Applying Gradient Descent in Convolutional Neural Networks

Nan Cui¹, *

¹Electrical and Computer Engineering, University of Massachusetts Lowell, Lowell, 01854, United States

*834450215@qq.com

Abstract. With the development of the integrated circuit and computer science, people become caring more about solving practical issues via information technologies. Along with that, a new subject called Artificial Intelligent (AI) comes up. One popular research interest of AI is about recognition algorithm. In this paper, one of the most common algorithms, Convolutional Neural Networks (CNNs) will be introduced, for image recognition. Understanding its theory and structure is of great significance for every scholar who is interested in this field. Convolution Neural Network is an artificial neural network which combines the mathematical method of convolution and neural network. The hieratical structure of CNN provides it reliable computer speed and reasonable error rate. The most significant characteristics of CNNs are feature extraction, weight sharing and dimension reduction. Meanwhile, combining with the Back Propagation (BP) mechanism and the Gradient Descent (GD) method, CNNs has the ability to self-study and in-depth learning. Basically, BP provides an opportunity for backward-feedback for enhancing reliability and GD is used for self-training process. This paper mainly discusses the CNN and the related BP and GD algorithms, including the basic structure and function of CNN, details of each layer, the principles and features of BP and GD, and some examples in practice with a summary in the end.

1. Introduction

Human beings have a complex visual recognition system. People distinguish and classify objects independently. Whenever people see an object or scene, it is a picture that is reflected in the brain; then the brain will go through a series of operations and processing to tell people what was seen. The whole process is object recognition. For the computer field, object detection is a very large area of research. This piece of content involves neural networks and machine learning, and deep learning also helps people build more rigorous and efficient identification algorithms. Artificial neural network refers to a computer algorithm that is generated based on animal nerve propagation, which is mainly used to improve the computing speed. From the 1960s, many scientists have begun to study the neural network [1, 2, 3, 4]. Until 1986, with the use of the Back Propagation mechanism (BP), the neural network began to have practical significance at the practical level [5, 6]. Until today, the neural network has become the core algorithm in machine learning and in-depth learning to solve problems such as picture recognition, pattern recognition, speech recognition, natural language processing and video processing [7]. Convolutional Neural Networks (CNNs) is one of the most common algorithms in neural networks. Its biggest characteristics are feature extraction, weight sharing and dimension reduction. The Convolutional Neural Network utilizes the mathematical properties of the convolution, extracts the useful information from the target picture, as well as the propagation characteristics of the neural network, to finally output the result through the classifier. The convolutional neural network
combines a variety of algorithms and has the ability of self-machine learning. Nowadays, it has become the most crucial neural network algorithm in depth learning. At the same time, with the development of new technology, Convolutional Neural Network has also been strengthened, where Region with CNN (R-CNN), Fast R-CNN, and Faster R-CNN are the representatives. With the development and progress of CNN, it has been widely used in many fields. To better understand the structure and algorithm of CNN, this paper will focus on CNN itself and its associated BP algorithm and Gradient Descent (GD). The article then further introduces each level of CNNs, mainly including its structure and function. Detailed information about BP and GD is presented as well. The last part is the practical examples and the summary.

2. Convolutional Neural Networks
The Convolutional Neural Networks (CNNs) is a kind of mathematical structure for analysis datasets, images and so on. The primary function of CNNs is extracting features from samples with different requests in fast speed.

2.1. Layers
The Convolutional Neural Network has so many layers which like a long shelf. There are enough quantity computing units or elements in each layer, dealing with datasets at the same time. Computing units or elements in the same layer have the same function working on input data.

Although the CNN is a complex structure, it has three kinds of main layers: convolutional layers, pooling layers, and output layers. Firstly, the features of input data transmitted into the convolution layer will be extracted, and also the size of input data is shrunk. Then, after passing through several convolution layers, the data will be delivered to pooling layers, for enhancing features, and the data would be truncated again. Finally, results of CNNs come out from the output layers. The following will discuss the three layers respectively.

2.1.1 Convolutional Layers. Here is a single computing unit of a convolutional layer, as shown in Figure 1.

“Xn” refers to the input dataset. The circle is the computing unit which includes a weighted function in it. The weighted function can be regarded as a filter. The equation of the filter is following.

\[ h_{w,b}(x) = f(W^Tx) = f(\sum_{t=1}^{3} W_t x_t + b) \]  

In the equation (1), “W” is weighted functions designed by programmers, for some time it is a “convolutional window”, which is a small matrix and elements of the matrix are depended on processes requests. “b” is a fixed element belonged to weighted function, which can be removed sometimes.

Then, if there is a two-layer structure, the first layer can be seen as the input items and the second layer is a convolutional layer, which is shown in Figure 2.
Figure 2. Two-Layer Basic Convolutional Network

In the second layer, each circle has a different weighted function. Therefore, every piece of input data should be calculated with all weighted functions in layer 2, and transmit the results to the layer 3, which is called fully connected convolutional layer. The expressions in details are followings:

\[
\begin{align*}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{align*}
\]

The fully connected convolutional layer can guarantee all information can be transmitted precisely. However, it also produces huge computational burden. For instance, there are 10000 pieces of data as input units and 100 computed units in a convolutional layer at second row in the network. Then, the calculation will be about \(10^6\) times, from layer 1 (input layer) to layer 2.

To save operating time and cost, the input items will be passed through a part of convolutional layers units instead of all convolutional units, which is called non-connection convolutional layer. While non-connection convolutional layers would lose information when it is extracting features, the layers miss some input units from the last layer. Because the CNNs is a structure for extracting data features. It can be suffered from some degree of distortion. The dropping information will be affordable in CNNs. People set the first convolutional layer as full connected. After several non-fulled connection convolutional layers, it also needs to place a full connected convolutional layer.

2.1.2 Pooling Layers. Before discussing pooling layers, the concept of the classifier will be introduced, which is a standard step in programming, classifying data statistically into various categories. The data cannot be analysed directly, without categorising. To convince classifiers operating, we need to extract features further by using pooling layers and also pooling steps can save the space of data.

Pooling means picking up features from different parts of the results at high layers. It is not similar to convolutional layer’s function. For convolutional layers, the weights in filtering equations in units should be fixed again and again by Back Propagation and input data can overlap in different weighted units. For pooling layers, it just wants to pick a sample feature fastly in various groups, such as maximum and mean. And sampling data should not overlap in each pooling groups. The brief working process of pooling layers is shown in Figure 3.
2.1.3 Output Layers. After filling with so many convolutional layers, the results should be output by classes. Convolutional layers and pooling layers are focusing on extracting features and reducing unnecessary factors. Then, a fully connected layer will work as an output layer to produce suitable quantity and variety results. The categorization cannot be realized only by convolutional layers. There is always a classifier and computing unit for calculating loss function in the output layer. The loss function is for amending weighted factor in above convolutional layers. Once the CNNs finishes forward passway, the loss function also begins to work on backward passway which fixes error and loss during weighted functions.

2.2. Process
As the CNNs are always applied on image processing field, data is displayed as pixel map like a huge matrix.

2.2.1. Convolution. For most weighted functions in convolutional layers, they are designed as smaller pixel maps according to need, which is called Kernel or convolution window. Figure 4 shows how the pixel map works with the kernel in a convolutional way.

![Convolution Example](image)

**Figure 4. An Example of Convolution**

The convolution is the sum of the products between each image pixel and kernel pixel. For example, in Figure 4, each of 9 pixels can be extracted as 1 pixel to stands feature value. The higher value of result means this area’s data has a higher relationship with a convolutional window so that kernel can be regarded as a filter. Besides various values in kernel’s pixels, the convolution process is also changed by edge and stride. Edge is easy to understand it is just the length of the side of convolution window. It describes the size of kernels. When a kernel finishes a convolution process with a part of input data, it should move to another place for doing the next convolution. The moving
distance of kernels depends on the value of stride that the distance of kernels is moving during each convolutional processes.

2.2.2 Pooling. When the picture is too large to pass through the classifier, the pooling layers will be used. The only function of pooling layers is reducing the size of input data or picture. The normal way is grouping the input data and picking the maximum in each group which is called max-pooling. Figure 5 displays the max-pooling.

![Convolutional Feature and Max-Pooled Feature](image)

**Figure 5.** An Example of Max-Pooling

3. Gradient Descent and Backpropagation

There are two passways in the Convolutional Neural Networks: forward passway and backward passway. The first two sections in the paper are talking about the forward passway. In this section, backward passway will be discussed. Because of the initialization of the weighted function at first, in the CNNs process, the weighted function will not be confirmed, whether satisfying the request of structure’s precision. The function should be fixed again and again. The Back Propagation (BP) is for transmitting the fixing error to the lower layers from the upper layers. And then the lower layers can maintain the weighted functions by fixed errors. How can we find “the fixed errors”? Here comes the Gradient Descent method.

3.1. Gradient Descent method

Gradient Descent (GD) method is one of the most common optimisation algorithms in machine learning field. As we know, the most of the mathematical models have errors because of the estimating factors. There is a concept called loss function, to describe the total error between samples and function output. We can assume a mathematical model is a linear regression equation and give it a hypothesis function: \( h_\theta(x) = \theta_0 + \theta_1 x \), \( \theta_0 \) and \( \theta_1 \) are factors of the equation. Its sample is \( (x_i, y_i) \) \( (i = 1, 2, ..., n) \), where every \( x_i \) corresponds a \( y_i \). Then, the loss function of hypothesis function should be:

\[
J(\theta_0, \theta_1) = \sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2
\]

Gradient Descent method aims to focus on optimising mathematical model and loss function, which means amending the factors of original functions and reduce the value of loss functions.

Now, let us make the hypothesis function more complicated than before, such as \( h_\theta(x_1, x_2, ..., x_n) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \ [\theta_i = (i = 1, 2, ..., n)] \), we can add a feature value \( x_0 = 1 \), then we simplify it as: \( h_\theta(x_1, x_2, ..., x_n) = \sum_{i=0}^{n}\theta_i x_i \).

Then, the loss function is:
\[ J(\theta_0, \theta_1, ..., \theta_n) = \frac{1}{2m} \sum_{i=0}^{m} (h_{\theta}(x_i, x_1, ..., x_n) - y_i)^2 \] (7)

Because we want to reduce the value of loss function, we should make microcosmic it at first to analyse changes of functions. The partial derivatives are good chances. Here we should mention some concepts for initialising the Gradient Descent method.

Gradient, which measures the trend of aim function, we regard the partial derivative result of \( \theta_i \) to loss functions as a gradient, the equation is: \( \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, ..., \theta_n) \). Recursion terminal distance \( \varepsilon \) is the functions’ precision usually a difference between mathematical model output and sample. When the difference is less than or equals to the recursion terminal distance, the weighted functions in the model are satisfied to the request of process. Then, the process can stop. Step size \( \alpha \), a factor controls how much percent of gradient used to update new \( \theta_i \). The updating function expression is: \( \theta_{i \text{new}} = \theta_{i \text{old}} - \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, ..., \theta_n) \). If the goal function is a convex function, finally it can be optimized.

The step size is also a significant element during programming. When the step size is too small, a process can figure out the most optimal option, however, the convergence speed is very slow. Large step size can improve convergence speed extremely, while it cannot guarantee obtaining the optimal option, because one large step size sometimes already covers a small convergence space. Then it cannot detect a optimal point in this convergence space. Figure 6 shows the comparison among different step size’s loss functions convergence situations.

**Figure 6. The Comparison of Different Step Size**

Therefore, the general gradient descent algorithm is following:
Initialise \( \theta_0, \theta_1, ..., \theta_n \) in aim function, recursion terminal distance \( \varepsilon \), step size \( \alpha \).
Calculate the gradient, the partial derivative of the loss function of aim function, \( \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, ..., \theta_n) \).
If \( J(\theta_0, \theta_1, ..., \theta_n) \leq \varepsilon \), stop the algorithm, otherwise, continue the process.
Use the gradient multiplying with step size \( \alpha \).
Renew all \( \theta \) by \( \theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, ..., \theta_n) \), and then back to step II.

3.2. Backpropagation
Besides updating weight functions, the updating fixed errors should be passed through the layers in the CNNs. It seems like combining Gradient Descent algorithm, the chain rule and Back Propagation algorithm.
Figure 7. An Example of Backpropagation in Single Calculating Unit

Figure 7 is a diagram of a single neural cell. We assume training sample is \((x, y)\), \(x\) is input, which passes through activation function \(h_{w,b}(x)\). And the unit gets an output "a", then the result "a" goes through the loss function, the output of loss function is the cost \(J\).

The expression of activation function or hypothesis function is:

\[
h_{w,b}(x) = a = \text{sigmoid}(x) = \text{sigmoid}(\sum_{i=0}^{n}(x_iw_i + b))
\]  

(8)

The equation of loss function is:

\[
J(W, b, x, y) = \frac{1}{2}\|y - h_{w,b}(x)\|^2
\]  

(9)

In the activation function (equation (8)), the sigmoid function \(\sigma(z) = \frac{1}{1+e^{-z}}\), is a hypothesis function. We can use other functions as an activation function either. The sigmoid function has a property is convinced for calculating: \(a = \sigma(z) \); \(\frac{\partial a}{\partial x} = a(1 - a)\).

During the training, we should figure out the gradients of the cost \(J\) by partial derivatives. There are two gradients; one is for the factor \(W\), another one is for the factor \(b\). Then we are going to take the partial derivative concerning \(W\) and \(b\). However, we cannot calculate the partial derivative from cost \(J\) to \(w\) and \(b\). The chain rule would help to solve this problem. It can display as \(\frac{\partial J}{\partial w} = \frac{\partial J}{\partial z} \frac{\partial z}{\partial w} \). Now, we should figure out the partial derivative from cost \(J\) to intermediate variable "a" and \(z\) at first:

\[
\delta^{(a)} = \frac{\partial}{\partial a} J(W, b, x, y) = -(y - a)
\]  

(10)

\[
\delta^{(b)} = \frac{\partial}{\partial z} J(W, b, x, y) = \frac{\partial J}{\partial a} \frac{\partial a}{\partial z} = \delta^{(a)} a(1 - a)
\]  

(11)

Then, according to the chain rule, obtain factor \(W\) and \(b\)'s gradients:

\[
\nabla_w J(W, b, x, y) = \frac{\partial}{\partial w} J = \frac{\partial J}{\partial z} \frac{\partial z}{\partial w} = \delta^{(z)} x^T
\]  

(12)

\[
\nabla_b J(W, b, x, y) = \frac{\partial}{\partial b} J = \frac{\partial J}{\partial z} \frac{\partial z}{\partial b} = \delta^{(z)}
\]  

(13)

During the above processes, we take \(\frac{\partial J}{\partial z}\) firstly, then figure out \(\frac{\partial J}{\partial a}\) finally we get \(\frac{\partial J}{\partial w}\) and \(\frac{\partial J}{\partial b}\).

Therefore, we can find that this is a procedure which propagates the increment \(\partial J\) of the cost function from the back forward, it called back propagation.

4. Applications
4.1 License Plate Recognition
The most important application field of CNN is in picture recognition, and a very common application scene in picture recognition is vehicle license plate recognition. In the same area, the number and style of vehicle license plate numbers or characters are fixed, so the training system cost is greatly reduced. At the same time, the characteristics of vehicle license plate, such as the unified size of plate shape, make the license plate easily to be located and recognized. However, there are also problems and limitations in license plate recognition. For instance, as for high-speed vehicles, it is difficult to use the ordinary camera for image recognition. Similarly, under some extreme weather conditions, such as fog, rain and snow weather, it is extremely hard to obtain clear vehicle license plate images. These problems increase the difficulty of license plate recognition. Therefore, the vehicle license plate recognition can only be applied in a specific scene, such as the underground garage. With the upgrading of hardware technology, vehicle license plate recognition technology is bound to have more development and application.

4.2 Semantic Recognition
In addition to image recognition, CNN can also be used for semantic recognition. Most of the applications of CNN in semantic recognition exist in meaning detection, such as spam detection or topic classification. By using word2vec to translate semantic words into word vectors, an input matrix is constructed, followed by a convolution layer consisting of several filters, as well as a maximal pool layer and a Softmax classifier. Training mainly exists in the adjustment of word2vec, with the increase of training set, the semantic differentiation becomes more obvious.

5. Conclusion
This paper introduces the Convolutional Neural Network algorithm, which is very popular in the field of machine learning and image recognition. It combines the Back Propagation mechanism and the Gradient Descent method to discuss its basic structure and function. At the same time, the practical application and role are discussed.

With the continuous development of technology, the neural network is bound to become an important tool in people's lives. The structure of the neural network will be more complete and efficient. For the neural network, the biggest challenges in the future may not only exist in the more optimized system, but also in the quality and quantity of the training set. A high-quality training set is an efficient neural network guarantee. Therefore, in the subsequent research of machine learning and in-depth study, the competition of algorithms is likely to become a competition of the quality and scale of data.

References
[1] Rosenblatt, F. (1962). Principles of neurodynamics.
[2] Kabrisky, M. (1964). A Proposed Model for Visual Information Processing in the Human Brain.
[3] Giebel, H. (1971). Feature extraction and recognition of handwritten characters by homogeneous layers. In Zeichenerkennung durch biologische und technische Systeme/Pattern Recognition in Biological and Technical Systems (pp. 162-169). Springer, Berlin, Heidelberg.
[4] Fukushima, K. (1975). Cognition: A self-organizing multilayered neural network. Biological cybernetics, 20(3-4), 121-136.
[5] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science.
[6] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.
[7] Koushik, J. 2016. Understanding convolutional neural networks.