Applied Intelligence (2021) 51:4106–4126
https://doi.org/10.1007/s10489-020-02015-5

Research on improved wavelet convolutional wavelet neural networks

Jing-Wei Liu[1,2] · Fang-Ling Zuo[3] · Ying-Xiao Guo[1] · Tian-Yue Li[1] · Jia-Ming Chen[2]

Accepted: 9 October 2020 / Published online: 27 November 2020

Abstract
Convolutional neural network (CNN) is recognized as state of the art of deep learning algorithm, which has a good ability on the image classification and recognition. **The problems of CNN are as follows:** the precision, accuracy and efficiency of CNN are expected to be improved to satisfy the requirements of high performance. The **main work is as follows:** Firstly, wavelet convolutional neural network (wCNN) is proposed, where wavelet transform function is added to the convolutional layers of CNN. Secondly, wavelet convolutional wavelet neural network (wCwNN) is proposed, where fully connected neural network (FCNN) of wCNN and CNN are replaced by wavelet neural network (wNN). Thirdly, image classification experiments using CNN, wCNN and wCwNN algorithms, and comparison analysis are implemented with MNIST dataset. **The effect of the improved methods are as follows:** (1) Both precision and accuracy are improved. (2) The mean square error and the rate of error are reduced. (3) The complexitie of the improved algorithms is increased.

Keywords Wavelet convolutional neural network · Convolutional neural network · Wavelet neural network · Deep learning · Image analysis

1 Introduction

Convolutional neural network (CNN) is a typical deep learning method which is based on feature extraction of convolution calculation [9]. It is widely applied to fields of prediction, classification [14] etc. CNN can solve high-dimensional problems which are difficult for traditional machine learning methods [19]. The ability to minimize the system error between the label and the inference [22] of CNN is much more powerful especially in the application of image processing. The neuron weights [12] of CNN are modified by forward propagation and error back propagation [15]. In recent years, the ability of CNN becomes more powerful because the distributed computing power has been greatly improved. Apart from image recognition [3], CNN are also applied in the other fields [11] such as text classification [26], control system [1] and target tracking [21].

The development history of CNN is as follows The earliest study about CNN can be traced back to Fukushima, who mimicked the visual cortex of an organism and proposed the Neocognition model [7]. Time-Delay Neural Network (TDNN) was proposed by Alexander Waibel et al. in 1987 [27]. It is proved in TDNN that more hidden layers have greater feature extraction capabilities, which becomes the foundation of further optimization of CNN. After a series of improvements, He-Kaiming et al. released ResNet in 2015 [8]: the network manages to skip some neuron nodes to achieve higher performance. In 2017, Gao Huang et al. proposed DenseNet.

Problems of CNN can be summarized as follows (1) The precision, accuracy and efficiency of CNN are expected to be improved. (2) High-dimensional information contains more details, which is difficult to be learned such as datasets of MINIST and CIFAR. Even human brain also tends to ignore the high-dimensional information. (3) CNN is more complex than classical neural network, but the trained model of CNN cannot be well explained. It is proved that randomly generated

The first two authors contributed equally to this work.

✉ Jing-Wei Liu
liujingwei@cueb.edu.cn

✉ Fang-Ling Zuo
zuofangling@cueb.vip

1 Information College, Capital University of Economics and Business, Beijing 100070, China

2 Information Department, Beijing university of technology, Beijing 100124, China

3 School of Statistics and Mathematics, Central University of Finance and Economics, Beijing 100081, China

network of CNN can solve difficult problems better than the carefully designed network sometimes. More intelligent module which can identify more detailed information is expected.

Wavelet transform (WT) is often used in deep learning [5, 16, 24]. Many features can be obtained by the discrete wavelet transform which have been improved by researches. The application fields based on WT and deep learning methods are image classification [10, 23], computer vision [4, 17], texture classification [6], etc.

The applications based on wavelet neural network (WNN) in deep learning are as follows In 2019, Pengju Liu et al. proposed a Multi-level Wavelet Convolutional Neural Networks(MWCNN) [16], which is proved to increase the receptive field by reducing the number of map. The Multi-Path Learnable Wavelet Neural Network for Image Classification was introduced by De Silva et al. [5]. This model introduces a multi-path layout with several levels of wavelet decompositions. In the domain of prediction, a convolutional LSTM network using the wavelet decomposition has been proposed in 2018 [28]. It takes the wavelet decomposition as the method of feature extraction rather than the manual feature extraction, which has been also proved by Kiskin et al. in 2017 [13].

The advantages of wavelet analysis are as follows Wavelet analysis has been widely used in signal processing and analysis. Wavelet analysis method is called mathematical microscope [2, 18], which is considered as a powerful tool for zooming details of sound, image, etc. Although the wavelet transformation has some complexity [32], the powerful detail extraction ability of wavelet transformation is helpful and important to solve the above problems of CNN [20].

The motivation of this research is to solve the CNN's problems based on the advantages of the WT. The importance of the research is that the improvements of CNN neurons are focused. Different from the ability of network with deeper layers, it is believed that the improvements of each neuron of CNN can improve the features identification and learning ability of the whole CNN [30]. Wavelet analysis is adopted [29] to improve the CNN network in this study.

The contributions of this study are as follows (1) The wavelet-based Convolutional Neural Network (wCNN) is proposed, where the wavelet transformation is adopted as the activation function in Convolutional Pool Neural Network (CPNN) of CNN. (2) Based on wCNN, the wavelet-ased Convolutional wavelet Neural Network (wCwNN) is proposed, where the Fully connected Neural Network (FCNN) of wCNN is replaced by wavelet Neural Network (wNN). (3) Comparative experiments between CNN, wCNN and wCwNN are implemented on the MNIST dataset.

The following sections are organized as follows The traditional CNN model is described in the second section. The improved wCNN is proposed in the third section. The improved wCwNN is proposed in the fourth section. The performance of CNN, wCNN and wCwNN is verified, analyzed and compared respectively with MNIST dataset in the fifth section. Discussion, conclusions and further research are given in the sixth section.

2 Model of convolutional neural network (CNN)

2.1 Structure of CNN

The structure of classical CNN is shown in Fig. 1. There are two parts in CNN: the first part is CPNN, and the second part is FCNN. In CPNN, the first layer is an input layer, and the following layers of CPNN are several pairs of convolutional layers and pooling layers. In FCNN, the first layer is an input

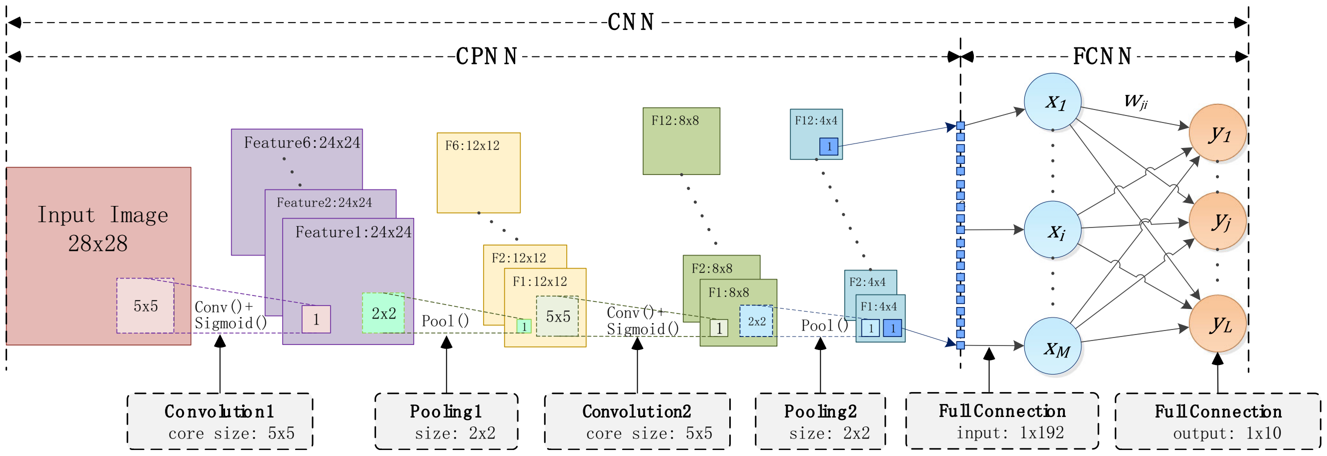


Fig. 1 Structure of CNN

layer, and the second layer of FCNN is an output layer, both layers of FCNN are fully connected.

The relation and features of CPNN and FCNN are as follows. (1) The input layer of CPNN is the input layer of CNN; (2) The last layer of CPNN is the input layer of FCNN; (3) The output layer of FCNN is the output layer of CNN; (4) The activation function of the convolutional layer in CPNN and the output layer in FCNN is sigmoid function; (5) There are not any activation functions in the input layer and pooling layer of CPNN and the input layer of FCNN.

2.2 Algorithm of CNN

The algorithm of CNN can be described as follows: (1) Initializing weights between layers and bias of neurons. (2) Forward propagating. (3) Calculating the mean square error (MSE) of all samples according to the loss function. (4) Calculating the errors of back propagating for each layer, which are the results of derivation by the chain rule. (5) Applying gradient to adjust the weights and bias according to the back-propagated errors. (6) Repeating the step (2) to step (5) until the MSE is small enough. (6) Evaluating the accuracy, precision and efficiency.

2.2.1 Forward propagation of CNN

Forward propagation of CNN is the calculation process from the input layer to the output layer, which can be described as follows: (1) The input layer of CNN is filled by a two-dimensional matrix of pixels of an image. (2) Forward propagation is calculated in convolutional and pooling layers (CPNN). (3) Forward propagation is calculated in fully connected layer (FCNN).

Definition 1: net^l and O^l are the input and the output of neurons in layer l. The output of each neuron can be calculated according to the input and the activation function of each neuron. l is the layer number, e.g. $l = 1$ stands for the first layer, and $l = -1$ stands for the last layer. i and j are the row number and column number respectively.

According to the above definition, net^{-1}, net^{-2} and net^{-3} stand for the input of the last FCNN layer, the input of the first FCNN layer and the input of the layer before FCNN (i.e. the last layer of pooling layers) respectively. The data structures of net^l and O^l of each layer of CPNN are two-dimensional matrix, while the net^l and O^l of each layer in FCNN are one-dimensional vectors.

Definition 2: w^l_{ij} and b^l_j are the weights and bias of layer l. w^{-1}_{ij} and b^{-1}_j are the weights and bias of the last layer of FCNN. If the layer l is a convolutional layer or a pooling layer, the size of the convolutional kernel or the pooling windows can be expressed as $size^l \times size^l$. If layer l is a fully-connected layer, the number of neurons is expressed as $size^l$.

Definition 3: int(x) is the function for getting the integer part of x, e.g., int(5.1) = int(5.7) = 5.

Forward propagation of convolutional layer The input of convolutional layer (net^l) can be calculated according to Eq. (1). The net^l_{mn} stands for each input value of neurons in layer l. The convolution(O^{l-1}, w^l, m, n) is the function for convolution calculations. The O^{l-1} is the output of the previous layer. The w^l is the matrix of weights between the input of layer l (net^l) and the output of the previous layer (O^{l-1}). The b^l is the bias of layer l.

$$\begin{aligned} net^l_{mn} &= \text{convolution}\left(O^{l-1}, w^l, m, n\right) + b^l \\ &= \sum_{i=0}^{size^l-1}\sum_{j=0}^{size^l-1}\left(O^{l-1}_{m+i,n+j}\cdot w^l_{i,j} + b^l\right) \end{aligned} \tag{1}$$

An example of convolution operation is provided. If $x = \begin{matrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{matrix}$, $y = \begin{matrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{matrix}$, the formula of convolution(x, y) can be expressed as Eq. (2):

$$\text{convolution}(x,y) = \begin{matrix} x_{11}y_{11} + x_{12}y_{12} + x_{21}y_{21} + x_{22}y_{22} & x_{12}y_{11} + x_{13}y_{12} + x_{22}y_{21} + x_{23}y_{22} \\ x_{21}y_{11} + x_{22}y_{12} + x_{31}y_{21} + x_{32}y_{22} & x_{22}y_{11} + x_{23}y_{12} + x_{32}y_{21} + x_{33}y_{22} \end{matrix} \tag{2}$$

The output of the convolutional layer l (O^l_{mn}) can be calculated as Eq. (3), where sigmoid() is the activation function.

$$O^l_{mn} = \text{F}\left(net^l_{mn}\right) = \text{sigmoid}\left(net^l_{mn}\right) = \frac{1}{1 + e^{-net^l_{mn}}} \tag{3}$$

Forward propagation of pooling layer Definition 4: The function pool(x) represents the average pooling of matrix x. The formula of pool(x) can be expressed as Eq. (4). The $size^l$ stands for the size of the pooling window.

$$y_{ij} = \text{pool}(x,i,j) = \frac{\sum_{m=1}^{size^l}\sum_{n=1}^{size^l} x^{l-1}_{size^l\times(i-1)+m,\, size^l\times(j-1)+n}}{size^l \times size^l} \tag{4}$$

An example of average pooling is provided: If $x = \begin{matrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{matrix}$, The pooling result is calculated as Eq. (5).

$$\text{pool}(x) = \begin{matrix} \frac{x_{11}+x_{12}+x_{21}+x_{22}}{4} & \frac{x_{13}+x_{14}+x_{23}+x_{24}}{4} \\ \frac{x_{31}+x_{32}+x_{41}+x_{42}}{4} & \frac{x_{33}+x_{34}+x_{43}+x_{44}}{4} \end{matrix} \quad (5)$$

According to Eq. (4), the output of the pooling layer O^l is according to the output of the previous layer (O^{l-1}). In other words, the input of the pooling layer l (net^l) is as same as the output of the previous layer (O^{l-1}).

Forward propagation of fully-connected layer The total number of neurons in the first layer of FCNN ($size^{-2} \times 1$) is equal to the number of neurons of in the last layer of CPNN ($size^{-3} \times size^{-3}$), which can be expressed as $size^{-2} \times 1 = size^{-3} \times size^{-3}$. The output of the first layer of FCNN (O_i^{-2}) is transformed from the output of the last layer of CPNN (O_{mn}^{-3}). The transform relation between O_i^{-2} and O_{mn}^{-3} can be expressed as Eq. (6).

$$O_i^{-2} = O_{mn}^{-3}, \quad m = \text{int}\left(\frac{i}{size^{-2}}\right) + 1, n = i - size^{-2} \times (m-1) \quad (6)$$

The result of forward propagation is $\widehat{y}_n$, which can be formulated in Eq. (7) to Eq. (9).

$$net_j^{-1} = \sum_{i=1}^{size^{-2}} \left(O_i^{-2} \cdot w_{ij}^{-1} + b^{-1}\right), \quad j = 1, 2, \ldots, size^{-1} \quad (7)$$

$$O_j^{-1} = \text{F}\left(net_j^{-1}\right) = \text{sigmoid}\left(net_j^{-1}\right) = \frac{1}{1+e^{-net_i^{-1}}} \quad (8)$$

$$\widehat{y}_n = O^{-1} \quad (9)$$

Back propagation of CNN There are three kinds of back propagation (BP) in CNN algorithm: BP in fully-connected layer, BP in pooling layer and BP in convolutional layer.

Definition 5: δ^l is defined as the input error of layer l.

Definition 6: L is the mean square error (MSE) of all samples, which can be formulated as Eq. (10). The closer the values of $\widehat{y}_n$ and y_n, the better the training effect is, because $\widehat{y}_n$ is the prediction of x_n, and y_n is the label of x_n. If each value of $\widehat{y}_n$ is very close to y_n, the value of L will be very small, which means that the training effect is very good and the trained model has a good fitting.

$$L = \frac{1}{2} \sum_{n=1}^{N} \left(\widehat{y}_n - y_n\right)^2 \quad (10)$$

Back propagation of fully-connected layer According to the above definition, δ_j^{-1} is defined as the input error of the last layer of FCNN, which is formulated as Eq. 11. y_n is the labels of training and testing samples. $\widehat{y}_n$ is the predictive result of samples. n is number (ID) of the samples. N is the total number of samples.

$$\delta_j^{-1} = \frac{\partial L}{\partial net_j^{-1}} = \frac{1}{N} \sum_{n=1}^{N} \left(\widehat{y}_n - y_n\right)\left(1 - net_j^{-1}\right) net_j^{-1} \quad (11)$$

δ_i^{-2} is defined as the input error of the first layer of FCNN. The size of δ_i^{-2} is $size^{-2} \times 1$. δ_{mn}^{-3} is defined as the back propagation error of previous layer of FCNN (the last layer of CPNN before the first layer of FCNN). The size of δ_{mn}^{-3} is $size^{-3} \times size^{-3}$. The transform relation between δ_i^{-2} and δ_{mn}^{-3} can be expressed as Eq. (12):

$$\delta_{mn}^{-3} = \delta_i^{-2}, \quad i = size^{-2} \times (m-1) + n \quad (12)$$

The error back propagation from the first layer of FCNN to the last pooling layer in CPNN is expressed as Eq. (13):

$$\begin{aligned} \delta_i^{-2} &= \frac{\partial L}{\partial net_i^{-2}} = \frac{\partial L}{\partial O_i^{-2}} = \frac{\partial L}{\partial net_j^{-1}} \cdot \frac{\partial net_j^{-1}}{\partial O_i^{-2}} \\ &= \sum_{j=1}^{size^{-2}} \delta_j^{-1} \cdot w_{ij}^{-1} \; i = 1, 2, \ldots, size^{-1} \end{aligned} \quad (13)$$

Backpropagation of pooling layer If the layer l is a convolutional layer, the layer $l+1$ is a pooling layer. Functions of pool calculations can be expressed as Eq. (14) to Eq. (16):

Definition 7: Function of padding(x): matrix x can be expanded with 0 around as Eq. (14):

$$x = \begin{matrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{matrix}, \quad padding(x) = \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & x_{11} & x_{12} & 0 \\ 0 & x_{21} & x_{22} & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \quad (14)$$

Definition 8: Function of rotate(x): matrix x can be rotated 180 degrees as Eq. (15):

$$x = \begin{matrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{matrix}, \quad rotate(x) = \begin{matrix} x_{22} & x_{21} \\ x_{12} & x_{11} \end{matrix} \quad (15)$$

The input error of convolutional layer is calculated by Eq. 16:

$$\begin{aligned} \delta_{mn}^{l} &= \frac{\partial L}{\partial net_{mn}^{l}} = \frac{\partial L}{\partial net_{mn}^{l+1}} \cdot \frac{\partial net_{mn}^{l+1}}{\partial O_{mn}^{l}} \cdot \frac{\partial O_{mn}^{l}}{\partial net_{mn}^{l}} \\ &= convolution\left(padding\left(\delta^{l+1}\right), rotate\left(w^l\right)\right) \end{aligned} \quad (16)$$

Backpropagation of convolutional layer Definition 9: Function poolExpand(x): the size and data of the output of pooling layer is expanded to the input of pooling layer. For example, matrix x_{uv} (output of pooling layer) is replaced by matrix y_{mn} (input of pooling layer) according to the function poolExpand(x) which is expressed as Eq. (17). $size^l$ is the size of pooling window.

$$y_{mn} = poolExpand(x_{uv}) = \frac{1}{size^l \times size^l} \cdot x_{uv}, u = int\left(\frac{m-1}{size^l \times size^l}\right) + 1, v = int\left(\frac{n-1}{size^l \times size^l}\right) + 1 \quad (17)$$

For example, if $x = \begin{matrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{matrix}$, the result of poolExpand(x) is calculated as Eq. (18):

$$\text{poolExpand}(x) = \begin{matrix} \frac{x_{11}}{4} & \frac{x_{11}}{4} & \frac{x_{12}}{4} & \frac{x_{12}}{4} \\ \frac{x_{11}}{4} & \frac{x_{11}}{4} & \frac{x_{12}}{4} & \frac{x_{12}}{4} \\ \frac{x_{21}}{4} & \frac{x_{21}}{4} & \frac{x_{22}}{4} & \frac{x_{22}}{4} \\ \frac{x_{21}}{4} & \frac{x_{21}}{4} & \frac{x_{22}}{4} & \frac{x_{22}}{4} \end{matrix} \quad (18)$$

If the layer l is a pooling layer, then the layer $(l+1)$ is a convolutional layer, and the input error of pooling layer can be calculated as the Eq. (19),:

$$\delta^l_{mn} = \frac{\partial L}{\partial net^l_{mn}} = \frac{\partial L}{\partial net^{l+1}_{mn}} \cdot \left(\frac{\partial net^{l+1}_{mn}}{\partial O^l_{mn}}\right) \cdot \left[\frac{\partial O^l_{mn}}{\partial net^l_{mn}}\right] = \text{poolExpand}\left(\delta^{l+1}\right) \cdot \left(\frac{\partial net^{l+1}_{mn}}{\partial O^l_{mn}}\right) \cdot \left[F'\left(net^l_{mn}\right)\right]$$
$$= \text{poolExpand}\left(\delta^{l+1}\right) \cdot (1) \cdot \left[\left(1-net^l_{mn}\right) \cdot net^l_{mn}\right] = \text{poolExpand}\left(\delta^{l+1}\right) \cdot \left(1-net^l_{mn}\right) \cdot net^l_{mn} \quad (19)$$

Adjustment of weights and parameters of CNN The change value of weights and bias can be calculated as Eq. (20) to Eq. (21):

$$\Delta w^l = \frac{\partial L}{\partial w^l} = \frac{\partial L}{\partial net^l} \times \frac{\partial net^l}{\partial w^l} = \delta^l \cdot O^{l-1} \quad (20)$$

$$\Delta b^l = \frac{\partial L}{\partial b^l} = \frac{\partial L}{\partial net^l} \times \frac{\partial net^l}{\partial b^l} = \delta^l \quad (21)$$

$$\Delta w^{-1}_{ij} = \frac{\partial L}{\partial w^{-1}_{ij}} = \frac{\partial L}{\partial net^{-1}_j} \times \frac{\partial net^{-1}_j}{\partial w^{-1}_{ij}} = \delta^{-1}_j \cdot O^{-2}_i \quad (22)$$

$$\Delta b^{-1} = \frac{\partial L}{\partial b^{-1}_j} = \frac{\partial L}{\partial net^{-1}_j} \times \frac{\partial net^{-1}_j}{\partial b^{-1}_j} = \frac{1}{size^l} \sum_{j=1}^{size^{-1}} \delta^{-1}_j \quad (23)$$

The updated value of weights and bias can be calculated as Eq. (24) to Eq. (27). η_CPNN is the learning rate of CNN:

$$w^l(t+1) = w^l(t) - \eta\_CPNN \times \Delta w^l \quad (24)$$

$$b^l(t+1) = b^l(t) - \eta\_CPNN \times \Delta b^l \quad (25)$$

$$w^{-1}_{ij}(t+1) = w^{-1}_{ij}(t) - \eta\_CPNN \times \Delta w^{-1}_{ij} \quad (26)$$

$$b^{-1}(t) = b^{-1}(t) - \eta\_CPNN \times \Delta b^{-1}_j \quad (27)$$

Pseudocode of CNN Features and labels are contained in training set, which are learned by the model of CNN. Weights and bias are adjusted in the training process.

(1) Definition of Adjustment cycle and Simulation process

Definition 10: Adjustment cycle (AC): In each AC, all the weights and biases are adjusted one time according to Eq. (20) to Eq. (27).

Definition 11: Simulation process (SP): Each SP is a complete training process. 1SP contains many continuous ACs. Each SP starts from the first AC (for example: $t = 1$) to the last AC (for example: $t = 6000$).

The relationship between 1SP and 1 AC is that 1SC is composed of ***n***ACs.

(2) Training algorithm of CNN

The effect of each 1SP is measured by the loss function, which is expressed as Eq. (28):

$$\boldsymbol{L}(t) = \frac{1}{2} \sum_{n=1}^{N} (train\_p(n) - train\_y(n))^2 \quad (28)$$

In Eq. (28), n is the number of each training sample, and N represents the total number of training samples. $train\_p(n)$ is the result of forward propagation of the n^{th} training sample, which is also expressed as $\widehat{y}_n$. $train\_y(n)$ is the label of n^{th} training sample, which is also expressed as y_n.

The pseudocode of CNN is listed in Algorithm 1.

```
Algorithm 1  Training algorithm of CNN
Input:
  train_x, train_y: features and labels of Training Set
  test_x, test_y: features and labels of Test Set
Output:
  $w_{ij}^l$, $b_j^l$,: weights and bias of Convolution and Pooling Neural Network (CPNN).
  $w_{jk}$, $b_{jk}$: weights and bias of Full Connection Neural Network (FCNN, FCNN have 2 layers)
Required parameters:
  max_time: maximum value n of nACs in every 1SP
  target_error: when the current training error is less than target error, the training is finished
  $\eta\_CPNN$: the learning rate of CPNN
Initialization work:
  $w_{ij}^l$, $b_j^l$, $w_{jk}$, $b_{jk}$: weights and scaling parameters of CNN (CPNN+FCNN) are set as random numbers.
  t: t is the current simulation time, which is initialized as t=1 before the training loop.
  $L(t)$: $L(t)$ is the mean square error at simulation time t. $L(t)$ is initialized as $L(1) = 1$>target_error.
Begin:
1:  Set the required parameters and complete the initialization work
2:  while t < max_time and L(t) > target_error
3:    for all trainingSet:
4:      train_p (prediction of label) is calculated according to train_x and forward calculation formula 1-9
5:    end for
6:    $L(t)$ is re-calculated as $L(t) = \frac{1}{2}\sum_{n=1}^{N}(train\_p(n) - train\_y(n))^2$, N is the total number of trainingSet.
7:    $\Delta w^l$ , $\Delta b^l$ , $\Delta w_{ij}^{-1}$, $\Delta b_j^{-1}$ are updated according to the formula 20-23
8:    $w^l\ (t), b^l\ (t), w_{ij}^{-1}(t), b_j^{-1}(t)$ are adjusted according to the formula 24-27
9:    t++
10: end while
End
```

3 Wavelet transform

Wavelet transform (WT) is an ideal method to process details of signals. WT provides a Time-Frequency Window which can capture higher and lower resolution of details of signals. The problem of Fourier transform [25] (FT) is that the window size cannot be changed when the frequency is changed. This problem can be solved by WT. The $\psi(a, b)$ is called wavelet generating function, which can be expressed as $\psi(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} f(t) * \varphi\left(\frac{t-b}{a}\right)$, where a and b are the scale parameters which control the extension and translation of function.

$\psi(a, b)$ is designed according to the following conditions (1) Only in a very small domain, the function value is not 0, and other domains are 0. In other words, translating the signal in timeline is same to adding a window on the original signal. (2) The integral value of function in the x axis must be 0. (3) The transform must be reversible. **There are many wavelet generating functions such as:**

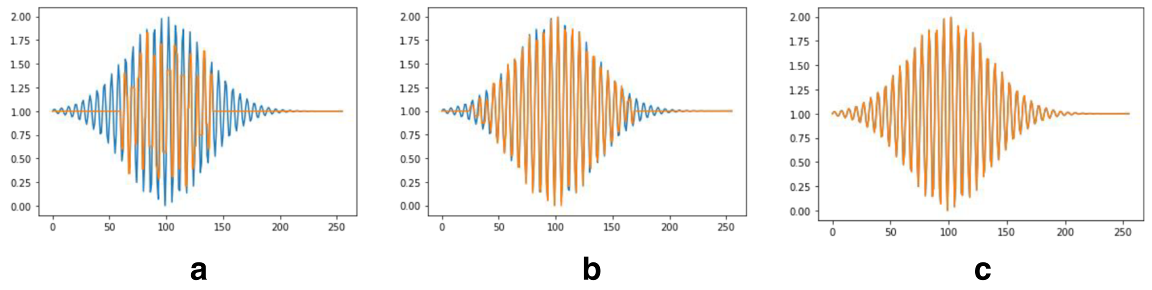


Fig. 2 Different effects of WT. **a**. scale = 20, error = 2.08. **b**. scale = 100, error = 0.044. **c**. scale = 200, error = 0.00045

(1) haar wavelet, (2) db wavelet [31], (3) sym wavelet [15], (4) coif series wavelet, etc. The wavelet function of this study is $\varphi(x) = \cos(1.75t)*e^{-\frac{t^2}{2}}$.

The processes of wavelet transform are visualized as follows In Fig. 2, the error is the difference between the signal of wavelet inverse transformation and the original signal. The scale is the parameter of wavelet function, which controls the extension of wavelet function. When the scale parameter is changed, the wavelet transform's ability of information extraction to original signal is changed.

In summary, by adjusting the scale and translation, wavelet transform can learn the different feature. So, richer feature can be learned by adding the wavelet transformation into the CNN.

4 Model of wavelet convolutional neural network (wCNN)

4.1 Structure of wCNN

The improvement of the proposed wCNN is that: the activation function F() of the convolutional layer in CNN is replaced by the Ψ(). The F() of CNN is sigmoid function, and the Ψ() of wCNN is wavelet scale transformation function.

The structure of proposed wCNN is that: The first part of wCNN is Wavelet Convolutional Pooling Neural Network (wCPNN), and the second part is Fully Connected Neural Network (FCNN). The structure of wCNN is shown in Fig. 3.

4.2 Algorithm of wCNN

The difference between wCNN and CNN is the activation function of convolution layer.

The training algorithm of wCNN also has three steps: (1) forward propagation of wCNN; (2) back propagation of wCNN; (3) weight and bias adjustment of wCNN.

4.2.1 Forward propagation of wCNN

The forward propagation of wCNN is same as CNN. The input of wCNN is the feature of training samples. The output of wCNN is calculated from the first layer of wCNN (input of wCNN) through convolutional layer, pooling layer and fully connected layer.

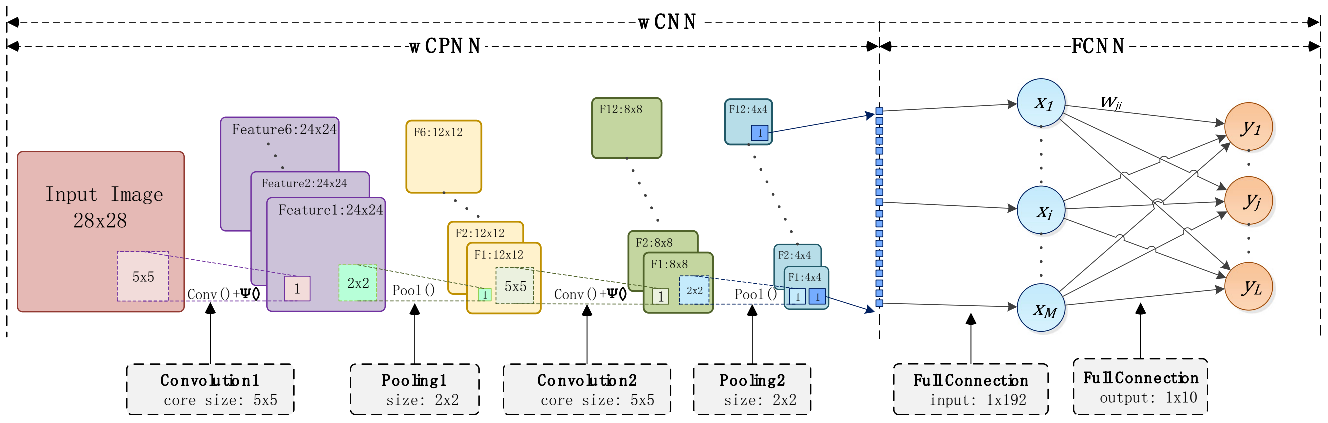


Fig. 3 Structure of wCNN

(1) Forward propagation of convolution layer

If the layer l is the convolutional layer, the input of this layer (net^l_{mn}) is calculated by Eq. (29):

$$net^l_{mn} = \text{convolution}\left(O^{l-1}, w^l\right) + b^l = \sum_{i=0}^{size^l-1} \sum_{j=0}^{size^l-1} \left(O^{l-1}_{m+i,n+j} \cdot w^l_{i,j}\right) \tag{29}$$

The output of this layer ($O^l_j(t)$) is calculated by Eq. (30). $ac^l$and bc^l is the parameters of the scale transformation in activation function:

$$O^l_{mn} = \Psi\left(\frac{net^l_{mn} - ac^l}{bc^l}\right) \tag{30}$$

In the convolutional layer of wCNN, the activation function $\Psi_{wc}(x)$ is expressed as Eq. (31):

$$\Psi_{wc}(x) = cos(1.75x) \cdot e^{-\frac{x^2}{2}} \tag{31}$$

(2)Forward propagation of Pooling layer and Fully connected layer.

Forward propagation in the pooling layer and the fully-connected layer of wCNN is the same as CNN, which are shown in Eq. (4) and Eq. (7).

4.2.2 Back propagation of wCNN

The predicted values $\widehat{y}_n$ of the training samples can be calculated by forward propagation of wCNN, then the MSE (mean square error) of all the training samples can be calculated by loss function as Eq. (10).

Back propagation of error is necessary for weights and bias adjustment, which is calculated in fully connected layer, pooling layer and convolutional layer. The back propagation in fully connected layer and convolutional layer are same as CNN, while back propagation in pooling layer is different:

If the layer l is pooling layer, the layer $(l+1)$ is a convolutional layer, and the error of the input of the pooling layer is expressed as Eq. (32):

$$\delta^l_{mn} = \frac{\partial L}{\partial net^l_{mn}} = \frac{\partial L}{\partial net^{l+1}_{mn}} \cdot \frac{\partial net^{l+1}_{mn}}{\partial O^l_{mn}} \cdot \frac{\partial O^l_{mn}}{\partial net^l_{mn}} = \frac{1}{bc^l} \text{poolExpand}\left(\delta^{l+1}_{uv}\right) \cdot \Psi'\left(\frac{net^l_{mn} - ac^l}{bc^l}\right) \tag{32}$$

$$\Psi'(x) = -1.75 \cdot sin(1.75x) \cdot e^{-\frac{x^2}{2}} - x \cdot cos(1.75x) \cdot e^{-\frac{x^2}{2}} \tag{33}$$

Gradient descent method is applied to calculate the changed values of weights and bias such as w^l,ac^l,bc^l, which can be expressed as Eq. (34) to Eq. (36):

$$\Delta w^l = \frac{\partial L}{\partial w^l} = \frac{\partial L}{\partial net^l} \times \frac{\partial net^l}{\partial w^l} = \delta^l \cdot O^{l-1} \tag{34}$$

$$\Delta ac^l = \frac{\partial L}{\partial ac^l} = \frac{\partial L}{\partial net^l} \times \frac{\partial net^l}{\partial ac^l} = -\delta^l \cdot \frac{1}{bc^l} \tag{35}$$

$$\Delta bc^l = \frac{\partial L}{\partial bc^l} = \frac{\partial L}{\partial net^l} \times \frac{\partial net^l}{\partial bc^l} = -\frac{1}{{bc^l}^2} \cdot \delta^l \cdot \left(net^l_{mn} - ac^l\right) \tag{36}$$

The adjusted results such as w^{-1}_{ij} and b^{-1}_j can be expressed as Eq. (37) to Eq. (41), where η_CPNN is the learning rate of wCNN:

$$w^l(t+1) = w^l(t) - \eta\_wCPNN \times \Delta w^l \tag{37}$$

$$bc^l(t+1) = bc^l(t) - \eta\_wCPNN \times \Delta bc^l \tag{38}$$

$$ac^l(t+1) = ac^l(t) - \eta\_wCPNN \times \Delta ac^l \tag{39}$$

$$w^{-1}_{ij}(t+1) = w^{-1}_{ij}(t) - \eta\_CPNN \times \Delta w^{-1}_{ij} \tag{40}$$

$$b^{-1}_j(t) = b^{-1}_j(t) - \eta\_wCPNN \times \Delta b^{-1}_j \tag{41}$$

4.3 Pseudocode of wCNN

The training process of wCNNwCNN is similar to CNN, while the activation function of wCNN is different from CNN. The pseudocode of wCNN is listed in Algorithm 2.

Algorithm 2 Training process of wCNN (wCNN=wCPNN+FCNN)

```
Input:
  train_x, train_y, test_x and test_y are set same as the pseudocode of CNN
Output:
  $wc_{ij}^{l}$, $bc_{j}^{l}$, $ac_{j}^{l}$: weights and bias of wCPNN ($l = 2, 4$, wCPNN have 5 layers)
  $w_{jk}$, $b_k$ : weights and bias of FCNN (FCNN have 2 layers)
Required parameters:
  max_time and target_error are set same as the pseudocode of CNN
  $\eta$_wCPNN,: learning rate of wCPNN
Initialization work:
  $t$=1 and $loss(1) = 1$ are set same as the pseudocode of CNN
  $wc_{ij}^{l}, ac_{j}^{l}, bc_{j}^{l}, w_{jk}, b_k$ : weights and bias of wCNN are set as random number.
Begin:
1:  Set the required parameters and complete the initialization work
2:  while $t$ < max_time and loss($t$) > target_error
3:    for all trainingSet:
4:      train_p (prediction of label) is calculated according to train_x and forward calculation formula 29-31 and 4-9.
5:    end for
6:    $loss(t)$ is re-calculated as $loss(t) = \frac{1}{2}\sum_{n=1}^{N}(train\_p(n) - train\_y(n))^2$, $N$ is the total number of trainingSet.
7:    $\Delta w_{ij}^{-1}, \Delta b_{j}^{-1}$ and $\Delta w^{l}$ , $\Delta ac^{l}$ , $\Delta bc^{l}$ are calculated according to the formula 22-23 and 34-36
8:    $w_{ij}^{-1}(t), b_{j}^{-1}(t)$ and $w^{l}(t), ac^{l}(t), bc^{l}(t)$ are adjusted according to the formula 37-41
9:    $t$++
10: end while
End
```

5 Model of wavelet convolutional wavelet neural network(wCwNN)

5.1 Structure of wCwNN

Based on wCNN, the improvement of wCwNN is that: the fully connected network (FCNN) is replaced by a wavelet Neural Network(wNN). The structure of wCwNN has two parts: (1) wavelet Convolutional Pooling Network(wCPNN), and (2) wavelet Neural Network (wNN). In the convolutional layer of wCPNN and the hidden layer of the wNN, all the activation functions are wavelet scale transformation functions. The structure of wCwNN is drawn as Fig. 4.

5.2 Algorithm of wCwNN

The first part of wCwNN is wCPNN, which is same as wCNN. The second part of wCwNN is wNN, which is different from the second part of wCNN (FCNN). In this section, the forward propagation and back propagation of wNN are described in detail.

Definitions for wNN are listed as follows.

(1) The number of the last layer (output layer) of wNN is expressed as $l = -1$. $size^{-1}$ is the number of neurons in this output layer.
(2) The number of the second layer (the second to last layer, hidden layer) of wNN is expressed as $l = -2$. $size^{-2}$ is the number of neurons in this hidden layer.
(3) The number of the first layer (the third to last layer, input layer) of wNN is expressed as $l = -3$. $size^{-3}$ is the number of neurons in this input layer.
(4) The number of the last layer of wCPNN, which is the previous layer of the input layer of wNN, is defined as $l = -4$. $size^{-4} \times size^{-4}$ is the number of neurons in this output layer of wCPNN.

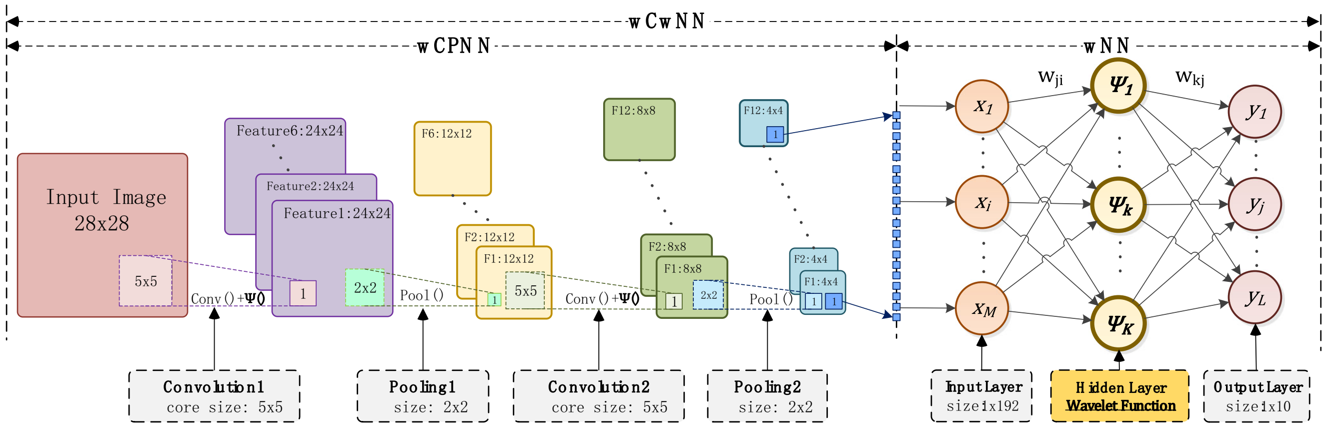


Fig. 4 Structure of wCwNN

5.2.1 Forward propagation of wCwNN

The input of the input layer in wNN (O_i^{-3}) comes from the output of the last layer in wCPNN (O_{mn}^{-4}), both two layers have the same numbers of neurons: $size^{-3} = size^{-4} \times size^{-4}$.

The dimension of matrix O_{mn}^{-4} is $m \times n$, and the dimension of matrix of O_i^{-3} is $i \times 1$. The correspondence between O_{mn}^{-4} and O_i^{-3} can be expressed as Eq. 42:

$$O_i^{-3} = O_{mn}^{-4}, m = \text{int}\left(\frac{i}{size^{-3}}\right) + 1, n = i - size^{-3} \times (m-1) \tag{42}$$

In the hidden layer of wNN, the input matrix is net_j^{-2}, and the output matrix is O_j^{-2}. O_j^{-2} can be calculated by Eq. (43) to Eq. (44):

$$net_j^{-2} = \sum_{i=1}^{size^{-3}} \left(O_i^{-3} \cdot w_{ij}^{-2}\right), \quad j = 1, 2, \ldots, size^{-2} \tag{43}$$

$$O_j^{-2} = \Psi\left(net_j^{-2}\right) = \Psi\left(\frac{net_j^{-2} - ac^{-2}}{bc^{-2}}\right), \quad j = 1, 2, \ldots, size^{-2} \tag{44}$$

In the output layer of wNN, the input matrix and the output matrix are net_j^{-1} and O_j^{-1} respectively, and the predicted result of wCwNN is $\widehat{y}_n$. O_j^{-1} and $\widehat{y}_n$ can be calculated by Eq. (45) to Eq. (47):

$$net_k^{-1} = \sum_{j=1}^{size^{-2}} \left(O_j^{-2} \cdot w_{kj}^{-1} + b_k^{-1}\right), \qquad k = 1, 2, \ldots, size^{-1} \tag{45}$$

$$O_k^{-1} = F\left(net_k^{-1}\right) = \text{sigmoid}\left(net_k^{-1}\right) = \frac{1}{1 + e^{-net_k^{-1}}}, \qquad k = 1, 2, \ldots, size^{-1} \tag{46}$$

$$\widehat{y}_n = O^{-1} \tag{47}$$

5.2.2 Back propagation of wCwNN

In wNN, the back propagation of input errors (δ_i^{-3} in the output layer, δ_j^{-2} in the hidden layer and δ_k^{-1} in the input layer) can be calculated as Eq. (48) to Eq. (50).

$$\delta_k^{-1} = \frac{\partial L}{\partial net_k^{-1}} = \frac{1}{N} \sum_{n=1}^{N} \left(\widehat{y}_n - y_n\right)\left(1 - net_k^{-1}\right) net_k^{-1}, \qquad k = 1, 2, \ldots, size^{-1} \tag{48}$$

$$\begin{aligned}\delta_j^{-2} &= \frac{\partial L}{\partial net_j^{-2}} = \frac{\partial L}{\partial net_k^{-1}} \cdot \frac{\partial net_k^{-1}}{\partial O_j^{-2}} \cdot \frac{\partial O_j^{-2}}{\partial net_j^{-2}} \\ &= \frac{1}{bc^{-2}} \sum_{k=1}^{size^{-1}} \delta_k^{-1} \cdot \Psi'\left(\frac{net_j^{-1} - ac^{-2}}{bc^{-2}}\right) \cdot w_{kj}^{-1}, \qquad j = 1, 2, \ldots, size^{-2}\end{aligned} \tag{49}$$

$$\begin{aligned}\delta_{mn}^{-3} = \delta_i^{-3} &= \frac{\partial L}{\partial net_i^{-3}} = \frac{\partial L}{\partial O_i^{-3}} = \frac{\partial L}{\partial net_j^{-2}} \cdot \frac{\partial net_j^{-2}}{\partial O_i^{-3}} \\ &= \sum_{j=1}^{size^{-2}} \delta_j^{-2} \cdot w_{ij}^{-2}, \qquad i = 1, 2, \ldots, size^{-3}\end{aligned} \tag{50}$$

Gradient descent method is applied to adjust the weights (w_{ij}^{-2} and w_{kj}^{-1}) and bias (ac^{-2}, bc^{-2} and b_k^{-1}) of wCwNN. The changed values of the above weights and bias can be calculated by Eq. (51) to Eq. (55).

$$\Delta w_{ij}^{-2} = \frac{\partial L}{\partial w_{ij}^{-2}} = \frac{\partial L}{\partial net_j^{-2}} \times \frac{\partial net_j^{-2}}{\partial w_{ij}^{-2}} = \delta_j^{-2} \cdot O_i^{-3} \tag{51}$$

$$\begin{aligned}\Delta ac^{-2} &= \frac{\partial L}{\partial ac^{-2}} = \frac{\partial L}{\partial net_j^{-2}} \times \frac{\partial net_j^{-2}}{\partial ac^{-2}} \\ &= \frac{1}{size^{-2}} \sum_{j=1}^{size^{-2}} -\delta_j^{-2} \cdot \frac{1}{bc^{-2}}\end{aligned} \tag{52}$$

$$\begin{aligned}\Delta bc^{-2} &= \frac{\partial L}{\partial bc^{-2}} = \frac{\partial L}{\partial net_j^{-2}} \times \frac{\partial net_j^{-2}}{\partial bc^{-2}} \\ &= -\frac{1}{size^{-2}} \sum_{j=1}^{size^{-2}} \frac{1}{\left(bc^{-2}\right)^2} \cdot \delta_j^{-2} \cdot \left(net_j^{-1} - ac^{-2}\right)\end{aligned} \tag{53}$$

$$\Delta w_{kj}^{-1} = \frac{\partial L}{\partial w_{kj}^{-1}} = \frac{\partial L}{\partial net_k^{-1}} \times \frac{\partial net_k^{-1}}{\partial w_{kj}^{-1}} = \delta_k^{-1} \cdot O_j^{-2} \tag{54}$$

$$\Delta b_k^{-1} = \frac{\partial L}{\partial b_k^{-1}} = \frac{\partial L}{\partial net_k^{-1}} \times \frac{\partial net_j^{-1}}{\partial b_k^{-1}} = \delta_k^{-1} \quad (55)$$

The adjustive result of the above weights and bias are expressed as Eq. (56) to Eq. (60), where α_wNN, η_wNN, and η_wCPNN are the inertia coefficient of wNN, the learning rate of wNN, and the learning rate of wCPNN, respectively.

$$w_{ij}^{-2}(t+1) = w_{ij}^{-2}(t) - \Delta w_{ij}^{-2} \times \eta\_wNN + w_{ij}^{-2}(t) \times \alpha\_wNN \quad (56)$$

$$ac^{-2}(t+1) = ac^{-2}(t) - \Delta ac^{-2} \times \eta\_wNN + ac^{-2}(t) \times \alpha\_wNN \quad (57)$$

$$bc^{-2}(t+1) = bc^{-2}(t) - \Delta bc^{-2} \times \eta\_wNN + bc^{-2}(t) \times \alpha\_wNN \quad (58)$$

$$w_{kj}^{-1}(t+1) = w_{kj}^{-1}(t) - \Delta w_{kj}^{-1} \times \eta\_wCPNN \quad (59)$$

$$b_k^{-1}(t+1) = b_k^{-1}(t) - \Delta b_k^{-1} \times \eta\_wCPNN \quad (60)$$

5.2.3 Pseudocode of wCwNN

Training algorithm of wCwNN is similar to wCNN, while the difference is that FCNN in wCNN is replaced by wNN in wCwNN. The pseudo code of wCwNN is listed in Algorithm 3.

Algorithm 3 Training process of wCwNN (wCwNN= wCPNN + wNN)

```
Input:
  train_x, train_y, test_x and test_y are set same as the pseudocode of CNN and wCNN
Output:
  $wc_{ij}^l$, $ac_j^l$, $bc_j^l$: weights and bias of wCPNN ($l = 2, 4$, wCPNN have 5 layers)
  $w_{ij}^2$, $w_{jk}^3$, $a_j$, $b_j$: weights and bias of wNN (wNN have 3 layers)
Required parameters:
  max_time and target_error are set same as the pseudocode of CNN and wCNN
  $\eta\_wCPNN$: learning rate of wCPNN
  $\eta\_wNN$, $\alpha\_wNN$: learning rate and inertia coefficient of wNN
Initialization work:
  t=1 and $loss(1) = 1$ are set same as the pseudocode of CNN and wCNN
  $wc_{ij}^l, bc_j^l, ac_j^l, w_{ij}^2, w_{jk}^3, a_j, b_j$: weights and bias of wCwNN (wCPNN+wNNs) are set as random number.
Begin:
1:  Set the required parameters and complete the initialization work
2:  while t < max_time and loss(t) > target_error
3:    for all trainingSet:
4:      train_p is calculated according to train_x and wCwNN forward calculation formula 29-31, 4-5, and .42-47
5:    end for
6:    $loss(t)$ is re-calculated as $loss(t) = \frac{1}{2}\sum_{n=1}^{N}(train\_p(n) - train\_y(n))^2$, $N$ is the total number of trainingSet.
7:    $\Delta w_{ij}^{-2}, \Delta ac^{-2}, \Delta bc^{-2}, \Delta w_{kj}^{-1}, \Delta b_k^{-1}$ of wNN are updated according to the formula 51-55
      $\Delta w^l$, $\Delta ac^l$, $\Delta bc^l$ of wCPNN wNN are calculated according to the formula 34-36
8:    $w_{ij}^{-2}(t), ac^{-2}(t), bc^{-2}(t), w_{kj}^{-1}(t), b_k^{-1}(t)$ of CNN are adjusted according to the formula 56-60
      $w^l(t), ac^l(t), bc^l(t)$ of wCPNN are adjusted according to the formula 37-39
9:    t++
10: end while
End
```

6 Experiment

The objectives of the experiment are as follows: (1) verify the viability of each algorithm (convergence), (2) improve accuracy of three algorithms(reduce the minimum mean square error), (3) improve the accuracy rate (reduce the error rate), (4) analyze the efficiency of the algorithms (5) find an algorithm with greater classification capacity.

The contents of experiment are as follows: (1) record error of three algorithms in every AC, then plot time-error curve (2) calculate the mean square error of all test samples after training (precision); (3) calculate error rate (accuracy rate) of the test sample after training; (4) record the time of training process (5) analyze the results of each algorithm; (6) analyze the results of each experiment.

6.1 Datageset introduction

Dataset of MNIST and CIFAR-10 are adopted for the comparative experiments. MNIST is well known from the National Institute of Standards and Technology. The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples.. Training set and test set are shown in Fig. 5 below:

CIFAR-10 is an open dataset, which has 60,000 images. The resolution of the images is 32*32. The images are divided into 10 categories, each category contains 6000 images. There are 50,000 images for training, 10,000 images for testing. The data set is shown in Fig. 6 below. In this experiment, two kinds images are selected.

6.2 Experiment design of CNN

For the following comparative experiments, the structure of CNN is designed as follows: the first layer of CNN is an input layer; the 2nd to 5th layers are two pairs of convolutional-pooling layers; the 6th to 7th layers are fully connected (FCNN).

Firstly, parameters of the structure of CNN are set as follows:

(1) Input layer of CNN: The size is 28×28.
(2) The first convolutional layer: the input size is 28×28. The size of convolutional kernel is set to 5×5. The output size is 24×24. The feature number of the output is set to 6. The activation function is Sigmoid.
(3) The first pooling layer: the input size is set to 24×24. The size of the pooling window is 2×2. The output size is 12×12. The number of output features is 6.
(4) The second convolutional layer: The input size is 12×12. The convolutional kernel size is 5×5. The output size is 8×8. The activation function is Sigmoid. The number of output feature is 12.
(5) The second pooling layer: the input size is 8×8. The size of pool window is 2×2, the output size is 4×4, the number of output feature is 12.
(6) Input layer of FCNN: The size is 192, which is equal to $4 \times 4 \times 12$.
(7) Hidden layer of FCNN: The size is 10, the activation function is Sigmoid.
(8) Output layer of FCNN: The size is 10, which can represent 10 different classes. For example: if the predicted result is class 1, the output is: [1,0,0,0,0,0,0,0,0,0], and if the predicted result is class 2, the output is: [0,1,0,0,0,0,0,0,0,0].

Secondly, parameters for simulations are configured as follows:

(1) Learning rate: $\eta = 0.01$ or $\eta = 0.1$.
(2) Total number of training processes (SP): $max\_SPs = 10$.
(3) Total number of adjustment cycles (AC): $max\_ACs = 6000$.
(4) Target error: $target\_err = 0.0000001$. In each SP, current error is calculated by $current\_err = L()$. $L()$ is loss function, where adopts MSE in this study. When $current\_err$ is smaller than $target\_err$, SP will be stop though the current AC is smaller than max_ACs.

In each SP, the current error is continuously reduced. Therefore, in order to get smaller error, $target\_err$ is set

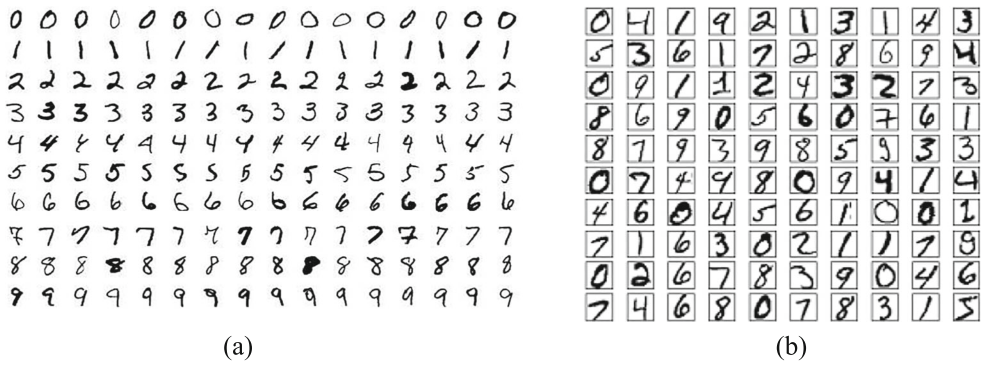
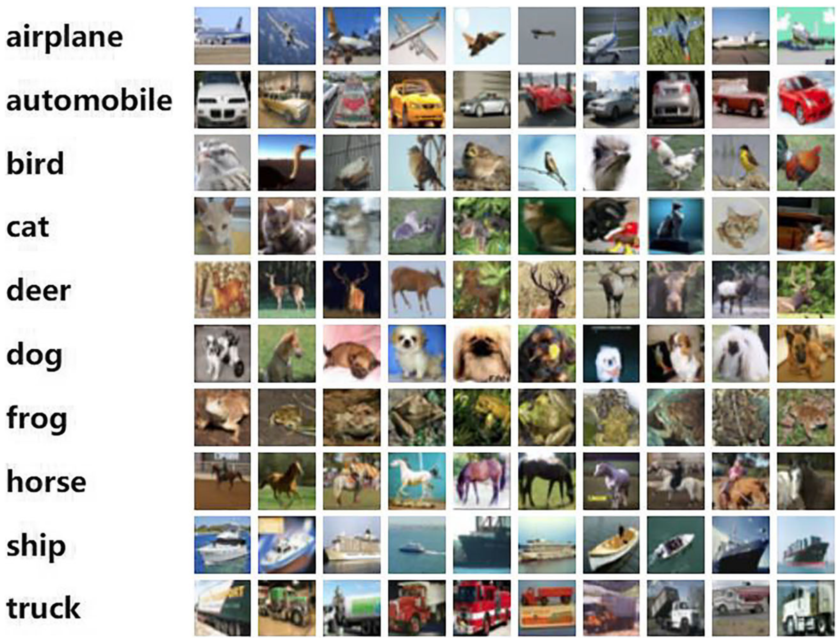


Fig. 5 Data Set of MNIST. **a** Training set. **b** Test set

Fig. 6 Data Set of CIFAR-10



to a very small value which cannot be reached within max_ACs in each SP.

(5) Total number of training samples taken in each AC: $BatchSize = 10$.

6.3 Experiments design of wCNN and wCwNN

Configuration of parameters of network structures and simulations of wCNN and wCwNN are listed in Table 2. There are two groups of experiments with different learning rate $\eta = 0.1$ and $\eta = 0.01$ respectively.

Comparative results will be recorded as follows: (1) feasibility (whether the algorithm is convergent), (2) minimum MSE (precision), (3) correct rate (accuracy) of all the test data, (4) running time (efficiency). The structure and experimental parameters configuration of wCNN, wCwNN are shown in Table 1.

Table 2 shows that the most significant difference is that: (1) The activation function of CPNN in wCNN and wCwNN is wavelet function, while the activation function of CNN is sigmoid function. (2) The second part of neural network (after CPNN) of wCwNN is wavlet neural network (wNN), while the second part of neural network of CNN and wCNN is FCNN.

7 Results

7.1 Results of the experiment of CNN

The learning rate is set as $\eta = 0.01$ and $\eta = 0.1$ respectively. Results are recorded as follows: (1) current error (MSE) of each AC in each SP; (2) the error rate of all test samples after each SP; (3) time spent in each SP. The above results are recorded in Table 2:

In the experiment of CNN, each SP has 6000 ACs. All the errors of 6000 ACs are recorded and drawn in Fig. 7. All the points of error values are drawn into the orange line and fitted to a blue line by linear regression. The blue line indicates the downward trend of the orange line. Fig. 7a shows the result of 1SP when $\eta = 0.01$. Fig. 7b shows the result of 1SP when $\eta = 0.1$. Fig. 7c shows the result of 10SPs when $\eta = 0.01$. Fig. 7d shows the result of 10SPs when $\eta = 0.1$.

7.2 Results of the experiments of wCNN

In experiment of the wCNN, learning rate is set as $\eta = 0.01$ and $\eta = 0.1$ respectively. Each SP contains 6000ACs. All the MSE in each ACs are recorded, which is shown in Fig. 8:

Statistic results of the above simulations are listed in Table 3, which includes: (1) the final MSE of each SP, (2) the error rate of each SP, (3) Consumed time of each SP.

7.3 Results of the experiments of wCwNN

In the experiment of the wCwNN, learning rate is set as $\eta = 0.01$ and $\eta = 0.1$ respectively. Each SP contains 6000ACs. All the MSE in each ACs are recorded, which is shown in Fig. 9.

Statistic results of the above simulations are listed in Table 4, which includes: (1) the final MSE of each SP, (2) the error rate of each SP, (3) consumed time of each SP.

Table 1 Configurations of CNN, wCNN and wCwNN experiments

No.	Parameter type	Parameter name	CNN	wCNN	wCwNN
1	1st to 5th layers	First type of NN	CPNN	wCPNN	wCPNN
2	2nd,4th layers	Activation function of convolutional layer	Sigmoid	Wavelet	Wavelet
3	1st layer	Dimension of the 1st layer	28 × 28	28 × 28	28 × 28
4	2nd layer	Dimension of 1st convolutional layer	28 × 28	28 × 28	28 × 28
5	3th layer	Dimension of 1st pooling layer	24 × 24	24 × 24	24 × 24
6	2nd,3th layers	Number of features	6	6	6
7	4th layer	Dimension of 2nd convolutional layer	12 × 12	12 × 12	12 × 12
8	5th layer	Dimension of 2nd1 pooling layer	8 × 8	8 × 8	8 × 8
9	4th,5th layers	Number of features	12	12	12
10	-1st, −2nd, −3th layers	Second type of NN	FCNN	FCNN	wNN
11	-3th layer	Dimension of -3th input layer	192	192	192
12	-2nd layer	Dimension of -2nd hidden layer	None	None	10
13	-2nd layer	Activation function of hidden layer	None	None	Wavelet
14	-1st layer	Dimension of -1st output layer	10	10	10
15	-1st layer	Activation function of output layer	Sigmoid	Sigmoid	Sigmoid
16	hyperparameters	Learning rate η	0.01 and 0.1	0.01 and 0.1	0.01 and 0.1
17	hyperparameters	Coefficient of inertia α	None	None	0.2
18	hyperparameters	max_SPs	10	10	10
19	hyperparameters	max_ACs	6000	6000	6000
20	hyperparameters	$target\_err$	0.0000001	0.0000001	0.0000001
21	hyperparameters	$BatchSize$	10	10	10

8 Discussion

8.1 Discussion of results of CNN

According to the experimental results of the CNN presented in Table 2 and Fig. 7, the main findings are as follows:

(1) Classification of MNIST can be completed by CNN, the correct rate is 90.03% (the error rate is 9.97%)

(2) CNN has a good ability for image classification, and the maximum correct rate can reach 90.8% (the error rate is at least 9.2%).

Table 2 Experiment results of CNN

Numbers of SPs and statistical item	Total Numbers of ACs	MSE $\eta = 0.01$	MSE $\eta = 0.1$	Error rate $\eta = 0.01$	Error rate $\eta = 0.1$	time $\eta = 0.01$	time $\eta = 0.1$
1	6000	0.317	0.095	30.22%	10.20%	82.86	88.53
2	6000	0.280	0.094	30.47%	10.09%	82.52	87.68
3	6000	0.299	0.091	26.93%	9.35%	83.20	89.84
4	6000	0.294	0.092	27.86%	9.96%	82.55	87.96
5	6000	0.310	0.089	33.45%	9.46%	82.79	89.70
6	6000	0.270	0.115	24.10%	12.05%	82.40	93.47
7	6000	0.269	0.090	25.45%	9.22%	82.36	95.83
8	6000	0.278	0.092	26.45%	10.15%	85.24	91.51
9	6000	0.288	0.092	27.31%	10.02%	85.12	96.90
10	6000	0.282	0.088	28.49%	9.20%	84.83	88.96
Minimum	6000	0.269	0.088	24.10%	9.20%	82.36	87.68
Maximum	6000	0.317	0.115	33.45%	12.05%	85.24	96.90
Average	6000	0.289	0.094	28.07%	9.97%	83.39	91.04

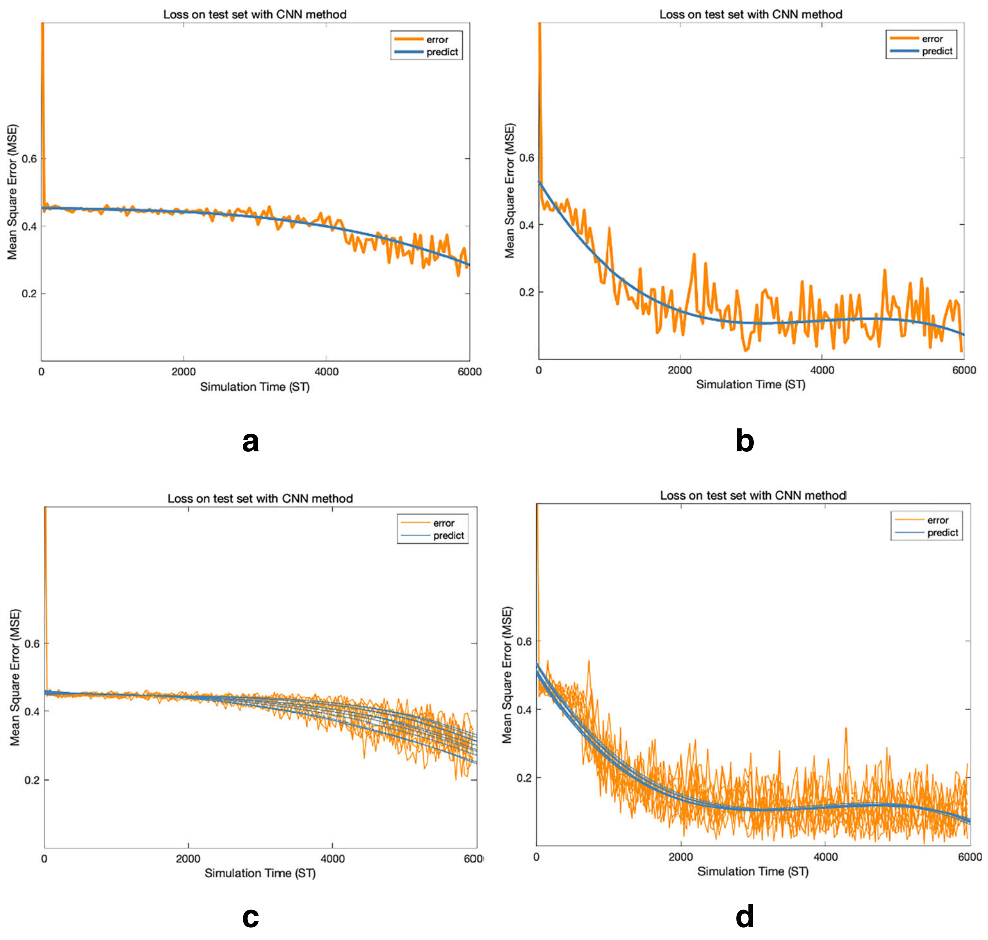


Fig. 7 MSE plot of CNN. **a** Result of 1SP when $\eta = 0.01$. **b** Result of 1SP when $\eta = 0.1$. **c** Result of 10SPs when $\eta = 0.01$. **d** Result of 10SPs when $\eta = 0.1$

(3) When the learning rate is increased (from $\eta = 0.01$ to $\eta = 0.1$), the MSE is significantly decreased (from 0.289 to 0.094), and the error rate is significantly decreased (from 28.07% to 9.97%). Time spent in simulation ($\eta = 0.1$) is lightly increased (from 83.39 to 91.04).

(4) The descending process of MSE is stable, and the results among 10SPs are close.

8.2 Discussion of the wCNN results

According to the experimental results of the wCNN presented in Table 3 and Fig. 8, the following findings can be drawn:

(1) wCNN algorithm is convergent. Classification of the MNIST can be completed by wCNN. The average correct rate is 92.78% (the average error rate of 7.22%).

(2) wCNN has a good (better than CNN) ability for image classification, and the maximum correct rate can reach 95.66% (the error rate is at least 4.34%).

(3) When the learning rate of wCNN is increased (from $\eta = 0.01$ to $\eta = 0.1$), the MSE is significantly decreased (from 0.184 to 0.088), and the error rate is significantly decreased (from 13.91% to 7.22%). Time spent in simulation ($\eta = 0.1$) is lightly increased (from 258.53 to 268.56).

(4) MSE reduction processes have differences, variance of final error of wCNN is greater than CNN. wCNN achieves a very low error rate (4.34%), but there are some high error rates (such as 18.71% and 18.35%) of wCNN experiments. With the research of Liu in 2015 [15], the wavelet network's ability to make the training MSE smaller is a result of sacrificing network stability, because it has the advantage of jumping out of the local minimum, which is not available in classical BP network and RBF network. However, this advantage also brings the disadvantage that the error decline during the learning process is more oscillating.

8.3 Discussion of experimental results of wCwNN

According to the experimental results of wCwNN presented in Fig. 9 and Table 4, the following conclusions can be drawn:

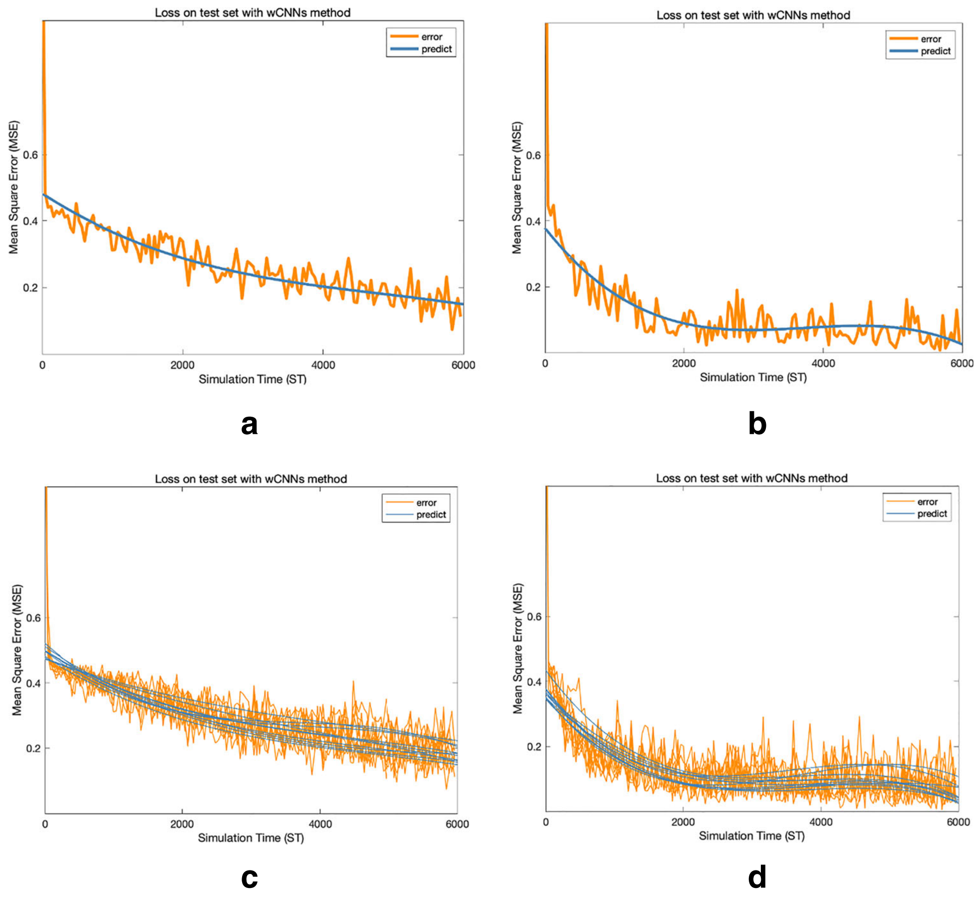


Fig. 8 MSE plot of wCNN. **a** Result of 1SP when $\eta = 0.01$. **b** Result of 1SP when $\eta = 0.1$. **c** Result of 10SPs when $\eta = 0.01$. **d** Result of 10SPs when $\eta = 0.1$

(1) wCwNN is convergent. It can complete the task of classification for MNIST dataset, and the average accuracy is 96.57% (the average error rate is 3.43%).

(2) wCwNN has a good ability (better than CNN and wCNN) of image classification, and the maximum correct rate can reach 97.04% (the minimal error rate is 2.96%).

(3) When the learning rate of wCwNN is increased (from $\eta = 0.01$ to $\eta = 0.1$), the MSE is decreased (from 0.068 to 0.054), and the error rate is significantly decreased

Table 3 Experiment results of wCNN

Numbers of SPs and statistical item	Total Numbers of ACs	MSE $\eta = 0.01$	MSE $\eta = 0.1$	Error rate $\eta = 0.01$	Error rate $\eta = 0.1$	time $\eta = 0.01$	time $\eta = 0.1$
1	6000	0.152	0.074	9.79%	6.96%	249.18	266.60
2	6000	0.169	0.098	9.47%	7.85%	256.24	280.40
3	6000	0.182	0.085	14.52%	7.62%	255.28	278.98
4	6000	0.211	0.101	18.35%	8.62%	262.09	279.67
5	6000	0.169	0.077	12.03%	6.44%	261.10	276.26
6	6000	0.179	0.087	14.11%	7.43%	260.90	262.74
7	6000	0.213	0.056	17.26%	4.34%	268.64	260.43
8	6000	0.184	0.056	14.30%	4.57%	258.81	261.49
9	6000	0.227	0.177	18.71%	12.62%	259.27	260.78
10	6000	0.159	0.069	10.54%	5.76%	253.79	258.25
Minimum	6000	0.152	0.056	9.47%	4.34%	249.18	258.25
Maximum	6000	0.227	0.177	18.71%	12.62%	268.64	280.40
Average	6000	0.184	0.088	13.91%	7.22%	258.53	268.56

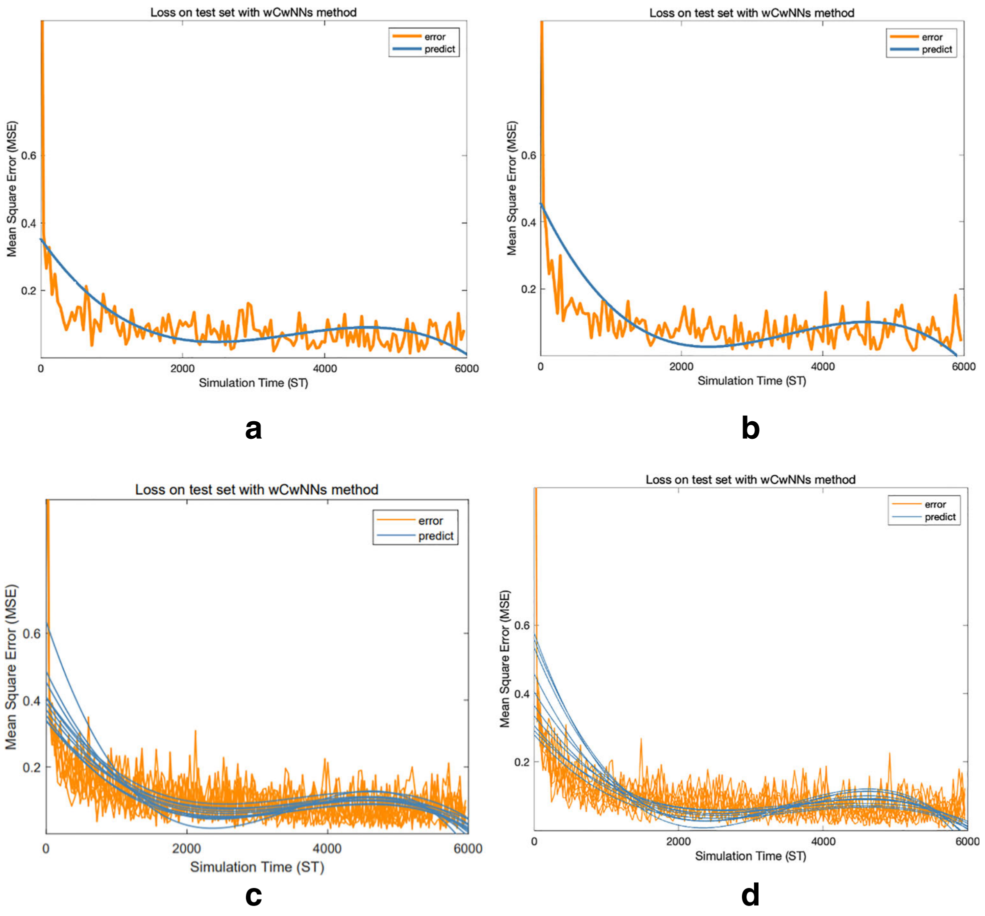


Fig. 9 MSE plot of wCwNN. **a** Result of 1SP when $\eta = 0.01$. **b** Result of 1SP when $\eta = 0.1$. **c** Result of 10SPs when $\eta = 0.01$. **d** Result of 10SPs when $\eta = 0.1$

(from 5.17% to 3.43%). Time spent in simulation ($\eta = 0.1$) is lightly increased (from 357.23 to 368.81).

(4) The descending process of MSE is stable, and the differences among 10SPs are not significant.

9 Conclusion

Firstly, MNIST dataset is adopted to verify the proposed methods in this study, and CNN is implemented to finish the

Table 4 Experiment results of wCwNN

Numbers of SPs and statistical item	Total Numbers of ACs	MSE $\eta = 0.01$	MSE $\eta = 0.1$	Error rate $\eta = 0.01$	Error rate $\eta = 0.1$	Time (s) $\eta = 0.01$	Time (s) $\eta = 0.1$
1	6000	0.056	0.053	4.04%	3.21%	344.84	367.55
2	6000	0.061	0.060	4.41%	3.95%	352.99	378.07
3	6000	0.064	0.049	4.83%	3.23%	354.61	377.98
4	6000	0.073	0.058	5.80%	3.39%	363.21	378.13
5	6000	0.069	0.055	4.88%	2.96%	360.81	375.22
6	6000	0.063	0.052	4.59%	3.63%	359.04	360.77
7	6000	0.076	0.054	6.78%	3.09%	368.92	363.55
8	6000	0.069	0.045	5.09%	3.08%	357.07	362.65
9	6000	0.092	0.053	7.41%	3.88%	361.10	365.03
10	6000	0.062	0.064	3.90%	3.89%	349.74	359.20
Minimum	6000	0.056	0.045	3.90%	2.96%	344.84	355.51
Maximum	6000	0.092	0.064	7.41%	3.95%	368.92	378.13
Average	6000	0.068	0.054	5.17%	3.43%	357.23	368.81

Table 5 Comparative results of CNN, wCNN and wCwNN experiments (Learning Efficiency = 0.01)

Numbers of SPs and statistical item	Total Numbers of ACs	MSE of CNN	MSE of wCNN	MSE of wCwNN	Error rate of CNN	Error rate of wCNN	Error rate of wCwNN	Time of CNN (s)	Time of wCNN (s)	Time of wCwNN (s)
1	6000	0.317	0.152	0.056	30.2%	9.8%	4.0%	82.86	249.18	344.84
2	6000	0.280	0.169	0.061	30.5%	9.5%	4.4%	82.52	256.24	352.99
3	6000	0.299	0.182	0.064	26.9%	14.5%	4.8%	83.20	255.28	354.61
4	6000	0.294	0.211	0.073	27.9%	18.4%	5.8%	82.55	262.09	363.21
5	6000	0.310	0.169	0.069	33.5%	12.0%	4.9%	82.79	261.10	360.81
6	6000	0.270	0.179	0.063	24.1%	14.1%	4.6%	82.40	260.90	359.04
7	6000	0.269	0.213	0.076	25.5%	17.3%	6.8%	82.36	268.64	368.92
8	6000	0.278	0.184	0.069	26.5%	14.3%	5.1%	85.24	258.81	357.07
9	6000	0.288	0.227	0.092	27.3%	18.7%	7.4%	85.12	259.27	361.10
10	6000	0.282	0.159	0.062	28.5%	10.5%	3.9%	84.83	253.79	349.74
Minimum	6000	0.269	0.152	0.056	24.1%	9.5%	3.9%	82.36	249.18	344.84
Maximum	6000	0.317	0.227	0.092	33.5%	18.7%	7.4%	85.24	268.64	368.92
Mean	6000	0.289	0.184	0.068	28.1%	13.9%	5.2%	83.39	258.53	357.23
Variance	0	0.000262	0.000616	0.000104	0.000744	0.001186	0.000135	1.41	28.16	48.66

task of classification. The correct rate of CNN is more than 90%. Secondly, wCNN is proposed, the activation function of the convolutional network in CNN is replaced by the wavelet function. Thirdly, the FCNN of CNN and wCNN is replaced by wNN, wCwNN is proposed. With the same hyperparameters, the comparative results of experiments among CNN, wCNN and wCwNN are shown in Tables 5 and 6.

The MSE of CNN, wCNN and wCwNN in each ACs are drawn in Fig. 10.

We took the same experiments on both the MNIST dataset and the CIFAR10 dataset. Results are shown in the Experiment section. **The comparison between MNIST dataset and the CIFAR-10 dataset is as follows**:

In the experiment of CIFAR-10 dataset, wCNN's performance(the Mean MSE is 0.144948, the mean error rate is 0.20095) is better than CNN(the Mean MSE is 0.29845, and the mean error rate is 0.205107), while the wCwNN's performance(the Mean MSE is 0.134675, and the mean error rate is 0.18145) is better than wCNN.

Table 6 Comparative results of CNN, wCNN and wCwNN experiments (Learning Efficiency = 0.1)

Numbers of SPs and statistical item	Total Numbers of ACs	MSE of CNN	MSE of wCNN	MSE of wCwNN	Error rate of CNN	Error rate of wCNN	Error rate of wCwNN	Time of CNN (s)	Time of wCNN (s)	Time of wCwNN (s)
1	6000	0.095	0.074	0.053	10.20%	6.96%	3.21%	88.53	266.60	344.84
2	6000	0.094	0.098	0.060	10.09%	7.85%	3.95%	87.68	280.40	352.99
3	6000	0.091	0.085	0.049	9.35%	7.62%	3.23%	89.84	278.98	367.55
4	6000	0.092	0.101	0.058	9.96%	8.62%	3.39%	87.96	279.67	378.07
5	6000	0.089	0.077	0.055	9.46%	6.44%	2.96%	89.70	276.26	377.98
6	6000	0.115	0.087	0.052	12.05%	7.43%	3.63%	93.47	262.74	378.13
7	6000	0.090	0.056	0.054	9.22%	4.34%	3.09%	95.83	260.43	375.22
8	6000	0.092	0.056	0.045	10.15%	4.57%	3.08%	91.51	261.49	360.77
9	6000	0.092	0.177	0.053	10.02%	12.62%	3.88%	96.90	260.78	363.55
10	6000	0.088	0.069	0.064	9.20%	5.76%	3.89%	88.96	258.25	362.65
Minimum	6000	0.088	0.056	0.045	9.20%	4.34%	2.96%	87.68	258.25	365.03
Maximu	6000	0.115	0.177	0.064	12.05%	12.62%	3.95%	96.90	280.40	359.20
Mean	6000	0.094	0.088	0.054	9.97%	7.22%	3.43%	91.04	268.56	355.51
Variance	0	0.000060	0.001214	0.000029	0.000069	0.000554	0.000014	10.89	83.57	131.67

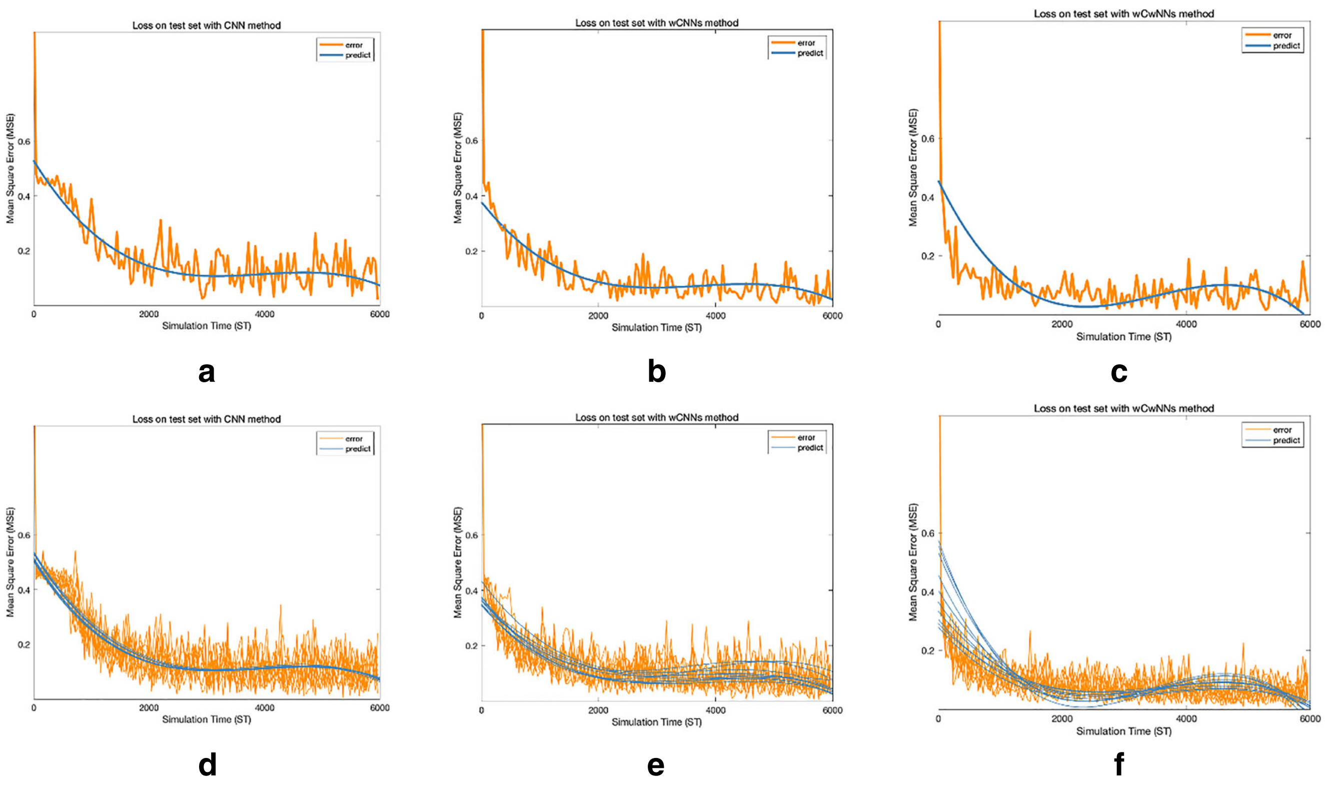


Fig. 10 MSE plot of CNN, wCNN and wCwNN (Learning efficiency is 0.1). **a** Result of 1SP of CNN. **b** Result of 1SP of wCNN. **c** Result of 1SP of wCwNN. **d** Results of 10SPs of CNN. **e** Results of 10SPs of wCNN. **f** Results of 10SPs of wCwNN

According to the comparative experimental results of CNN, wCNN and wCwNN shown in Tables 5, 6 and 7 **and** Fig. 10**. The following findings can be drawn:**

(1) CNN, wCNN and wCwNN are convergent, the task of classification for MNIST dataset can be completed by all above methods.
(2) When the learning rate of CNN, wCNN and wCwNN are increased (from $\eta = 0.01$ to $\eta = 0.1$), the MSE of each algorithm is decreased significantly, and the error rate of each experiment is decreased significantly, while the consumed time is increased slightly.
(3) wCwNN has the highest precision (the minimum MSE is 0.045), and the precision of wCNN (the minimum MSE is 0.088) is less than the precision of CNN (the minimum MSE is 0.094).
(4) The wCwNN has the highest accuracy (the minimum error rate is 3.43%, when the $\eta = 0.1$), the error rate of wCNN (the minimum error rate is 7.22%, when the $\eta = 0.1$), both of them are less than the error rate of CNN (9.97%).
(5) The variance of MSE of trained wCwNN is the smallest (the variance is 0.000014). The variance of MSE of trained wCNN is the largest (the variance is 0.000554). The variance of MSE of trained CNN is 0.000069.
(6) The wCwNN (average time of SP is 131.67) consumes more time than wCNN (average time of SP is 83.57), and wCNN consumes much more time than CNN (average time of SP is 10.89).

Table 7 Comparative results of different datasets (Learning Efficiency = 0.1)

Network	Statistics	MINIST	CIFAR-10
CNN	Minimum MSE	0.088	0.26
	Minimum Error rate	0.0092	0.188416
	Maximu MSE	0.115	0.4465
	Maximu Error rate	0.1205	0.261128
	Mean MSE	0.094	0.29845
	Mean Error rate	0.0997	0.205107
	MSE Variance	0.007346	0.052135
	Error rate Variance	0.007892	0.019824
wCNN	Minimum MSE	0.056	0.119091
	Minimum Error rate	0.119091	0.17
	Maximu MSE	0.0434	0.167035
	Maximu Error rate	0.17	0.244
	Mean MSE	0.177	0.144948
	Mean Error rate	0.167035	0.20095
	MSE Variance	0.1262%	0.014222
	Error rate Variance	0.244	0.024691
wCwNN	Minimum MSE	0.045	0.109378
	Minimum Error rate	0.0296	0.1495
	Maximu MSE	0.064	0.148659
	Maximu Error rate	0.0395	0.2035
	Mean MSE	0.054	0.134675
	Mean Error rate	0.0343	0.18145
	MSE Variance	0.005139	0.01195
	Error rate Variance	0.003569	0.017453

In summary (1) the proposed wCwNN is successfully improved based on wCNN, and the proposed wCNN is successfully improved based on CNN. (2) Both wCwNN and wCNN have higher precession (MSE is smaller) than CNN, and the precision of wCwNN is the highest. (3) Both wCwNN and wCNN have higher accuracy (error rate is smaller) than CNN, and the accuracy of wCwNN is the highest. Both improvements of wCNN and wCwNN lead to more time comsuming in each SPs.

In the future, the research we continue to do is as follows (1)We should find a mechanism so that wavelet function will not overextend the minimum value while maintaining the ability to jump out of the local minimum value. (2) We would improve the learning ability of CNN, wCNN and wCwNN by expanding the network structure like depth of network. Furthermore, the proposed wCNN and wCwNN can be used as a neuron to build a deeper neural network. (3) We would conduct more experiments to verify the performances of the improved methods. .

References

1. Bateux Q, Marchand E, Leitner J, Chaumette F, Corke P (2017) Visual servoing from deep neural networks. arXiv preprint arXiv:1705.08940
2. Cao L, Hong Y, Fang H, He G (1995) Predicting chaotic time series with wavelet networks. Physica D 85(1–2):225–238
3. Chang J, Sitzmann V, Dun X, Heidrich W, Wetzstein G (2018) Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. Sci Rep 8(1):1–10
4. Chen H, He X, Qing L, Xiong S, Nguyen TQ (2018) Dpw-sdnet: Dual pixel-wavelet domain deep cnns for soft decoding of jpeg-compressed images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 711–720
5. De Silva D, Vithanage H, Fernando K, Piyatilake I (2020) Multi-path learnable wavelet neural network for image classification. In: Twelfth International Conference on Machine Vision (ICMV 2019), vol. 11433, p. 114331O. International Society for Optics and Photonics Res improv wavelet convolutional wavelet neural netw 35
6. Fujieda S, Takayama K, Hachisuka T (2017) Wavelet convolutional neural networks for texture classification. arXiv preprint arXiv:1707.07394
7. Fukushima K (1980) Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 36(4):193–202
8. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778
9. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
10. Jiang Y, Chen L, Zhang H, Xiao X (2019) Breast cancer histopathological image classification using convolutional neural networks with small se-resnet module. PLoS One 14(3):e0214587
11. Khan A, Sohail A, Zahoora U, Qureshi AS (2019) A survey of the recent architectures of deep convolutional neural networks. arXiv preprint arXiv:1901.06032
12. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
13. Kiskin I, Orozco BP, Windebank T, Zilli D, Sinka M, Willis K, Roberts S (2017) Mosquito detection with neural networks: the buzz of deep learning. arXiv preprint arXiv:1705.05180
14. LeCun Y, Bengio Y, Hinton G (1988) Deep learning. Nature 521(7553), 436–444 (2015) 15. LeCun, Y., Touresky, D., Hinton, G., Sejnowski, T.: a theoretical framework for backpropagation. In: proceedings of the 1988 connectionist models summer school, vol. 1, pp. 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann
15. Liu J (2014) research of adaptive wavelet neural network (awnn) and ann based control system intelligent applications
16. Liu P, Zhang H, Lian W, Zuo W (2019) Multi-level wavelet convolutional neural networks. IEEE Access 7:74973–74985
17. Liu Y, Li Q, Sun Z (2019) Attribute-aware face aging with wavelet-based generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 11877–11886
18. Mallat S (2008) A wavelet tour of signal processing: The sparse way (academic, burlington ma)
20. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5(4):115–133
21. Pati YC, Krishnaprasad PS (1993) Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations. IEEE Trans Neural Netw 4(1):73–85
22. R, G.: Fast r-cnn. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit pp. 1440–1448 (2015)
23. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by backpropagating errors. nature 323(6088):533–536
24. Sakkari M, Zaied M (2015) An architecture of distributed beta wavelet networks for large image classification in mapreduce. In: 2015 15th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 523–527. IEEE
25. Savareh BA, Emami H, Hajiabadi M, Azimi SM, Ghafoori M (2019) Wavelet-enhanced convolutional neural network: a new idea in a deep learning paradigm. Biomed Engin Biomedizinische Technik 64(2):195–205
26. Sifuzzaman M, Islam M, Ali M (2009) Application of wavelet transform and its advantages compared to fourier transform
27. Song Y, Hu QV, He L (2019) P-cnn: enhancing text matching with positional convolutional neural network. Knowl-Based Syst 169: 67–79
28. Waibel A, Hanazawa T, Hinton G, Shikano K, Lang KJ (1989) Phoneme recognition using time-delay neural networks. IEEE Trans Acoust Speech Signal Process 37(3):328–339
29. Wang F, Yu Y, Zhang Z, Li J, Zhen Z, Li K (2018) Wavelet decomposition and convolutional lstm networks based improved deep learning model for solar irradiance forecasting. Appl Sci 8(8):1286 36 Jing-Wei LIU1,2 et al.

30. Wang JZ (2001) Wavelets and imaging informatics: a review of the literature. J Biomed Inform 34(2):129–141
31. Wang X Ma S YB (2014) Effects of visual perception training on hippocampal neural cell plasticity and changes in learning and memory functions. J Zhejiang Univ (Med Sci) pp. 601–604
32. Xiao H, H.T.D.Z.: A damage detection method for grout sleeve splicingat assembly column based on the sym wavelet and bp neural network. Structural Engineers (2018)
33. Zhang Q (1997) Using wavelet network in nonparametric estimation. IEEE Trans Neural Netw 8(2):227–236

Jingwei Liu was born in Beijing China in 1982. He received Ph.D. degree in Control Science and Engineering (Pattern recognition and intelligent system) from the Department of Electronic Information and Control Engineering, Beijing University of Technology, Beijing, China in 2014. From 2014, he has been an Associate Professor in Information College, Capital University of Economics and Business. His research interests include artificial intelligence, automatic control, intelligent control and intelligent systems. He served as Director of the Software R&D Center of Information College of Capital University of Economics and Business. He has published 3 academic monographs, more than 20 academic articles and 20 patents. He received the first prize of teaching achievements, the first prize of teaching basic skills competition, the most popular student recognition award.

Fangling Zuo was born in Jiangxi China in 1997. She is studying for her M.S. degree in School of Statistics and Mathematics, Central University of Finance and Economics. She received her B.S. degree in Information College, Capital University of Economics and Business in 2019. Her research interests include artificial intelligence, neural network, big data distributed computing, Bayesian inference. She has published some patents and academic articles. She received the national scholarship, President Scholarship of Capital University of Economics and Business, the honor of Beijing Merit Student and Beijing Outstanding Graduate.

Yingxiao Guo was born in Beijing, China in 1999. She is studying for her B.S. degree in Information System, from Capital University of Economics and Business. She is a research member of the Big Data and Intelligent Transportation Research Center of Management Engineering College. Her research interests include natural language processing and deep learning.

Tianyue Li was born in Hebei, China in 1997. She is studying for her M.S. degree in Management Engineering College, Capital University of Economics and Business. She received her B.S. degree in Quality and Technical Supervision College, Hebei University in 2019. She is a research member of the Big Data and Intelligent Transportation Research Center of Management Engineering College. Her research interests include artificial intelligence, deep learning and intelligent control.

Jiaming Chen was born in Beijing, China in 1994. Since 2019, he has been pursuing the PhD in computer science in the Beijing University of Technology. He received the M.S. degree in software engineering from the Capital Normal University, Beijing, China, in 2019. In 2018, he was a researcher and an assistant tutor in the Big Data College of the Huike Education Corporation. From 2016 to 2018, he was the frontend architect of the Chengdu iDelta Technology Co., Ltd. His research interest includes artificial intelligence, brain computer interface, and data mining. He received the scholarship of the Capital Normal University from 2016 to 2018 and won the third prize in the contest of Lan Qiao Cup in 2017.