Neural Networks 168 (2023) 484–495

Contents lists available at ScienceDirect

Neural Networks

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

Understanding neural network through neuron level visualization

Hui Dou [a,b], Furao Shen [a,d,*], Jian Zhao [c,**], Xinyu Mu [a,d]

[a] State Key Laboratory for Novel Software Technology, China
[b] Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China
[c] School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China
[d] School of Artificial Intelligence, Nanjing University, Nanjing 210023, China

ARTICLE INFO

Keywords:
Interpretability
Neural network
Visualization

ABSTRACT

Neurons are the fundamental units of neural networks. In this paper, we propose a method for explaining neural networks by visualizing the learning process of neurons. For a trained neural network, the proposed method obtains the features learned by each neuron and displays the features in a human-understandable form. The features learned by different neurons are combined to analyze the working mechanism of different neural network models. The method is applicable to neural networks without requiring any changes to the architectures of the models. In this study, we apply the proposed method to both Fully Connected Networks (FCNs) and Convolutional Neural Networks (CNNs) trained using the backpropagation learning algorithm. We conduct experiments on models for image classification tasks to demonstrate the effectiveness of the method. Through these experiments, we gain insights into the working mechanisms of various neural network architectures and evaluate neural network interpretability from diverse perspectives.

1. Introduction

Artificial Neural Networks (ANNs) are among the most widely used artificial intelligence methods nowadays (Bodria et al., 2021; Zhou, Sun, Bau, & Torralba, 2018). However, despite their success in various applications, the interpretability of ANN models remain an open problem and the decision-making process is unreliable in many practical applications (Bau, Zhou, Khosla, Oliva, & Torralba, 2017; Rudin et al., 2021). In some areas that require high precision and reliability in decision-making, e.g., medical treatment (Zhang, Xie, Xing, McGough, & Yang, 2017; Zhu, Ma, Yuan, & Zhu, 2022), autonomous driving (Kim & Canny, 2017; You, Leskovec, He, & Xie, 2020) and criminal justice (Liang, Li, Yan, Li, & Jiang, 2021; McGough, 2018), every decision must have sufficient evidences and convincing reasons. However, ANN models were historically regarded as “black box" models. In the whole process, users do not understand the basis for the judgment of the result, nor can they judge whether the results produced by the model are correct (Lipton, 2018). Therefore, it is difficult for users to completely trust and rely on decisions made by the model (Ghorbani, Abid, & Zou, 2019) and it is necessary to add an interpretation module into the neural network model to explain how it works (Nam, Gur, Choi, Wolf, & Lee, 2020).

Visualizing features learned by individual neurons is an important way to understand the working mechanism of neural networks. There are different architectures of ANNs, e.g., Fully Connected Networks (FCNs) and Convolutional Neural Networks (CNNs). Some of their components have the ability to localize objects, e.g., the convolutional layer and the pooling layer, while others have no localization ability, e.g., the fully-connected layer (Peng et al., 2021; Zhou, Khosla, Lapedriza, Oliva, & Torralba, 2016). There are several methods to visualize the learned features of the components with localization ability (Zhang, Tiňo, Leonardis, & Tang, 2021). However, it is hard to know the information that the components without localization ability have learned. This makes it difficult to compare the output results of different types of components.

In this paper, we propose a method named *Neural Network Scanner (NNS)*. It is used to visualize neuron learning results for different types of neural networks. The features learned by individual neurons can be combined flexibly to analyze different components of the neural network. The results of different components are visualized in a unified way. So we can compare the differences in the operating mechanisms of different models. In the medical field, researchers use external devices (e.g., the functional magnetic resonance imaging (fMRI)) to capture images of the brain and determine the activations of cerebral neurons. After the human brain receives information, fMRI captures the changes in hemodynamics caused by neuron activities. As the fMRI scans human brains, The proposed NNS is a method to scan ANN models. Given an

* Corresponding author at: State Key Laboratory for Novel Software Technology, China.
** Corresponding author.
E-mail addresses: huidou@smail.nju.edu.cn (H. Dou), frshen@nju.edu.cn (F. Shen), jianzhao@nju.edu.cn (J. Zhao).

https://doi.org/10.1016/j.neunet.2023.09.030
Received 16 April 2023; Received in revised form 9 August 2023; Accepted 17 September 2023
Available online 22 September 2023

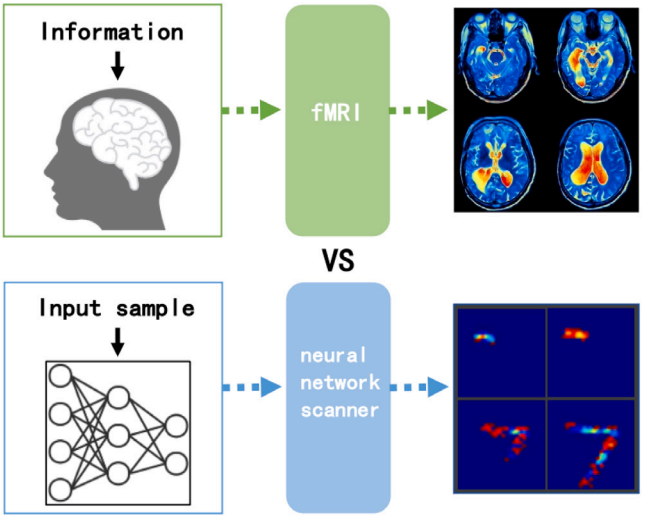

Fig. 1. As the fMRI scans the human brain, the proposed NNS is a method to scan the ANN model. After the human brain receives information, fMRI captures the changes in hemodynamics caused by neuron activities. Given an ANN model and an input sample, NNS gets the features learned by each neuron of the model.

ANN model and an input sample, NNS gets the features learned by each neuron of the neural network. We retain useful information for the neural network interpretation from the learned features of individual neurons. The working principle of the fMRI and the NNS are illustrated in Fig. 1.

The contributions of the proposed method can be summarized as follows:

- NNS visualizes the learning process of neurons. It is used to show features learned by each neuron in a human-understandable form. During the operation, NNS keeps the object localization ability.
- NNS is neuron-based. The learned features of individual neurons are combined flexibly to analyze the working mechanism of different components of the neural network. As the results of different components are visualized in a unified way, NNS is applicable to the neural network composed of artificial neurons without requiring architecture changes. Experiments on different neural network architectures (FCNs and CNNs) are performed to show the results of the proposed method. We discuss the differences of the operating mechanisms of different architecture models.
- Several experiments are conducted to evaluate the interpretability of CNNs. We demonstrate the effectiveness of the proposed method by evaluating the features obtained in a layer. Then, we evaluate the interpretability of the highly activated neurons. Finally, we compare the visualization results of NNS with existing approaches.

The rest of the paper is organized as follows. Section 2 gives a brief summary of the related work, where we review the neural network interpretation methods from different perspectives. In Section 3, we introduces the principle of NNS. We propose the concept of the "learning image" and the process of using the learning image to explain the neuron information. To demonstrate the applicability of our method, we apply our method to a collection of models with different neural network architectures in Section 4. Experiments are provided for scanning an FCN, a simple CNN and a ResNet in order to analyze their operating mechanism. Then, NNS is applied to different CNN models for evaluating interpretability from different perspectives. Finally, we summarize this paper and discuss the future work in Section 5.

2. Related work

In recent years, there have been many attempts to visualize, explain, and understand the inner working principles of neural networks (Piano, 2020). The study of the interpretability of neural networks has received considerable attention (Thiebes, Lins, & Sunyaev, 2021). One intuitive approach is to visualize the value of the neural network directly. Various methods have been proposed to display the activation values of neurons in neural networks (Zhang, Tiňo, et al., 2021). Yosinski, Clune, Nguyen, Fuchs, and Lipson (2015) introduce a tool that visualizes the activation produced on each feature map in CNN. The activation values changes dynamically according to the input samples. Wang et al. (2020) presents a CNN visualization tool, called the CNN explainer. The CNN explainer integrates a model overview that summarizes a CNN's architecture, and displays the operation of each component of CNN dynamically. This tool helps users understand the role of each layer and the interplay between two adjacent layers. There is a simulation tool named 3D Multilayer Neural Network Simulation.[1] It produces a visualization result of an FCN represented in 3D. These tools are simple and easy to understand. However, we cannot obtain the information that each neuron has learned by using these tools.

To explain the neural network in more depth, some researches shows the representations of models through visualization (Samek, Montavon, Lapuschkin, Anders, & Müller, 2021). Currently, there are two major groups of existing methods that explain neural networks with visualization. One method is the activation maximization (Simonyan, Vedaldi, & Zisserman, 2013), which tries to find an input pattern that maximally activates the inspected neuron. This method generates a sample by maximizing the activation of a certain output neuron. The sample presents what a certain neuron is interested in. By using backpropagation, this sample is iteratively optimized. Finally, the sample with the maximum activation value of a classified neuron is obtained. Nguyen, Dosovitskiy, Yosinski, Brox, and Clune (2016) add an image generator network to synthesize the image that is close to the original image. In order to avoid the high-frequency noise, Wang, Liu, and Cheng (2018) apply image blurring and image deblurring to generate images. Another method to explain neural networks is the saliency map (Simonyan et al., 2013). For a given input sample, the saliency map attempts to assign an importance score to each pixel. The class activation mapping method generates a map to show the region of interest in CNN. Zhou et al. (2016) propose Class Activation Mapping (CAM) to use global average pooling (GAP) instead of fully-connected layers in CNN. Through this architecture, the weights of the output layer are projected back to the convolution feature maps, thus recognizing the important areas of the image. Based on the basic architecture of Zhou et al. (2016), Gradient-weighted Class Activation Mapping (Grad-CAM) is proposed in Selvaraju et al. (2017) to highlighting important regions in the image for predicting the concept. Muhammad and Yeasin (2020) use the principal components of the learned representations from the convolutional layers to create the visual explanations. Zeiler and Fergus (2014) explain CNN by features learned through the deconvolution model. It can be seen that the neural network mainly learns simple edge features in the low-level layers and learns some or all objects in the high-level layers. Ren, Zhang, Wang, Hu, and Zuo (2020) produce images of satisfactory visual quality, where non-blind deconvolution is not a necessary choice. Zhang, Wang, et al. (2021) propose to explain CNN by extracting a graphical model. These methods automatically extract different parts of the image to describe the content of interest to the filter.

The activation maximization methods focus on the visual patterns learned by neurons. These methods are not suitable for explaining the

[1] The 3D Multilayer Neural Network Simulation is in https://tutorials.retopall.com/index.php/2019/02/16/3d-multilayer-neural-network-simulation

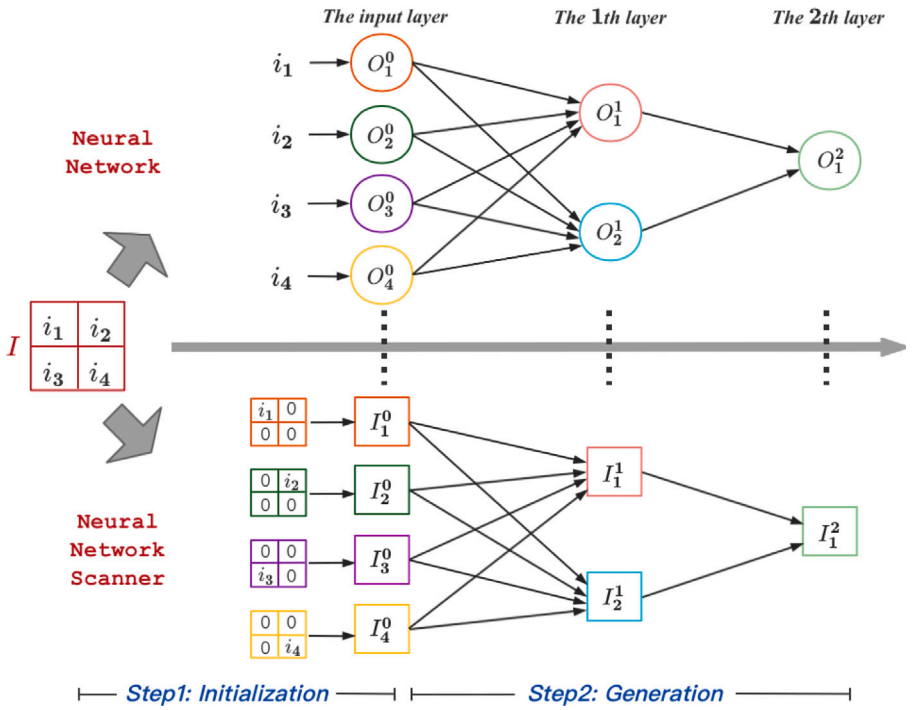

Fig. 2. The working procedure of NNS. NNS visualizes the neural network at the neuron level. Here we set the dimension of the input sample to be 4. The neurons and the corresponding learning images are connected by a dotted line and in the same color. (1) Initialization: we construct the learning image based on the input sample for each neuron in the input layer. In the neural network, elements of I are values of the input neurons. In NNS, we generate several images based on I. Each image keeps a part of the value information and the location information of I. These images are distributed to the input neurons as their learning images. (2) Generation: we generate learning images for neurons in hidden layers and the output layer. We calculate the contributions of the input neurons to the specified neuron. Then the learning image of this neuron is generated based on the learning images of the input neurons.

result of a single input sample. The saliency map methods show which areas are visually important to an input sample. While methods of this kind are closely related to the fixed neural network models. It is difficult to compare operating mechanisms of different neural network architectures and analyze the functions of different components of the model. The method proposed in this paper treats each neuron as a unit to explain the information that the model has learned. Unlike the saliency map, which shows the area where the model is interested in, the proposed method shows the features that the model has learned from the input sample. The proposed method can be used to compare different models under a unified framework.

3. The proposed method

We propose NNS which scans the neural network and visualizes each neuron in the model. For an input sample, the location information of objects in the image is as important as the value information. In the fully-connected layer, only the value information is used, while the location information is discarded. This makes the fully-connected layer lack the ability to locate objects. The location information is preserved in NNS. We generate an input-resolution image for each neuron which is called the "*learning image*" of this neuron. The learning image of a neuron is a representation of the features it has learned from the input sample. The generation procedure of learning images for different neurons is illustrated in Fig. 2.

The workflow of NNS is listed as follows:

- Learning image initialization. In this step, we construct the learning image based on the input sample for each neuron in the input layer. We split the input image into several images. Each splitted image keeps a part of the features of the input sample. The features include the value information and the location information. Then the splitted images are assigned to the input neurons in turn.

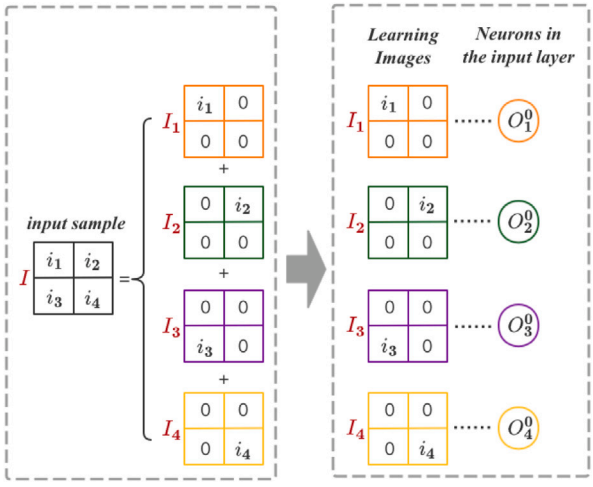

Fig. 3. Learning image initialization. We take a 4-dimensional input sample as an example. First, we split the input sample into 4 images with the same dimension. Each splitted image contains a part of the features of the original input sample. Second, we distribute splitted images to neurons in the input layer as their learning images.

- Learning image generation. In this step, we generate the learning images for all neurons in the hidden layers and the output layer. By obtaining the contributions of the input neurons to the specified neuron, the learning image of this neuron is generated based on the learning images of the input neurons. Throughout the whole process, NNS always keeps the ability to locate the object.

Next, we discuss the detailed steps of NNS.

3.1. Learning image initialization

In NNS, the value and the location information of each pixel in the input image is used in the input learning image. As shown in Fig. 3, We construct the learning images for neurons in the input layer.

3.1.1. Split the sample

First, we regard the input sample as the vector $I = [i_1\ i_2\ \dots\ i_N]^T$. The dimension of the input sample I is N, where we set N is 4 in Fig. 3. We split the input sample I into N images $I_1, I_2, \dots, I_p, \dots, I_N$:

$$I = \sum_{p\in[1,N]} I_p. \tag{1}$$

The pth splitted image I_p is an image whose pixel values are all 0 except for the pth pixel. The pth pixel value of I_p is equal to the pth pixel value of the input sample I. The pth splitted image is represented as follows:

$$I_p(x) = \begin{cases} i_p, & \text{if } x == p \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where x means the pixel number of the image I_p and $x \in [1, N]$. Each splitted image contains a part of the features of the original input sample, and preserves the location information corresponding to the features.

3.1.2. Distribute learning images

In NNS, we set a learning image for each neuron in the model. Given the input image I, we distribute the splitted images to neurons in the input layer. For the pth input neuron O_p^0, its value is x_p^0 and its learning image is I_p^0. We assign the splitted images to input neurons in turn. So the pth neuron O_p^0 is associated with the pth splitted image I_p:

$$I_p^0 = I_p, \qquad p \in [1, N]. \tag{3}$$

If the input sample is an RGB image, this sample is regarded as an independent image in each channel. The image is split in each channel. We get $3\times N$ splitted images. They are assigned to input neurons in turn as the learning images.

With the above analysis, we present the step of the learning image initialization in Algorithm 1.

Algorithm 1 Learning Image Initialization

Initialization: $I = [i_1\ i_2\ \dots\ i_N]^T$ is the input sample, $I_p^0 \in \mathbb{R}^N$ is the learning image of the pth neuron O_p^0 in the input layer.
set $p = 1$.
repeat
  set $I_p \in \mathbb{R}^N$, $I_p = [0\ 0\ \dots\ 0]^T$
  $I_p(p) = i_p$
  $I_p^0 = I_p$
  $p = p + 1$
until $p = N$

3.2. Learning image generation

In this step we generate the learning images for neurons in hidden layers and in the output layer. We calculate the contributions of the input neurons to the specified neuron. Then the learning image of this neuron is generated based on the learning images of the input neurons. The procedure for generating the learning images is illustrated in Fig. 4.

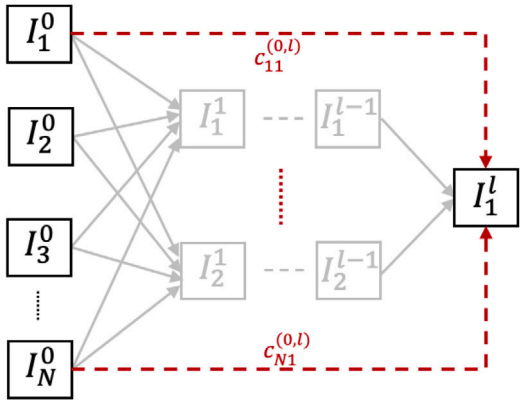

Fig. 4. Learning image generation. We generate the learning images for neurons in hidden layers and the output layer. We calculate contributions of the input learning images to the specified learning image. Then the learning image is generated based on the input learning images.

3.2.1. Learning image generation in the special case

The weight value is regarded as the contribution value to calculate the learning image in the second layer. We take the qth neuron in the second layer as an example. We set $c_{pq}^{(0,1)}$ as the contribution value of the pth learning image in layer 0 (the input layer) to the qth learning image in layer 1 (the second layer). The contribution value $c_{pq}^{(0,1)}$ is equal to the weight value between the pth neuron and the qth neuron. The learning image become $I' = [i_1'\ i_2'\ \dots\ i_N']^T$ through the linear transformation:

$$I' = \sum_{p\in[1,N]} c_{pq}^{(0,1)} \times I_p^0. \tag{4}$$

The tth pixel value of I' is calculated as:

$$i_t' = \sum_{p\in[1,N]} c_{pq}^{(0,1)} \times i_{pt}^0, \tag{5}$$

where i_{pt}^0 is the tth pixel value of I_p^0. After the linear transformation, the features learned by neurons are saved with the location information in NNS.

Then, a non-linear transformation is applied to neurons through a non-linear activation function f. In this paper, we set ReLU as the activation function. The qth learning image in the second layer is I_q^1, which is set as follows:

$$I_q^1 = \begin{cases} I', & \text{if } x' > 0 \\ \mathbf{0}, & \text{otherwise} \end{cases}, \tag{6}$$

where x' is the value of the neuron. If x' is greater than 0, the neuron is activated. So its learning image is activated too. If the value of the neuron is 0 after f, the neuron is not activated. The contribution of this neuron to neurons in the next layer is zero. So its learning image is $\mathbf{0}$ and does not contribute to the learning images in the next layer.

The algorithm of the learning image generation in the special case is shown in Algorithm 2.

3.2.2. Learning image generation in the general case

Up to now, we have introduced the generation of the learning image in the second layer. For the learning image in other layers, we obtain the contributions of the input learning images to this learning image. In FCN, the weight matrix between the layer $l-1$ and the layer l is $W^l \in \mathbb{R}^{n\times m}$:

$$W^l = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots & w_{1m}^l \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1}^l & w_{n2}^l & \cdots & w_{nm}^l \end{bmatrix}, \tag{7}$$

where n is the neuron number of layer l and m is the neuron number of layer $l-1$. The contribution matrix of the learning images in layer

Algorithm 2 Learning Image Generation in the Special Case

Initialization: I_p^0 is the learning image of neuron p in layer 0, x_q^1 is the value of neuron q in layer 1, $c_{pq}^{(0,1)}$ is the contribution value of the pth learning image to the qth learning image, w_{pq} is the weight between p and q, the input neuron number is m, the output neuron number is n.
set $p = 1$, $q = 1$, $c_{pq}^{(0,1)} = w_{pq}$.
repeat
  $I' \in \mathbb{R}^N$
  repeat
    $I' = I' + c_{pq}^{(0,1)} \times I_p^0$
    $p = p + 1$
  until $p = m$
  if $x_q^1 > 0$ **then**
    $I_q^1 = I'$
  else
    $I_q^1 = 0$
  end if
until $q = n$

$l-1$ to the learning images in layer l is $C^{(l-1,l)} \in \mathbb{R}^{n\times m}$:

$$C^{(l-1,l)} = \begin{bmatrix} c_{11}^{(l-1,l)} & c_{12}^{(l-1,l)} & \cdots & c_{1m}^{(l-1,l)} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1}^{(l-1,l)} & c_{n2}^{(l-1,l)} & \cdots & c_{nm}^{(l-1,l)} \end{bmatrix}. \tag{8}$$

In order to achieve non-linear transformation, we use a piecewise linear function S as follows:

$$S(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

The contribution value $c_{pq}^{(l-1,l)}$ that the pth learning image in layer $l-1$ to the qth learning image in layer l is calculated as follows:

$$c_{pq}^{(l-1,l)} = S(x_q^l) w_{pq}^l, \tag{10}$$

where x_q^l is the value of the qth neuron in layer l. The contribution matrix of the input learning images to the qth learning image I_q^l in layer l is represented as $C_q^{(0,l)}$:

$$C_q^{(0,l)} = W_q^l \cdot \prod_{i=1}^{l-1} C^{(i-1,i)}, \tag{11}$$

where $W_q^l \in \mathbb{R}^{1\times m}$ is the weight matrix between layer $l-1$ and the qth neuron in layer l.

All the input learning images are represented as a matrix $I_{sum}^0 \in \mathbb{R}^{N\times N}$:

$$\begin{aligned} I_{sum}^0 &= (I_1^0, I_2^0, \ldots, I_N^0) \\ &= \begin{bmatrix} i_{11}^0 & i_{21}^0 & \cdots & i_{N1}^0 \\ \vdots & \vdots & \ddots & \vdots \\ i_{1N}^0 & i_{2N}^0 & \cdots & i_{NN}^0 \end{bmatrix}, \end{aligned} \tag{12}$$

where N is the input learning image number. The qth learning image in layer l is I_q^l:

$$I_q^l = I_{sum}^0 \cdot C_q^{(0,l)}. \tag{13}$$

Carrying out this process, we obtain the learning images for all neurons.

3.3. Additional discussion on CNN

In the fully-connected layer, the output neurons apply a linear transformation to the input neurons through a weights matrix. With an artificial neuron as the fundamental unit, various types of neural network layers can be transformed into fully-connected layers. We take CNN as an example. The equivalence relationship between the convolutional layer and the fully-connected layer is shown in Fig. 5. The outputs of a convolutional layer are feature maps. Neurons in the same feature map correspond to the same convolution filter, i.e., these neurons share the same weights. The numerical matrix of a filter can be converted to a weight matrix between two layers. The convolution operation is equivalent to the linear operation. In the pooling layer, neurons in the feature maps are screened, which can be regarded as a special convolution process. Therefore, a pooling layer is also equivalent to a fully-connected layer. We regard each unit (pixel) in the convolutional and pooling layers as a neuron and build the learning image for each neuron. The learning image of the neuron changes as the value of the neuron changes. Neurons in the pooling layer are selected from the previous layer. During this process, the values of the neurons do not change. The learning images of these selected neurons are collected from the previous layer and stay the same.

The size of the input image is $M \times M \times D$. First, we split the input image into $M \times M$ images. For one unit (x, y), its splitted image is $I_{(x,y)} \in \mathbb{R}^{M\times M\times D}$. In each channel, the splitted image is generated as discussed in Section 3.1.1. Then the splitted images is distributed to each neuron in the input layer as discussed in Section 3.1.2:

$$I_{(x,y)}^0 = I_{(x,y)}, \qquad x, y \in [1, M]. \tag{14}$$

As shown in Fig. 5, we convert the convolution and pooling operations into linear operations, and then obtain the learning images in the convolutional and pooling layers in the same way as calculating the learning images in the fully-connected layer. In the whole process, the size of the learning images is $M \times M \times d$, where d is the channel number of the corresponding neuron. In this way, the learning images for neurons in CNN can be obtained.

Table 1
The detailed description of neural networks.

Architecture	Dataset	Learning algorithm
FCN CNN	MNIST	BP
FCN with skip connections ResNet10	MNIST	BP
AlexNet VGGNet	ImageNet CIFAR-10, CIFAR-100	BP

4. Experiments

In CNN, we regard each unit of the feature maps as a neuron. In order to clearly distinguish different pixel intensities in the images, we use the form of heat map to represent the image. The heatmap uses different colors to represent different pixel intensities. The colormap of the heatmap is shown in Fig. 6. The larger the pixel value is in the image, the closer the color of the pixel is to "red". The smaller the pixel value is, the closer the color of the pixel is to "blue". To demonstrate the applicability of our method, we apply our method to a collection of models with different neural network architectures. The detailed description of the neural network are shown in Table 1. In Section 4.1, the neural networks with and without skip connections are trained on MNIST. In Section 4.2, the models are trained on ImageNet, CIFAR-10 and CIFAR-100. All models are trained by backpropagation (BP) learning algorithm.

4.1. Scanning different neural network architectures

To analyze the differences of the operating mechanisms of different architecture models, we perform experiments on an FCN model and a simple CNN model. The FCN model used here has an input layer (784 neurons), two hidden layers (500 neurons) and an output layer (10 neurons). The simple CNN consists of three convolutional layers (10 filters), two pooling layers, a flatten layer, a fully-connected layer (64

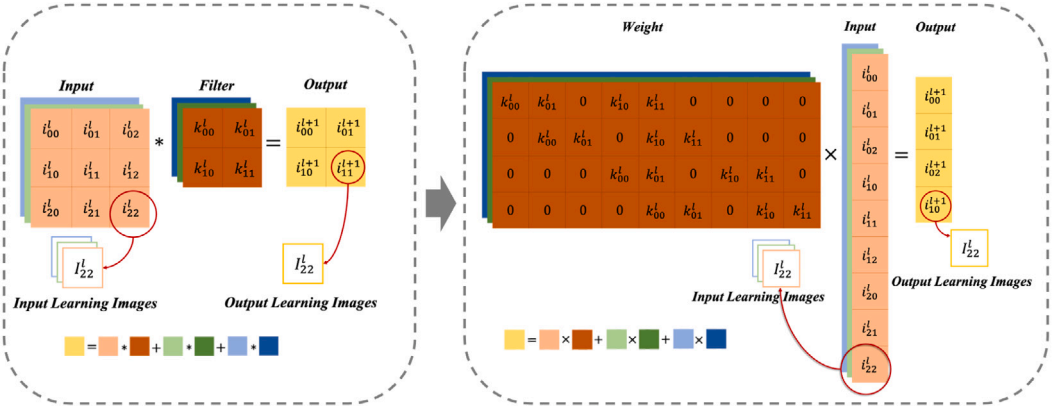

Fig. 5. The equivalence relationship between convolution and linear operations. Neurons in the same feature map correspond to the same convolution filter, i.e., these neurons share the same weights. The numerical matrix of a filter can be converted to a weight matrix between two layers. The convolution operation is equivalent to the linear operation. Then, we obtain the learning images in the convolutional layer in the way of calculating the learning images in the fully-connected layer.

Fig. 6. The colormap of the heatmap.

neurons) and an output layer (10 neurons). We choose a simple dataset so that different models are trained well. Both models are trained on the MNIST handwritten digits training set.

In the following experiments, first we verify the effectiveness of the proposed method on different neural networks and analyze their operating mechanisms. Second, we clarify the neuron learning rules for different models. Then, we explore the nature of the features learned by the winning neuron. Finally, we compare the learning process between different architectures with skip connections.

4.1.1. Operating mechanism comparison

In this part, we compare operating mechanisms of different models. Visualizing all neurons of a trained neural network can gain an insight into the operation of the model. Given the same input sample, we select one activated neuron in each layer to show its learning image. The scanning results from each layer show the layered characteristics of neural network learning. In FCN, we observe from Fig. 7(a) that the network always learns global features in each layer. For each neuron, we see the edge outlines of the input sample. In layer 1, the features learned by FCN are represented by only two colors in the learning images. In layer 3, the learned features are represented by multiple colors. The low-level layers learn monotonous features with small color changes in the learning images. The high-level layers learn abundant features with big color changes in the learning images. In CNN, we observe that the network always learns abundant features with big color changes in each layer. As shown in Fig. 7(b), in low-level layers (the first and second convolutional layers), only a few neurons are activated. The low-level layers respond to features in the small part of the image. The convolutional layer 3 learns features in a larger part. The neurons in the pooling layer are selected from the neurons in the previous convolutional layer. The values of the selected neurons do not change, so the learning images of the selected neurons remain the same. The fully-connected layer in CNN learns global features as well.

We analyze the learning images by measuring the knowledge quantity learned by neurons in each layer. The knowledge quantity is evaluated from two aspects: the activation quantity and the activation intensity. For one learning image, the activation quantity is the number of activation pixels, the activation intensity is the sum of the pixel intensity. For comparison purposes, the activation quantity of one layer is set as the percentage of the average activation quantity of all neurons to the activation quantity of the input image. Similarly, the activation intensity of one layer is set as the percentage of the average activation intensity of all neurons to the activation intensity of the input image. The larger the activation quantity and the activation intensity, the more knowledge the neural network learns from the input samples.

Fig. 8 shows the knowledge quantity of FCN and CNN. As shown in Fig. 8(a), the activation quantity of each layer always keeps at a very high value in FCN. The activation intensity increases significantly with the increase of the number of layers. As shown in Fig. 8(b), the activation quantity of the convolutional layer increases with the increase of the number of layers. Comparing with activation intensities of other convolutional layers, the activation intensity of the first convolutional layer has the largest value. Activation intensities of other convolutional layers do not change significantly. In CNN, the changing trend of the knowledge quantity at the fully-connected layer is similar to that of the fully-connected layer in FCN.

Through experiments, we analyze the working mechanism of FCN and CNN. In FCN, we observe that changes of the knowledge quantity are mainly realized by changes of the activation intensity. Although global features have been learned in the first layer, they are only preliminary features. With the number of layers increases, the activation intensity of the learning image becomes large. The complexity of the features learned by FCN at the same position of the image increases with the increase of the number of layers. In CNN, we observe that changes of the knowledge quantity are mainly achieved through changes of the activation quantity. Neurons in the low-level layers learn simple local features and neurons in the high-level layers learn complex global features.

4.1.2. Neuron learning rules

In order to clarify the features that neurons pay attention to, we obtain the similarity between the learning images and the samples in the dataset. We compare the similarity of the learning image to the corresponding input sample with the similarity of the learning image to other irrelevant samples. We measure the similarity between two images by calculating the distance between them. The distance between the input sample I and the learning image I_o of the neuron o is calculated as follows:

$$dist_o^I = \left\| I_o - I \right\|. \tag{15}$$

The larger the distance, the lower the similarity between images, and vice versa. As shown in Fig. 9, we calculate the distance between the learning image and its corresponding input sample. For comparison and analysis, we calculate the distance between the same learning image and other irrelevant samples.

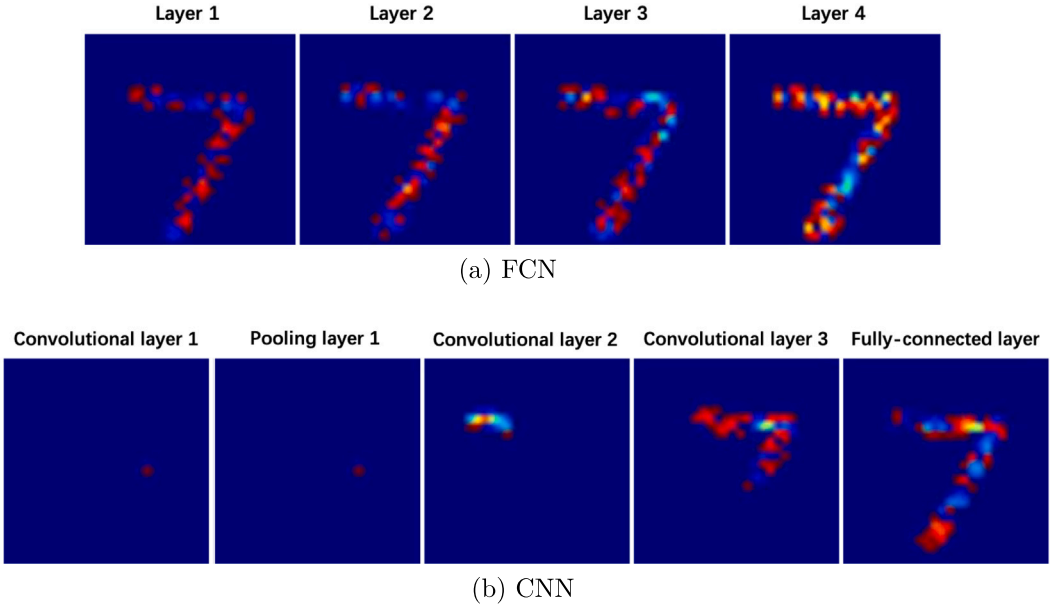

Fig. 7. The visualization of the learning process of neurons. (a) NNS for FCN: the neural network always learns global features in each layer. The low-level layers learn monotonous features with small color changes in the learning images. The high-level layers learn abundant features with big color changes in the learning images. (b) NNS for CNN: the convolutional layer 1 and layer 2 respond to simple features such as corners and edge conjunctions. The convolutional layer 3 learns more complex features. As neurons in the pooling layer are selected from neurons in the previous convolutional layer, the learning images of neurons do not change.

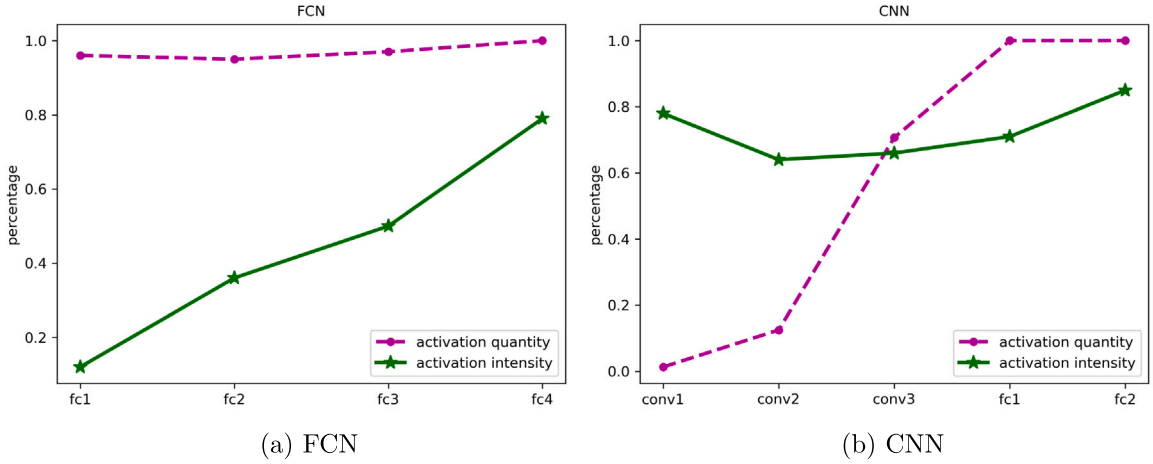

Fig. 8. Comparisons of the knowledge quantity of FCN and CNN in each layer. In FCN, changes of the knowledge quantity is mainly realized by changes of the activation intensity. In CNN, changes of the knowledge quantity are mainly achieved through changes of the activation quantity.

In FCN, as shown in Fig. 9(a), regardless of the layer, the distance between the learning image and other irrelevant samples is far greater than the distance between the learning image and its input sample. Neurons in the fully-connected layer have no fixed learning features. The features learned by neurons always have a very large relationship with the input sample.

In CNN, as the pooling operation does not change the learning images of neurons, we only show the learning images of neurons in the convolutional layers and the fully-connected layers. As shown in Fig. 9(b), in the convolutional layer 1 and the convolutional layer 2, the distance between the learning image and other irrelevant samples is similar to the distance between the learning image and the input image. Neurons in the low-level convolutional layer tend to learn fixed features. In the convolutional layer 3, the distance between the learning image and the input image is smaller than the distance between the learning image and other irrelevant samples. Compared with the learning image of the low-level convolutional layer, the learning image of the high-level convolutional layer is more similar to the input image. The changing trend of the learning images of neurons in the fully-connected layer is similar to that of the fully-connected layer in FCN.

Through experiments, we analyze the learning rules of different neural network architectures. The features learned by neurons in the fully-connected layer always maintain a high similarity with the features in the input samples in each layer. The features learned by neurons in the fully-connected layer are not fixed and change with the input sample. Neurons in the fully-connected layer learn global features and obtain global contour information, which is the obvious difference between the specified input sample and other irrelevant samples. Neurons in the convolutional layer tend to learn fixed features while this property decreases as the number of layers increases. Neurons in the low convolutional layer are interested in the specified

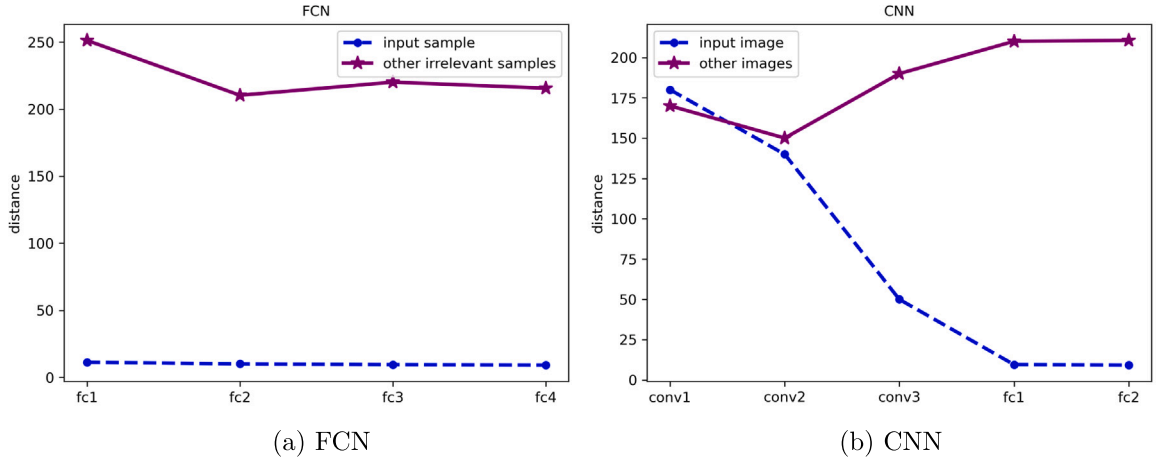

Fig. 9. The similarity between the learning images and the samples in the dataset. We compare the similarity of the learning image to the corresponding input sample with the similarity of the learning image to other irrelevant samples. The larger the distance, the lower the similarity between images, and vice versa. The features learned by neurons in the fully-connected layer always maintain a high similarity with the features in the input samples in each layer. Neurons in the convolutional layer tend to learn fixed features while this property decreases as the number of layers increases.

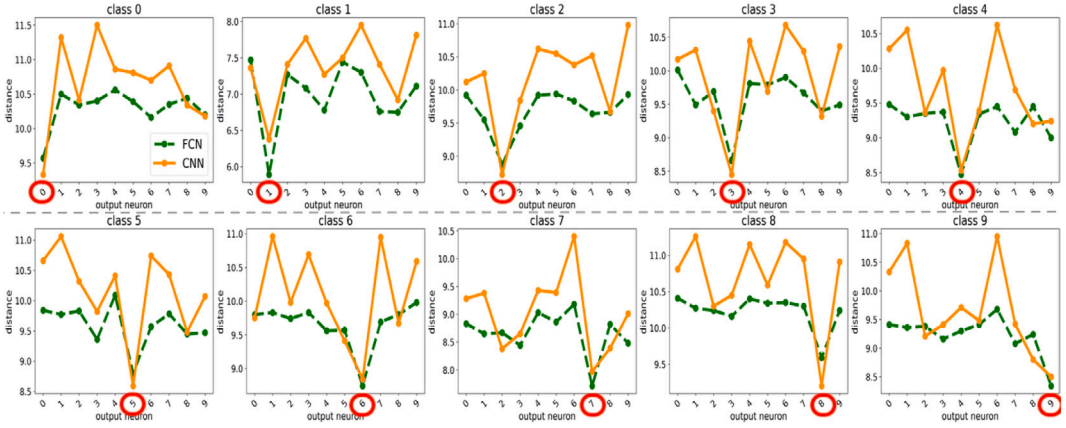

Fig. 10. The distances between the input samples and the learning images of the output neurons. The cth subgraph presents the distances between the input samples belonging to class c and the different output neurons. In each subgraph, compared with the learning images of other output neurons, the learning image of the winning neuron is closer to the input sample in both FCN and CNN. The features learned by the winning neuron are the most similar to the input sample. In other words, compared with other neurons in the output layer, the winning neuron recovers the input sample better.

local features in a small part of the image. These features bear little relationship with the global contour of the input sample. In the high-level convolutional layer, neurons learn more complex features in a larger part. The similarity between the learned features and the input sample becomes larger.

4.1.3. Winning neuron analysis

In this experiment, we explore how discriminative the features in the output layer of the neural networks are. For the same input sample, we compare the abilities to restore the input sample of the different neurons in the output layers. The winning neuron is the output neuron corresponding to the correct class. The distance between the input sample I and the learning image I^o of the specified output neuron o is calculated as that in Eq. (15). All the input samples belonging to class c are I_c. The distance between I^c and a specified neuron o is represented as the average of the distance between each sample in class c and the neuron o:

$$DIST_o^c = \frac{\sum_{I \in I^c} dist_o^I}{N}, \tag{16}$$

where N is the input sample number of class c.

The distances between the input samples and the learning images of the output neurons are shown in Fig. 10. In the cth subgraph, the distances between the input samples belonging to class c and the different output neurons are shown. The distance is minimum when $o = c$. Compared with the learning images of other output neurons, the learning image of the winning neuron is closer to the input sample for both FCN and CNN in each subgraph. The features learned by the winning neuron are the most similar to the input sample. In other words, compared with other neurons in the output layer, the winning neuron recovers the input sample better.

In order to analyze the property of the winning neuron, we measure winning neurons on three neural networks with different accuracies for both FCN and CNN in Table 2. First we analyze the relationship between distance and accuracy. The distance between the learning images of winning neurons and input samples is shown. For both FCN and CNN, the higher the accuracy of the model, the smaller the distance. In other words, the features learned by the neural network with higher accuracy are more similar to input sample. Then we measure the activation intensity and activation quantity of winning neurons' learning images. As Table 2 shows, for winning neurons of both FCN and CNN, the activation intensity increases with the increase of model accuracy. The activation quantity is always 1 in different neural network for FCN. The activation quantity increases with the increase of accuracy and always keeps a high value for CNN. For both FCN and CNN, winning neurons of neural networks with different

Table 2
The results of winning neurons.

	FCN			CNN		
Accuracy	40	70	90	40	70	90
Distance	9.10	8.84	8.47	8.91	8.52	8.35
Activation intensity	0.589	0.648	0.665	0.427	0.453	0.678
Activation quantity	1.0	1.0	1.0	0.995	0.996	0.998

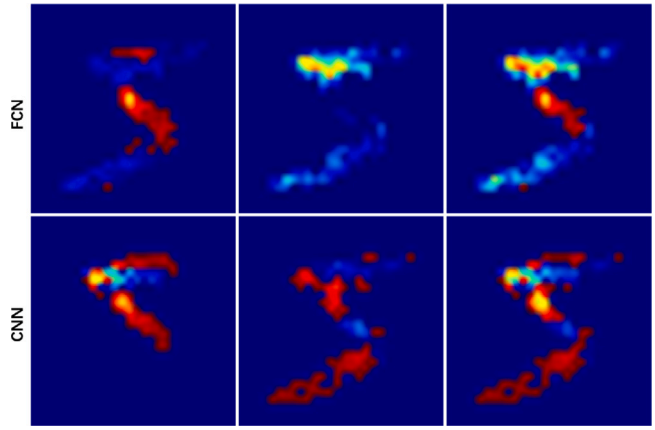

Fig. 11. Comparisons of a fully-connected architecture with skip connections (top) and a convolutional architecture with skip connections (bottom). The learning images of neurons before the residual block (left), in the residual block (middle), after the residual block (right).

accuracies all obtain global feature information. The winning neuron of the neural network with higher accuracy learns features with higher activation intensity.

4.1.4. Architectures with skip connections

In this part, we compare the learning process between different architectures with skip connections. We design experiments on a fully-connected architecture with one residual block and a ResNet10 model. We compare the learning images of neurons before the residual block, in the residual block and after the residual block. Changes in the learning images of neurons at the same location are shown in Fig. 11. Neurons in the residual block learn features that are different from the features learned by neurons before the residual block. Fig. 12 shows the knowledge quantity of different architectures with skip connections. For FCN, the activation quality and the activation intensity of neurons in the residual block are lower than those before the residual block. After adding the residual block, both the activation quality and the activation intensity of neurons increase significantly. For CNN, the activation quality always maintains a growing trend. The activation intensity in the residual block decreases compared to those before the residual block. After adding the residual block, the activation quality increases significantly and the activation intensity has a slight increase. For both FCN and CNN, residual blocks increase the knowledge quantity of the neural networks.

4.2. Evaluating interpretability of CNNs

In the following experiments, we evaluate the interpretability of CNNs. We use AlexNet and VGGNet as representative models to analyze CNNs. These models are trained on ImageNet. We begin by validating the effectiveness of NNS by evaluating interpretability of the model in a layer. Then, we analyze the learning ability of highly activated neurons in the same layer. Finally, we compare the visualization results of NNS with existing approaches.

4.2.1. Evaluating interpretability by layer

We conduct experiments to analyze whether it is meaningful to generate a learning image for individual neurons. The feature map has the ability to localize objects. So we regard the feature map as a baseline. We evaluate the interpretability by measuring the correlation between the learning image and the feature map in a single layer of the CNN model. The learning image of a layer is the sum of the learning images of all neurons in that layer. We find that the learning image of one layer and the corresponding feature map are highly correlated. In other words, the features learned by one layer are the sum of the features learned by all neurons that make up the layer.

The models we used are AlexNet and VGGNet. For the input image, $F^{k,l}$ is the feature map of the filter k in layer l. $f^{k,l}_{(x,y)}$ represents the activation value of $F^{k,l}$ at spatial location (x, y). $I^{k,l}_{(x,y)}$ is the learning image of the neuron at (x, y) of $F^{k,l}$. We use $I^{k,l} = \sum_{x,y} I^{k,l}_{(x,y)} \times f^{k,l}_{(x,y)}$ to represent the learning image of the filter k in layer l. To compare a low-resolution feature map to the input-resolution learning image, the feature map is scaled up to the mask resolution $S^{k,l}$ from $F^{k,l}$ using bilinear interpolation. The correlation between the feature map and the learning image is reported as:

$$C^{k,l} = \frac{S^{k,l} \cap I^{k,l}}{I^{k,l}}. \tag{17}$$

Eq. (17) is a variant of intersection over union. As there are more detail features in the learning image than in the feature map, we replace $S^{k,l} \cup I^{k,l}$ with $I^{k,l}$ to make the equation more sensitive to changes. The correlation value between the feature map and the learning image in one layer is:

$$C^{l} = \frac{\sum_{k \in K} C^{k,l}}{K}, \tag{18}$$

where K is the number of filters in the layer. In layer l, the larger the C^l, the higher the correlation between the feature map and the learning image, and vice versa.

According to the models, different layers are selected to measure the correlations between the feature maps and the learning images. For AlexNet, we choose pooling layer 1, pooling layer 2, convolutional layer 3, convolutional layer 4 and convolutional layer 5. For VGGNet, we choose pooling layer 1, pooling layer 2, pooling layer 3, pooling layer 4 and pooling layer 5.

The correlation performance is shown in Table 3. For convenience, we will describe these layers as layer 1, layer 2, layer 3, layer 4 and layer 5, respectively. For both AlexNet and VGGNet, the feature maps and the learning images always keep a high degree of correlations in different layers. The performance of these two models is comparable. In the low-level layers (layer 1 and layer 2), the consistencies between the feature maps and the learning images are very high. As the feature map is scaled up to the input-resolution using bilinear interpolation. Compared with the learning image, the feature map lacks details. Therefore, in the high-level layers (layer 3, layer 4 and layer 5), the consistency is reduced.

Fig. 13 shows the image comparison results of feature maps and their corresponding learning images in each layer in AlexNet. The learning images are highly correlated with their corresponding feature maps. The information in the learning image is more detailed than that in the corresponding feature map. We conclude that the input-resolution learning image shows the features learned by the model. The features learned by one layer are the sum of the features learned by all neurons that make up this layer.

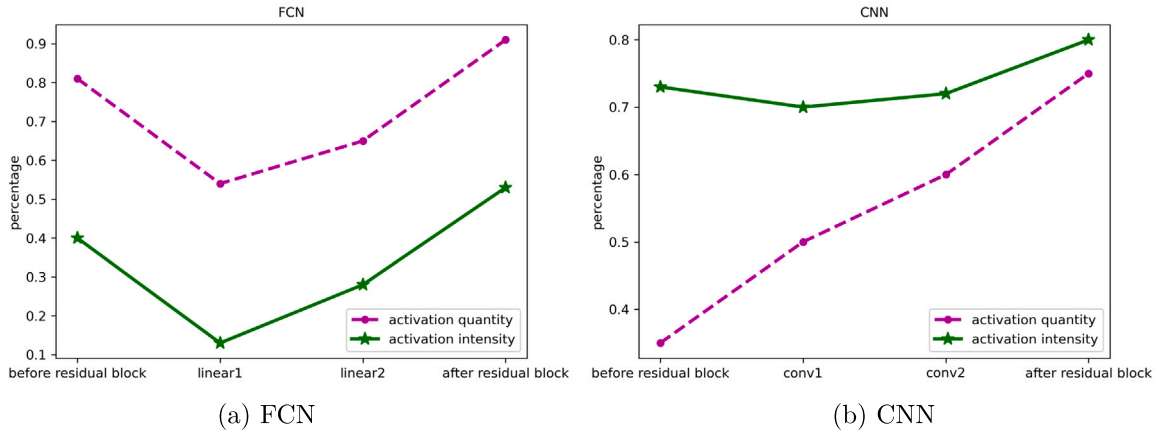

Fig. 12. Comparisons of the knowledge quantity between different architectures with skip connections. For FCN, the activation quality and the activation intensity of neurons in the residual block are lower than those before the residual block. After adding the residual block, both the activation quality and the activation intensity of neurons increase significantly. For CNN, the activation quality always maintains a growing trend. The activation intensity in the residual block decreases compared to those before the residual block. After adding the residual block, the activation quality increases significantly and the activation intensity has a slight increase.

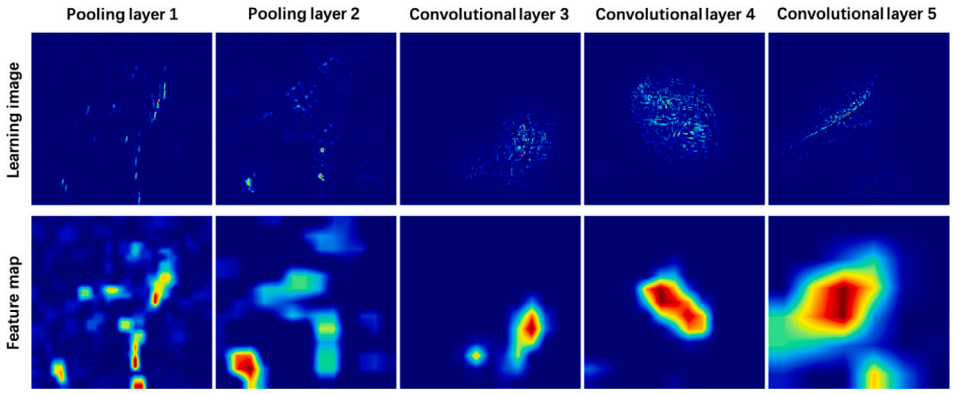

Fig. 13. The learning images (top) and their corresponding feature maps (bottom) in AlexNet. The learning image and the corresponding feature map are highly correlated in each layer. There is more detailed information in the learning image than the corresponding feature map.

Table 3
The correlation between the feature map and the learning image.

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
AlexNet	0.998	0.872	0.641	0.634	0.623
VGGNet	0.952	0.923	0.688	0.620	0.538

Table 4
The learning image correlation of highly activated neurons.

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
AlexNet	0.999	0.953	0.761	0.676	0.615
VGGNet	0.997	0.990	0.873	0.835	0.804

4.2.2. Evaluating interpretability by neurons

By using NNS, we analyze and compare the learning images of neurons in the same convolutional layer. For the input image, we select neurons with top 2 activation values in each convolution result to show their learning images. These neurons are the most representative ones in one layer. We find that the learning images of two representative neurons are highly similar and these neurons tend to learn features at similar locations in the image.

The learning images of two neurons with top 2 activation values of the filter k in layer l are $I_1^{k,l}$ and $I_2^{k,l}$. We use intersection over union to measure the correlation between these two neurons:

$$IoU^{k,l} = \frac{I_1^{k,l} \cap I_2^{k,l}}{I_1^{k,l} \cup I_2^{k,l}}. \tag{19}$$

The correlation value of these neurons in one layer is:

$$IoU^{l} = \frac{\sum_{k \in K} IoU^{k,l}}{K}, \tag{20}$$

where K is the number of filters in the layer.

The correlations of two neurons with top 2 activation values are summarized in Table 4. Same as previous experiments, different layers are selected to measure the correlations in different models. In the low-level layers (layer 1 and layer 2), the correlations are high. Highly activated neurons tend to learn the same features at a similar location. Although the correlation decreases in the high-level layers (layer 3, layer 4 and layer 5), the features learned by the highly activated neurons still have consistency. The highly activated neurons in VGGNet consistently maintain a high degree of correlations across different layers. The neurons in VGGNet show stronger correlations than those in AlexNet. The highly activated neurons in the low-level layers tend to learn similar features. The learning abilities of neurons in the high-level layers vary significantly among different models.

4.2.3. Evaluating visualization result

In this section, we compare NNS with CAM and Grad-CAM. The visual comparison results on CIFAR-10, CIFAR-100 and ImageNet are shown in Fig. 14. For NNS, we show the learning image of the winning neuron in the output layer. The heat map generated by NNS is the same size as the original image. CAM and Grad-CAM generate small-sized heatmaps and then interpolate and enlarge them to the original image size, so their heatmaps are smoother than NNS's heatmaps.

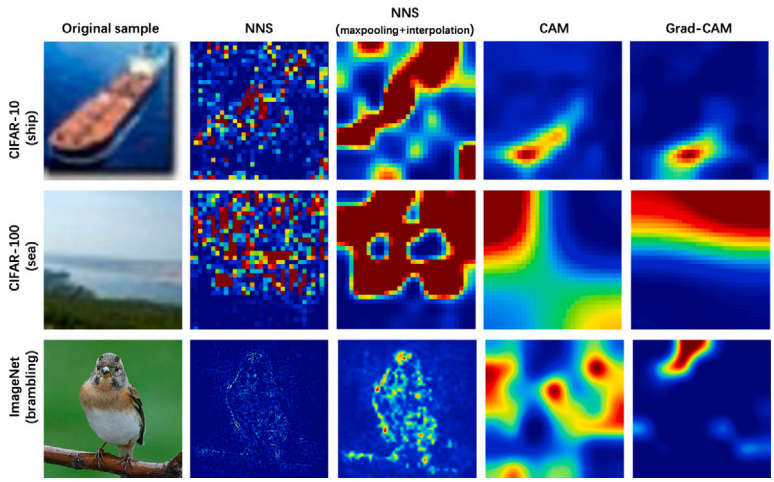

Fig. 14. Comparison of visualization results of different approaches. CAM and Grad-CAM generate small-sized heatmaps and then interpolate and enlarge them to the original image size, so their heatmaps are smoother than NNS's heatmaps. In order to reduce the impact of this phenomenon on the comparison results, we perform max pooling and interpolation on the results of NNS. On CIFAR-10 and CIFAR-100, NNS displays object feature information. On ImageNet, NNS describes the contours and the important details of the input sample.

Compared with the large-sized ImageNet dataset, this phenomenon is more significant on the small-sized CIFAR-10 and CIFAR-100. In order to reduce the impact of this phenomenon on the comparison results, we perform max pooling and interpolation on the results of NNS. NNS shows what the model has learned from the input sample and how the model describes the sample. The other algorithms show the regions of interest for the model. On CIFAR-10 and CIFAR-100, NNS displays object feature information. On ImageNet, we see that NNS describes the contours and the important details of the input sample.

5. Conclusions and future work

In this paper, we propose a method to interpret the working process of neural networks. Given a neural network, the proposed NNS visualizes the features learned by the individual neuron of the model without requiring any changes to the model architecture. The results of different components are visualized in a unified way, enabling us to compare the operating mechanisms of different models. Experiments are carried out by scanning FCN and CNN. By scanning the model, we gain a deep understanding of the neural networks. Firstly, we visualize the individual neuron of the trained neural network to verify the effectiveness of the proposed NNS and analyze the operating mechanism of different models. Secondly, we clarify the neuron learning rules for different models by analyzing the similarity between the learning image and different samples. Then, we explore the nature of the features learned by the winning neuron in the output layer. Lastly, we compare the learning process between different architectures with skip connections. Moreover, we evaluate neural network interpretability from different perspectives of CNN models. We validate the effectiveness of NNS by evaluating interpretability of the model in a layer. Then, we analyze the learning ability of highly activated neurons. Finally, we compare the visualization results of NNS with existing approaches. NNS aids us in exploring the characteristics of various neural networks.

This NNS is suitable for all neural network architectures composed of artificial neurons. In future work, we will apply NNS to more neural network models (e.g., Recurrent Neural Network) to analyze their working mechanisms. Furthermore, based on the results of NNS, we will explore how to make a deeper analysis of the neural network, so as to diagnose and optimize the neural network.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Furao Shen reports financial support was provided by the STI 2030-Major Projects of China under Grant 2021ZD0201300, the National Science Foundation of China under Grant 62276127.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by the STI 2030-Major Projects of China under Grant 2021ZD0201300, and by the National Science Foundation of China under Grant 62276127.

References

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6541–6549).

Bodria, F., Giannotti, F., Guidotti, R., Naretto, F., Pedreschi, D., & Rinzivillo, S. (2021). Benchmarking and survey of explanation methods for black box models. arXiv preprint arXiv:2102.13076.

Ghorbani, A., Abid, A., & Zou, J. (2019). Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence, vol. 33, no. 01* (pp. 3681–3688).

Kim, J., & Canny, J. (2017). Interpretable learning for self-driving cars by visualizing causal attention. In *Proceedings of the IEEE international conference on computer vision* (pp. 2942–2950).

Liang, Y., Li, S., Yan, C., Li, M., & Jiang, C. (2021). Explaining the black-box model: A survey of local interpretation methods for deep neural networks. *Neurocomputing, 419*, 168–182.

Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, *16*(3), 31–57.

McGough, M. (2018). How bad is Sacramento's air, exactly? Google results appear at odds with reality, some say. *Sacramento Bee*, *7*.

Muhammad, M. B., & Yeasin, M. (2020). Eigen-cam: Class activation map using principal components. In *2020 International joint conference on neural networks* (pp. 1–7). IEEE.

Nam, W.-J., Gur, S., Choi, J., Wolf, L., & Lee, S.-W. (2020). Relative attributing propagation: Interpreting the comparative contributions of individual units in deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 03* (pp. 2501–2508).

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., & Clune, J. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *Advances in Neural Information Processing Systems*, *29*, 3387–3395.

Peng, Z., Huang, W., Gu, S., Xie, L., Wang, Y., Jiao, J., et al. (2021). Conformer: Local features coupling global representations for visual recognition. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 367–376).

Piano, S. L. (2020). Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. *Humanities and Social Sciences Communications*, *7*(1), 1–7.

Ren, D., Zhang, K., Wang, Q., Hu, Q., & Zuo, W. (2020). Neural blind deconvolution using deep priors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3341–3350).

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., & Zhong, C. (2021). Interpretable machine learning: Fundamental principles and 10 grand challenges. arXiv preprint arXiv:2103.11251.

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., & Müller, K.-R. (2021). Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, *109*(3), 247–278.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

Thiebes, S., Lins, S., & Sunyaev, A. (2021). Trustworthy artificial intelligence. *Electronic Markets*, *31*(2), 447–464.

Wang, F., Liu, H., & Cheng, J. (2018). Visualizing deep neural network by alternately image blurring and deblurring. *Neural Networks*, *97*, 162–172.

Wang, Z. J., et al. (2020). CNN explainer: Learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, *27*(2), 1396–1406.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579.

You, J., Leskovec, J., He, K., & Xie, S. (2020). Graph structure of neural networks. In *International conference on machine learning* (pp. 10881–10891). PMLR.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). Springer.

Zhang, Y., Tiňo, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Zhang, Q., Wang, X., Cao, R., Wu, Y. N., Shi, F., & Zhu, S.-C. (2021). Extraction of an explanatory graph to interpret a CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(11), 3863–3877.

Zhang, Z., Xie, Y., Xing, F., McGough, M., & Yang, L. (2017). Mdnet: A semantically and visually interpretable medical image diagnosis network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6428–6436).

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2921–2929).

Zhou, B., Sun, Y., Bau, D., & Torralba, A. (2018). Interpretable basis decomposition for visual explanation. In *Proceedings of the European conference on computer vision* (pp. 119–134).

Zhu, Y., Ma, J., Yuan, C., & Zhu, X. (2022). Interpretable learning based dynamic graph convolutional networks for alzheimer's disease analysis. *Information Fusion*, *77*, 53–61.