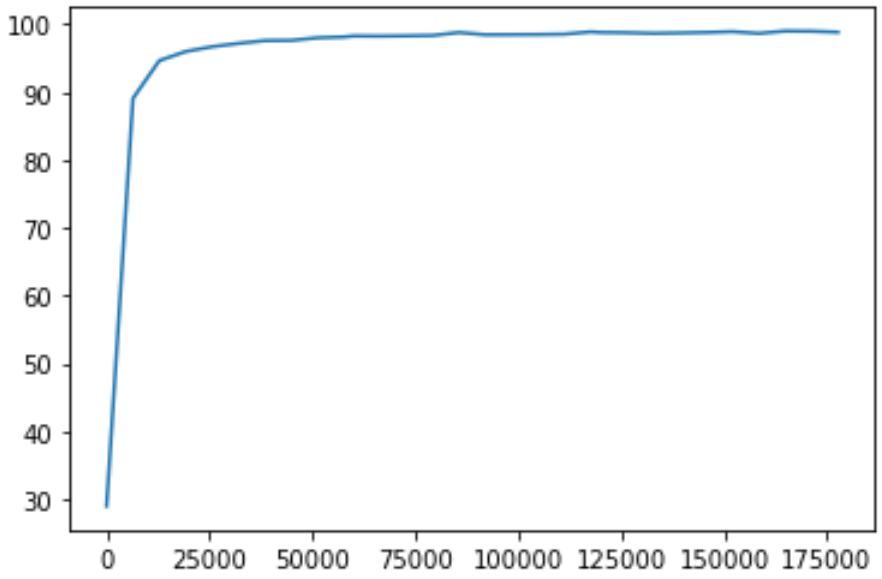
$$\Delta\omega_{jk} = \alpha \cdot E(k) \cdot o_k(1 - o_k) \cdot o_{jp} \tag{6}$$

By repetitively using this function to adjust weight parameters between neighboring layers, the neuron network will eventually evolve into a model system that is capable of yielding

5 Implementation

In normal steps, for convenience in processing dataset, it is firstly necessary to scale each picture into a unified size, namely 28 pixels both in length and width. Then the scaled pictures are supposed to be converge into long arrays, with each of the colored pixels extracted as a feature that range from 0 to 255. Nonetheless, as the MNIST.csv imported from Torchview already offer us with preprocessed data, this dataset can be directly processed by the CNN.

To fully utilize the given dataset, we split it into the training set and the inference set in the proportion of 4:1. The range of the digits of number suggests that there should be 10 outputs in our CNN, each suggesting the possibility of the tested picture being one of the numbers from 0 to 9. With the kernel initiated, the training set is run for 3 epochs at the speed of 64 pictures per batch. At the end of 3 full rounds of training, the CNN have achieved an accuracy of 98.90% within the inference dataset, exceeding the expectation for handwritten number classification



6 Conclusion

With the aid of Convolutional Neuron Network, the model in this paper have achieved an accuracy of 98.90% in identifying handwritten numbers, successfully simulating the handwritten number recognition function found in modern smart devices. This success, while innovating a way for a significant technological breakthrough, emphasizes the possibility lies within Deep Learning. It could be expected that in the near future, studies in Deep Learning will be able to create machines that match or even exceed human intelligence [MKS$^+$15].

Acknowledgement

I would like to thank Professor Guillermo Goldsztein for guiding me through the research process and teaching me how to write a research paper.

References

[MCZ+21] Mahdi Mahdavi, Hadi Choubdar, Erfan Zabeh, Michael Rieder, Safieddin Safavi-Naeini, Zsolt Jobbagy, Amirata Ghorbani, Atefeh Abedini, Arda Kiani, Vida Khanlarzadeh, et al. A machine learning based exploration of covid-19 mortality risk. *Plos one*, 16(7):e0252384, 2021.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[NAS+20] Maria Nicola, Zaid Alsafi, Catrin Sohrabi, Ahmed Kerwan, Ahmed Al-Jabir, Christos Iosifidis, Maliha Agha, and Riaz Agha. The socio-economic implications of the coronavirus pandemic (covid-19): A review. *International journal of surgery*, 78:185–193, 2020.

[WMG+17] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *Advances in neural information processing systems*, 30, 2017.