It may be time to perfect the neuron of artificial neural network

Gang Liu [1,1,1,1,1,1,1,1,1,1,2]

[1]Xian Jiaotong University
[2]Zhengzhou University; Xian Jiaotong University;

November 8, 2023

Abstract

In recent years, artificial neural networks (ANNs) have won numerous contests in pattern recognition, machine learning, and artificial intelligence. The basic unit of an ANN is to mimic neurons in the brain. Neuron in ANNs is expressed as f(wx+b) or f(wx).This structure does not consider the information processing capabilities of dendrites. However, recently, studies shown that dendrites participate in pre-calculation in the brain. Concretely, biological dendrites play a role in the pre-processing to the interaction information of input data. Therefore, it's time to perfect the neuron of the neural network. This paper, add dendrite processing section, presents a novel artificial neuron, according to our previous studies (CR-PNN or Gang transform). The dendrite processing section can be expressed as WA.X. Because I perfected the basic unit of ANNs-neuron, there are so many networks to try, this article gives the basic architecture for reference in future research.

It may be time to improve the neuron of artificial neural network

Gang Liu
E-mail: gangliu_@zzu.edu.cn

Abstract

Artificial neural networks (ANNs) have won numerous contests in pattern recognition, machine learning, and artificial intelligence in recent years. The neuron of ANNs was designed by the stereotypical knowledge of biological neurons 70 years ago. Artificial Neuron is expressed as $f(wx+b)$ or $f(WX)$. This design does not consider dendrites' information processing capacity. However, some recent studies show that biological dendrites participate in the pre-calculation of input data. Concretely, biological dendrites play a role in extracting the interaction information among inputs (features). Therefore, it may be time to improve the neuron of ANNs. In this study, some dendritic modules with excellent properties are proposed and added to artificial neurons to form new neurons named Gang neurons. E.g., The dendrite function can be expressed as $W^{i,i-1}A^{i-1} \circ A^{0|1|2|\dots|i-1}$. The generalized new neuron can be expressed as $f(W(W^{i,i-1}A^{i-1} \circ A^{0|1|2|\dots|i-1}))$.The simplified new neuron be expressed as $f(\sum(WA \circ X))$. After improving the neurons, many networks can be tried.This paper shows some basic architecture for reference in the future. **Up to now, others and the author have applied Gang neurons to various fields, and Gang neurons show excellent performance in the corresponding fields.**

Interesting things: (1) The computational complexity of dendrite modules ($W^{i,i-1}A^{i-1} \circ A^{i-1}$) connected in series is far lower than Horner's method. Will this speed up the calculation of basic functions in computers? (2) The range of sight of animals has a gradient, but the convolution layer does not have this characteristic. This paper proposes receptive fields with a gradient. (3) The networks using Gang neurons can delete Fully-connected Layer. In other words, the parameters in Fully-connected Layers are assigned to a single neuron, which reduces the parameters of a network for the same mapping capacity. (4) ResDD(ResDD modules+One Linear module) can replace the current ANNs' Neurons. ResDD has controllable precision for better generalization capability.

Index Terms

Neural network, neuron, DD, transform, Gang neuron, Receptive fields with gradient, Acceleration algorithm

CONTENTS

1. **Current applications of Gang neurons in various fields can be viewed in Section VIII.** This study will be continuously updated in the future.
2. Some code and information can be found on GitHub: https://github.com/liugang1234567/Gang-neuron
Explanation video:https://space.bilibili.com/176499425/channel/seriesdetail?sid=1820868

Gang neuron (the combination of dendrite in this paper and function $f(.)$) was published on **June 16, 2020**.
Citation:[1] Liu, Gang (2020): It may be time to improve the neuron of artificial neural network. IEEE TechRxiv. DOI:10.36227/techrxiv.12477266

I. Introduction

AN artificial neural network(ANNs) is an algorithmic architecture that imitates the biological brain [1], [2]. ANNs are gradually changing our lives and improving the world [3]. Artificial neural networks, particularly "deep learning" [4], have recently made impressive advances in many fields, such as machine vision, speech recognition, autonomous vehicles, and machine translation. Thus, in the tech world today, optimism is high. Some people think ANNs are approaching human intelligence gradually [2], [3].

Neurons are the fundamental units of the biological brain. Accordingly, the fundamental units of ANNs are artificial neurons. With the development of technology, people have a deeper understanding of the biological brain and neurons. However, the artificial neuron has maintained its original architecture. It may be time to improve the neuron of the artificial neural network.

The following briefly introduces the necessary knowledge and latest developments in artificial neural and biological neural networks.

A. Artificial neural networks

An ANN consists of many interconnected functional units or neurons and can learn the relationship between input space and output space. Thus, they are usually used to solve classification and regression problems [5]. Over the years, based on the characteristics of the application and data, different ANN architectures have been developed, such as Convolutional Neural Networks (CNN) [6], [7] in computer vision, Recurrent Neural Networks (RNN) [8], or Long Short Term Memory Networks (LSTM) [9] in sequence and time series modeling, and Generative Adversarial Network (GANs) [10].

In the last decade, some newer architectures have been developed to endure the need to develop human-like efficient machines in different application areas. From the perspective of information, they can be divided into a network focusing on spatial information [CNN, the representative field is computer vision (CV).] and a network focusing on temporal information [RNN, the representative field is natural language processing (NLP).]. There are several newer CNN architectures and efficient mechanisms: Alexnet [7], VGG [11], Googlenet [12], Inception-V3 [13], ResNet [14], ResNeXt [15], Convolutional Block Attention Module (CBAM) introduced by Woo et al. [16], and competitive squeeze-excitation (SE) mechanism introduced by Hu et al. [17]. There are several newer RNN architectures and efficient mechanisms: Deep Recurrent Attentive Writer (DRAW) [18], Grid Long Short-Term Memory [19], gating mechanism introduced by Jing et al. [20], and factorized recurrent network architecture introduced by Belletti et al. [21].

In addition, ANN learning strategies that approximate the relationship between input and output spaces by changing the weight distribution are investigated. Lately, success has been achieved in many techniques, such as L1 and L2 regularization [22], batch normalization [23], a good collection of weight initialization techniques [24], [25], and a good set of activation functions [5].

B. Biological neural network

The human brain has approximately 100 billion biological neurons, and neurons are connected via specialized structures known as synapses to sense stimulations and pass signals to other neurons [26]. A neuron is the fundamental structural and functional unit of the neural information network and comprises a cell body (soma), dendritic trees, and an axon [27], [28]. The most extended parts of many neurons are dendrites [29], and the active electrical properties of dendrites shape neuronal input.

In the field of biology, researchers have been studying the mechanism of the element of neurons over the years [27], [30]–[32]. Recently, a study discovered a class of calcium-mediated dendritic action potentials (dCaAPs) [32], [33]. Here, we quote the original words in the literature: "These dCaAPs enabled the dendrites of individual human neocortical pyramidal neurons to classify linearly nonseparable inputs—a computation conventionally thought to require multilayered networks." [32] ***Computing exists at the intersection of dendrites. This means that biological dendrites participate in pre-calculation in a biological neural network.***

Reviewing the previous ANNs, I found few studies highlight improving the neuron of ANNs in applications(Up to 2020).

II. Traditional artificial neuron

In 1943, McCulloch and Pitts proposed ANN models with adjustable weights [34]. More than a decade later, Rosenblatt put forward the Perceptron Learning Algorithm [35]. In 1986, Rumelhart et al. proposed learning representations by back-propagating errors [36]. By then, a typical artificial neuron was established.

Fig. 1 shows the traditional artificial neuron.

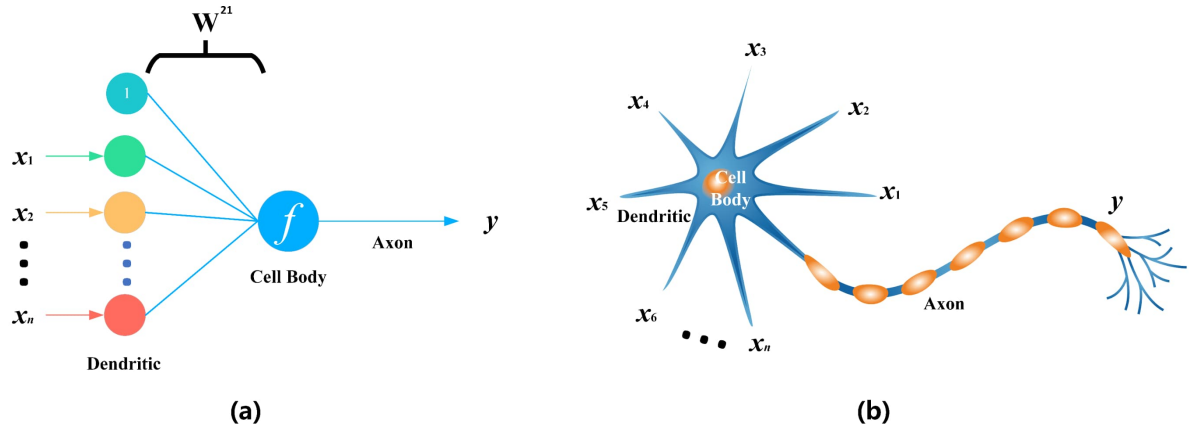


Fig. 1. Traditional neuron. (a) Traditional artificial neuron. (b) Traditional imitation for biological neuron.

This architecture ignores the pre-calculation of dendrites. If X is the input and f is the nonlinear activation function, the output y can be represented by,

$$y = f(W^{21}X) \tag{1}$$

Where W^{21} is the weight matrix, $\{1, x_1, x_2, \cdots, x_n\} = X$. In the back-propagation procedure, the weight matrix can be updated with,

$$W^{21}_{new} = W^{21} - \eta \frac{\partial E}{\partial W^{21}} \tag{2}$$

Where W^{21}_{new} is the updated weight matrix for W_{21}, E is the cost function or errors from the later layer in back-propagation, and η is the learning rate.

III. Novel Gang neuron

For the biological neurons, the XOR operation is performed in the dendrites with dCaAPs [32], [33], and AND/OR operations are performed at the soma and at tuft and basal dendrites with sodium and NMDA spikes [37]–[39]. The XOR/AND/OR operation means the pre-processing to the interaction information of input data. In the task of understanding a picture, it refers to the relationship between parts of an input-picture. In the task of understanding an article or a speech, it refers to the relationship between words of sentences. However, traditional artificial neurons only simulate the cell body.

In this section, some dendritic modules with excellent properties are proposed and added to artificial neurons to form new neurons named Gang neurons(Dendrite+Cell body, see Fig. 2).

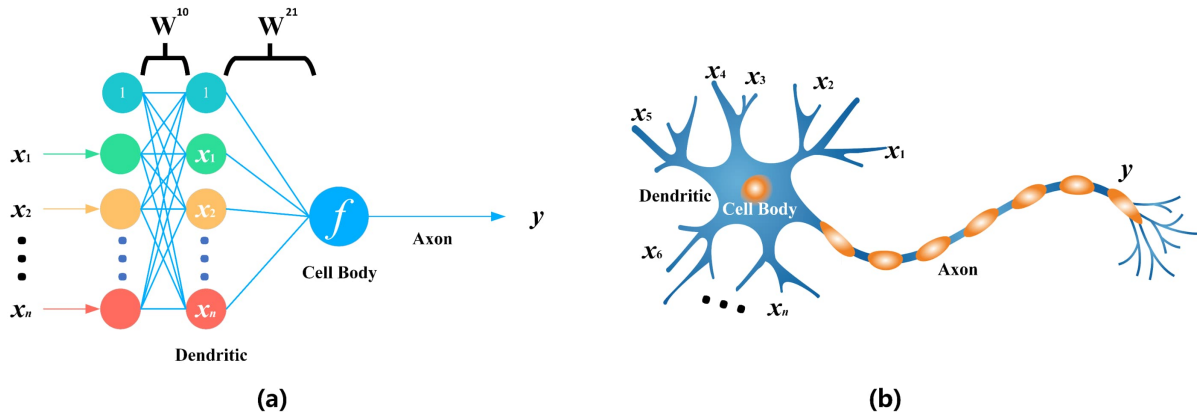


Fig. 2. Novel neuron. (a) Gang neuron. (b) Novel Imitation for biological neuron.

A. Novel Dendrite

1) Dendrite module ($\circ X$):

In previous studies, Gang et al. [40] presented a dendrite module named DD module, and introduced the logical relation of inputs [40], [41]. DD module has low computational complexity,and solves linearly inseparable problems by Hadamard product. The module is represented as follows.

$$A^i = W^{i,i-1}A^{i-1} \circ X \tag{3}$$

Where A^{i-1} and A^i are the input and output of the module, respectively, and X is the original input. $\circ$ denotes Hadamard product.

In Fig. 2a, the dendrite is simulated by one module. The "dendrite" contains the interaction item between both inputs and can be represented as follows.

$$A = W^{10}X \circ X \tag{4}$$

Where A is the output of dendrite or the input of the cell body. $\circ$ denotes Hadamard product.

Additionally, the dendrite can also be simulated by more modules. The "dendrite" in Fig.3 contains the interaction items between multiple inputs. The number of modules plus 1 is the number of interactive inputs. For example, if we use two modules to simulate the dendrite, the "dendrite" contains the interaction items among three inputs and can be represented as follows.

$$\begin{cases} A^1 = W^{10}X \circ X \\ A^2 = W^{21}A^1 \circ X \end{cases} \tag{5}$$

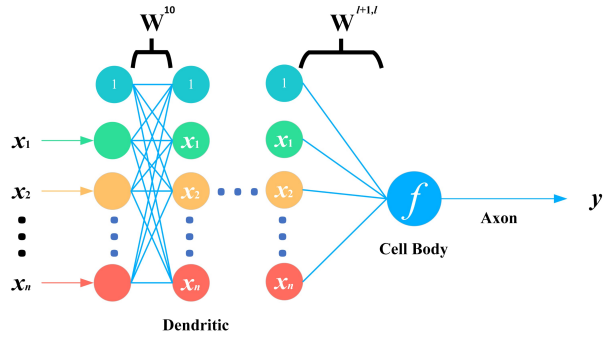


Fig. 3. Gang neuron with multiple dendrite modules.

Where A^2 is the output of dendrite or the input of the cell body. $\circ$ denotes Hadamard product. The mapping capacity of Gang neuron can be adjusted by the number of dendrite modules.

2) Generalized Dendrite module ($\circ A^i$):

Fig. 4 shows a generalized Dendrite module. The module is represented as follows.

$$A^i = W^{i,i-1}A^{i-1} \circ A^{0|1|2|...|i-1} \tag{6}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. $A^{0|1|2|...|i-1}$ is any of A. $\circ$ denotes Hadamard product.

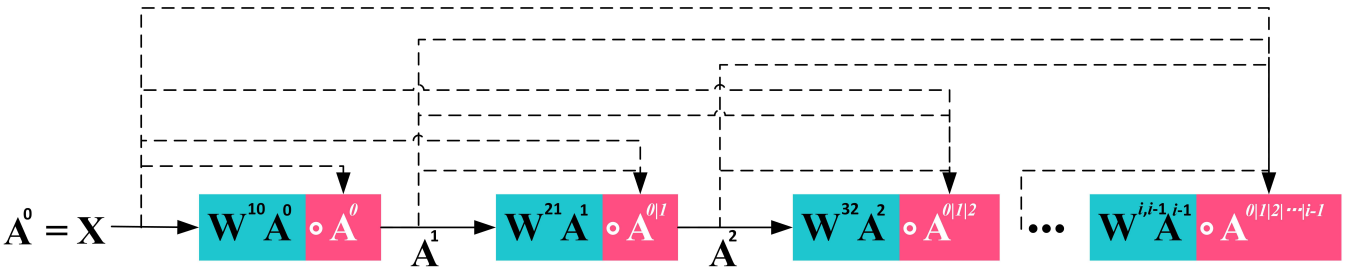


Fig. 4. Generalized Dendrite module. '|' denotes 'or'. The dotted line represents any of them.

We can adjust the promotion degree or the number of interactive variables added for the single module by $A^{0|1|2|...|i-1}$. When we add a module, the degree of network can increase by at least one and by i at most.

When $A^{0|1|2|...|i-1} = A^0$, the degree increases by 1 for one module, which is similar to Horner's method. The degree increases with the number of modules linearly. For a certain degree of network, the computational complexity is $n-1$, , where n denotes the number of modules.

When $A^{0|1|2|...|i-1} = A^{i-1}$, the degree increases by i for one module. The degree increases with the number of modules exponentially. For a certain degree of network, the computational complexity is approximate log_2^n, where n denotes the number of modules. This is similar to exponentiation by squaring. However, exponentiation by squaring is only used for calculation, and the modules in this paper are used for approximation. The weight matrix in Fig. 4 can be solved for by error backpropagation.

Here, I give a off-topic special USES. When this module is connected in series, it is used for polynomial calculation (Fig. 4). When $A^{0|1|2|...|i-1} = A^{i-1}$, there is no doubt that its computational complexity is far lower than Horner's method.This can change many things. The typical functions (e.g., sin) are stored in the form of polynomial coefficients in the computer. The value of the function is calculated by polynomial when we invoke these functions. We can solve for the polynomial coefficients of the new module ($A^{0|1|2|...|i-1} = A^{i-1}$) and store these coefficients in the computer. The value of the function is calculated by the network in Fig. 4 when we invoke these functions. *Note:For deeper levels, we can use ResDD or DenseDD (see the ResDD and DenseDD section).*

3) Generalized Residual Dendrite module ($\circ A^i$):

In order to make DD deeper, a residual strategy can be used (see Fig. 5) [42]. The module is represented as follows.

$$A^i = W^{i,i-1}A^{i-1} \circ A^{0|1|2|...|i-1} + W^{i,i-1}A^{i-1} \tag{7}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. $A^{0|1|2|...|i-1}$ is any of A. $W^{i,i-1}$ is a weight matrix. $\circ$ denotes Hadamard product.

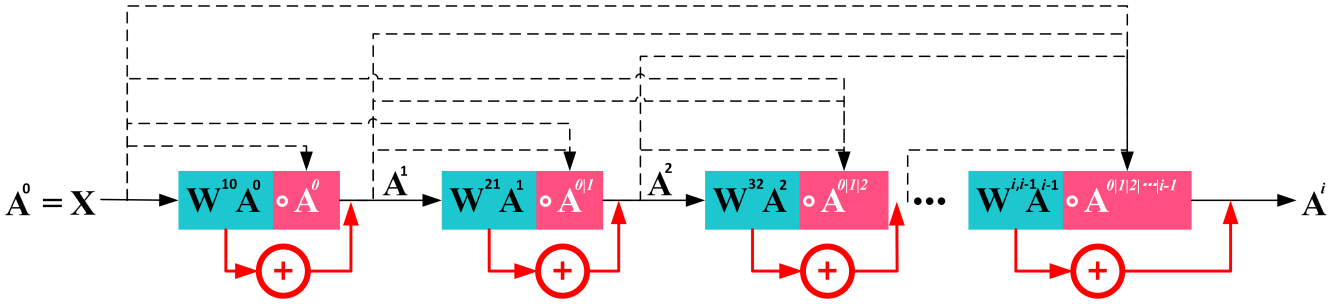


Fig. 5. Generalized Residual Dendrite module. '|' denotes 'or'. The dotted line represents any of them.

4) ResDD-X or Residual Dendrite module ($\circ X$):

Fig. 6 shows the Residual Dendrite module-X(ResDD-X). The module is represented as follows.

$$A^i = W^{i,i-1}A^{i-1} \circ X + W^{i,i-1}A^{i-1} \tag{8}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. X is the inputs of DD. $W^{i,i-1}$ is a weight matrix. $\circ$ denotes Hadamard product.

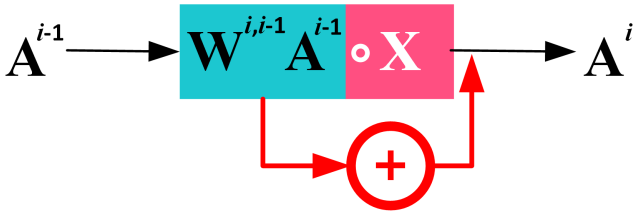


Fig. 6. ResDD-X or Residual Dendrite module ($\circ X$).

Fig. 8 (a) shows ResDD-X code for MNIST dataset. The number of modules can be adjusted according to the task.

In addition, we can also add a weight matrix to the short-circuit route. Fig. 7 shows the Residual Dendrite module-XW(ResDD-XW). The module is represented as follows.

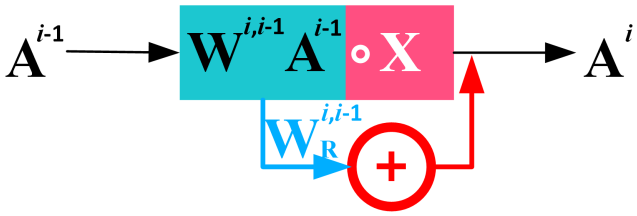


Fig. 7. Residual Dendrite module-XW(ResDD-XW).

$$A^i = W^{i,i-1}A^{i-1} \circ X + W_R^{i,i-1}A^{i-1} \tag{9}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. X is the inputs of DD. $W^{i,i-1}$ and $W_R^{i,i-1}$ are two different weight matrixes. $\circ$ denotes Hadamard product.

Fig. 8 (c) shows ResDD-XW code for MNIST dataset. The number of modules can be adjusted according to the task.

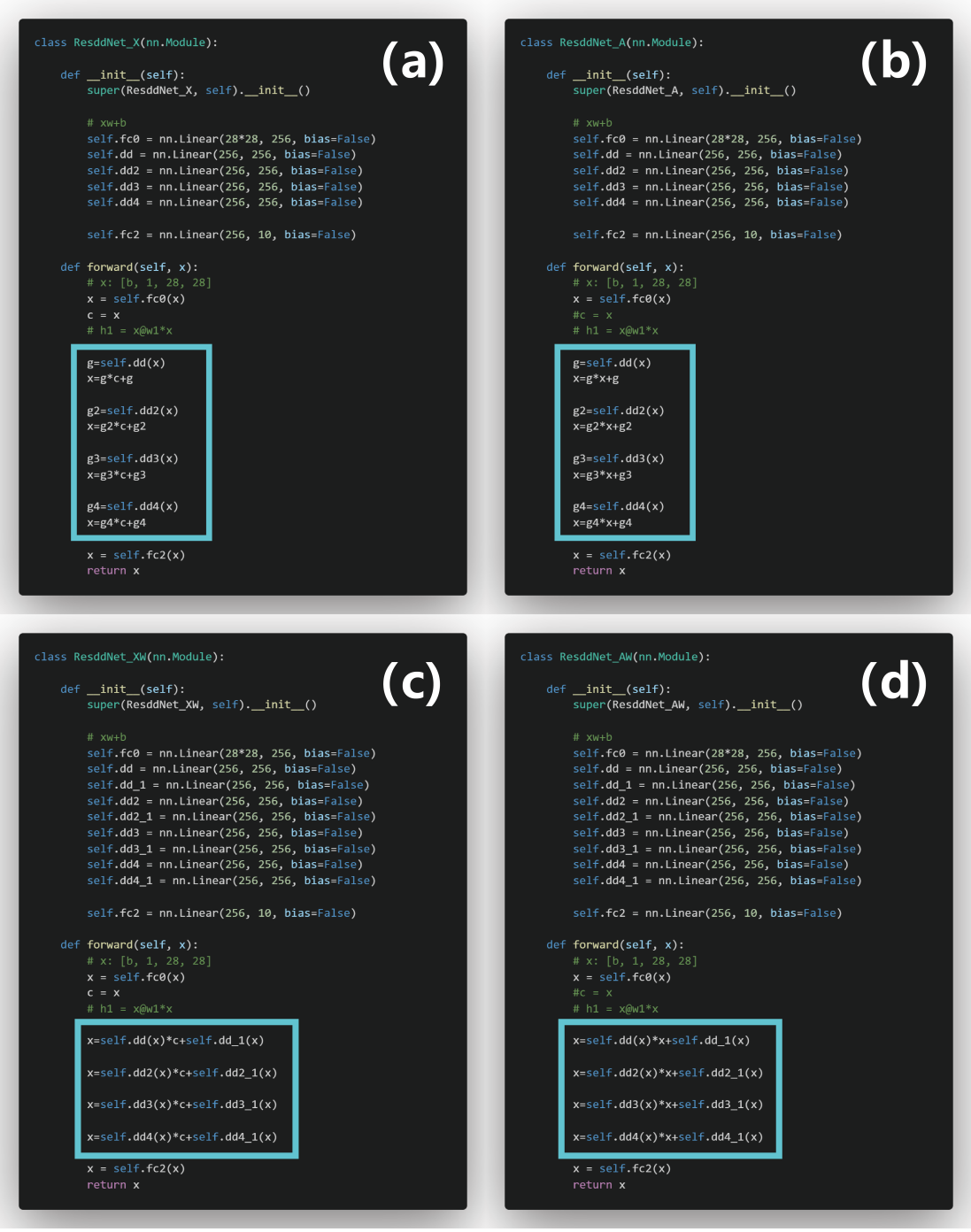


Fig. 8. ResDD code for MNIST dataset. (a) shows $A^i = W^{i,i-1}A^{i-1} \circ X + W^{i,i-1}A^{i-1}$. (b) shows $A^i = W^{i,i-1}A^{i-1} \circ A^{i-1} + W^{i,i-1}A^{i-1}$. (c) shows $A^i = W^{i,i-1}A^{i-1} \circ X + W_R^{i,i-1}A^{i-1}$. (d) shows $A^i = W^{i,i-1}A^{i-1} \circ A^{i-1} + W_R^{i,i-1}A^{i-1}$.

5) ResDD-A or Residual Dendrite module ($\circ A^{i-1}$):

Fig. 9 shows the Residual Dendrite module-A(ResDD-A). The module is represented as follows.

$$A^i = W^{i,i-1}A^{i-1} \circ A^{i-1} + W^{i,i-1}A^{i-1} \tag{10}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. $W^{i,i-1}$ is a weight matrix. $\circ$ denotes Hadamard product.

Figure 8 (b) shows ResDD-A code for MNIST dataset. The number of modules can be adjusted according to the task.

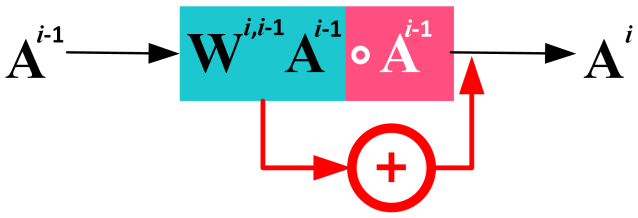


Fig. 9. Residual Dendrite module-A(ResDD-A).

In addition, we can also add a weight matrix to the short-circuit route. Fig. 10 shows the Residual Dendrite module-AW(ResDD-AW). The module is represented as follows.

$$A^i = W^{i,i-1}A^{i-1} \circ A^{i-1} + W_R^{i,i-1}A^{i-1} \tag{11}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. $W^{i,i-1}$ and $W_R^{i,i-1}$ are two different weight matrixes. $\circ$ denotes Hadamard product.

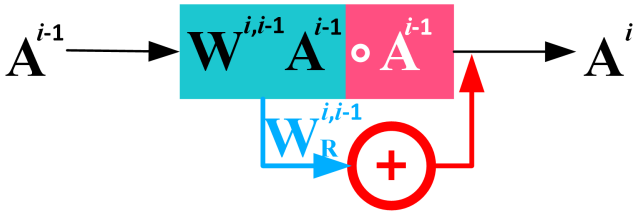


Fig. 10. Residual Dendrite module-AW(ResDD-AW).

Fig. 8 (d) shows ResDD-AW code for MNIST dataset. The number of modules can be adjusted according to the task.

6) Shared weight of Dendrite module:

If convolution is the shared weight of the "plane", then this section denotes the shared weight of depth. This section is described in the form of *Python* code (example).

Fig. 11 shows the DD module code for MNIST (example).

Fig. 12 shows the Shared ResDD module code for MNIST (example).

(a)

```python
class ddNet(nn.Module):

    def __init__(self):
        super(ddNet, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(28*28, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)
        self.dd2 = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        c = x
        # h1 = x@w1*x

        x=self.dd(x)*c
        x=self.dd2(x)*c

        x = self.fc2(x)
        return x
```

(b)

```python
class ddNet(nn.Module):

    def __init__(self):
        super(ddNet, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(28*28, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        c = x
        # h1 = x@w1*x
        for i in range(2):
            x=self.dd(x)*c

        x = self.fc2(x)
        return x
```

Fig. 11. DD module code for MNIST dataset. (a) Unshared DD module.(b) Shared DD module.

```python
class ResddNet_X(nn.Module):

    def __init__(self):
        super(ResddNet_X, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(28*28, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        c = x
        # h1 = x@w1*x
        for i in range(4):
            g=self.dd(x)
            x=g*c+g
        x = self.fc2(x)
        return x
```

```python
class ResddNet_A(nn.Module):

    def __init__(self):
        super(ResddNet_A, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(28*28, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        #c = x
        # h1 = x@w1*x
        for i in range(4):
            g=self.dd(x)
            x=g*x+g
        x = self.fc2(x)
        return x
```

```python
class ResddNet_XW(nn.Module):

    def __init__(self):
        super(ResddNet_XW, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(28*28, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)
        self.dd_1 = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        c = x
        # h1 = x@w1*x
        for i in range(4):
             x=self.dd(x)*c+self.dd_1(x)
        x = self.fc2(x)
        return x
```

```python
class ResddNet_AW(nn.Module):

    def __init__(self):
        super(ResddNet_AW, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(28*28, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)
        self.dd_1 = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        #c = x
        # h1 = x@w1*x
        for i in range(4):
             x=self.dd(x)*x+self.dd_1(x)
        x = self.fc2(x)
        return x
```

Fig. 12. Shared ResDD module code for MNIST dataset.

B. Simplified neuron(Dendrite+Cell body)

1) Architecture-Gang neuron with one module:

Figure 2 shows the Gang neuron with one module. The architecture of Gang neuron can be represented as

$$\begin{cases} A = W^{10}X \circ X \\ y = f(W^{21}A) \end{cases} \tag{12}$$

Where W^{10} and W^{21} are the weight matrix, $\{1, x_1, x_2, \cdots, x_n\} = X$. $\circ$ denotes Hadamard product. In the back-propagation procedure, the weight matrix can be updated with,

$$W^{21}_{new} = W^{21} - \eta^{21}\frac{\partial E}{\partial W^{21}} \tag{13}$$

$$W^{10}_{new} = W^{10} - \eta^{10}\frac{\partial A}{\partial W^{10}}\frac{\partial E}{\partial A} \tag{14}$$

Where W^{21}_{new} and W^{10}_{new} are the updated weight matrix for W^{21} and W^{10} respectively, and E is the cost function or errors from the later layer in back-propagation. η^{21} and η^{10} are the learning rates. The learning rates of "dendrite" and cell body can be different.

2) Architecture-Gang "neuron" of unicellular organism:

In nature, there are unicellular organisms, such as bacteria and fungus. The first organisms to appear on Earth were presumably single-celled [43]. Thus, Fig. 3 shows the Gang "neuron" with multiple interactions to imitate unicellular organisms. The architecture can realize excellent nonlinear expression [40], [41] and can be represented as follows.

$$\begin{cases} A^1 = W^{10}X \circ X \\ A^2 = W^{21}A^1 \circ X \\ \quad\vdots \\ A^l = W^{l,l-1}A^{l-1} \circ X \\ y = f(W^{l+1,l}A^l) \end{cases} \tag{15}$$

Where l represents the number of the dendrite modules. $\circ$ denotes Hadamard product.

3) Architecture-simplified Gang neuron:

The weight matrix of the cell body W^{21} in Fig.2 or $W^{l+1,l}$ in Fig. 3 is defined as a constant matrix where all its elements are 1.The architecture in Fig. 2 can be represented as

$$y = f(W_{ones}(W^{10}X \circ X)) \tag{16}$$

And the architecture in Fig. 3 can be represented as

$$\begin{cases} A^1 = W^{10}X \circ X \\ A^2 = W^{21}A^1 \circ X \\ \quad\vdots \\ A^l = W^{l,l-1}A^{l-1} \circ X \\ y = f(W_{ones}A^l) \end{cases} \tag{17}$$

C. ResGangneuron(RGN)

ResGangneuron module is the combination of ResDD and residual cell body(see Fig. 13). The formula can be expressed as follows.

$$\begin{cases} A^1 = W^{10}A^0 \circ A^0 + W^{10}A^0 \\ \quad\vdots \\ A^i = W^{i,i-1}A^{i-1} \circ A^{0|1|2|\ldots|i-1} + W^{i,i-1}A^{i-1} \\ Y = f(W^{i+1,i}A^i) + A^i \end{cases} \tag{18}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. $A^{0|1|2|\ldots|i-1}$ is any of A. $W^{i,i-1}$ is a weight matrix. i represents the number of the dendrite modules. $\circ$ denotes Hadamard product.

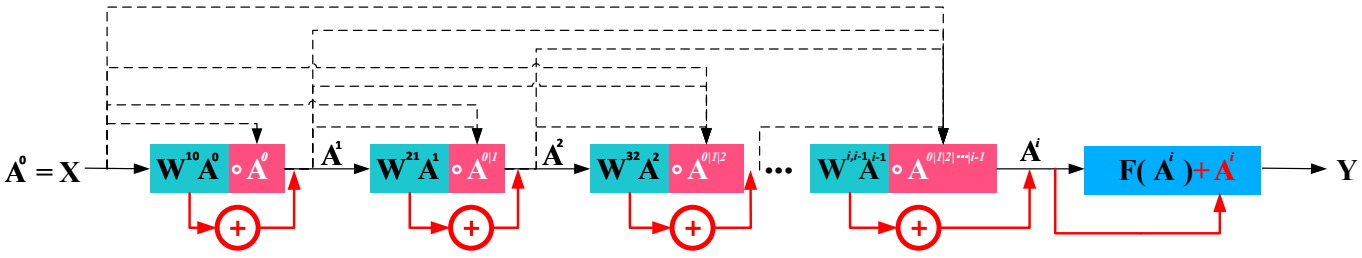


Fig. 13. ResGangneuron module. '|' denotes 'or'. The dotted line represents any of them.

Fig. 14 shows a simple example of using four ResGangneuron.

```python
class ResGangneuron_A(nn.Module):

    def __init__(self):
        super(ResGangneuron_A, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(3*32*32, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)
        self.dd2 = nn.Linear(256, 256, bias=False)
        self.dd3 = nn.Linear(256, 256, bias=False)
        self.dd4 = nn.Linear(256, 256, bias=False)

        self.c1 = nn.Linear(256, 256, bias=False)
        self.c2 = nn.Linear(256, 256, bias=False)
        self.c3 = nn.Linear(256, 256, bias=False)
        self.c4 = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        #c = x
        # h1 = x@w1*x

        g=self.dd(x)
        x=g*x+g
        x= F.relu(self.c1(x))+x

        g2=self.dd2(x)
        x=g2*x+g2
        x= F.relu(self.c2(x))+x

        g3=self.dd3(x)
        x=g3*x+g3
        x= F.relu(self.c3(x))+x

        g4=self.dd4(x)
        x=g4*x+g4
        x= F.relu(self.c4(x))+x

        x = self.fc2(x)
        return x
```

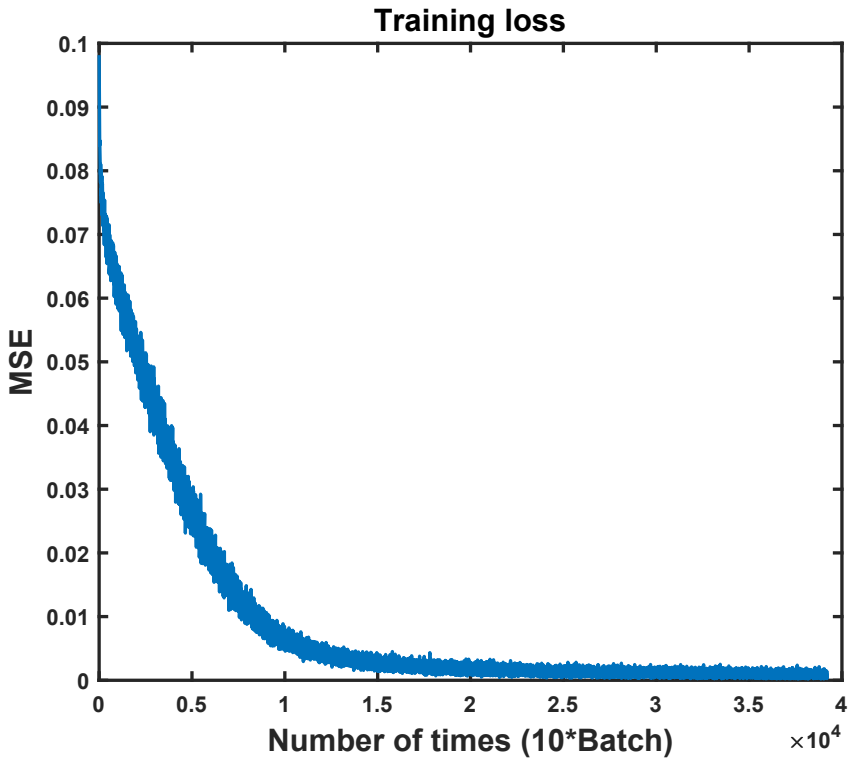


Fig. 14. Approximation using ResGangneuron (RGN-A) for CIFAR-10 dataset (example). Training loss minimum: 2.117×10^{-4}. Note that this is just a simple example without tuning.

D. DenseGangneuron (DGN)

1) Dense Dendrite module (DenseDD) :

In order to make ResDD denser, Generalized Dense Dendrite module can be designed (see Fig. 15). The module is represented as follows.

$$A^i = W^{i,i-1}A^{i-1} \circ A^{0|1|2|...|i-1} + X \tag{19}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. $A^{0|1|2|...|i-1}$ is any of A. $W^{i,i-1}$ is a weight matrix. X denotes the DD's inputs. $\circ$ denotes Hadamard product.

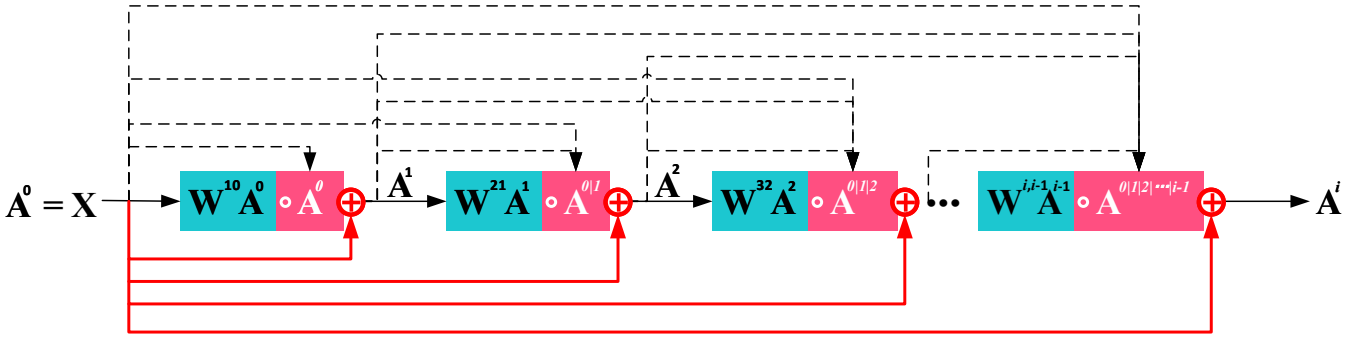


Fig. 15. Generalized Dense Dendrite module. '|' denotes 'or'. The dotted line represents any of them.

Additionally, similar to ResDD module, the code for the special case is shown in Fig. 16.

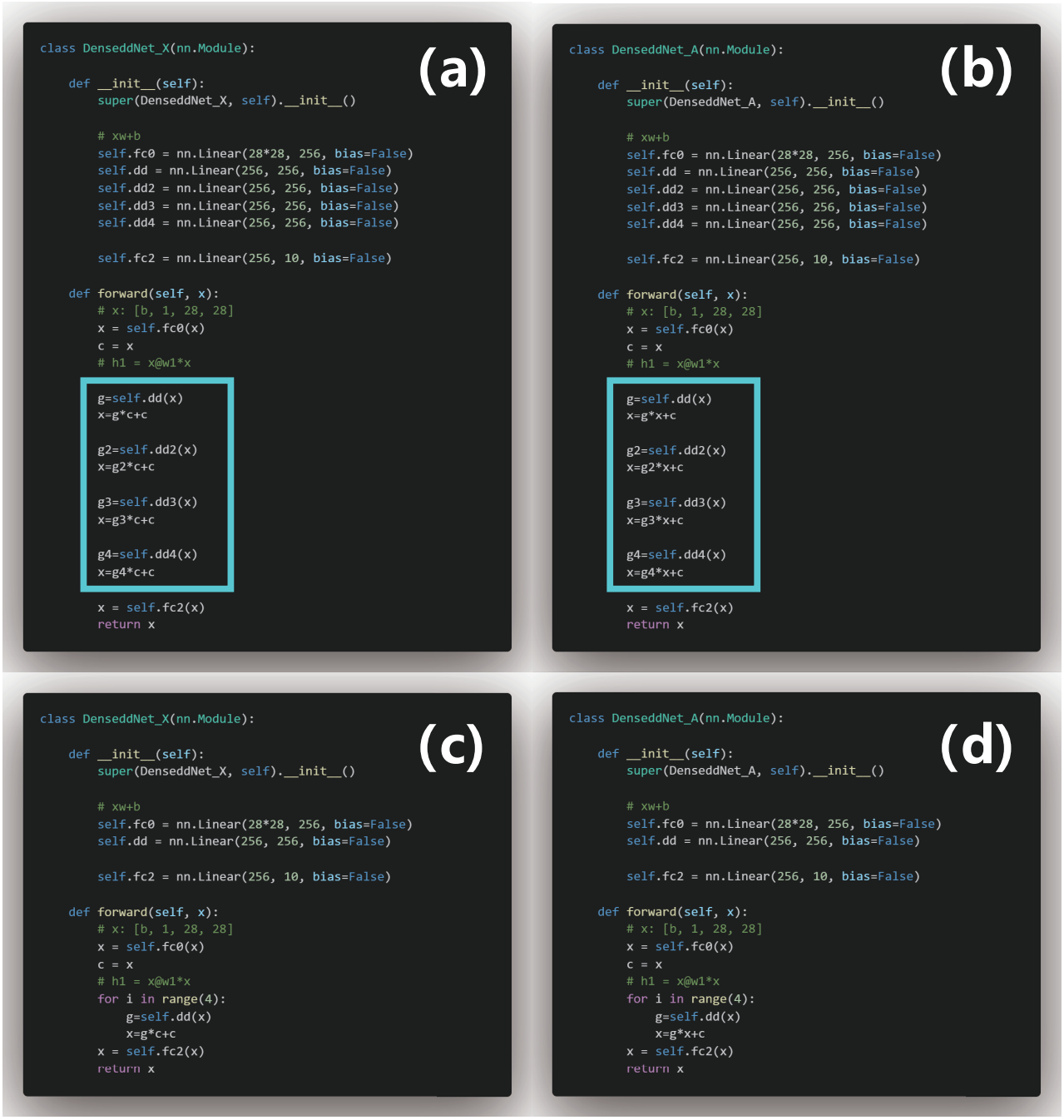


Fig. 16. Dense Dendrite module code (example). (a) and (b) are with the form of the unshared weights. (c) and (d) are with the form of the shared weights.

2) DenseGangneuron: DenseGangneuron module is the combination of DenseDD and Dense cell body. Here, I show two special examples.

Example 1 (DenseGangneuron-X) can be expressed as follows.(see Fig. 17)

$$\begin{cases} A^1 = W^{10}A^0 \circ A^0 + X \\ \quad\vdots \\ A^i = W^{i,i-1}A^{i-1} \circ A^{0|1|2|\ldots|i-1} + X \\ Y = f(W^{i+1,i}A^i) + X \end{cases} \tag{20}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. $A^{0|1|2|\ldots|i-1}$ is any of A. $W^{i,i-1}$ is a weight matrix. i represents the number of the dendrite modules. X denotes the DD's inputs. $\circ$ denotes Hadamard product.

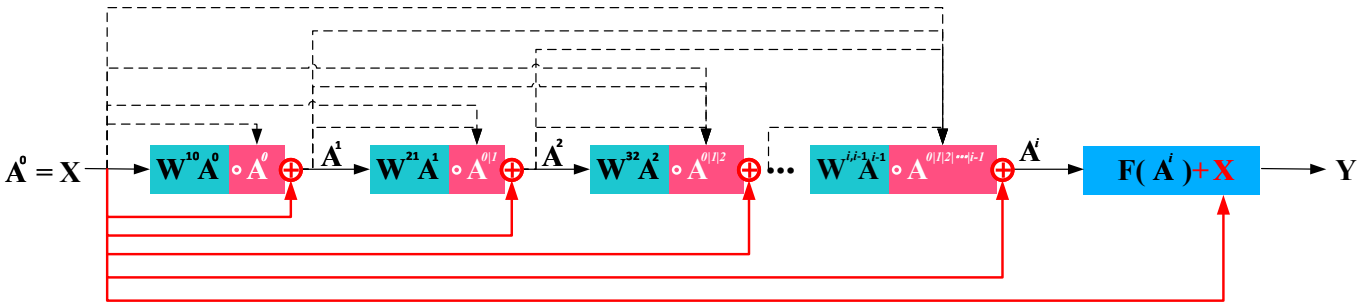


Fig. 17. ResGangneuron-X module. '|' denotes 'or'. The dotted line represents any of them.

Example 2 (DenseGangneuron-A) can be expressed as follows.(see Fig. 18)

$$\begin{cases} A^1 = W^{10}A^0 \circ A^0 + X \\ \qquad \vdots \\ A^i = W^{i,i-1}A^{i-1} \circ A^{0|1|2|\ldots|i-1} + X \\ Y = f(W^{i+1,i}A^i) + A^i \end{cases} \tag{21}$$

Where A^{i-1} and A^i are the inputs and outputs of the module, respectively. $A^{0|1|2|\ldots|i-1}$ is any of A. $W^{i,i-1}$ is a weight matrix. i represents the number of the dendrite modules. X denotes the DD's inputs. $\circ$ denotes Hadamard product.

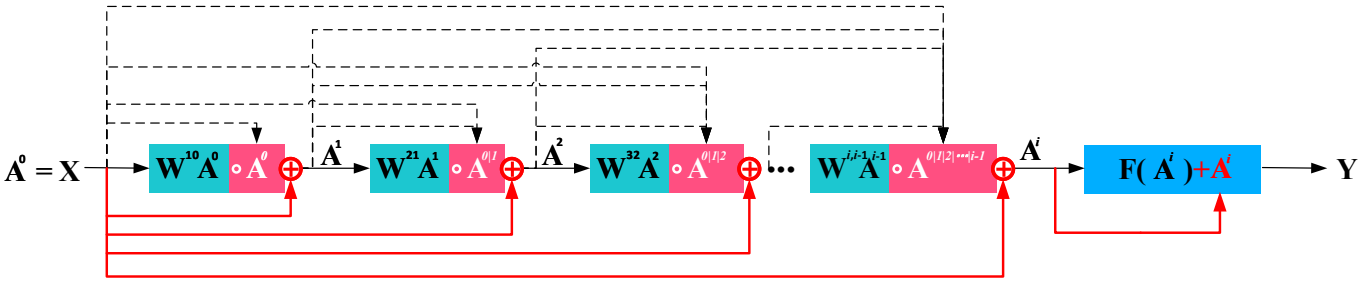


Fig. 18. ResGangneuron-A module. '|' denotes 'or'. The dotted line represents any of them.

As an example, Fig. 19 shows the code when $\circ A^{i-1}$ is used in the dendrite module.

```python
class DenseGangneuron_AX(nn.Module):

    def __init__(self):
        super(DenseGangneuron_AX, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(3*32*32, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)
        self.dd2 = nn.Linear(256, 256, bias=False)
        self.dd3 = nn.Linear(256, 256, bias=False)
        self.dd4 = nn.Linear(256, 256, bias=False)

        self.c1 = nn.Linear(256, 256, bias=False)
        self.c2 = nn.Linear(256, 256, bias=False)
        self.c3 = nn.Linear(256, 256, bias=False)
        self.c4 = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        c = x
        # h1 = x@w1*x

        g=self.dd(x)
        x=g*x+c

        x= F.relu(self.c1(x))+c

        g2=self.dd2(x)
        x=g2*x+c

        x= F.relu(self.c2(x))+c

        g3=self.dd3(x)
        x=g3*x+c

        x= F.relu(self.c3(x))+c

        g4=self.dd4(x)
        x=g4*x+c

        x= F.relu(self.c4(x))+c

        x = self.fc2(x)
        return x
```

```python
class DenseGangneuron_AA(nn.Module):

    def __init__(self):
        super(DenseGangneuron_AA, self).__init__()

        # xw+b
        self.fc0 = nn.Linear(3*32*32, 256, bias=False)
        self.dd = nn.Linear(256, 256, bias=False)
        self.dd2 = nn.Linear(256, 256, bias=False)
        self.dd3 = nn.Linear(256, 256, bias=False)
        self.dd4 = nn.Linear(256, 256, bias=False)

        self.c1 = nn.Linear(256, 256, bias=False)
        self.c2 = nn.Linear(256, 256, bias=False)
        self.c3 = nn.Linear(256, 256, bias=False)
        self.c4 = nn.Linear(256, 256, bias=False)

        self.fc2 = nn.Linear(256, 10, bias=False)

    def forward(self, x):
        # x: [b, 1, 28, 28]
        x = self.fc0(x)
        c = x
        # h1 = x@w1*x

        g=self.dd(x)
        x=g*x+c

        x= F.relu(self.c1(x))+x

        g2=self.dd2(x)
        x=g2*x+c

        x= F.relu(self.c2(x))+x

        g3=self.dd3(x)
        x=g3*x+c

        x= F.relu(self.c3(x))+x

        g4=self.dd4(x)
        x=g4*x+c

        x= F.relu(self.c4(x))+x

        x = self.fc2(x)
        return x
```

Fig. 19. DenseGangneuron code (example). The left figure is DenseGangneuron-X. The right figure is DenseGangneuron-A.

For ease of description, this paper describes the Gang neuron with the best-represented architecture ($\circ X$). In use, we can replace ($\circ X$) by ($\circ A^{0|1|2|\ldots|i-1}$) or using ResDD, DenseDD, ResGangneuron, or DenseGangneuron (see the ResGangneuron section and the DenseGangneuron section). Linear modules (WA) can be inserted anywhere in Gang neuron to adjust the dimensionality. E,g., the DD module ($W^{i,i-1}A^{i-1} \circ A^{i-1}$) is inserted into a linear module and can be written as $W^{i,i-1}A^{i-1} \circ W_L^{i,i-1}A^{i-1}$; Insert linear modules before the dendrite module to adjust the dimensionality.

IV. Some typical ANN architecture using Gang neuron

A. Single-layer network

Fig. 20 shows the single-layer network. This architecture imitates the intricate dendrite of multiple neurons. Multiple neurons may obtain the same signal [32], [33], [43]. "$f(...)$" in Fig. 20 can be set as the function that we plan to use for application. For example, when "$f(...)$" is set as $softmax$, it can be used for classification.

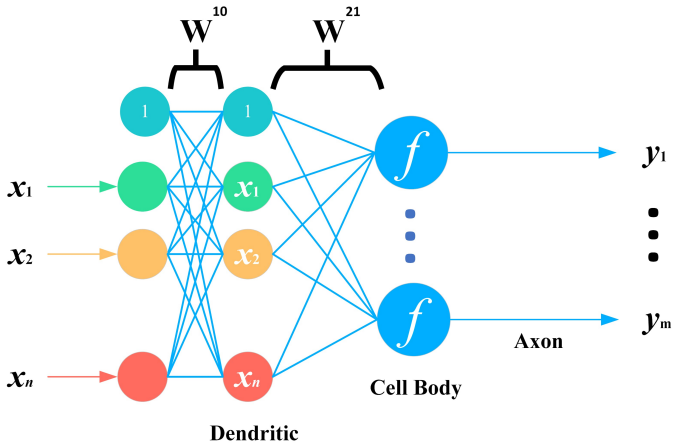


Fig. 20. Single-layer network.

B. Multi-layer perceptron (MLP) using Gang neuron

MLP is the most typical neural network. Here, Fig. 21 shows a MLP using Gang neuron with one module. The forward propagation can be represented as follows.

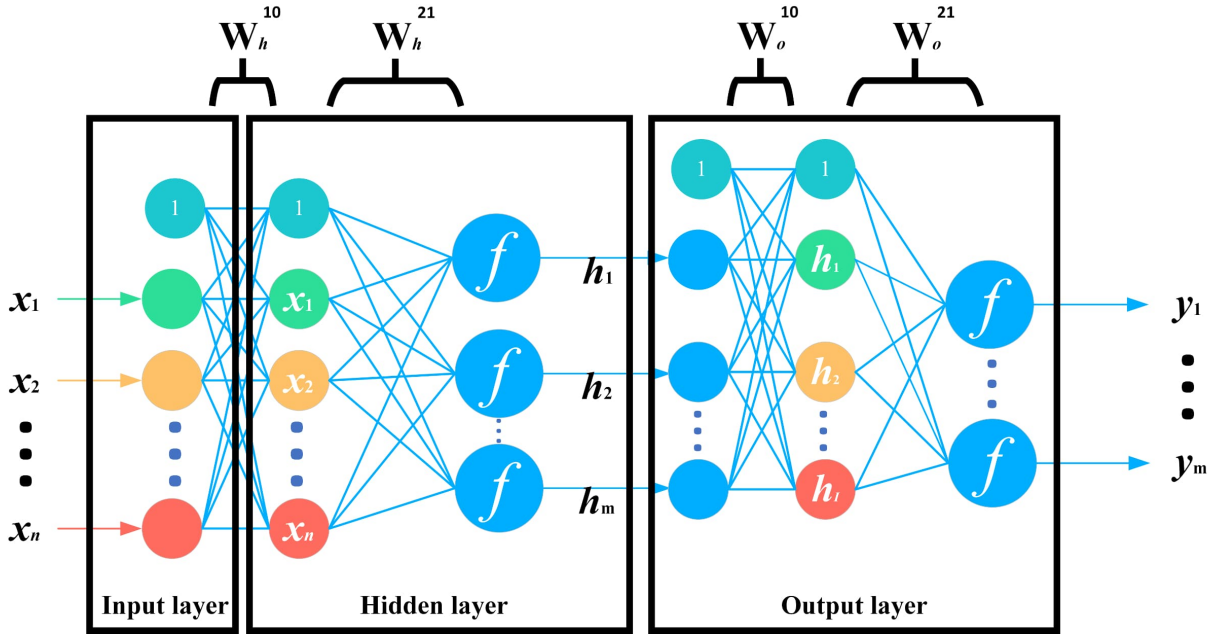


Fig. 21. MLP using Gang neuron.

$$\begin{cases} A_h = W_h^{10} X \circ X \\ H_h = f(W_h^{21} A_h) \end{cases} \tag{22}$$

$$\begin{cases} A_o = W_o^{10} H_h \circ H_h \\ Y_o = f(W_o^{21} A_o) \end{cases} \tag{23}$$

Where $H = [1; H_h]$, W_h^{10} , W_h^{21} , W_o^{10} , and W_o^{21} are the weight matrix, $\{1, x_1, x_2, \cdots, x_n\} = X$. $\circ$ denotes Hadamard product. The backpropagation can be represented according to formula 13 and 14.

Additionally, when the simplest form of Gang neuron is used, MLP can be represented as follows.

$$\begin{cases} H_h = f(\sum(W_h^{10} X \circ X)) \\ Y_o = f(\sum(W_o^{10} H_h \circ H_h)) \end{cases} \tag{24}$$

C. Information fusion network

Figure 22 shows the information fusion of different neurons. This information interaction is common in the biological brain [37]–[39]. The architecture in Fig. 22 can be expressed as follows.

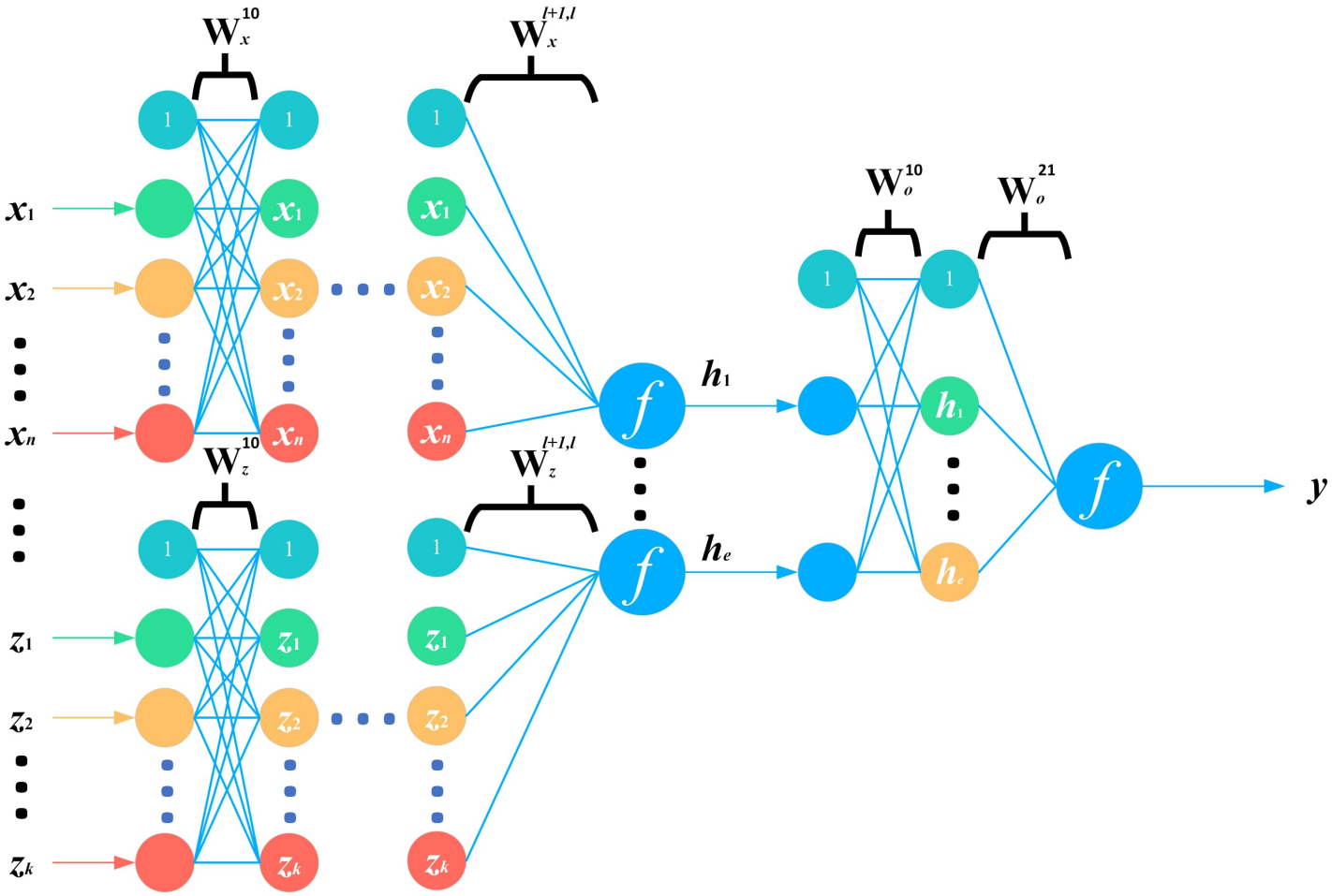


Fig. 22. Information fusion network.

$$\begin{cases} \begin{cases} A_x^1 = W_x^{10} X \circ X \\ A_x^2 = W_x^{21} A_x^1 \circ X \\ \quad\vdots \\ A_x^l = W_x^{l,l-1} A_x^{l-1} \circ X \\ h_1 = f(W_x^{l+1,l} A_x^l) \end{cases} \\ \quad\vdots \\ \begin{cases} A_z^1 = W_z^{10} X \circ X \\ A_z^2 = W_z^{21} A_z^1 \circ X \\ \quad\vdots \\ A_z^l = W_z^{l,l-1} A_z^{l-1} \circ X \\ h_e = f(W_z^{l+1,l} A_z^l) \end{cases} \end{cases} \tag{25}$$

$$\begin{cases} A_h = W_h^{10} H \circ H \\ y = f(W_h^{21} A_h) \end{cases} \tag{26}$$

Where $\{1, x_1, x_2, \cdots, x_n\} = X$,$\{1, z_1, z_2, \cdots, z_k\} = Z$, and $\{1, h_1, h_2, \cdots, h_e\} = H$. e is the number of neurons. $\circ$ denotes Hadamard product. The network architecture can be trained by error backpropagation according to formula 13 and 14.

In fact, these network architectures using Gang neuron can be trained by $PyTorch$ expediently. [Note that the inputs and outputs are normalized to $[-1, 1]$.] The architectures presented in this paper are like some plugins and can be used to other larger architectures where we want to design to improve networks' mapping capacity.

D. Convolution layer using Gang neuron

Fig. 23 compares the architecture of using a Gang neuron with a traditional neuron in the convolution layer (an example). The Gang neuron extracts interaction information between inputs.

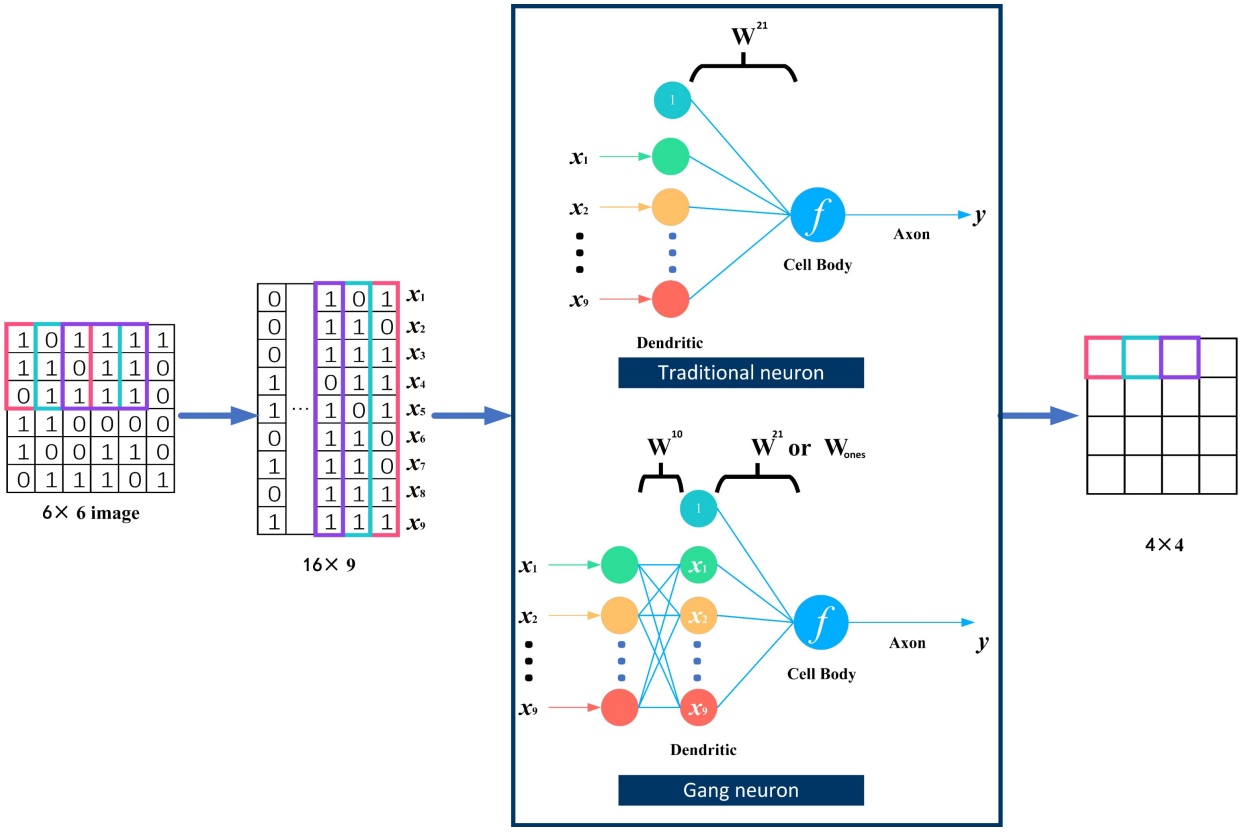


Fig. 23. Convolution layer using Gang neuron or traditional neuron.

The network using Gang neurons can delete the Fully-connected Layers of traditional networks. In other words, the parameters of the Fully-connected Layer are assigned to a single neuron, which reduces parameters of network for the same mapping capacity. **The Fully-connected Layers do not exist in human brain.**

For example, the number of parameters in the Fully-connected Layer is 10000×100. When the parameters are assigned to a single neuron, the number of parameters can be $\mathbf{100} \times 10 \times 10$. [**100** denotes the number of Gang neurons.]

E. Schematic diagram of RNN using Gang neuron

Fig. 24 compares the architecture of using a Gang neuron with a traditional neuron in RNN. The Gang neuron extracts the interaction information between inputs about time. Where x represents the currents input and h represents the outputs of the previous time.

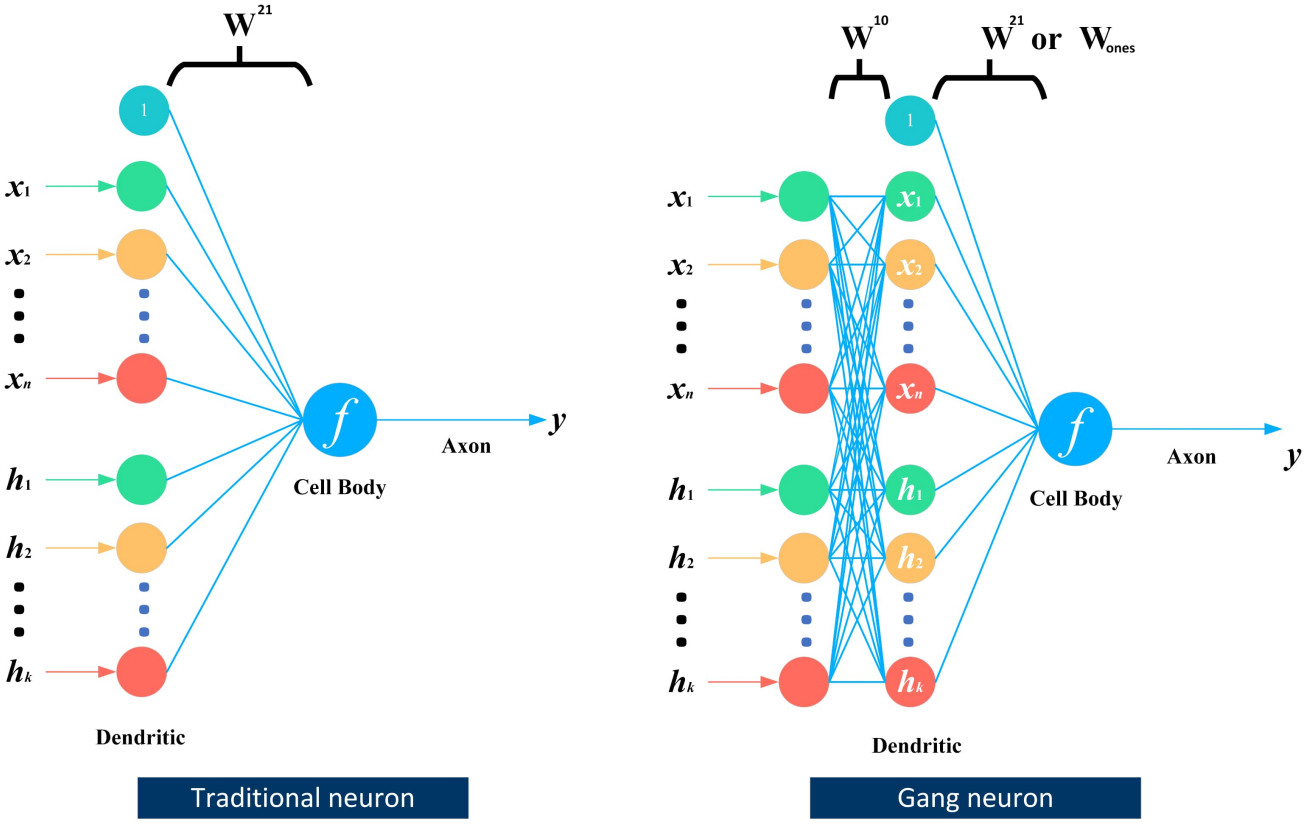


Fig. 24. Schematic diagram of RNN using Gang neuron.

V. Biomimetic architecture using Gang neuron

A. Module 1: Retina,attention(focus),and convolution-like scan

Fig. 25 shows a retina module. This module imitates the focus mechanism of the human eye when we observe things. At the focus, our eye extracts the multiple relationships of the picture, and then, the extracted texture is more and more rough along the direction of focus diffusion. Additionly, **Attention** in Fig. 25 can scan the entire picture, which happens to be similar to convolution. However, my architecture contains gradients related to focus, and the maximum receptive field can be set as the entire picture (Of course, we can also set a small range of receptive field with gradient). I show a 4th-order gradient in Fig. 25. The number of gradients can be adjusted by dendrite modules appropriately. For specific details, please see the specific formula in the Fig. 25.

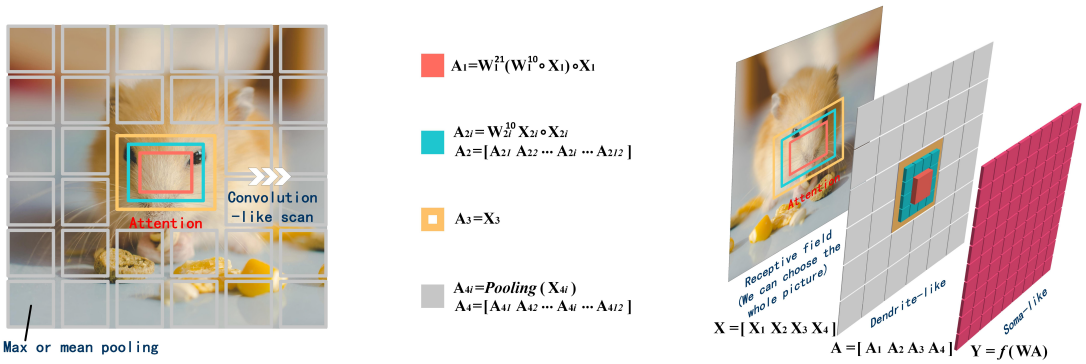


Fig. 25. Module 1: Retina,attention(focus),and convolution-like scan. The gap between the 'squares' only aims to distinguish the 'squares' in drawing. The calculation of each color block consists of two steps(see Fig. 23): 1) Transform X into a 1-dimensional input; 2) use the dendrite modules. In terms of the activation function, we can use what we want, such as $relu$. Besides, linear activation function (e.g.,$Y = WA$) also can be used because the dendrite module has the nonlinear mapping capacity. Additionally, as an example, the figure shows the calculation process of one channel. For more channels, each channel can be calculated separately. Then, the outputs of channels are averaged in the same position. Note that the above formulas express one operation like a convolution kernel. The later layer is generated by a convolution-like scan.

B. Module 2: Population coding in brain

The dendrites of neurons are crisscrossed. In population coding of biological brain, the primary afferent fibre whose receptive field centre is closest to the point of stimulation will produce more action potentials than those on the periphery (Fig. 26, the green fibre has a greater action potential frequency than the blue fibres) [44].

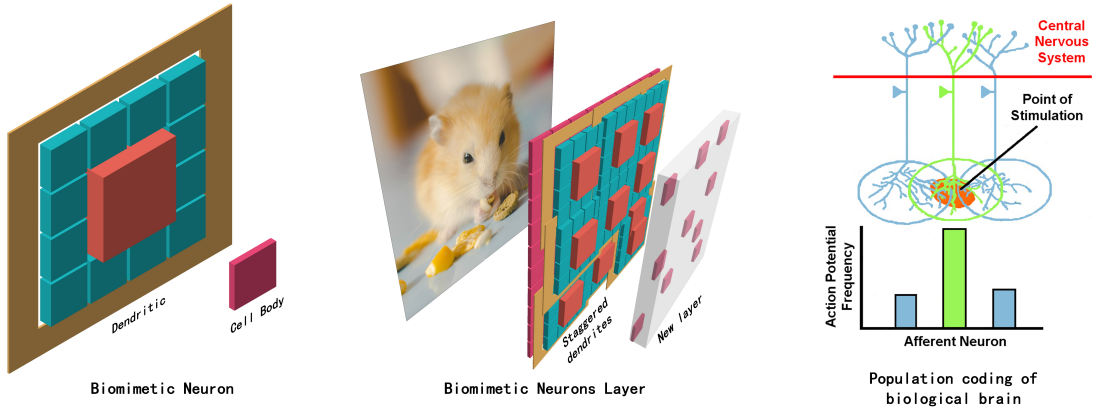


Fig. 26. Module 2: Population coding in brain. Refer to Fig. 15 for the calculation of neurons. Besides, as an example, the figure shows the calculation process of one channel. For more channels, each channel can be calculated separately. Then, the outputs of channels are averaged in the same position.

Fig. 26 shows a biomimetic population coding module. Two points are important to emphasize.

1)Biomimetic dendrite is staggered; 2)When we design a network by multilayer Module 2 in Fig. 26, the dimension of output decreases as the layer.

C. Application methods

This paper shows three usage.

1) In the existing networks, the convolutional layers can be replaced by module 1.(Compared with the existing convolution, module 1 has a larger receptive field and can extract the position information.)

2) Multi-layer modules 2 are interconnected to build a new neural network.

3) Module 1 and Module 2 are used together. There is a special architecture imitating the biological brain. The first layer is designed by module 1, and all the later layers are designed by module 2.

VI. Tips of Gang neuron

A. Avoiding curse of dimensionality and improving generalization ability

1) Adjusting the range of inputs: When the absolute value of the dendrites' inputs is less than 1, there is an inhibitory effect on higher-order terms.

$$\lim_{n\to+\infty} x^n = 0 \tag{27}$$

Where $|x| < 1$, $n \in \mathbb{N}^+$. Simultaneously, $i, j \in \mathbb{N}^+$, when $i < j$, $x^i > x^j$. The simulation example is shown in Fig. 27 . The degree of suppression of higher-order terms can be changed by adjusting the range of x (see Fig. 27).

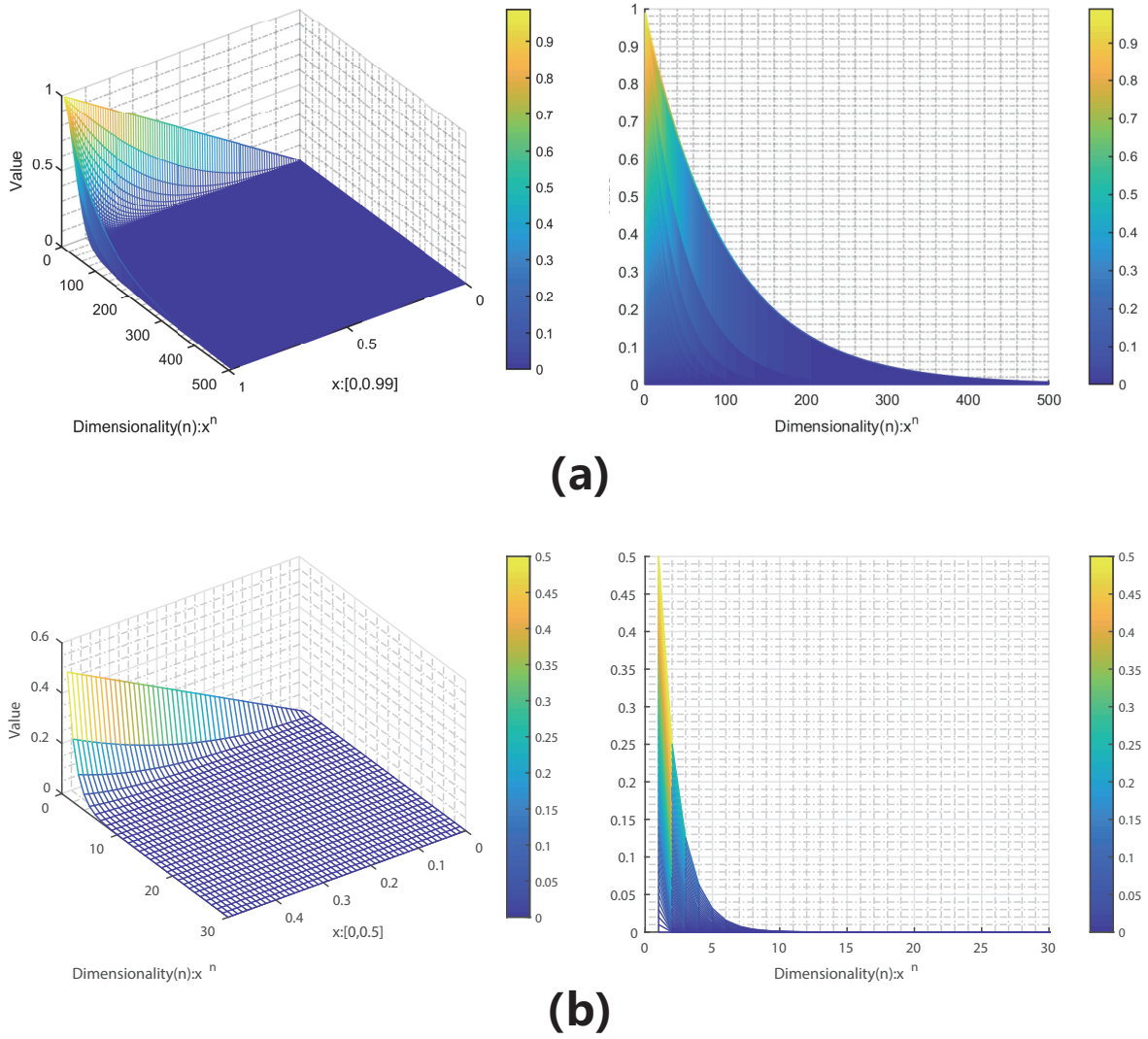


Fig. 27. $\lim_{n\to+\infty} x^n = 0$. (a) $x \in [0, 0.99]$. (b) $x \in [0, 0.5]$.

```
# Regularization L1
Regularization_loss=0

for param in net.parameters():
    Regularization_loss += torch.sum(torch.abs(param))

loss = F.mse_loss(out, y_onehot)+a*Regularization_loss
```

Fig. 28. L1-regularization (Example,$PyTorch$ code).

2) Regularization: Regularization (e.g.,L1-regularization [45]) is used to balance the weight distribution, which makes Gang neuron more inclined to approximate the data set by using low-order terms. Gang neuron (e.g., ResDD) uses lower-order terms as much as possible, and then higher-order terms are used to approximate details, which is consistent with what we expect. Here, I show an example of L1-regularization(see Fig. 28).

VII. Some theories for Gang neurons

The following content is only to explain Gang neurons and does not contain strict proof.

A. Polynomial, Convolution, and Dendrite in this paper

Understanding dendrites for feature extraction from the perspective of convolution

Convolution:

The convolution of f and g is written $f * g$, denoting the operator with the symbol $*$. It is defined as the integral of the product of the two functions after one is reversed and shifted. As such, it is a particular kind of integral transform:

$$\int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{28}$$

We can express it in discrete form.

$$\sum f[k]g[x-k] \tag{29}$$

Polynomial multiplication:

E.g: $(x \times x + 3 \times x + 2)(2 \times x + 5)$

The calculation process is as follows:

$$\begin{aligned}
&(x \times x + 3 \times x + 2)\,(2 \times x + 5) \\
&= (x \times x + 3 \times x + 2) \times 2 \times x + (x \times x + 3 \times x + 2) \times 5 \\
&= 2 \times x \times x \times x + 3 \times 2 \times x \times x + 2 \times 2 \times x + 5 \times x \times x + 3 \times 5 \times x + 10
\end{aligned} \tag{30}$$

Then, merge the same terms:

$$\begin{aligned}
&2x \times x \times x \\
&3 \times 2 + 1 \times 5x \times x \\
&2 \times 2 + 3 \times 5x \\
&2 \times 5 \\
&- - - - \\
&2 \times x \times x \times x + 11 \times x \times x + 19 \times x + 10
\end{aligned} \tag{31}$$

The polynomial can be regarded as a vector space, and its basis can be selected as $\{1, x, x \times x, x \times x \times x, \cdots\}$. Then any polynomial can correspond to a coordinate vector in an infinite-dimensional space.

$(x \times x + 3 \times x + 2)$ corresponds to $(1 \quad 3 \quad 2)$; $(2 \times x + 5)$ corresponds to $(2 \quad 5)$.

We perform coordinate calculation according to the definition of convolution:

$(1 \quad 3 \quad 2) * (2 \quad 5)$ is expressed as follows.

$$\begin{array}{cccccc} 2 & 3 & 1 & & & \\ - & - & 2 & 5 & & \\ - & - & - & - & - & - \\ & & & 2 & & \end{array} \tag{32}$$

$$\begin{array}{cccccc} 2 & 3 & 1 & & & \\ - & 2 & 5 & & & \\ - & - & - & - & - & - \\ & 6 & + 5 = 11 & & & \end{array} \tag{33}$$

$$\begin{array}{cccccc} 2 & 3 & 1 & & & \\ 2 & 5 & & & & \\ - & - & - & - & - & - \\ 4 & + 15 = 19 & & & & \end{array} \tag{34}$$

$$\begin{array}{cccccc} - & 2 & 3 & 1 & & \\ 2 & 5 & & & & \\ - & - & - & - & - & - \\ & 10 & & & & \end{array} \tag{35}$$

e.i., $(1 \quad 3 \quad 2) * (2 \quad 5) = (2 \quad 11 \quad 19 \quad 10)$

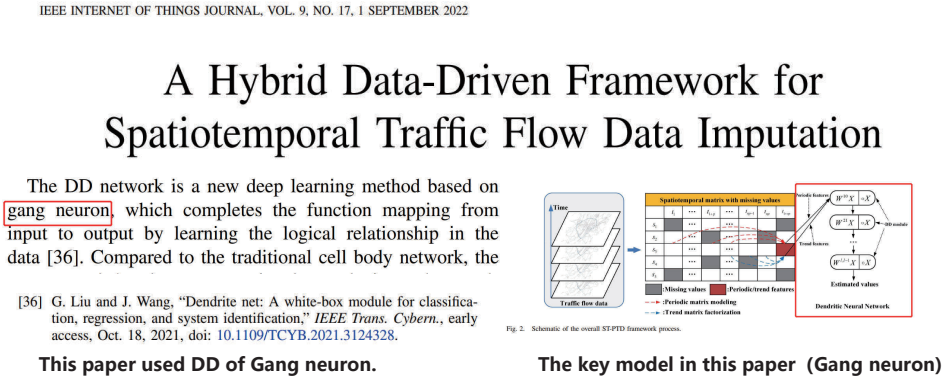
Now we look at Eq.31: $(x \times x + 3 \times x + 2)(2 \times x + 5) = 2 \times x \times x \times x + 11 \times x \times x + 19 \times x + 10$. Polynomial multiplication is equivalent to convolution of coefficient vectors. Therefore, we can also understand the calculation of Dendrite modules from the perspective of convolution.

VIII. CURRENT APPLICATIONS OF GANG NEURONS

A. *Traffic:IEEE Internet of Things Journal (Journal, IF:10.238)*

Study using Gang neurons [46]:"A Hybrid Data-Driven Framework for Spatiotemporal Traffic Flow Data Imputation"(see Fig. 29)

An accurate estimation of missing data in traffic flow is crucial in urban planning, intelligent transportation, economic geography, and other fields. Thus, improving the data quality of traffic flow is a necessary step in data modeling. Most existing studies use data-driven models to determine spatiotemporal patterns in traffic flow data and fill in the missing information automatically. However, simple data-driven models have unsatisfactory results for describing complex patterns in traffic flow and filling in missing data. This study treated the complex patterns in traffic flow as integrating multiple simple patterns and proposed a hybrid missing data imputation framework called ST-PTD based on **Gang neurons**. The results showed that the ST-PTD framework outperformed the eight existing baselines in terms of imputation accuracy.



IEEE INTERNET OF THINGS JOURNAL, VOL. 9, NO. 17, 1 SEPTEMBER 2022

A Hybrid Data-Driven Framework for Spatiotemporal Traffic Flow Data Imputation

The DD network is a new deep learning method based on gang neuron, which completes the function mapping from input to output by learning the logical relationship in the data [36]. Compared to the traditional cell body network, the

[36] G. Liu and J. Wang, "Dendrite net: A white-box module for classification, regression, and system identification," *IEEE Trans. Cybern.*, early access, Oct. 18, 2021, doi: 10.1109/TCYB.2021.3124328.

Fig. 2. Schematic of the overall ST-PTD framework process.

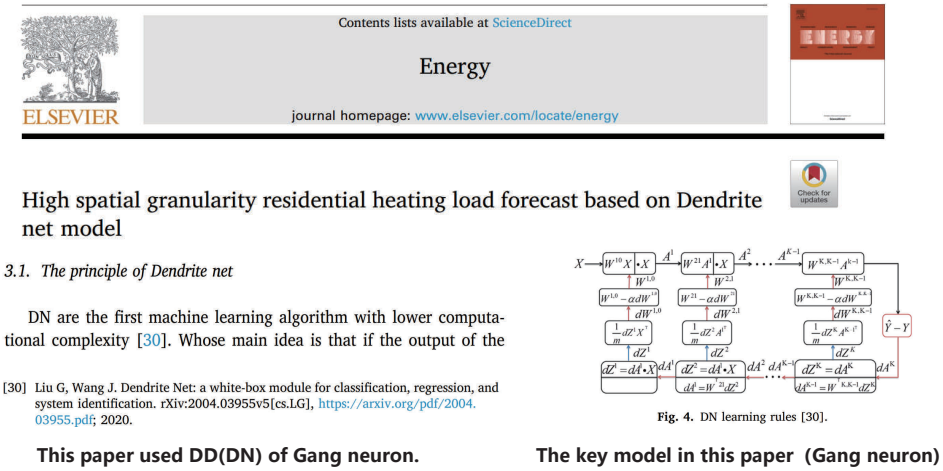
This paper used DD of Gang neuron. **The key model in this paper (Gang neuron)**

Fig. 29. **Traffic:***IEEE Internet of Things Journal* (Journal, IF:10.238)

B. *Energy:Energy (Journal, IF:8.857)*

Study using Gang neurons [47]:"High Spatial Granularity Residential Heating Load Forecast Based on Dendrite Net Model"(see Fig. 30)

With the application of smart meters, more information is available from residential buildings for support heat load forecast. Yet, there is still a lack of an effective method to exploit the value of the high spatial granularity information, particularly for residential communities with high randomness in human behaviors. To fill this gap, this paper proposes a data-driven heat load forecast method based on **Gang neurons**. The results demonstrate that the method shows higher forecast accuracy, the root-mean-square error is 0.0029, in some research cases the coefficient of determination can reach 0.99–1, and wide scope of application in the area of heat load forecast.



Contents lists available at ScienceDirect

Energy

journal homepage: www.elsevier.com/locate/energy

ELSEVIER

High spatial granularity residential heating load forecast based on Dendrite net model

3.1. The principle of Dendrite net

DN are the first machine learning algorithm with lower computational complexity [30]. Whose main idea is that if the output of the

[30] Liu G, Wang J. Dendrite Net: a white-box module for classification, regression, and system identification. rXiv:2004.03955v5[cs.LG], https://arxiv.org/pdf/2004.03955.pdf; 2020.

Fig. 4. DN learning rules [30].

This paper used DD(DN) of Gang neuron. **The key model in this paper (Gang neuron)**

Fig. 30. **Energy:***Energy* (Journal, IF:8.857)

C. ***Energy:***Applied Energy *(Journal, IF:11.446)*

Study using Gang neurons [48]:"Energy saving of buildings for reducing carbon dioxide emissions using novel dendrite net integrated adaptive mean square gradient"(see Fig. 31)

In this paper, **Gang neurons** is used to energy saving and emission reducing of the construction industry. Compared with other methods, the experimental results show the availability of Gang neurons. Through predicting the heating and cooling loads based on Gang neurons, the construction plan is adjusted to decrease the energy consumption of buildings. Moreover, the appliances energy consumption is predicted and analyzed by Gang neurons to improve energy efficiency and cut carbon dioxide emissions.

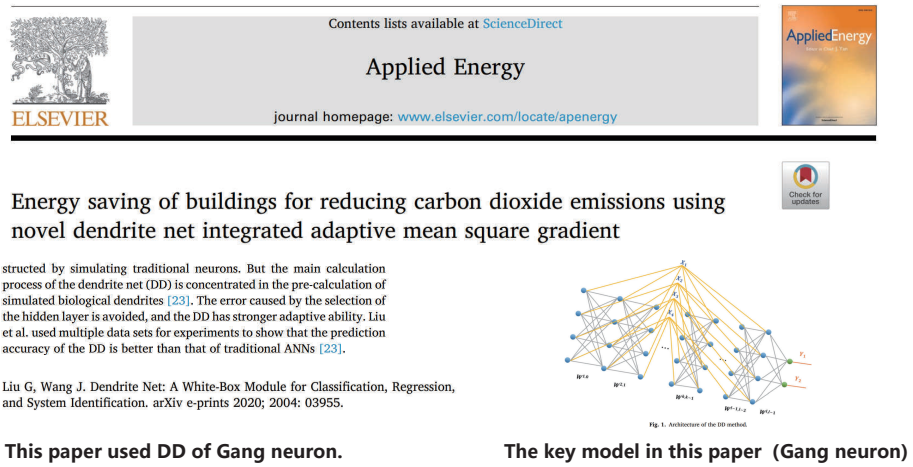


Fig. 31. **Energy:***Applied Energy* (Journal, IF:11.446)

D. ***Mechanical Engineering:***Engineering Optimization *(Journal, IF:2.616)*

Study using Gang neurons [49]:"A radial sampling-based subregion partition method for dendrite network-based reliability analysis"(see Fig. 32)

In sampling-based reliability analysis, a constraint with multiple failure points may lead to an inefficient iteration process and inaccurate results. To examine this problem, a novel analysis method based on **Gang neurons** is proposed in this article, which achieves multiple failure point-based constraint model construction. In the proposed method, a radial sampling-based subregion partition method is presented to locate the potential failure subregions that may have a failure point and to construct a subregion partition model to find the model refinement points in parallel. In addition, a new machine learning algorithm, the **Gang neuron**, is adopted to construct the constraint model and the subregion partition model, and a network-matched learning function is designed to assist model refinement. Test results demonstrate that the number of training samples is decreased compared with other citation methods.

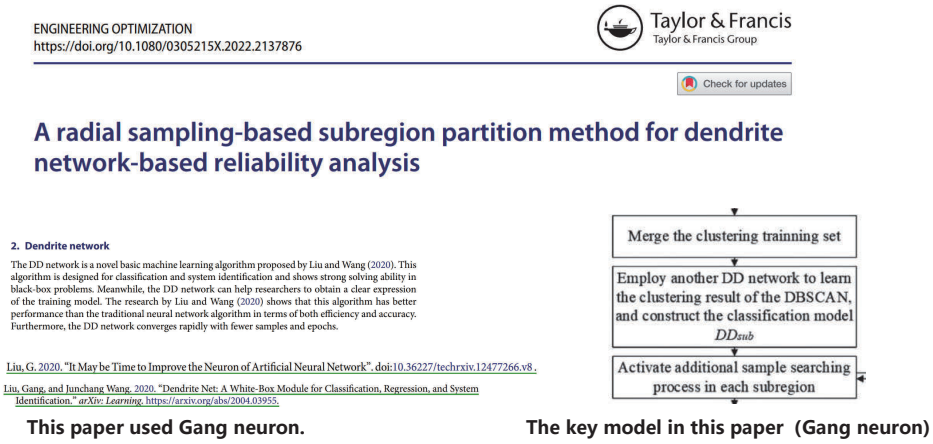


Fig. 32. **Mechanical Engineering:***Engineering Optimization* (Journal, IF:2.616)

E. ***Meteorology:****Frontiers in Plant Science (Journal, IF:6.627)*

Study using Gang neurons [50]:"An Algorithm for Precipitation Correction in Flood Season Based on Dendritic Neural Network"(see Fig. 33)

In recent years, the National Climate Center has developed a dynamic downscaling prediction technology based on the Climate-Weather Research and Forecasting (CWRF) regional climate model and used it for summer precipitation prediction, but there are certain deviations, and it is difficult to predict more accurately. The CWRF model simulates the summer precipitation forecast data from 1996 to 2019 and using **Gang neurons** to conduct a comparative analysis of summer precipitation correction techniques. The results show that the correction effect based on Gang neurons is better than the CWRF historical return, in which, the anomaly correlation coefficient (ACC) and the temporal correlation coefficient (TCC) both increased by 0.1, the mean square error (MSE) dropped by about 26%, and the overall trend anomaly (Ps) test score was also improved.

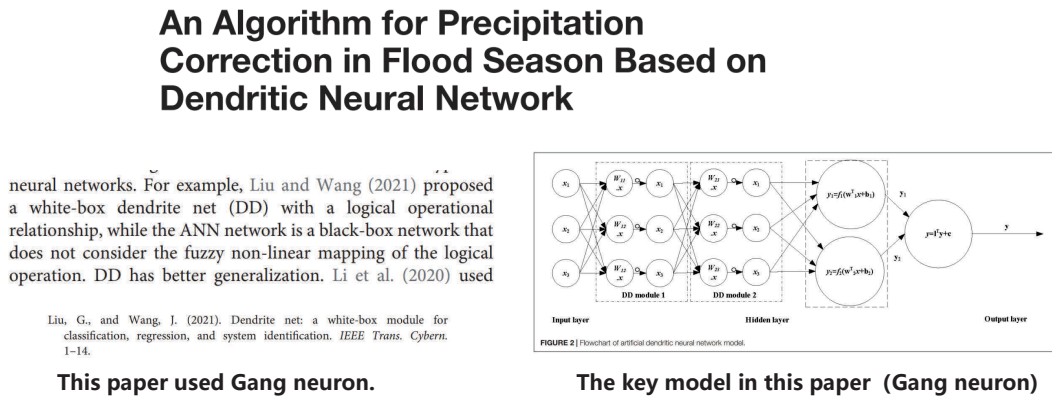


Fig. 33. **Meteorology:***Frontiers in Plant Science* (Journal, IF:6.627)

F. ***Brain-Computer Interface:****IEEE Transactions on Neural Systems and Rehabilitation Engineering (Journal, IF:4.528)*

Study using Gang neurons [51]:"An Algorithm for Precipitation Correction in Flood Season Based on Dendritic Neural Network"(see Fig. 34)

Modeling the brain as a white box is vital for investigating the brain. However, the physical properties of the human brain are unclear. Therefore, BCI algorithms using EEG signals are generally a data-driven approach and generate a black- or gray-box model. This paper presents the first EEG-based BCI algorithm (EEG-BCI using **Gang neurons**, EEGG) decomposing the brain into some simple components with physical meaning and integrating recognition and analysis of brain activity. EEGG decomposed the biological EEG-intention system of this paper into the relation spectrum inheriting the Taylor series (in analogy with the data-driven but human-readable Fourier transform and frequency spectrum), which offers a novel frame for analysis of the brain.

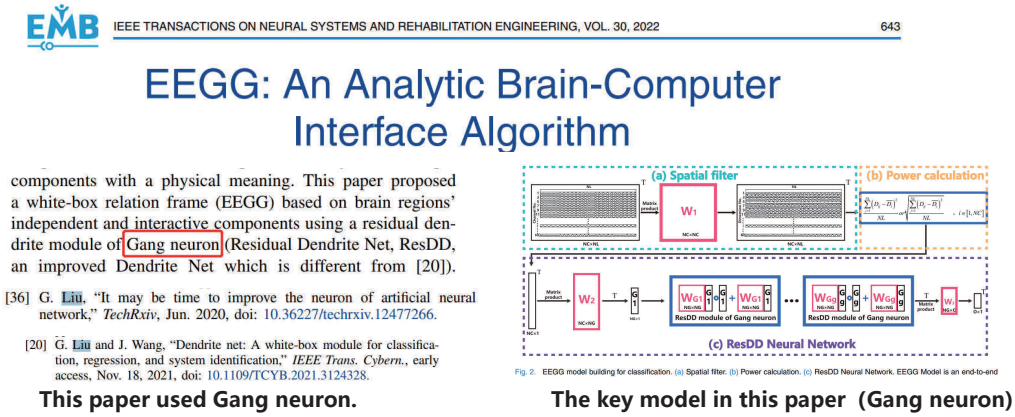


Fig. 34. **Brain-Computer Interface:***IEEE Transactions on Neural Systems and Rehabilitation Engineering* (Journal, IF:4.528)

G. *Myocardial Infarction:IEEE Sensors Journal (Journal, IF:4.325)*

Study using Gang neurons [52]:"Application of Convolutional Dendrite Net for Detection of Myocardial Infarction Using ECG Signals"(see Fig. 35)

Myocardial infarction (MI) is a sudden-onset medical emergency. Without timely diagnosis and treatment, mortality is very high. **Gang neuron** realizes the classification of images by extracting the logical relationship between features. The method based on Gang neuron is verified by Physikalisch Technische Bundesanstalt (PTB) dataset. The results show that the average detection accuracy and time of one patient are 98.95% and 2.09 ms, respectively.

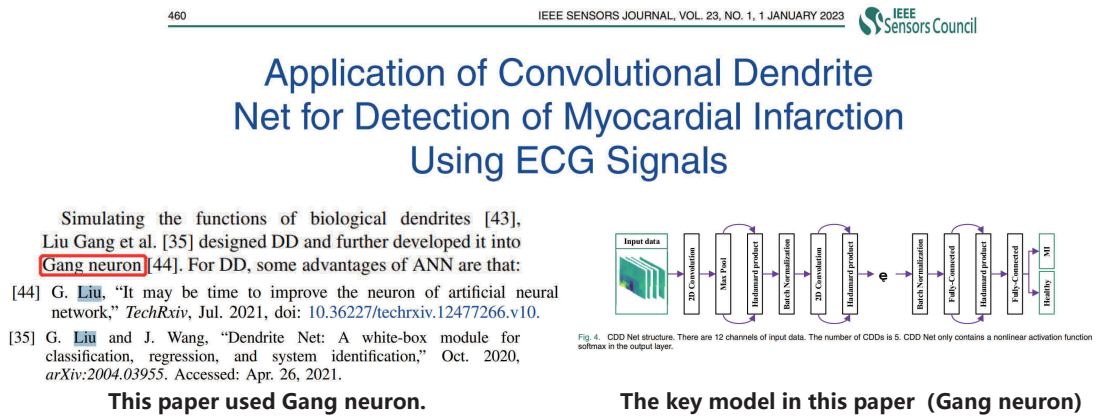


Fig. 35. **Myocardial Infarction:***IEEE Sensors Journal* (Journal, IF:4.325)

H. *Mechanical Engineering:Energies (Journal, IF:3.252)*

Study using Gang neurons [52]:"An Accuracy Prediction Method of the RV Reducer to Be Assembled Considering Dendritic Weighting Function"(see Fig. 36)

There are many factors affecting the assembly quality of rotate vector reducer, and the assembly quality is unstable. Matching is an assembly method that can obtain high-precision products or avoid a large number of secondary rejects. Selecting suitable parts to assemble together can improve the transmission accuracy of the reducer. In the actual assembly of the reducer, the success rate of one-time selection of parts is low, and "trial and error assembly" will lead to a waste of labor, time cost, and errors accumulation. In view of this situation, a **Gang neuron** model based on mass production and practical engineering applications has been established. The size parameters of the parts that affected transmission error of the reducer were selected as influencing factors for input. The key performance index of reducer was transmission error as output index. After data standardization preprocessing, a quality prediction model was established to predict the transmission error. The experimental results show that the dendritic neural network model can realize the regression prediction of reducer mass and has good prediction accuracy and generalization capability. The proposed method based on Gang neuron can provide help for the selection of parts in the assembly process of the RV reducer.

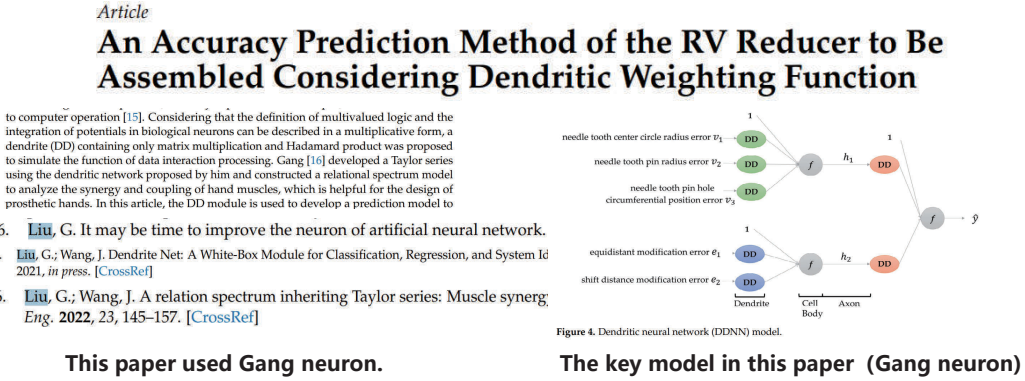


Fig. 36. **Mechanical Engineering:***Energies* (Journal, IF:3.252)

I. Mechanical Engineering:2022 MLISE

Study using Gang neurons [53]:"Unteady aerodynamics modeling method based on dendrite-based gated recurrent neural network model"(see Fig. 37)

System identification-based aerodynamic reduced-order model (ROM) is an effective method for solving nonlinear unsteady aerodynamics prediction. However, most of the ROMs are shallow neural network architecture which is difficult to capture nonlinear aerodynamic behavior with large samples and Varying Mach Numbers. This paper proposes a dendrite-based(**Gang neurons**) gated recurrent neural network (DD-GRU) fusion model from deep learning theory to improve the capability of system identification-based ROM in varying flow conditions. DD-GRU network can process time-series data, which is also suitable for capturing the time-delay effect of unsteady aerodynamics. Unlike other recurrent models, DD-GRU uses the dendrite method to re-extract the logic of information of GRU. The approach is evaluated by predicting NACA64A01 airfoil transonic aerodynamic loads across multiple flow conditions. The example demonstrates that the model accurately predicts the harmonic aerodynamic response in the frequency and time domains with different flow conditions.

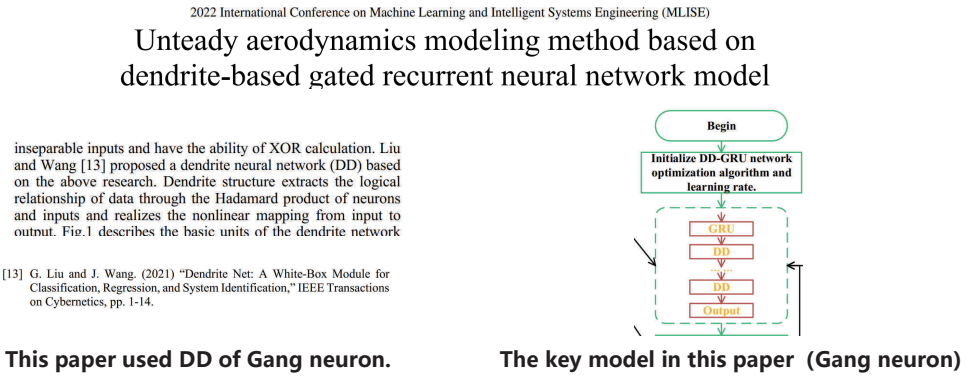


Fig. 37. **Mechanical Engineering:***2022 MLISE*

J. Epilepsy(Magnetoencephalography):2022 ICRCV

Study using Gang neurons [54]:"An Efficient CADNet for Classification of High-frequency Oscillations in Magnetoencephalography"(see Fig. 38)

Epilepsy is a chronic neurological disease, and locating the lesions precisely is crucial to the success of epilepsy surgery. The high-frequency oscillations (HFOs) in magnetoencephalography (MEG) of epileptic patients can be used to detect seizures. Due to the inefficient and error-prone operation of traditional HFOs detection, it is necessary to develop an approach for the detection of HFOs, which can automatically classify HFOs in MEG. This paper proposed a novel deep learning-based CADNet(**Gang neuron**) for the classification of HFOs in MEG. The model was evaluated on MEG dataset, and the accuracy, precision, recall, and F1-score of the optimized model reached 0.97, 0.98, 0.97, 0.97. This paper compared the proposed CADNet with other deep learning models, the result demonstrates that CADNet based on Gang neuron outperforms others.

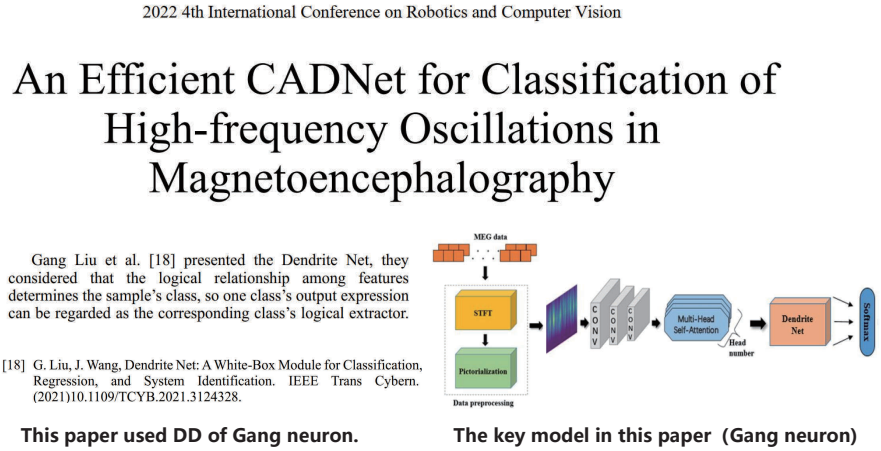


Fig. 38. **Epilepsy(Magnetoencephalography):***2022 ICRCV*

K. *Mechanical Engineering:Processes (Journal, IF:3.352)*

Study using Gang neurons [55]:"Multi-Objective Optimization for the Radial Bending and Twisting Law of Axial Fan Blades"(see Fig. 39)

The performance of low-pressure axial flow fans is directly affected by the three-dimensional bending and twisting of the blades. A new blade design method based on **Gang neurons** is adopted in this work, where the radial distribution of blade angle and blade bending angle is composed of standard-form rational quadratic Bézier curves. The simulation results show that the optimal blade load distribution along the radial direction becomes uniform, and the suction surface separation vortex and passage vortex are restrained. On the other hand, the tip leakage vortex is enhanced and moves toward the blade leading edge. According to the experimental results, the maximum efficiency increases by 5.44%, and the maximum total pressure coefficient increases by 2.47% after optimization.

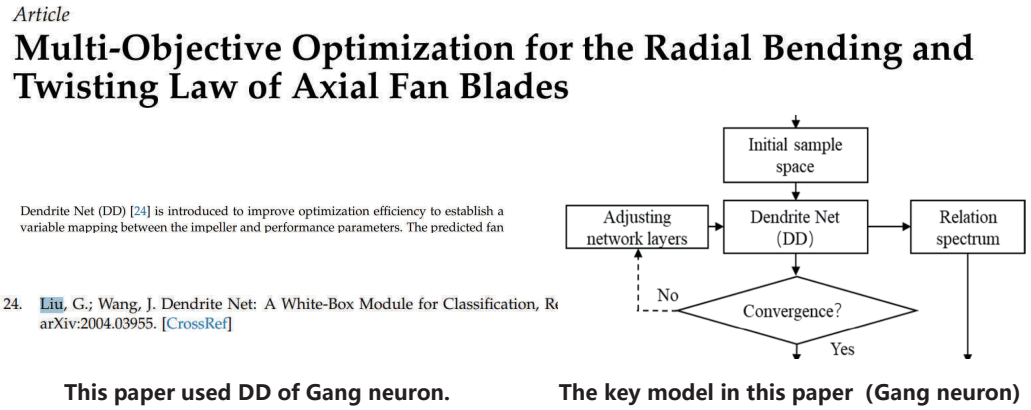


Fig. 39. **Mechanical Engineering:***Processes* (Journal, IF:3.352)

Gang neurons have been used in many fields and has demonstrated excellent characteristics. Here, I will not list them all due to the limited space of the article.

IX. Discussion and conclusion

In this paper, I compared the development of artificial neural networks with biological neural networks. As expected, the direction of their research fields is different. Researchers in ANNs pay attention to the architectures for different applications, while researchers in biology focus on exploring neural mechanisms. Both directions achieve success. It may be time to learn from each other.

As we know, artificial neural networks are inspired by biological neural networks. Seventy years ago, people designed artificial neurons with the knowledge about biological neurons at that time. Today, we have a better understanding of how the work of neurons, especially dendrites.

The traditional artificial neurons ignore that dendrites participate in pre-calculation in a biological neuron or biological neural network. More specifically, biological dendrites play a role in the pre-processing of the interaction information of input data. Two examples can be mentioned in this regard. In understanding a picture, biological dendrites play a role in extracting the relationship between parts of an input picture. In understanding an article, biological dendrites play a role in extracting the relationship between parts of an article.

Therefore, this paper improved the neuron of ANNs by introducing the dendrite module and presented a novel neuron named Gang neuron.

X. Outlook

This paper shows some basic architecture using Gang neurons. These network architectures using Gang neurons can be trained by $PyTorch$ expediently. [Note that the inputs and outputs are normalized to $[-1, 1]$.] These architectures in this paper are like some plugins. They can be used for other larger architectures that we want to design to improve the mapping capacity of networks.

There is an interesting phenomenon. Some data augmentation methods improve networks' performance, such as flipping and zooming in or out pictures. I believe this shows that the existing networks can not fully extract some simple information, such as flipping and zooming in or out pictures. However, our brains do this well.

REFERENCES

[1] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 335–346, 2020.

[2] A. M. Zador, "A critique of pure learning and what artificial neural networks can learn from animal brains," *Nature Communications*, vol. 10, p. 7, 2019.

[3] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. Atiah, V. Ravi, and A. Peters, "A review of deep learning with special emphasis on architectures, applications and recent trends," *Knowledge-Based Systems*, vol. 194, p. 33, 2020.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the Ieee*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the Acm*, vol. 60, no. 6, pp. 84–90, 2017.

[8] T. Mikolov, M. Karafiat, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *conference of the international speech communication association*, pp. 1045–1048, 2010.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] I. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *neural information processing systems*, pp. 2672–2680, 2014.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *computer vision and pattern recognition*, 2014.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *computer vision and pattern recognition*, pp. 1–9, 2015.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *computer vision and pattern recognition*, pp. 2818–2826, 2016.

[14] Y. Bengio, "Learning deep architectures for al," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[15] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *computer vision and pattern recognition*, pp. 5987–5995, 2017.

[16] S. Woo, J. Park, J. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *european conference on computer vision*, pp. 3–19, 2018.

[17] Y. Hu, G. Wen, M. Luo, D. Dai, and J. Ma, "Competitive inner-imaging squeeze and excitation for residual network," *arXiv: Computer Vision and Pattern Recognition*, 2018.

[18] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv: Computer Vision and Pattern Recognition*, 2015.

[19] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid long short-term memory," *arXiv: Neural and Evolutionary Computing*, 2015.

[20] L. Jing, C. Gulcehre, J. Peurifoy, Y. Shen, M. Tegmark, M. Soljacic, and Y. Bengio, "Gated orthogonal recurrent units: On learning to forget," *Neural Computation*, vol. 31, no. 4, pp. 765–783, 2019.

[21] F. Belletti, A. Beutel, S. Jain, and E. H. Chi, "Factorized recurrent neural architectures for longer range dependence," in *international conference on artificial intelligence and statistics*, pp. 1522–1530, 2018.

[22] Y. Bengio, N. Boulangerlewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *international conference on acoustics, speech, and signal processing*, pp. 8624–8628, 2013.

[23] G. E. Dahl, T. N. Sainath, G. E. Hinton, and Ieee, *IMPROVING DEEP NEURAL NETWORKS FOR LVCSR USING RECTIFIED LINEAR UNITS AND DROPOUT*, pp. 8609–8613. International Conference on Acoustics Speech and Signal Processing ICASSP, New York: Ieee, 2013.

[24] D. Mishkin and J. Matas, "All you need is a good init," in *international conference on learning representations*, 2015.

[25] D. Sussillo and L. F. Abbott, "Random walk initialization for training very deep feedforward networks," *arXiv: Neural and Evolutionary Computing*, 2014.

[26] Z. Shi, S. Zhang, J. Yuan, B. Zhu, Y. Jiang, X. Shen, and Y. Wang, "Spatiotemporal summation and correlation mimicked in a four-emitter light-induced artificial synapse," *Scientific Reports*, vol. 8, no. 1, pp. 2159–2159, 2018.

[27] J. Defelipe, L. Alonsonanclares, and J. I. Arellano, "Microstructure of the neocortex: Comparative aspects," *Journal of Neurocytology*, vol. 31, no. 3, pp. 299–316, 2002.

[28] S. Sardi, R. Vardi, A. Sheinin, A. Goldental, and I. Kanter, "New types of experiments reveal that a neuron functions as multiple independent threshold units," *Scientific Reports*, vol. 7, no. 1, pp. 18036–18036, 2017.

[29] Y. Timofeeva, S. Coombes, and D. Michieletto, "Gap junctions, dendrites and resonances: A recipe for tuning network dynamics," *Journal of Mathematical Neuroscience*, vol. 3, no. 1, pp. 15–15, 2013.

[30] H. Mohan, M. B. Verhoog, K. K. Doreswamy, G. Eyal, R. Aardse, B. Lodder, N. A. Goriounova, B. Asamoah, A. B. C. B. Brakspear, and C. Groot, "Dendritic and axonal architecture of individual pyramidal neurons across layers of adult human neocortex," *Cerebral Cortex*, vol. 25, no. 12, pp. 4839–4853, 2015.

[31] Y. Deitcher, G. Eyal, L. Kanari, M. B. Verhoog, G. A. A. Kahou, H. D. Mansvelder, C. P. J. De Kock, and I. Segev, "Comprehensive morpho-electrotonic analysis shows 2 distinct classes of l2 and l3 pyramidal neurons in human temporal cortex," *Cerebral Cortex*, vol. 27, no. 11, pp. 5398–5414, 2017.

[32] A. Gidon, T. A. Zolnik, P. Fidzinski, F. Bolduan, A. Papoutsi, P. Poirazi, M. Holtkamp, I. Vida, and M. E. Larkum, "Dendritic action potentials and computation in human layer 2/3 cortical neurons," *Science*, vol. 367, no. 6473, pp. 83–+, 2020.

[33] P. Poirazi and A. Papoutsi, "Illuminating dendritic function with computational models," *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 303–321, 2020.

[34] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[35] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[37] M. E. Larkum, T. Nevian, M. Sandler, A. Polsky, and J. Schiller, "Synaptic integration in tuft dendrites of layer 5 pyramidal neurons: A new unifying principle," *Science*, vol. 325, no. 5941, pp. 756–760, 2009.

[38] A. Polsky, B. W. Mel, and J. Schiller, "Computational subunits in thin dendrites of pyramidal cells," *Nature Neuroscience*, vol. 7, no. 6, pp. 621–627, 2004.

[39] P. Poirazi, T. Brannon, and B. W. Mel, "Pyramidal neuron as two-layer neural network," *Neuron*, vol. 37, no. 6, pp. 989–999, 2003.

[40] G. Liu and J. Wang, "Dendrite Net: A White-Box Module for Classification, Regression, and System Identification," *arXiv e-prints*, p. arXiv:2004.03955, Apr. 2020.

[41] G. Liu and J. Wang, "A Polynomial Neural network with Controllable Precision and Human-Readable Topology II: Accelerated Approach Based on Expanded Layer," *arXiv e-prints*, p. arXiv:2006.02901, June 2020.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[43] A. V. Ritchie, S. Van Es, C. Fouquet, and P. Schaap, "From drought sensing to developmental control: Evolution of cyclic amp signaling in social amoebas," *Molecular Biology and Evolution*, vol. 25, no. 10, pp. 2109–2118, 2008.
[44] J. Fitzakerley, "Sensory physiology," *https://www.d.umn.edu/ jfitzake/Lectures/DMED/SensoryPhysiology/GeneralPrinciples/LateralInhibition.html*, 2014.
[45] C. C. Aggarwal *et al.*, *Neural networks and deep learning*. Springer, 2018.
[46] P. Wang, T. Hu, F. Gao, R. Wu, W. Guo, and X. Zhu, "A hybrid data-driven framework for spatiotemporal traffic flow data imputation," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16343–16352, 2022.
[47] L. Zhang, J. Li, X. Xu, F. Liu, Y. Guo, Z. Yang, and T. Hu, "High spatial granularity residential heating load forecast based on dendrite net model," *Energy*, p. 126787, 2023.
[48] Y. Han, J. Li, X. Lou, C. Fan, and Z. Geng, "Energy saving of buildings for reducing carbon dioxide emissions using novel dendrite net integrated adaptive mean square gradient," *Applied Energy*, vol. 309, p. 118409, 2022.
[49] L. Lu, Y. Wu, Q. Zhang, P. Qiao, and T. Xing, "A radial sampling-based subregion partition method for dendrite network-based reliability analysis," *Engineering Optimization*, pp. 1–20, 2022.
[50] T. Li, C. Qiao, L. Wang, J. Chen, and Y. Ren, "An algorithm for precipitation correction in flood season based on dendritic neural network," *Frontiers in Plant Science*, vol. 13, 2022.
[51] G. Liu and J. Wang, "Eegg: An analytic brain-computer interface algorithm," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 643–655, 2022.
[52] X. Ma, X. Fu, Y. Sun, N. Wang, and Y. Gao, "Application of convolutional dendrite net for detection of myocardial infarction using ecg signals," *IEEE Sensors Journal*, 2022.
[53] K. Liu, J. Huang, Z. Liu, and Q. Wang, "Unteady aerodynamics modeling method based on dendrite-based gated recurrent neural network model," in *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, pp. 437–441, IEEE, 2022.
[54] M. Zhang, J. Liu, C. Liu, T. Wu, and X. Peng, "An efficient cadnet for classification of high-frequency oscillations in magnetoencephalography," in *2022 4th International Conference on Robotics and Computer Vision (ICRCV)*, pp. 25–30, IEEE, 2022.
[55] Y. Ding, J. Wang, B. Jiang, Z. Li, Q. Xiao, L. Wu, and B. Xie, "Multi-objective optimization for the radial bending and twisting law of axial fan blades," *Processes*, vol. 10, no. 4, p. 753, 2022.

Author :

Gang Liu is with School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou 450001, China. He received the PhD degree from Xi'an Jiaotong University. He is the Advisor to IOP Publishing and reviews for IEEE Transactions on Cybernetics(Tcyb), IEEE Transactions on Signal Processing(TSP), IEEE Transactions on Neural Systems and Rehabilitation Engineering(TNSRE), Journal of Neural Engineering(JNE), etc.

He presented Dendrite Net, Relation Spectrum, and Gang neuron. His current research interests include machine learning, deep learning, computer vision, natural language processing, and brain-computer interface.