Neurocomputing 465 (2021) 63–70

Contents lists available at ScienceDirect

Neurocomputing

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

Neural networks based on vectorized neurons

Ji He [a], Hongwei Yang [b], Lei He [a], Lina Zhao [a,c,*]

[a] The College of Mathematics and Physics, Beijing University of Chemical Technology, Beijing 100029, China
[b] Information Center, Beijing University of Chemical Technology, Beijing 100029, China
[c] Interdisciplinary Research Center for Artificial Intelligence, Beijing University of Chemical Technology, Beijing 100029, China

ARTICLE INFO

Article history:
Received 6 August 2020
Revised 23 August 2021
Accepted 2 September 2021
Available online 08 September 2021
Communicated by Zidong Wang

Keywords:
Vectorized neuron
Neuron2vector
Attention mechanism
Neural function group
Image classification
Few-shot learning

ABSTRACT

As the main research content of artificial intelligence, the artificial neural network has been widely concerned because of its excellent performance in the fields such as computer vision and natural language processing since it was proposed in the 1940s. The neuron model of the traditional neural network was proposed by McCulloch and Pitts in 1943 (MP neurons), But MP neurons is too simple to representing biological neurons. Based on this, this paper studies the attention mechanism and proposes vectorized neuron and its activation function. Firstly, we propose vectorized neurons, then use the attention mechanism to dynamically generate connection weights between vectorized neurons. Nextly, we construct a new type of neural network with vectorized neurons, which we called neural functional group (NFG). Finally, we tested the proposed neural functional group model on two tasks: image classifcation and few-shot learning. The vectorized neuron can be conditionally activated through its activation function. Besides, the vectorized neuron has the potential of representing complex biological neurons, which is difficult for MP neuron. The experimental results show that it can achieve higher accuracy with fewer parameters than convolutional neural networks (CNN) and capsule networks in image classication task; it also competitive to CNN based feature extractor in few-shot learning task.

1. Introduction

Artificial neural networks originated from MP neurons proposed in 1943 (McCulloch-Pitts Neuron) [1], it is a computing model that simulates the human brain nervous system. MP neuron is shown in Fig. 1. Where x_i is the input of the first neuron in the previous layer, and w_i is the connection weight of the $i-th$ neuron in the previous layer to the current neuron. Where $y = f\left(\sum_{i=1}^{M} w_i x_i - \theta\right)$. f is activation function (Generally differentiable), θ is the threshold for MP neuron. MP neurons are the foundation of modern neural networks.

Artificial neural networks can be divided into three categories according to the connection between neurons, namely: feedforward network, memory network, and graph network [2].Among them, the feedforward network layered the network according to the calculation order, the output of the previous layer is used as the input of the next layer, and the information flow direction of the entire network is from front to back. For example, fully connected feedforward networks [3] and convolutional neural networks [4] are both feedforward networks (convolutional neural networks belongs to locally connected feedforward networks). The memory network is a network in which neurons have a connection relationship with themselves. Such a network can not only receive information from other neurons but also its own historical information, that is, it has a memory function for past historical information, such as recurrent neural networks [5]. A graph network [6] is a neural network defined on graph structure data. Each node in the graph is composed of one or a group of neurons. The connection between the nodes can be directed or undirected. Each node can receive information from neighboring nodes or itself [2]. Graph network is a relatively new neural network, it is proposed to deal with graph data (molecular structure, social network, etc.).

According to the general approximation theorem [7], the neural network can fit any continuous function, so to some extent, the neural network can be used as a "universal" function, it can be used to perform complex feature conversion, or approximate a complex Conditional distribution [2] Although the general approximation theorem shows that both feedforward networks and recurrent networks have strong capabilities, it is difficult for these networks to have general approximation capabilities due to computational resources and learning algorithms. Especially when dealing with complex tasks, such as the need to process a large amount of input

* Corresponding author at: The College of Mathematics and Physics, Beijing University of Chemical Technology, Beijing 100029, China.
E-mail addresses: hj@jimhe.cn (J. He), zhaoln@mail.buct.edu.cn (L. Zhao).

https://doi.org/10.1016/j.neucom.2021.09.006
0925-2312/

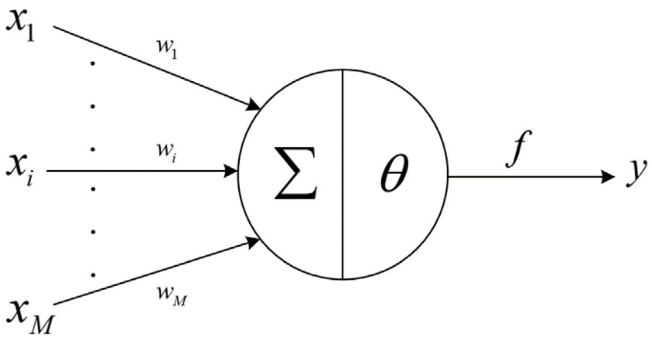

Fig. 1. MP Neuron.

information or complex calculation processes, the current computing power of computers is still the bottleneck that limits the development of neural networks [2].In order to reduce the computational complexity required by neural networks when facing large amounts of input information, people have proposed an attention mechanism [2], it draws on the attention that the human brain uses to process information, that is, it filters out a large amount of information that is irrelevant to the task and processes only important information. For example, max pooling [8] is an attention mechanism, it will only notice the most significant data in the input data. Recently, as attention mechanisms have been introduced into the field of natural language processing [9] and have surpassed the recurrent neural network model in more than 10 tasks such as machine translation, many researchers have also begun to pay attention to the attention mechanism and introduced the attention mechanism to many different fields such as robot path planning [10],recommendation system [11], Veličković and others in the field of graph network proposed graph attention network [12], etc. It can be said that the attention mechanism has been widely used in various neural network applications.

The above mentioned neural networks and their applications are actually neural networks based on MP neurons. However, because the MP model is too simple to simulate biological neurons [13], many researchers are not satisfied with the MP model. They proposed a kind of impulsive neuron more in line with biological neurons and built an impulsive neural network on this basis [14]. With the continuous development of neuroscience and artificial intelligence, it is foreseeable that not only will there be many studies in the future to expand neural networks from the vertical direction (based on the existing neural network), but there will also be a lot of work to expand from the horizontal direction Neural networks (propose new neurons and develop new neural network models). And this will greatly enrich the research of neural networks and even artificial intelligence.

In this paper, the attention mechanism and vectored neurons are studied. We propose vectored neurons and use the attention mechanism to dynamically generate connection weights between vectored neurons to construct a new type of neural network (called neural function group) and give its calculation formula. The vectorized neuron can be conditionally activated through its activation function. In addition, coding (representing) neurons with a learnable vector give us the potential to use the coding vector to represent complex biological neurons, which is difficult for MP models. We tested the neural functional group algorithm on two tasks: image classification and small data learning. The experimental results show that the neural functional group algorithm can achieve the effect comparable to the convolutional neural network and the capsule network with fewer parameters.

2. Neural Functional Group (neural network based on vectorized neurons)

Most major neuroanatomical studies support (in most mammals, especially primates) the existence of numerous columnar structures called Cortical minicolumn in the cerebral cortex, containing hundreds of neurons and internal stratification. This means that one layer in the human brain is not like the one in NN, but has a complex internal structure. According to the common existence of mini-column in the cerebral cortex [15], thought the minicolumn had something to do with it, and came up with a corresponding structure called capsule. The studies of the above researchers inspired this article, supervised training with vector as input feature has sufficient expressive ability to learn similarity between data and feature.

In the traditional neural network, the neurons are MP neurons, and the neurons themselves have no parameters. The parameters of the traditional neural network are the connection weights and bias terms between neurons. In fact, biological neurons are much more complicated than MP neurons, and MP neurons are also difficult to simulate biological neurons [14]. Therefore, we believe that neurons should require more complex parameters to describe, so we use vectors to encode a neuron. In this chapter, we will build a neural network different from the traditional neural network, a neural network based on vectorized neurons. The activation function of the traditional neural network is completely a function that takes input data as an independent variable. The activation function has nothing to do with the neuron itself, but in fact, the activation state of a neuron depends not only on the input but also on the neuron itself. For example, if the same input signal is input to different neurons, the activation state of different neurons may be different [16]. Therefore, we have defined a new type of activation function, which can be selectively activated according to the encoding (vector representation) of the neuron itself. Based on this, we propose a neural network architecture based on vectorized neurons, called the neural function group. Our experiments show that neuronal functional groups can learn good neuron representations and can use the attention mechanism to dynamically generate connection weights between vectored neurons. Finally, we tested our model on the two tasks of small data learning and image classification. In the image classification task, we can use parameters that are hundreds of times smaller than the Capsule Net to achieve more than CNN and capsule network effects. In the small data learning task, the effect of the neural function group model can be comparable to the CNN-based feature extractor. This also shows that the model based on the neural function group has great potential for future development.

2.1. Neural function group model

2.1.1. Neuron model and its activation function

The neuron model we defined is shown in the Fig. 2, where $y \in R^d$ is the neuron encoding, d is the length of the encoding, $x \in R^p$ is the input data, C is the context of input data $x, x \in C$.

We want to know whether x will cause the activation of neurons and how much of the activation amount. For this purpose, we first define the activation function as shown in Eq. 1

$$\sigma(x|y, C) = \rho(x|y, C)xW \tag{1}$$

where $W \in \mathbb{R}^{p\times *}$ is the weight matrix used to transform x, ρ is activation amount function determines the amount of activation, its value range is $\mathbb{R}^+$, the formula of ρ is shown in Eq. (2)

$$\rho(x|y, C) = \frac{\max(xW_x + yW_y, 0)\beta(C)}{\sum_{i=1}^{|C|} \max(x_iW_x + yW_y, 0) + \epsilon} \tag{2}$$

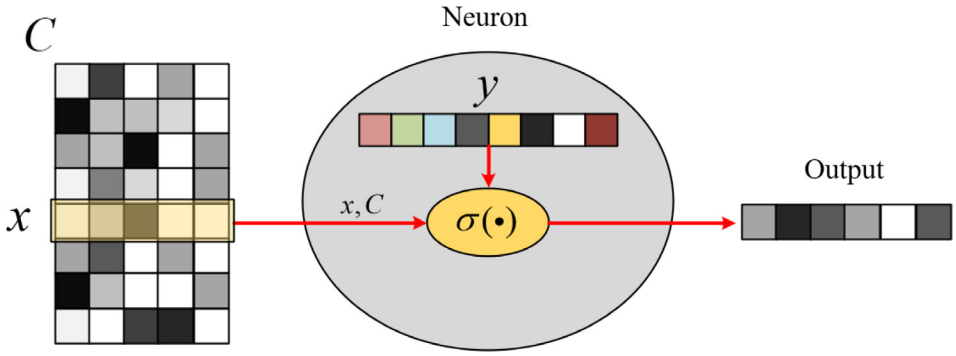

Fig. 2. Neuron model.

where $W_x \in \mathbb{R}^{p\times 1}, W_y \in \mathbb{R}^{d\times 1}$ are the learned weight matrix, $x_i \in C$ and $|C|$ is the number of elements in C, C is the context information of the current input x.It is easy to see that $\sum_{i=1}^{|C|}\rho(x_i|y,C)=\beta(C)$, where β is the budget function which we defined to control the activation amount according to the context C its value range is $\mathbb{R}^+$. For the context C, the sum of its activations is $\beta(C)$. In this article, we let $\beta(C)=|C|$. Besides, in Eq. (2) if $xW_x+yW_y=0$, then $\rho(x|y,C)=0$, say, the activation amount is 0, the input will not cause the activation of neurons (activation depends not only on the input, but also on the neuron itself, so it can be selective for input activation). In Eq. (2) $\max(x,0)$ is the ReLU function [17]. ϵ represents a small number used to prevent zero denominator.

If we want to process the input in batches, suppose we have m inputs to be input to m neurons separately, we use $X \in \mathbb{R}^{m\times p}$ to represent m inputs which represent the encodings of m neurons, then the activation function can be written as Eq. (3)

$$\rho(X|Y,C)=\frac{\max(XW_x \oplus YW_y,0)\odot\beta(C)}{\sum_{i=1}^{|C|}\max(x_iW_x \oplus YW_y,0)+\epsilon} \tag{3}$$

where $x_i \in C$, it is easy to see that $\rho(X|Y,C)$ is a m-dimensional vector. The symbol $\oplus$indicates the component-by-component addition between vectors; in addition, scalar and vector additions are allowed, which is to add the scalar to each component of the vector separately. $\odot$represents component-by-component multiplication between vectors; in addition, scalar and vector multiplication are allowed, which is to multiply the scalar and each component of the vector separately.

Fig. 3 shows the schematic diagram of batch processing of 5 input data by 5 neurons when $m=5$, where $C=(x_1,\cdots,x_9)^{\mathrm{T}}, X=(x_3,x_4,x_5,x_6,x_7)^{\mathrm{T}}$ we need to input X to the neuron group $Y=(y_1,y_2,y_3,y_4,y_5)^{\mathrm{T}}$, the corresponding position of X is input to the corresponding Y position, x_3 input to y_1, x_4 input to y_2, and so on.

Finally, we will get the output after the activation function σ, each neuron corresponds to an output, where y_1 corresponds to the output o_1 and $o_1=\sigma(x_3|y_1,C), o_2=\sigma(x_4|y_2,C)$ and so on, where $\sigma(X|Y,C)=\rho(X|Y,C)\odot XW$.

2.1.2. Neural function group and it's computation

A neural function group (NFG) is composed of multiple neurons, and each neuron has a vectorized encoding. We can concatenate the encodings of all neurons to form an encoding matrix. We also call this encoding matrix the neural encoding matrix of the neural function group. For example, Y in Fig. 3 can be regarded as a neural encoding matrix. But we know that there is a connection relationship between neurons, so a neuron function group requires both neurons encoding and the strength of the connection between neurons. Given a matrix $Y=(y_1,\cdots,y_m)^{\mathrm{T}}$ of m neurons and their neural encoding vectors, we now need to define the connection strength between the neurons. This strength needs to be a directed strength, such as y_1 to y_2 The connection strength is different from y_2 to y_1. For Y, there will be a connection strength between two elements, so all the connection strengths can form a matrix $A \in [0,1]^{m\times m}$.

We can think of a neural functional group as being composed of a neural encoding matrix and a neural connection matrix. Therefore, we define a neural functional group as a binary group consisting of a neural encoding matrix and a neural connection matrix. That is the neural function group $G=(Y,A)$, where Y is the neural coding matrix and A is the neural connection matrix.

A simple and effective method to represents the connection strength between neurons is to use the attention distribution in

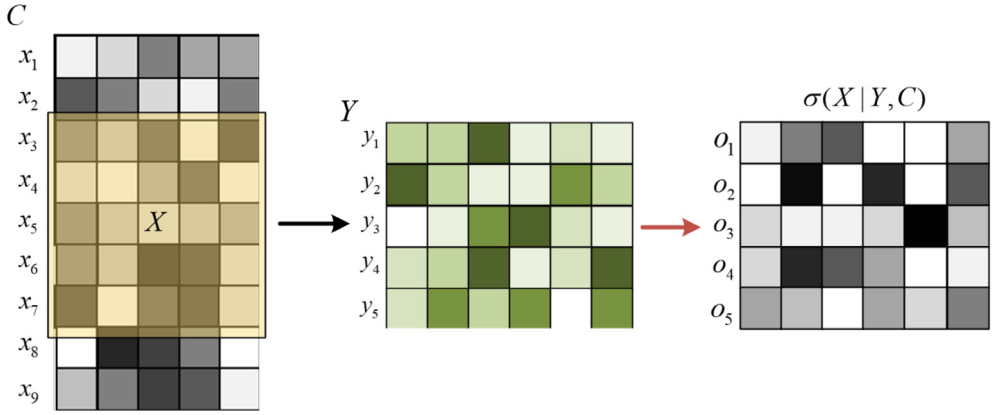

Fig. 3. Input X and it's context C into multiple neurons Y, then processing through activation function.

the attention mechanism. We know that calculating the attention distribution requires defining a scoring function and a regularization function. First, we define the scoring function $score : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}, score(y_i, y_j) = y_i W_Q + y_j W_K$, where, $W_Q \in \mathbb{R}^{d\times 1}$ is the weight matrix. We use *softmax* function as the regularization function. Therefore, the $i-th$, column of the connection strength A on Y is shown in Eq. (4).

$$(a_1, a_2, \cdots a_m) = soft\max(score(y_i, y_1), score(y_i, y_2), \cdots, score(y_i, y_m)) \tag{4}$$

For simplicity, we will record the function of calculating the connection strength between neurons as $Atten, Atten(Y) = A, A$ is defined as above.

We define a neural function group $G = (Y, A)$ as the input form is $I = (X, C)$, where X is the input data and C is the context information of the input data. The number of rows of Y is equal to the number of rows of X, and the data of the $i-th$ row of X is input to the $i-th$ neuron in Y. The computation process of the neural functional group is shown in Fig. 4. The computation of the neural functional group includes three steps:

The first step is data input, indicated by the black arrows in the Fig. 4.

The second step is to use the attention mechanism to calculate the connection strength matrix $A = Atten(Y)$ according to the neural coding matrix. At the same time, input the input data X, C and the neural encoding matrix into the activation function to obtain the activation function. Then output $\sigma(X|Y, C)$, this step is indicated by the red arrow in the Fig. 4.

The third step is to multiply the calculated connection strength matrix $A = Atten(Y)$ and $\sigma(X|Y, C)$ to obtain the output. This step is indicated by a yellow arrow. The third step, that is, multiplying the output of the activation function by the connection intensity matrix is that the output of the activated neuron is transmitted and fused among the neurons in the neuron function group through the connection intensity matrix A

Therefore, the computation formula of neural function group $G = (Y, A)$ on input $I = (X, C)$ can be written as Eq. (5)

$$G(I) = A\sigma(X|Y, C) \tag{5}$$

where $G(I)$ represents the calculation and role of the neurological group $G = (Y, A)$ on the input $I = (X, C)$. In particular, in this article we let $G(I) = Atten(Y)\sigma(X|Y, C)$.

Drawing inspirations from the multi-head attention in the attention mechanism, we define the multi-head neural functional group below. For the input $I = (X, C)$, if we use L neural functional groups $G_1, \ldots, G_L$ to process I separately, then we define an integration of these neural functional groups, namely the multi-headed neural functional group $G(I)$ and it's the calculation equation, see Eq. (6)

$$G(I) = (G_1(I) \oplus G_2(I) \oplus \cdots \oplus G_L(I))W_O \tag{6}$$

Among them, $\oplus$ represents the concatenate operation of vector, W_O is the weight matrix. The above-mentioned multi-head neural functional group is a parallel structure. We can also define a serial neural functional group structure, which is similar to a feedforward neural network. The output of the previous neural functional group is used as the input of the next neural functional group. Information is spread from front to back. We call the network constructed in this way a feedforward neural function group network. For the input $I = (X, C)$, suppose we have the $L-layer$ neural function group $G^{(1)}, \ldots, G^{(L)}$, where, $G^{(1)} = (Y^{(1)}, A^{(1)}), G^{(2)} = (Y^{(2)}, A^{(2)})$, $\ldots$, $G^{(L)} = (Y^{(L)}, A^{(L)})$. We let $G^{(1)}$ be the input neural functional group and G_L the output neural functional group. Without loss of generality, we give the calculation formula for layer $l, (1 <= l <= L+1)$, see Eq. (7)

$$C^{(l+1)} = X^{(l+1)} = A^{(l)}\sigma(X^{(l)}|Y^{(l)}, C^{(l)}) \tag{7}$$

where $A^{(l)} = Atten(Y^{(l)})$, we let $X^{(1)} = X$, $C^{(1)} = C$, finally output $X^{(L+1)}$.

2.1.3. Convolution based on neural function group

Convolutional neural networks (CNN) are very good at processing images, so we used CNN's approach when applying neural function groups (NFG) to the image field. For example, CNN only processing a small block on an image at a time and then traverses the image in a certain step size to extract features. Drawing inspiration on CNN, the neural function group will only process a small block at a time, and then traverse the entire image in certain step size. CNN relies on convolution kernels to extract features; how-

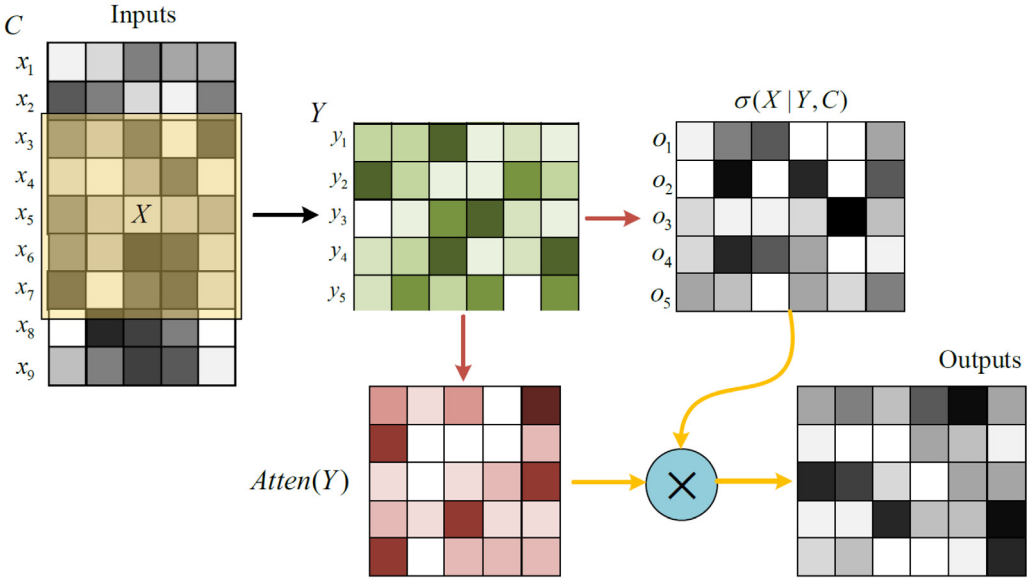

Fig. 4. Computational steps of neural function group.

ever, neural function groups use their "carried" neurons to extract features. This entire process is shown in Fig. 5.

In Fig. 5, C is an image, X is a (3×3) image block, let G have 9 neurons, and each corresponding pixel of X will be input to the corresponding neuron, for example, pixel 1 is input to neuron 1, pixel 2 is input to neuron 2. If the neural function group G traverses an image with a height of H, and a width of W with a window (two-dimensional) size (k_H, k_W) and step size (S_H, S_W), the neural function group will output O with width of $(W - k_W + 2 \times padding)/s_W$ and height of $(H - k_H + 2 \times padding)/s_H$. In addition, when applied to images, we use the output of a neuron to represent the output of the entire neuronal functional group. The calculation equation of the neural function group on the imagery input $I = (X, C)$ is determined by Eq. 4, the only difference is lies in activation amount function. In order to reduce the computation complexity, we use a the activation amount of a certain neuron in the neural function group to represents the activation amount of the entire neural function group on the input. So the activation amount function can also be written as Eq. (8)

$$\rho(X|y \in Y, C) = \frac{\max(XW_x \oplus YW_y, 0) \odot \beta(C)}{\sum^{|C|} \max(x_i W_x + yW_y, 0) + \epsilon} \tag{8}$$

Our experiments found that pooling methods like Maxpooling or Averagepooling could not improve the NFG, so we proposed a pooling method called Aggregator, which is shown in Fig. 6. Aggregator is a simple image feature collector used to provide a pixel and it's context information to each neuron of NFG. The output from Aggregator will be directly input into NFG. Aggregator actually expands all pixel features in a small window (such as a 3×3 small window) into a vector. Aggregator traverses an image with a height of H and a width of W with a window (two-dimensional) size (k_H, k_W) and step size (S_H, S_W), the Aggregator will output O with width of $(W - k_W + 2 \times padding)/s_W$ and height of $(H - k_H + 2 \times padding)/s_H$.

3. Experiment

We validated the neural functional group model on small data learning [18,19] and image classification experiments. Among them, few-shot learning aims to learn from very little data. We know that people can learn from small data. For example, we only need to look at very few pictures of cats and dogs, then we learn how to distinguish cats and dogs. Few-shot learning is trained on a dataset *Train* with C_{train} classes, then tested on a dataset *Test* with C_{test} new classes. During testing, only a few samples of each class have label.

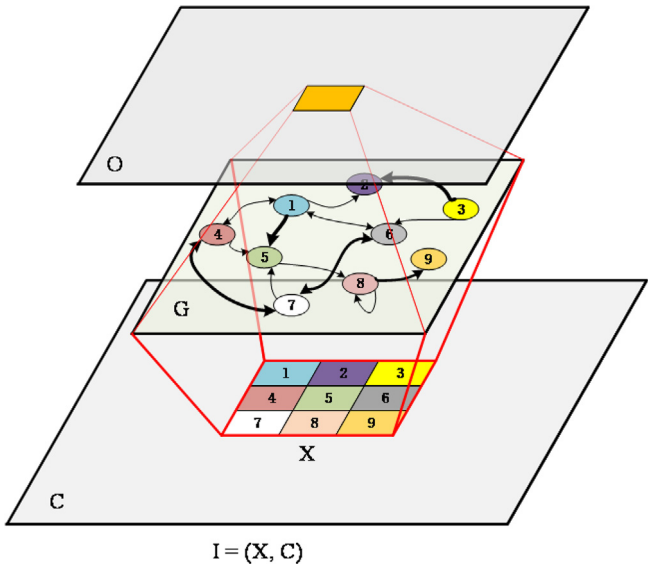

Fig. 5. Convolution based on neural functional group.

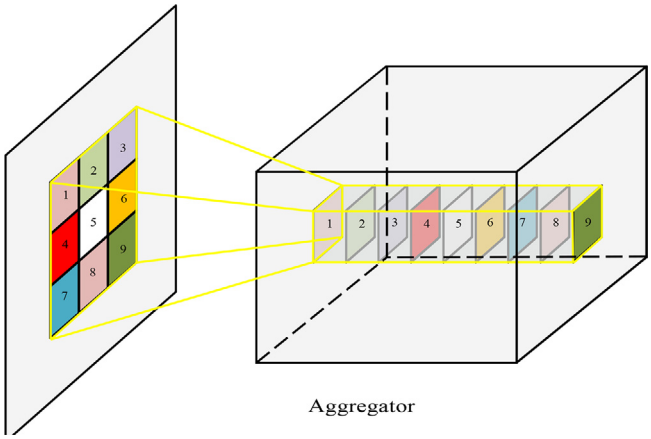

Fig. 6. The aggregate operation.

At each episode, we will select N classes of data from the C_{train} classes of *Train* to construct support set S and query set Q. Where S is a set of K elements selected from each of the N classes, and there are NK elements in total. We also call this way we constructing S as $N - way, K - shot$ setting. Q is a set of T elements randomly selected from the data of these N classes. At this time, the support set S is similar to the labeled training set, and the query set Q is similar to the test set. The purpose of this episode is to minimize the loss function on Q, then continue to the next episode of training, until maximum episode is reach.

3.1. Few-shot learning experiment

This experiment aims to compare few-shot learning based on CNN and neural function group (NFG). Our experiment was under the framework of the prototype network [20]. We performed few-shot target recognition experiments on the mini-ImageNet dataset, which was first proposed by Vinylals et al. in 2016 [19]. It is a new dataset composed of 100 classes of data selected from the ImageNet dataset [21], and each class contains 600 images. Recently, this dataset was divided into 64 classes, 16 classes, and 20 classes and was used as the training set, validation set, and test set respectively by Ravi Larochelle et al. [22]. We performed few-shot fine-grained image classification experiments on the CUB-200-2011 dataset [23] (referred to as the CUB dataset). The CUB dataset contains a total of 200 classes of data, a total of 11,788 images. According to the division rules of Hilliard et al. [24], we randomly select 100 classes as the training set, 50 classes as the verification set, and 50 classes as the test set for the CUB data set. The experimental results are shown in Table 1. In Table 1, each experiment was run 5 times, then averaged, and the standard deviation was recorded. ProtoNet (reported) and protoNet# (reported) are the results in the literature (see this paper [25] for details) ProtoNet (Our) and ProtoNet* (Our) are the results of our reproduced result, NFG- ProtoNet and NFG-ProtoNet* are models after replacing all CNN parts with NFG, and replacing Batch Normalization with Layer Normalization in ProtoNet and ProtoNet*. Among them, ProtoNet (reported), ProtoNet (Our), NFG-ProtoNet models are all trained in the way of 5-way, 5-shot, and ProtoNet* (reported), ProtoNet* (Our), NFG-ProtoNet* are all used 20-way, 5-shot training setting

Table 1
Comparative experiment of few-shot learning.

Methods	CUB		mini-ImageNet	
	1-shot	5-shot	1-shot	5-shot
ProtoNet(reported)	51.31 ± 0.91	70.77 ± 0.69	44.42 ± 0.84	60.24 ± 0.72
ProtoNet(Our)	45.67 ± 2.01	70.83 ± 1.09	44.05 ± 0.55	**63.80 ± 0.51**
NFG-ProtoNet	**56.03 ± 0.96**	**72.58 ± 0.75**	**47.11 ± 0.37**	63.62 ± 0.61
ProtoNet*(reported)	Not reported	Not reported	47.74 ± 0.84	66.68 ± 0.68
ProtoNet*(Our)	–	–	42.95 ± 0.74	66.42 ± 0.38
NFG-ProtoNet*	–	–	**51.04 ± 0.56**	**68.34 ± 0.40**

(the second set of experiments in Table 1). The number of episodes for each training is 200. All the above models are tested using 5-way, 1-shot, and 5-way, 5-shot setting. The test results are the average accuracy of each model on 500 randomly generated test episodes. Besides, the data augmentation method we used during training is different from the literature [25] The literature uses the random crop, left–right flip, and color jitter augmentation methods, but in our experiments, only left–right flip method is used. Therefore, if compared with the results in [25], the NFG-ProtoNet and NFG-ProtoNet# are underestimated, but even so, the performance of NFG-ProtoNet and NFG-ProtoNet# still surpass ProtoNet and ProtoNet# in most cases. In particular, in the 1-shot task, NFG-ProtoNet greatly surpass ProtoNet. For example, the accuracy of NFG-ProtoNet in the 1-shot task on the CUB dataset is absolutely improved 10.36 and 4.72 points over ProtoNet (Our) and ProtoNet (reported) respectively. The accuracy of NFG-ProtoNet on the 1-shot task on the mini-ImageNet dataset is absolutely improved 3.06 and 2.69 points over ProtoNet (Our) and ProtoNet (reported) respectively.

Fig. 7 shows the feature extractor used in ProtoNet (left) and NFG-ProtoNet (right). ProtoNet uses CNN-based feature extractor, while NFG-ProtoNet uses NFG-based feature extractor. For CNN-based feature extractor, the step size in both height and width directions is 1, for the NFG-based feature extractor, the step size of each layer of NFG in both height and width directions is 2. As for Aggregator, the window size of the first layer is 3×3, the step size is 1, otherwise, the window size is 2×2 and the step size is 1.

In Fig. 7, we have omitted the activation function (in fact, for CNN-based feature extractors, there will be a ReLU activation function after Batch Norm). In Fig. 7, Batch Norm represents Batch Normalization, and Layer Norm represents Layer Normalization. $3 \times 3\ conv\ 64$ represents a convolutional neural network layer with a convolution kernel size of 3×3, a number of convolution kernels of 64, and a step size of 1. $3 \times 3\ NFG\ 64$ means to input 3×3 squares to the neural function group layer (each neural function group in the neural function group layer is shown in Fig. 5), and the final output dimension is 64. It is worth noting that if the input size is a 3×3 block, then each neuronal group that receives the input data has 9 neurons. In this experiment, the neural functional group layer contains 8 neural functional group, and each neural functional group carries 9 neurons, the dimension of neuron coding vector of each neuron is 8. The step size of the neural functional group layer moving on the image is 2 (similar to the step size of the convolution layer).

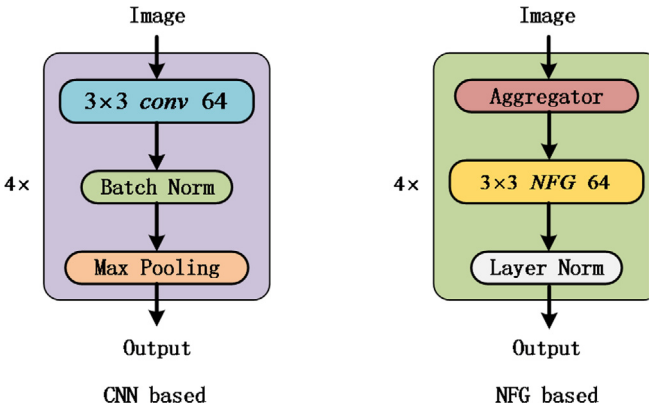

Fig. 7. The feature extractor of CNN based (left) and NFG based (right) in few-shot learning.

3.2. Image classification experiment

Image classification is the most basic visual task. This experiment aims to compare image classification algorithms based on neural function groups, convolutional neural networks, and capsule networks [15]. The datasets used in this experiment is the widely used MNIST [26] and Cifar10 [27] datasets. We only use a single model and no model ensembling. For the MNIST dataset, CNN [28] achieved a test error of 0.21% through model ensembling and data augmentation(rotation and scaling). Without model ensembling and rotation and scaling data augmentation strategies, CNN achieve a test error of 0.39% (only shift data augmentation is used). The capsule network [15] can achieve a test error of 0.25% through image reconstruction. Without image reconstruction, the capsule network can achieve a test error of 0.34%.

For fair comparison, we used the same data augmentation methods as Baseline [15], CNN [28], and Capsule Network [15] on the MNIST dataset, that is, randomly shifting the image in up to two pixels in all directions (shift data augmentation), and then fill the shifted image with 0 to the original size. We did not use any other data augmentation methods, nor did we use image reconstruction techniques. Baseline [15] is a standard 3-layer convolutional neural network. The number of convolution kernels in each layer is 256, 256, and 128. The size of each layer's convolution kernel is 5 and the step size is 1. After convolution It is a two-layer fully connected layer, and the number of neurons in each layer is 328 and 192, respectively. The neural functional group layer used in this experiment is basically the same as in Fig. 7, the only difference is that only 3 serially connected neural functional group layers are used in this experiment, and the few-shot learning experiment uses 4 serially connected neural functional group layer. The neural function group layer of this experiment contains 8 neuron function groups, each neuron function group contains 9 neurons, and the dimension of coding vector of each neuron is 8. The step size of the neural functional group layer is 2 (similar to the step size of the convolution layer). Our experimental results on MNIST are shown in Table 2.

It is easy to see that the accuracy of the 3-layer NFG is superior to Baseline, and the number of parameters of the 3-layer NFG is hundreds of times less than that of Baseline and capsule network. We did not use any data augmentation methods on the Cifar10 dataset. The 3-layer CNNs [28] contains three layers of convolutional neural networks. Each layer has a convolution kernel size

Table 2
Comparative experiment in MNIST dataset.

Methods	Test error (%)	#Parameters (million)
Baseline	0.39	35.40
CNN	0.39	Not reported
Capsule Networks	0.34	6.80
3-layer NFGs (Ours)	**0.30**	**0.067**

Table 3
Comparative experiment in Cifar10 dataset.

Methods	Test error (%)	#Parameters (million)	Time (hour)
3-layer CNNs	19.7	0.12	**1.81**
3-layer NFGs	**16.9**	**0.069**	8.84

of 5, and a step size of 1. The experimental results on Cifar10 are shown in Table 3. In Table 3, the computation time between CNN based model and NFG based model are alse compared. 3-layer CNNs [28] is roughly 5 times faster than 3-layer NFGs (we run 100 epochs with one NVIDIA Tesla P100 GPU). The reason why NFG based method is slower than CNN based method might be NFG based method has a relatively complex activation function (one solution to speed up the NFG based method is not to compute the neurons with zero activation, we will solve this in the future).

4. Conclusion

This paper studies the attention mechanism and the vectored neuron model. Vectored neurons can be conditionally activated through the conditional activation function. In addition, the vectored neuron we proposed uses a vector to encode the neuron so that it has the potential to simulate more complex biological neurons, which is difficult for MP neurons to do. Based on vectored neurons, we use the attention mechanism to generate connection weights between vectored neurons and construct a neural network, that is, a neural function group model. We tested the neural functional group model on the image classification and few-shot learning tasks. In the image classification task, the neural function group can achieve better classification results with hundreds of times less parameters than the CNN and the capsule network; In the few-shot learning task, the neural function group can be comparable to the CNN based Feature extractor. This also proves that vectorized neurons and neural functional group models have great potential (Maybe one day, people can encode a real-world neuron into a vector using some neuron2vector model, just like what word2vector model does. The "neuron encoding" computed through the neuron2vector model supposed to represents the neuron itself, in other words, make the neuron's behavior more predictable through the neuron encoding. And when the time is come, vectorized neurons may have some applications in the field).

Although the neural functional group model achieves comparable or even better results than the convolutional neural network and the capsule network in the image classification task, and the parameters used are less than 1/100 of the Baseline [4] (a convolutional neural network) and the capsule network [15], the speed of the neural functional group is slower than that of the convolutional neural network. Therefore, How to speed up the neural functional group is a technical problem that needs to be solved urgently (one solution that has come up at present is not to compute the neurons with zero activation in the neural functional group). In addition, the neural functional group model currently stays at the shallow layer (<=4 layer, due to lack of computing resources, we only stay at the shallow layer network at present). In the future, we hope to develop hundreds of layers or even deeper neural functional group models. Then the deep neural functional group model is compared with the more complex convolutional neural network (RESNET, etc.) on a more complex data set (ImageNet).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work is supported by the Fundamental Research Funds for the Central Universities (XK2020-03) and the National Natural Science Foundation of China 11301021.

References

[1] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The Bulletin of Mathematical Biophysics 5 (4) (1943) 115–133.
[2] X. Qiu, Neural Networks and Deep Learning, China Machine Press, Beijing, 2020.
[3] K. Hornik, M. Stinchcombe, H. White, et al., Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[4] K. Fukushima, Neocognitron: A hierarchical neural network capable of visual pattern recognition, Neural Networks 1 (2) (1988) 119–130.
[5] J.L. Elman, Finding structure in time, Cognitive Science 14 (2) (1990) 179–211.
[6] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems. .
[7] A.N. Gorban, I.I.D., The general approximation theorem, in: IEEE World Congress on Computational Intelligence the IEEE International Joint Conference on Neural Networks, 1998. .
[8] N. Murray, F. Perronnin, Generalized max pooling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2473–2480.
[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: NIPS, 2017.
[10] K. Fang, A. Toshev, L. Fei-Fei, S. Savarese, Scene memory transformer for embodied agents in long-horizon tasks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 538–547.
[11] Q. Chen, H. Zhao, W. Li, P. Huang, W. Ou, Behavior sequence transformer for e-commerce recommendation in alibaba, in: Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data, 2019, pp. 1–4.
[12] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903. .
[13] Z.Z.H. Shao-Qun Zhang, Flexible transmitter network, arXiv preprint arXiv:2004.03839v2. .
[14] W. Maass, Networks of spiking neurons: the third generation of neural network models, Neural Networks 10 (9) (1997) 1659–1671.
[15] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, in: Advances in Neural Information Processing Systems, 2017, pp. 3856–3866. .
[16] L. Tao, D. Porto, Z. Li, S. Fechner, S.A. Lee, M.B. Goodman, X.S. Xu, H. Lu, K. Shen, Parallel processing of two mechanosensory modalities by a single neuron in c. elegans, Developmental Cell 51 (5) (2019) 617–631.
[17] G.E. Dahl, T.N. Sainath, G.E. Hinton, Improving deep neural networks for lvcsr using rectified linear units and dropout, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 8609–8613. .
[18] R. Vilalta, Y. Drissi, A perspective view and survey of meta-learning, Artificial Intelligence Review 18 (2) (2002) 77–95.
[19] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: Advances in Neural Information Processing Systems, 2016, pp. 3630–3638. .
[20] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4077–4087. .
[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255. .
[22] S. Ravi, H. Larochelle, Optimization as a model for few-shot learning. .
[23] K. Duan, D. Parikh, D. Crandall, K. Grauman, Discovering localized attributes for fine-grained recognition, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 3474–3481.
[24] N. Hilliard, L. Phillips, S. Howland, A. Yankov, C.D. Corley, N.O. Hodas, Few-shot learning with metric-agnostic conditional embeddings, arXiv preprint arXiv:1802.04376. .
[25] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C.F. Wang, J.-B. Huang, A closer look at few-shot classification, arXiv preprint arXiv:1904.04232. .

[26] L. Deng, The mnist database of handwritten digit images for machine learning research [best of the web], IEEE Signal Processing Magazine 29 (6) (2012) 141–142.
[27] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images. .
[28] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus, Regularization of neural networks using dropconnect, in: International Conference on Machine Learning, 2013, pp. 1058–1066.

Ji He received the B.S. & Master degree from the Beijing University of Chemical Technology, Beijing, China, in 2020. His supervisor is Dr. L. Zhao. He is currently an investor and an Independent researcher. His interests include brain-inspired computing, general artificial intelligence, blockchain and cryptography, drug-discovery, etc.

Hongwei Yang received the Ph.D. degree from Zhejiang University, Hangzhou, China. She is currently a Lecturer with the Beijing University of Chemical Technology, Beijing, China. Her research interests include data mining, deep learning, and complex graph networks.

Lei He received the B.S. degree from the Beijing University of Chemical Technology, Beijing, China, in 2017, where he is currently pursuing the master's degree. His supervisor is Dr. L. Zhao. His research interests include remote sensing image processing and pattern recognition, tensor neural network, and artificial intelligence.

Lina Zhao received the Ph.D. degree in applied mathematics from the Chinese Academy of Sciences, Beijing, China, in 2004. She is currently an Associate Professor withthe Beijing University of Chemical Technology, Beijing, China. Her research interests include computational and applied mathematics, which includes geometric problems in computer vision, image processing, and information mining from 3-D Big Data.