applied sciences

MDPI

Article

A Neuronal Morphology Classification Approach Based on Locally Cumulative Connected Deep Neural Networks

Xianghong Lin * and Jianyang Zheng

College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China; peterzhengjianyang@gmail.com

* Correspondence: linxh@nwnu.edu.cn

Received: 30 August 2019; Accepted: 12 September 2019; Published: 16 September 2019

Abstract: Neurons are the basic building and computational units of the nervous system, and have complex and diverse spatial geometric structures. By solving the neuronal classification problem, we can further understand the characteristics of neurons and the process of information transmission. This paper presents a neuronal morphology classification approach based on locally cumulative connected deep neural networks, where 43 geometric features were extracted from two different neuron datasets and applied to classify types of neurons. Then, the effects of different parameters of deep learning networks on the performance of neuron classification were analyzed including mini-batch size, number of intermediate layers, and number of building blocks. The accuracy of the approach was also compared with that of the other mainstream machine learning approaches. The experimental results showed that the proposed approach is effective for solving complex neuronal morphology classification problems.

Keywords: neuron classification; geometric features; deep residual neural networks; locally cumulative connection

1. Introduction

The basic functions of neurons are to accept, integrate, transmit, and output information for information exchange. Neuronal clusters exchange information through various neurons to realize brain functions [1,2]. Studying the classification of neurons is an important way to thoroughly understand brain information processing [3,4]. The morphological and electrical characteristics of neurons are two important factors [5]. The electrical characteristic of neurons contains different action potential firing patterns. The potential spike pattern of the neuronal cells can be obtained in response to their puncture by a microelectrode; however, it is more complicated to distinguish the types of neurons by the spike firing characteristic of neuronal cells [6]. Dendritic and axonal trees stemming from the soma develop an extremely complex branching pattern that occupies and fills three-dimensional (3D) space [7]. The 3D morphological structure of neurons can be obtained by two-photon excitation fluorescence microscopy and fluorescence staining techniques. Since digitally reconstructed neuronal morphology is not only relatively simple to analyze, but also easier to obtain, researchers are more inclined to solve the neuron classification problem by analyzing the geometry data of neurons.

At present, some researchers have applied machine learning approaches to solve the morphology classification of neurons, which have achieved many important results. Alavi et al. [8] analyzed the two-dimensional (2D) and 3D microscopic images of neurons, and obtained the geometric information of neurons to classify dopaminergic neurons in the rodent brain. They used different machine learning approaches for neuron classification such as support vector machine (SVM), backpropagation neural network, and multi-class logistic regression (LR). The neuron classification performances of different

approaches have also been compared. Han and Zeng [9] proposed a neuron classification approach based on SVM where neurons were treated as irregular fragments, and the fractal dimension was calculated to describe the spatial structure of neurons. The fractal dimension was used as a new morphological feature of neurons and another 16 morphological features were added to the neuronal morphology classifier. The importance of fractal dimension feature to the neuron classification was validated by the experiments. Zhang [10] presented a neuron geometric classification approach based on a naive Bayesian (NB) classifier, which improves the accuracy of classification by the selection of classification attributes. Furthermore, Mihaljević et al. [11,12] used label Bayesian networks to develop the classification method for GABAergic interneurons. Hernández-Pérez et al. [13] proposed new features for neuron classification where three time series were derived from the 3D structure of the neuron. The selected neuron datasets were subjected to supervised classification and the results were compared by considering three sets of features: morphological, features obtained from the time series, and a combination of both. In addition, some neuron classification approaches did not apply to the category label information of neurons, so they used clustering techniques to realize the division of neuron morphologies. Zhang et al. [14] selected and analyzed 16 morphological features of neurons, extracted three main factors for cluster analysis, and then classified the neurons. Chao and Han [15] used the expectation maximization algorithm to classify 1907 human pyramidal neurons into six categories, and then sorted the six types of data according to the weights to predict the growth process of human pyramidal neurons.

From the above analysis, although researchers have proposed a number of neuron classification approaches, most of them are aimed at neuron datasets with a small number of morphological features, and the classification accuracy is not high. In this paper, we collected and selected the 3D neuron data of two different animals. Then, using the software tool L-Measure [16], the 43 morphological features were obtained for each 3D neuron. We propose a novel neuronal morphology classification approach based on deep neural networks, which contains two computational models: a locally cumulative connected deep neural network (LCCDNN) and a fully cumulative connected deep neural network (FCCDNN). In our approach, a large number of morphological features are extracted from the original 3D neuron data, and then trained by LCCDNNs and FCCDNNs to classify the neurons into proper subtypes according to their geometry structures. In addition, the current mainstream machine learning classifiers were compared to validate the effectiveness of the proposed approach in the selection datasets.

The rest of this paper is organized as follows. In Section 2, two 3D neuron datasets are introduced including C. elegans and zebrafish datasets. In addition, the feature extraction and feature selection methods of neuronal spatial structures are also introduced. In Section 3, the neuronal morphology classification approach based on the cumulative connected deep neural networks is presented. In Section 4, the flexibility and power of the proposed approach are showcased by the neuron classification experiments. Our conclusions are presented in Section 5.

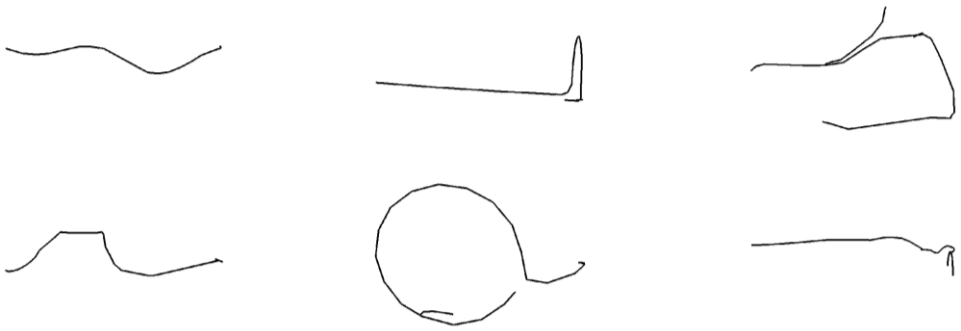
2. 3D Neuron Datasets and Morphological Feature Extraction

2.1. 3D Neuron Datasets

The 3D morphological structure of neurons can be described by the standard SWC file format, which has been widely used for analyzing neuronal morphologies or sharing neuron classifications [17,18]. The digitally reconstructed neurons are publicly available at the online NeuroMorpho.Org (http://www.neuromorpho.org) [19], which is a neuron database containing the publicly accessible 3D neuronal reconstructions and associated metadata. It collects neuron data from more than 200 laboratories around the world and continuously collects and publishes new data. So far, NeuroMorpho.org is the world's largest publicly accessible neural source database. SWC morphology files can be read by the publicly available tools CVAPP [20] and Neuromantic [21]. In order to verify the effectiveness of the proposed approach, two neuron classification datasets were constructed based on different types of neurons in different animals.

(1) C. elegans dataset

This dataset mainly performs different neuron classification within the somatic nervous system region of C. elegans. There are three types of selected neuron, namely interneuron, motor neuron (motoneuron), and somatic neuron. In the constructed neuron classification dataset, the number of samples for each subtype was 64, and the total number of samples was 192. The 2D projections of the morphological structures of interneurons, motoneurons, and somatic neurons are shown in Figure 1.



(**a**) (**b**) (**c**)

Figure 1. The 2D projections of geometric morphologies of the C. elegans neurons. (**a**) Interneuron, (**b**) Motoneuron, (**c**) Somatic neuron.

(2) Zebrafish dataset

This dataset focuses on different neuronal classifications within the main olfactory bulb region of zebrafish. We selected four types of neuron data for classification including 53 samples of each subtype, a total number of 212 neurons in the dataset, specific data sample names, and the thumbnails provided in NeuroMorpho.org. As shown in Figure 2, we provide the 2D projections of some of the geometric morphologies of neurons for the four types such as large, mitral, output, and small neurons. It can be seen that the morphological structures of different types of neurons are quite similar, and it is very challenging to effectively classify them.

(**a**) (**b**) (**c**) (**d**)

Figure 2. The 2D projections of geometric morphologies of the zebrafish neurons. (**a**) Large, (**b**) Mitral, (**c**) Output, (**d**) Small.

2.2. Morphological Feature Extraction

The morphological data file of neurons obtained from NeuroMorpho.org is in the SWC format, where a neuron, according to its spatial structure, can discretize a series of cylindrical segments (compartments). In the SWC format, each row contains seven values for a compartment: the integer label and type of compartment; x, y, and z coordinates; radius; and connectivity link, respectively. The connectivity link indicates the parental compartment (integer label) of the current coordinate point. As the SWC data does not directly reflect the morphological features of the neuron, the raw data need to be preprocessed. L-Measure is one of the subprojects of the computational neuroanatomy group in the Human Brain Project [22,23]. L-Measure can calculate a large number of morphological features from the 3D data of neurons such as the surface area of soma, the number of stems, the characteristics of compartments and branches, the properties of bifurcation points, and so on. We extracted 43 types of morphological feature for each 3D neuron using L-Measure; the names and simple descriptions of the 43 morphological features are shown in Table 1. For each morphological feature, we can obtain the maximum, minimum, average, and sum values as the original neuronal classification data. For example, the minimum length of the compartment was 1.11802 μm, the maximum length of the compartment was 656.878 μm, the average length was 10.8103 μm, and the total length of all compartments was 16,734 μm for a given input 3D neuron. The statistical analysis of each morphological feature can also be seen in [16]. In this paper, there were 172 values used as the input features for the neuronal morphology classification approaches.

Table 1. Neuronal morphological features extracted by L-Measure.

No.	Feature Name	The Simple Description of Feature
1	Soma surface	Surface of the soma
2	N_stems	Number of stems attached to the soma
3	N_bifs	Number of bifurcations for the given neuron
4	N_branch	Number of branches for the given neuron
5	N_tips	Number of terminal tips for the given neuron
6	Width	Difference of minimum and maximum x-values on the X-coordinate
7	Height	Difference of minimum and maximum y-values on the Y-coordinate
8	Depth	Difference of minimum and maximum z-values on the Z-coordinate
9	Type	Type of a compartment
10	Diameter	Diameter of a compartment
11	Diameter power	Compute the diameter raised to the power 1.5 for a compartment
12	Length	Length of a compartment
13	Surface	Surface of a compartment
14	Section area	Section area of a compartment
15	Volume	Volume of a compartment
16	Euclidean distance	Euclidean distance of a compartment with respect to the soma
17	Path distance	Path distance of a compartment from the soma point
18	Branch order	Order of the branch with respect to the soma
19	Terminal degree	Total number of tips that a compartment will terminate
20	Terminal segment	Compartment that ends as a terminal branch
21	Taper_1	Burke Taper which is measured between two bifurcation points
22	Taper_2	Hillman Taper which is measured between two bifurcation points
23	Branch path length	Sum of the length of all compartments from the given branch
24	Contraction	Ratio between Euclidean distance of a branch and its path length
25	Fragmentation	Total number of compartments that constitute a branch
26	Daughter ratio	Ratio between the bigger daughter and the other one
27	Parent daughter ratio	Ratio between the diameter of a daughter and its father
28	Partition asymmetry	Relation of the number of tips on the left and right on bifurcation
29	Rall power	Rall value is computed by linking the diameter of two daughter branches to the diameter of the bifurcating parent
30	Pk	This value is computed by Rall power
31	Pk classic	Same value as Pk, but with Rall power set to 1.5
32	Pk2	Same value as Pk, but with Rall Power set to 2
33	Bif_ampl_local	Angle between the first two compartments for a bifurcation

Table 1. *Cont.*

No.	Feature Name	The Simple Description of Feature
34	Bif_ampl_remote	Angle between two branches for a bifurcation
35	Bif_tilt_local	Angle between the previous compartment of bifurcating father and the two daughter branches of the same bifurcation
36	Bif_tilt_remote	Angle between the previous father compartment of the current bifurcating father and its two daughter compartments
37	Bif_torque_local	Angle between the plane of previous bifurcation and the current bifurcation
38	Bif_torque_remote	Angle between current plane of bifurcation and previous plane of bifurcation
39	Last_parent_diam	Diameter of last bifurcation before the terminal tips
40	Diameter threshold	Diameter of first compartment after the last bifurcation leading to a terminal tip
41	Hillman threshold	Compute the weighted average between 50% of father and 25% of daughter diameters of the terminal bifurcation
42	Helix	Choosing the 3 segments at a time (or four points at a time) and computes the normal form on the 3 vectors to find the 4th vector
43	Fractal dimension	Fractal dimension metric for branches of the dendrite trees

3. Neuronal Classification Approach Based on Deep Neural Networks

Artificial neural networks (ANNs) are a nonlinear and adaptive information processing system consisting of a large number of interconnecting neurons. It is an emerging interdisciplinary subject and imitates the structure and function of the human brain and draws on the research results of biological neuroscience to realize the processing of information. ANNs not only promote the application and development of intelligent computing, but have also revolutionized the research methods of information science and neurobiology. Recent studies in machine learning have shown that a deep or hierarchical neural architecture is useful to find highly nonlinear and complex patterns in data. The learning algorithms of deep neural networks have been successfully applied in image recognition, autonomous vehicles, speech recognition, question answering matching, precision push, and other applications, as deep learning technology is more and more adopted [24,25].

The general structure of deep neural networks consists of an input layer, multiple hidden layers, and an output layer. When adjacent layer neurons are connected by two, they form a fully connected deep neural network (FCDNN) [26], and its network structure is shown in Figure 3. The network has one input layer consisting of n input neurons, l hidden layers, and one output layer consisting of m output neurons. The matrix $\boldsymbol{W_i}$ represents the connection weights between layers, and the vector $\boldsymbol{b_i}$ represents the deviation of neurons in each layer. For complex pattern recognition problems, the researchers found that FCDNNs with the correct structures and synaptic weights achieved better results than the shallow neural networks [26]. FCDNNs enhance the feature representation ability from the input data by stacking a number of hidden layers. The explanation is that FCDNNs can learn and extract better features from the input samples than shallow neural networks [27].

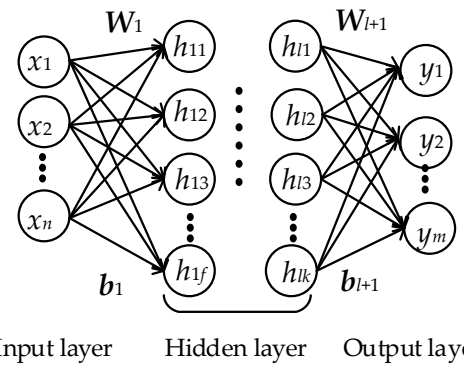


Figure 3. The fully connected deep neural network structure.

The input data in the input layer is transmitted from the hidden layer to the output layer, and the activities of the neurons for each layer are computed as:

$$\boldsymbol{h}_{i+1} = f(\boldsymbol{W}_{i+1}\boldsymbol{h}_i + \boldsymbol{b}_{i+1}) \tag{1}$$

where $f(\bullet)$ is the activation function. When $i = 0$, $\boldsymbol{h}_0 = \boldsymbol{x}$ represents the input layer, when $i = l$, $\boldsymbol{h}_{l+1} = \boldsymbol{y}$ represents the output layer.

For multi-classification problems, the l hidden layers generally use ReLU as the activation function, and the $l + 1$ layer uses softmax as the activation function. Since the layers in the network are slightly steerable, the FCDNN can be trained using the back propagation algorithm. The process is to minimize the error function, where the error function is the cross-entropy loss:

$$L = -\sum_{i=1}^{m} \hat{y}_i \log S_i \tag{2}$$

where $\hat{y}_i$ is desired output and S_i represents the probability that the sample belongs to the ith category.

3.1. Residually Connected Deep Neural Networks

Although deep neural networks have better feature representation capability, as the number of network layers increases, the gradient inevitably disappears [28]. With the introduction of batch normalization, the problem has been solved for simple network structures [29]. However, when the network structure is too complicated, the current mainstream optimizer is used to train the networks, and the degradation phenomenon remains unresolved. The residually connected deep neural network (RCDNN) model has been proposed to effectively solve this problem [30]. The residual deep network is composed of a number of inner layer blocks, and each block contains multiple neuron layers. The RCDNN structure is shown in Figure 4.

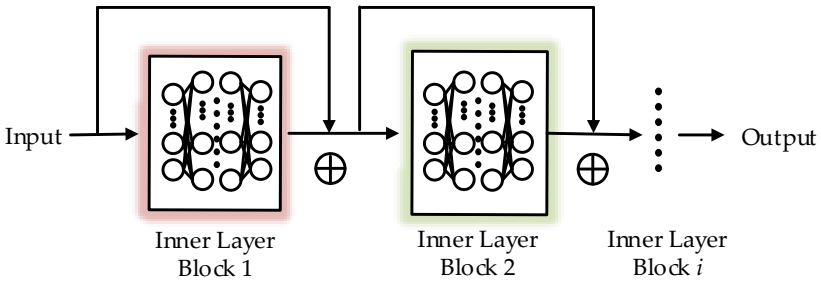


Figure 4. The structure of residually connected deep neural networks.

In the RCDNNs, each inner layer block is treated as a building block that contains multiple intermediate hidden layers. The intermediate layers are directly connected to each other in each building block, and the structure of the building blocks is shown in Figure 5. For the ith building block in the RCDNN,$\boldsymbol{x}_i$ represents the input feature vector, and $H_i(\boldsymbol{x}_i)$ represents the corresponding output of the ith building block. If the dimensions of input $\boldsymbol{x}_i$ and output $H_i(\boldsymbol{x}_i)$ are the same, the input for the next building block can be expressed as:

$$\boldsymbol{x}_{i+1} = H_i(\boldsymbol{x}_i) + \boldsymbol{x}_i \tag{3}$$

Otherwise, the input vector $\boldsymbol{x}_i$ is transformed by linear projection with $\boldsymbol{W}_s$, and the input for the next building block can be expressed as:

$$\boldsymbol{x}_{i+1} = H_i(\boldsymbol{x}_i) + \boldsymbol{W}_s\boldsymbol{x}_i \tag{4}$$

In the above connection method, the number of parameters and the computational complexity of the network is not increased. In other words, the residual structure can be applied to various existing models without changing the existing architecture of the networks.

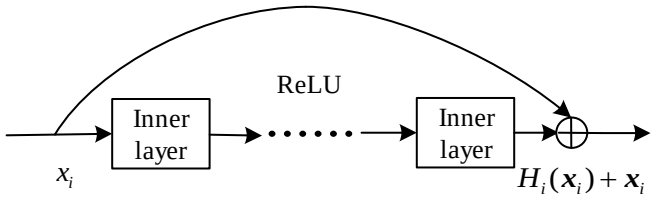


Figure 5. The structure of building blocks in the RCDNN model.

3.2. Cumulative Connected Deep Neural Networks

3.2.1. Locally Cumulative Connected Deep Neural Networks

Although RCDNNs can slow the disappearance of the gradient and further extract the hidden features, the training process in the neural networks still cannot avoid the loss of features, resulting in insufficient accuracy for data processing. In this paper, a locally cumulative connected deep neural network model was proposed based on the structure of RCDNNs. The characteristic of the LCCDNN model is that the input of each building block contains the feature vectors of multiple upper residual blocks. As shown in Figure 6, the structure of LCCDNNs is used for neuronal morphology classification, where the input feature vectors of two upper building blocks are cumulated as input for all residual blocks, except the first one.

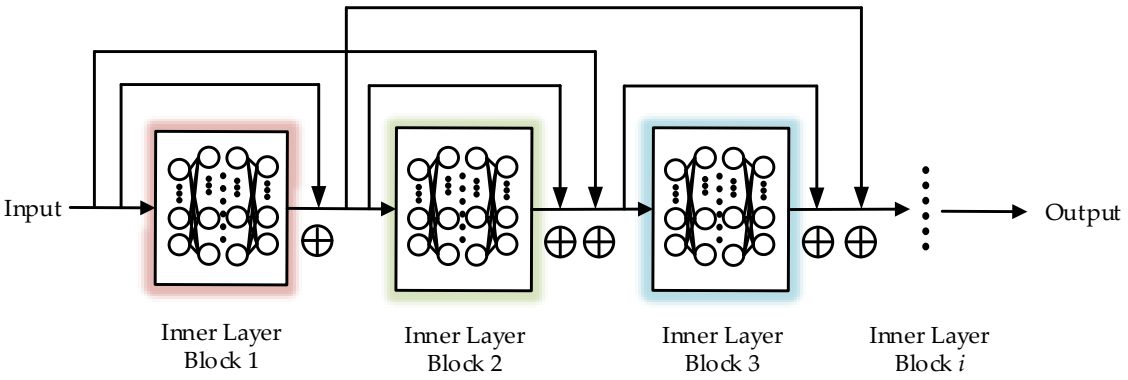


Figure 6. The structure of locally cumulative connected deep neural networks.

Compared with the traditional residual networks, the input of each building block in the LCCDNN is also accumulated with the output of the upper block and the inputs of the two upper blocks. The input of building block $I + 1$ is computed as:

$$\boldsymbol{x}_{i+1} = H_i(\boldsymbol{x}_i) + \boldsymbol{x}_i + \boldsymbol{x}_{i-1} \tag{5}$$

where $H_i(\boldsymbol{x}_i)$ represents the output of the ith building block and $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i-1}$ are the input feature vectors of the two upper blocks. In addition, the input of the first building block is calculated as Equation (3) or (4), and $\boldsymbol{x}_1$ is the sample feature vector.

3.2.2. Fully Cumulative Connected Deep Neural Networks

If the structure of the LCCDNN is extended over the entire building blocks of the network, that is, the input of each building block cumulates the feature vectors of all upper building block, the fully cumulative connected deep neural network model can be obtained. The FCCDNN structure is shown

in Figure 7, where it can be seen that the extended residual networks contain a number of building blocks through the fully cumulative compute mode. In the FCCDNNs, the input calculation of each inner layer block can be expressed as:

$$\boldsymbol{x}_{i+1} = H_i(\boldsymbol{x}_i) + \boldsymbol{x}_i + \sum_{j=1}^{j=i-1} \boldsymbol{x}_j \tag{6}$$

According to the first building block, the only input sample feature $\boldsymbol{x}_1$ is cumulative, and there are more and more cumulative feature vectors along with the increase in the number of building blocks.

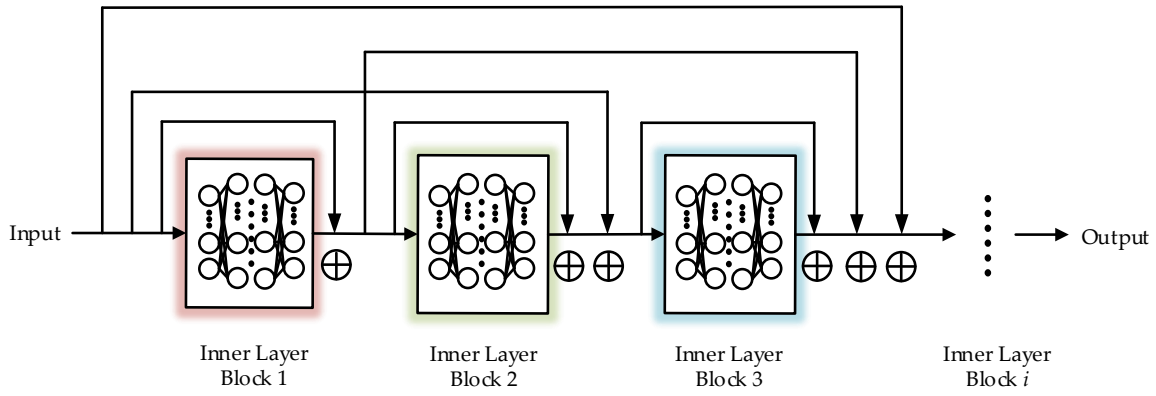


Figure 7. The structure of fully cumulative connected deep neural networks.

Although in theory the feature reuse capability of the FCCDNNs is stronger than that of the LCCDNNs, the networks may cause a certain degree of gradient explosion because the accumulation of all upper block features causes the input of the building block to be too large. In fact, the locally and fully cumulative connected deep neural network models are extended from the deep residual networks, the number of parameters is not increased, and the core idea is to increase the accuracy of the network train by increasing a certain amount of calculation.

3.3. Classification Approach of 3D Neurons

Normalizing the feature data can effectively reduce the influence of large numerical features, increase the training speed of machine learning methods, and reduce over-fitting [31]. For a feature x, the maximum $x_{\max}$ and minimum $x_{\min}$ can be found from all training sets, and the scaling value $\widetilde{x}$ can be expressed as:

$$\widetilde{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{7}$$

The neuron morphology classification process based on the LCCDNNs is shown in Figure 8. The neuron classification process consisted of the following parts: first, the selected neuronal dataset was divided into training and test sets, then the feature values were extracted by using the L-Measure tool; second, the normalized feature values were also obtained by feature scaling with Equation (7); and finally, the feature data of the training set were used to train LCCDNNs, and the test set was used to verify the result.

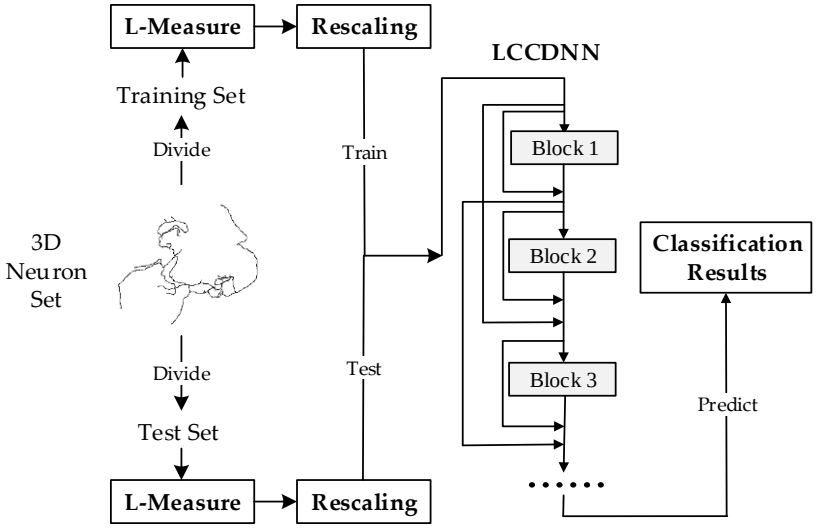


Figure 8. The neuronal morphology classification process based on the LCCDNNs.

4. Experiments and Results

In this section, several experiments are presented to demonstrate the classification capabilities of the LCCDNN and FCCDNN models. At first, the neuronal morphology classification approach was applied to the C. elegans and zebrafish datasets by demonstrating the neuron classification process. Furthermore, we analyzed the relevant parameters that may influence the classification performance of neuron morphology such as mini-batch size, the number of intermediate layers in a building block, and the number of building blocks. Finally, we compared the classification accuracy of the proposed deep neural network models and other current machine learning classifiers.

In order to evaluate the performance of this predictive model, the correct predictive score based on SKLEARN [32] was used as the classification evaluation criterion. If y_i is the prediction value of the ith sample and $\hat{y}_i$ is the corresponding true value, then the correct prediction score of the n_{sample} is defined as:

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{sample}}} \sum_{i=1}^{n_{\text{sample}}} 1(y_i = \hat{y}_i) \tag{8}$$

4.1. Experimental Parameter Settings

The deep neural network model used in the simulations was feedforward architecture with one input layer, multiple hidden layers, and one output layer. The number of units of the input layer was 172, which were used to input 43 neuronal morphological features, with each feature containing the maximum, minimum, average, and sum values. The number of units of the output layer was determined by the 3D neuron classification problem, which was three for the C. elegans dataset and four for the zebrafish dataset. The hidden layers were divided into many building blocks with the same number of intermediate layers, and the number of neurons of each intermediate layer (or hidden layer) was 200. Table 2 lists the base building block parameter settings of the LCCDNNs and FCCDNNs for different neuron datasets. The basic mini-batch size was 64 for the LCCDNN and FCCDNN models in the training processes.

The implementation of deep neural networks in this paper was based on the Keras framework [33]. All parameters, except the network structure parameters, used the Keras recommended parameters. The general parameters of the neural networks were: the last output layer activation function was softmax, the activation function of other hidden layers was Leaky ReLU; the initial weight distribution of each layer was random and uniform, which guarantees that good information is reserved; and the gradient was reduced by Adam [34] (learning rate 0.001, $\beta 1 = 0.9$, $\beta 2 = 0.999$, ε = 10E-8, decay = 0). Furthermore, each normalized layer was used after batch normalization to optimize training, and the

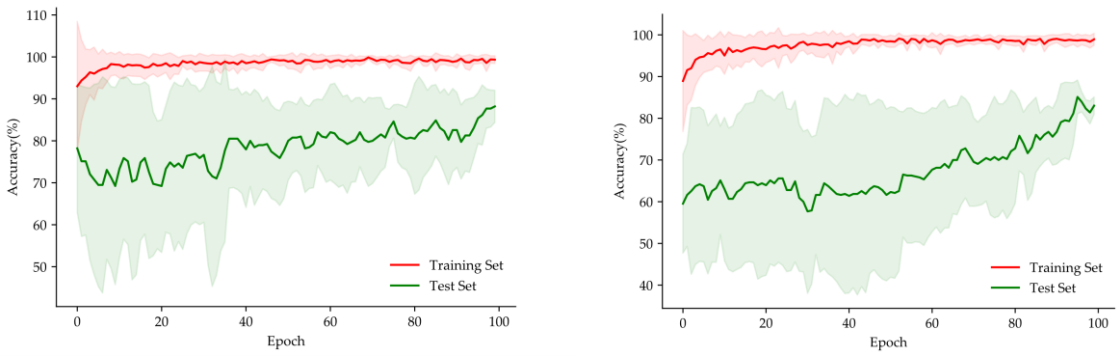
learning epoch was 100. In the results reported here, the training data and the test data were mainly divided by the number of random seeds at a ratio of 0.8 and 0.2 in each experiment. That is, for the C. elegans neuronal dataset, the number of training samples was 153, the number of test samples was 39; and for the zebrafish neuronal dataset, the number of training samples was 169, and the number of test samples was 43. Finally, the average and standard deviation of the statistical results after 10 repeated experiments were determined as the final results of the experiment to study the classification accuracy.

Table 2. The base building block parameter settings of the LCCDNNs and FCCDNNs.

Parameter	Dataset	Deep Neural Network Model	
		LCCDNNs	FCCDNNs
Number of intermediate layers	C. elegans	10	1
	Zebrafish	5	10
Number of building blocks	C. elegans	3	3
	Zebrafish	5	5
Mini-batch size		64	64

4.2. Neuron Classification Process Analysis Based on LCCDNNs and FCCDNNs

Figures 9 and 10 show the neuron classification accuracies of the training processes using the LCCDNN and FCCDNN models, where the red and green curves represent the average values of neuron classification, and the red and green areas were obtained by adding and subtracting the standard deviation values. Figure 9 shows the training process accuracy curves for classifying two kinds of neuron datasets with LCCDNNs. As shown in Figure 9a, it can be seen that the accuracy of the test set of the C. elegans dataset increased slowly during the whole process, and the fluctuation was larger before 40 learning epochs before it decreased. At the end of the training, the classification accuracy was 87.44% ± 4.09%. As shown in Figure 9b, it can be seen that the accuracy of test set of the zebrafish dataset increased slowly during the whole process, the fluctuation decreased gradually, and the final average accuracy was 81.16% ± 9.2%. Figure 10 shows the training process accuracy curves of the two neuron datasets with FCCDNNs. From Figure 10a, we can see that the accuracy of the C. elegans training set increased rapidly before 20 learning epochs, then reached a stable value, and the fluctuation of the est set was violent during the whole training process. As shown in Figure 10b, the classification accuracy fluctuation of the test set was concentrated to before 40 epochs in the zebrafish dataset, and the classification accuracy was 72.33% ± 6.05%.



(**a**) (**b**)

Figure 9. The change curve of accuracy during training by LCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

According to the above results, it can be seen that when LCCDNNs and FCCDNNs are used to solve the neuronal morphology classification problem, the constructed 3D neuron dataset has a greater impact

on the classification accuracy. For the zebrafish dataset, the morphological structures of the different types of neurons were more similar, so it is more difficult to classify neurons by the deep learning model. In addition, the neurons in the dataset had more types and more complex morphological structures, so the deep neural network model obtained a lower accuracy of neuron classification.

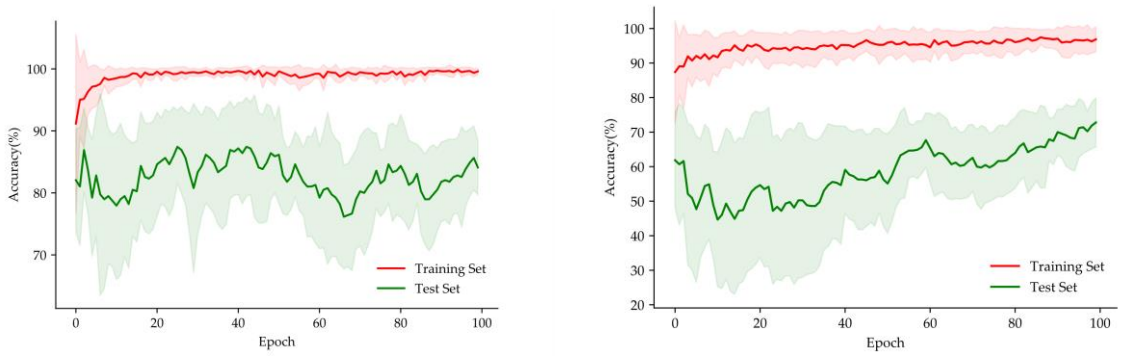


Figure 10. The change curve of accuracy during training by FCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

4.3. Effects of LCCDNN and FCCDNN Parameters on Neuron Classification

We also examined the effects of different parameters of deep learning networks on the performance of neuron classification including mini-batch size, number of intermediate layers, and number of building blocks. The average value of neuron classification accuracy was obtained after 10 repeated experiments, and the black line of the bar chart (Figures 11–16) represents the standard deviation of accuracy.

4.3.1. Accuracy Analysis of Different Mini-Batch Size

For neural networks, a reasonable mini-batch size plays a key role in the well-trained model results [35]. Figures 11 and 12 show the performance comparison of LCCDNN and FCCDNN models with a different mini-batch size. In order to facilitate parallel computing, the mini-batch size in this paper mainly used the power of 2. At the same time, considering the speed and accuracy, this paper set the base mini-batch size to 64 and then adjusted the parameters of the model. The selected training mini-batch sizes were 32, 64, and 128, respectively. It can be seen that LCCDNNs have a higher classification accuracy of neurons than FCCDNNs.

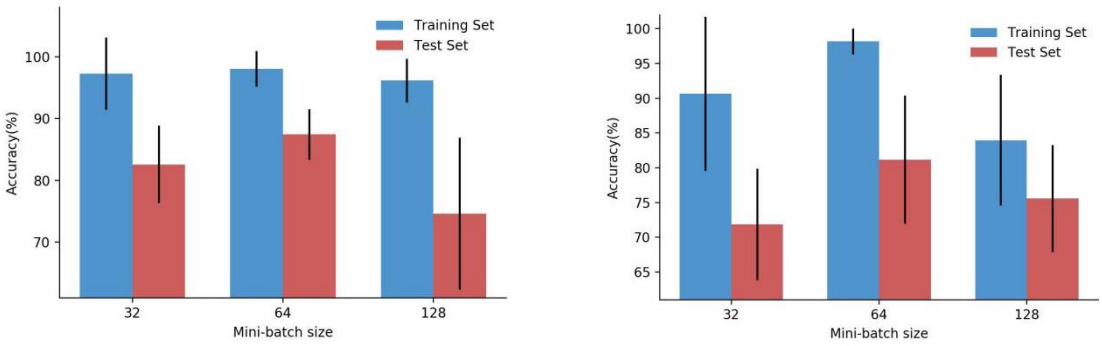


Figure 11. Comparison of accuracy with different mini-batch sizes by LCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

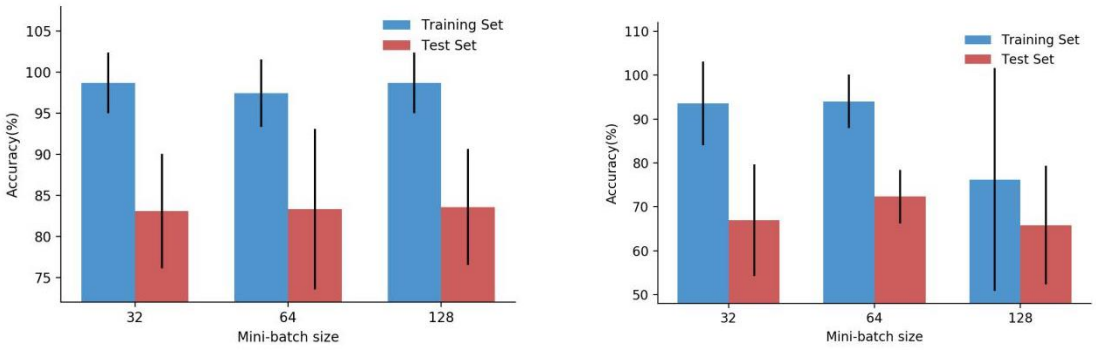


Figure 12. Comparison of accuracy with different mini-batch sizes by FCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

Figure 11 shows the variation in classification accuracy by using the LCCDNNs. The mini-batch sizes were 32, 64, and 128, while the other parameter settings remained the same. For the C. elegans dataset (Figure 11a), the accuracy of the training set was 97.25%, 98.03% and 96.14%, and the accuracy of the test set was 82.56%, 87.43%, and 74.62%, respectively. Figure 11b shows the accuracy variation in the zebrafish dataset with different mini-batch size, where the accuracy of the training set was 90.65%, 98.16%, and 83.96% and the accuracy of the test set was 71.86%, 81.16% and 75.58%. Figure 12 shows the classification results of FCCDNNs. When the mini-batch size was 32, 64, and 128, the accuracy of the test set of C. elegans was 83.08%, 83.33%, and 83.33%, and the accuracy of the test set of zebrafish was 66.98%, 72.33%, and 65.81%., respectively Therefore, when the mini-batch size was 64, the LCCDNN and FCCDNN models had the highest classification accuracy of neurons.

4.3.2. Accuracy Analysis of Different Numbers of Intermediate Layers

Figure 13 shows the classification results with different numbers of intermediate layers by using the LCCDNN learning model. The number of intermediate layers increased gradually, while the other parameter settings remained the same. Figure 13a shows the neuron classification accuracy after 100 learning epochs in the C. elegans dataset. When the number of intermediate layers was 5, 10, and 15, the accuracy of the training set was 98.24%, 98.04%, and 96.27%, and the accuracy of the test set was 85.13%, 87.44%, and 74.10%, respectively. Therefore, when the intermediate layer is 10, the LCCDNN model can learn with higher accuracy in the C. elegans dataset. Figure 13b shows the learning results in the zebrafish dataset with the intermediate layers of 1, 5, and 10, where the accuracy of the training set was 88.34%, 98.17%, and 91.72% the accuracy of the test set was 72.09%, 81.16%, and 64.65%, respectively. From Figure 13b, we can see that the classification accuracy increased at first, then decreased when the intermediate layer number of LCCDNNs increased, so the suitable intermediate layer number for the zebrafish dataset was five.

Furthermore, we investigated the effect of the number of intermediate layers on the neuron classification for the FCCDNN learning model. Figure 14 shows the change in accuracy with the different numbers of intermediate layers, while the number of building blocks and the mini-batch size remained the same. As shown in Figure 14a, the classification accuracy of the C. elegans dataset decreased when the number of intermediate layers increased gradually. When the number of intermediate layers was 1, 5, and 10, the accuracy of the training set was 97.45%, 96.34% and 95.16% and the accuracy of the test set was 83.33%, 81.03%, and 77.95%, respectively. From Figure 14b, we can see that the zebrafish dataset had the highest classification accuracy when the intermediate layer number was 10, the accuracy was 94.02% in the training set, and the accuracy was 72.33% in the test set. It can be seen that LCCDNNs have a higher classification accuracy of neurons than FCCDNNs.

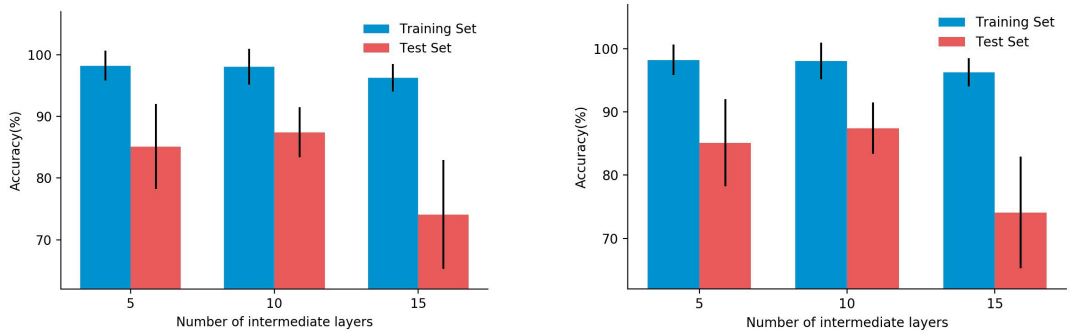


Figure 13. Comparison of accuracy with different numbers of intermediate layers by LCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

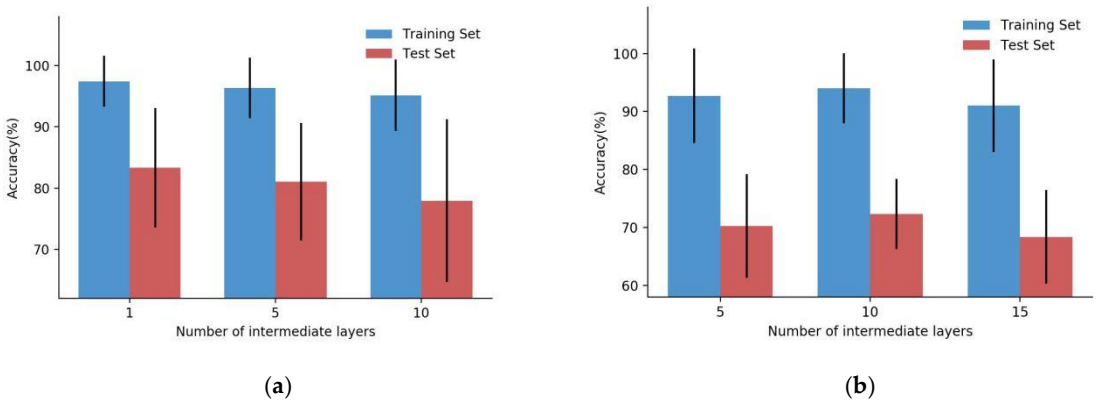


Figure 14. Comparison of accuracy with different numbers of intermediate layers by FCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

4.3.3. Accuracy Analysis of Different Numbers of Building Blocks

In this experiment, the performance comparison of different numbers of building blocks in the LCCDNN and FCCDNN models was tested for solving the neuronal morphology classification problem. Figure 15 shows the classification results with different numbers of building blocks in the LCCDNNs. The number of building blocks in the deep neural network was increased, while the other settings remained the same. Figure 15a shows the classification accuracy in the C. elegans dataset and Figure 15b shows the classification accuracy in the zebrafish dataset. With the increase in the number of building blocks, the classification accuracy increased. However, when the building block number increased further, the classification accuracy decreased gradually.

Figure 16a shows the variation of FCCDNN accuracy in the C. elegans dataset, when the number of building blocks was 3, 4, and 5, the accuracy of the training set was 97.45%, 96.86%, and 93.012%, and the accuracy of the test set was 83.33%, 82.82%, and 80.26%, respectively. Therefore, when the building block number is three, the FCCDNN model for the C. elegans dataset is reasonable. Figure 16b shows the classification accuracy in the zebrafish dataset; when the number of building blocks was 4, 5, and 6, the accuracy of the training set was 83.14%, 94.02%, and 78.34% and the accuracy of the test set was 66.28%, 72.33% and 65.12%, respectively. It can be seen that the zebrafish dataset had the highest classification accuracy when the number of building blocks was five.

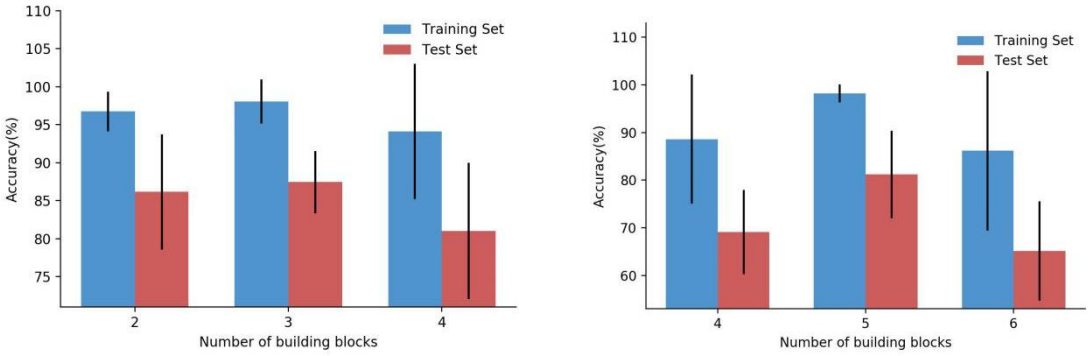


Figure 15. Comparison of accuracy with different numbers of building blocks by LCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

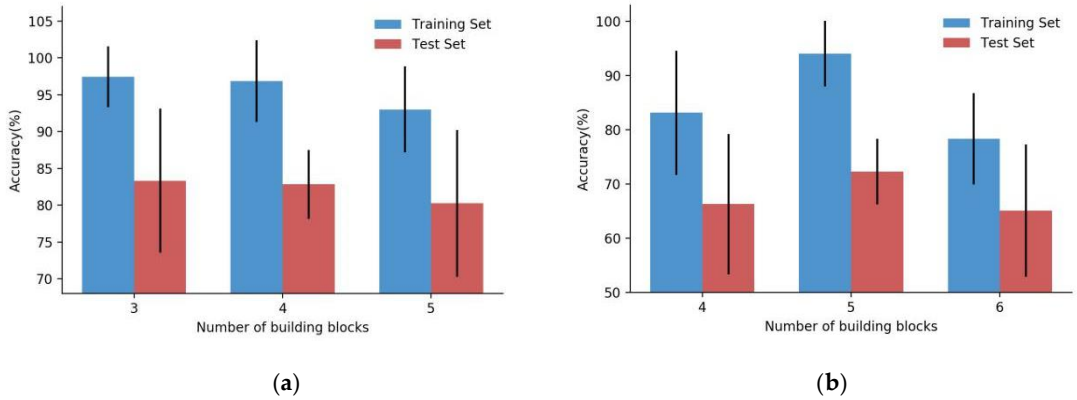
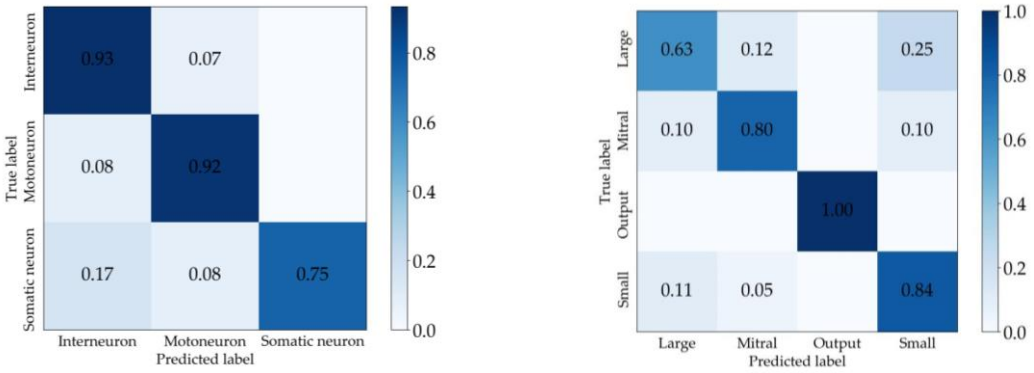


Figure 16. Comparison of accuracy with different numbers of building blocks by FCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

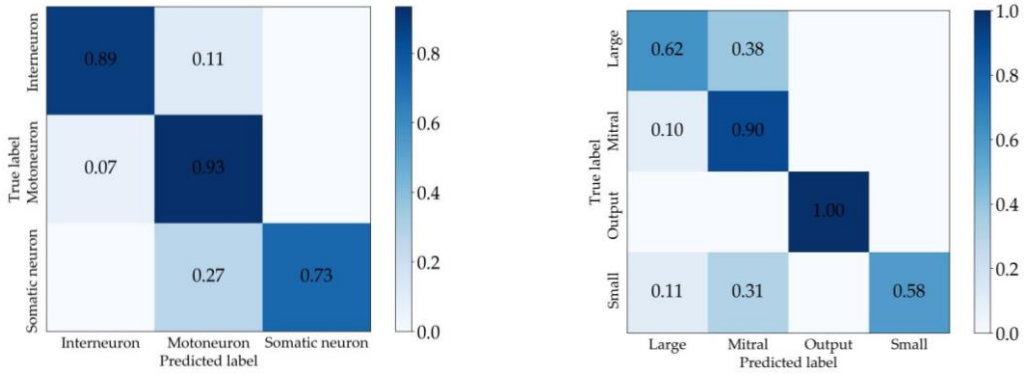
4.3.4. Comparison Experiment of Accuracy for Each Neuron Type

According to the above experiments, the overall classification accuracies of different types of neurons were compared for two kinds of neuron datasets by using LCCDNNs and FCCDNNs. To further validate the advantages of the cumulative connected deep neural networks, the accuracy of each neuron type is analyzed in this section. The confusion matrix was used to compare the accuracy of each neuron type for the two deep learning models in the C. elegans dataset and zebrafish dataset. Figures 17 and 18 are the confusion matrices of the test sets for the LCCDNNs and FCCDNNs, respectively. The values on the diagonals represent the classification accuracies of interneuron, motoneuron, and somatic neuron for the LCCDNN model in Figure 17a and the FCCDNN model in Figure 18a. Similarly, the values on the diagonals represent the classification accuracies of large, mitral, output, and small neuron types in Figures 17b and 18b. The color is bluer in the matrix, so the deep learning model had a better classification performance for the corresponding neuron type. It can be seen that the classification accuracy of interneurons and somatic neurons in LCCDNNs was higher than that in the FCCDNNs for the C. elegans dataset. For the zebrafish dataset, the recognition rate of the output neuron was 100% for the two deep learning models, the classification accuracy of small neurons by LCCDNNs was higher than the result obtained by FCCDNNs, but did not improve the classification accuracies of large and mitral neuron types.



(**a**) (**b**)

Figure 17. Confusion matrix of neuronal classification by LCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.



(**a**) (**b**)

Figure 18. Confusion matrices of neuronal classification by FCCDNNs. (**a**) C. elegans dataset, (**b**) zebrafish dataset.

4.4. Classification Performance Comparison with Other Machine Learning Methods

To verify the effectiveness of the LCCDNN model for the neuronal morphological classification problem, we compared the following mainstream machine learning approaches: logistic regression based on generalized linearity [8,36], naive Bayesian classifier based on prior probability estimation [10,12], decision tree (DT) based on information entropy [37], k-nearest neighbor (KNN) [38], support vector machine based on geometric distance [9], and long short-term memory (LSTM) network model [39]. In addition, the deep neural network models such as FCDNN and RCDNN are also used for classification performance comparison. These learning methods are implemented using SKLEARN [32], and the parameters of the learning methods have been tested and fine-tuned based on the recommended values in the simulation experiments. For LR, the 'liblinear' solver was used, which supports the L2 penalty, the tolerance for stopping criteria was 1e-4, the inverse of regularization strength was 1.0, the maximum number of iterations was 100, and other parameters were the default. For NB, the variance smoothing was 1e-09 and the other parameters were the default. For DT, the criteria = 'gini' and splitter = 'best', the minimum number of samples required to split an internal node was two, the minimum number of samples required to be at a leaf node was one, the threshold for early stopping in tree growth min_impurity_split was 1e-7, and other parameters were the default. For KNN, the number of neighbors was five, the metric was 'minkowski' with the power parameter of two, and the leaf size was 30. For SVM, the kernel was linear function, the penalty = 'L2' and loss = 'squared_hinge', the penalty parameter C of the error term was 1.0, and other parameters were the default.

Table 3 compares the classification accuracy of the LCCDNN and FCCDNN learning models against existing classifiers for the C. elegans and zebrafish datasets. The FCDNN model had the best effect on the C. elegans dataset with a layer parameter of 30 and a classification accuracy of 85.9% for the test set. When the layer number was 20, the FCDNN model obtained the best classification accuracy of 75.12% on the zebrafish dataset. For the RCDNN learning model, on the C. elegans dataset, the use of one building block and 10 intermediate layers per building block showed the best classification accuracy on the training set and the test set of 97.71% and 85.9%, respectively; on the zebrafish dataset, using four building blocks and 10 intermediate layers, the best classification accuracy on the training set and the test set was 93.79% and 78.14%, respectively. The LSTM network model has two layers and two step sizes, and the number of neurons in each layer was 50. The LSTM's classification accuracy on the two neuron datasets was 68.46% and 70.47% for the test set.

Table 3. Comparison of the accuracy of the classifiers.

Dataset	Classification Approach	Classification Accuracy	
		Training Set (%)	Test Set (%)
C. elegans	LR	87.52 ± 1.0	79.23 ± 8.24
	NB	80.85 ± 2.94	57.18 ± 10.54
	DT	100	63.85 ± 11.50
	KNN	88.04 ± 1.2	78.72 ± 4.53
	SVM	94.18 ± 1.55	80.77 ± 10.76
	LSTM	99.93 ± 0.21	68.46 ± 9.97
	FCDNN-30	89.02 ± 9.13	81.35 ± 15.14
	RCDNN-10	97.71 ± 4.19	85.9 ± 9.15
	LCCDNN-30	98.03 ± 2.89	87.44 ± 4.09
	FCCDNN-3	97.45 ± 4.14	83.33 ± 9.76
Zebrafish	LR	93.14 ± 1.12	74.65 ± 8.52
	NB	87.63 ± 1.38	46.28 ± 16.09
	DT	100	64.65 ± 11.44
	KNN	88.34 ± 1.15	76.51 ± 5.95
	SVM	96.27 ± 0.97	71.16 ± 8.29
	LSTM	97.81 ± 0.74	70.47 ± 0.74
	FCDNN-20	75.27 ± 10.21	75.12 ± 9.81
	RCDNN-40	93.79 ± 6.32	78.14 ± 8.15
	LCCDNN-25	98.16 ± 1.88	81.16 ± 9.2
	FCCDNN-50	94.02 ± 6.06	72.33 ± 6.05

For the two datasets, the classification accuracies of the LCCDNN model were comparable to that of the other learning methods. When LCCDNN contained three building blocks and 10 intermediate layers per building block, the classification accuracy on the training set and the test set of the C. elegans dataset was 98.03% and 87.44%, respectively. When the number of building blocks and the number of intermediate layers were both five, the classification accuracy of the LCCDNN on the training set and the test set of zebrafish dataset was 98.16% and 81.16%, respectively. It can be seen that the classification accuracy of the LCCDNN model was higher than that of the other approaches such as the traditional classifiers LR, NB, DT, KNN, and SVM, and the neural network models LSTM, FCDNN, RCDNN, and FCCDNN.

5. Conclusions

The classification of neurons based on their morphological structures is an important way to thoroughly understand brain information processing. This paper proposed a neuronal morphology classification approach based on the LCCDNNs. We first constructed two 3D neuron datasets including the C. elegans and zebrafish datasets, and 43 geometric features were extracted to classify the types of neurons. Then, by using the theory of the residually connected deep network model, we proposed two

cumulative connected deep neural networks, the LCCDNN and FCCDNN. Finally, the proposed deep learning networks were applied to a 3D neuron classification problem. In the experiments, through the analysis of the influence of relevant parameters, the performance of the better model was obtained. We also compared the classification performance of the LCCDNN approach with other mainstream machine learning approaches, which can learn and classify effective features for two different animal neuron datasets. However, these morphology-based classification approaches largely rely on the feature extraction and selection techniques of neuronal spatial structures, so a lot of useful information for neuronal classification may be lost. Future research will be based on 3D neuron data without feature extraction to study the classification of neurons.

Author Contributions: Conceptualization, X.L. and J.Z.; Data Curation, J.Z.; Funding Acquisition, X.L.; Methodology, X.L.; Software, J.Z.; Writing-Original Draft, J.Z.; Writing-Review & Editing, X.L.

Funding: The work is supported by the National Natural Science Foundation of China (grant no. 61762080); the Lanzhou Municipal Science and Technology Project (grant no. 2019-1-34), and the Program for Innovative Research Team in Northwest Normal University (grant no. 6008-01602).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Buckmaster, P.S.; Alonso, A.; Canfiled, D.R.; Amaral, D.G. Dendritic morphology, local circuitry, and intrinsic electrophysiology of principal neurons in the entorhinal cortex of macaque monkeys. *J. Comp. Neurol.* **2004**, *470*, 317–329. [CrossRef] [PubMed]
2. Perin, R.; Telefont, M.; Markram, H. Computing the size and number of neuronal clusters in local circuits. *Front. Neuroanat.* **2013**, *7*, 1. [CrossRef] [PubMed]
3. Bota, M.; Swanson, L.W. The neuron classification problem. *Brain Res. Rev.* **2007**, *56*, 79–88. [CrossRef] [PubMed]
4. Zeng, H.; Sanes, J.R. Neuronal cell-type classification: Challenges, opportunities and the path forward. *Nat. Rev. Neurosci.* **2017**, *18*, 530–546. [CrossRef] [PubMed]
5. Nishimura, K.; Ohta, M.; Saito, M.; Morita-Isogai, Y.; Sato, H.; Kuramoto, E.; Yin, D.X.; Maeda, Y.; Kaneko, T.; Yamashiro, T.; et al. Electrophysiological and morphological properties of α and γ motoneurons in the rat trigeminal motor nucleus. *Front. Cell Neurosci.* **2018**, *12*, 9. [CrossRef] [PubMed]
6. Oshio, K.I.; Yamada, S.; Nakashima, M. Neuron classification based on temporal firing patterns by the dynamical analysis with changing time resolution (DCT) method. *Biol. Cybern.* **2003**, *88*, 438–449. [PubMed]
7. Lin, X.; Li, Z.; Ma, H.; Wang, X. An evolutionary developmental approach for generation of 3D neuronal morphologies using gene regulatory networks. *Neurocomputing* **2018**, *273*, 346–356. [CrossRef]
8. Alavi, A.; Cavanagh, B.; Tuxworth, G.; Meedeniya, A.; Mackaysim, A.; Blumenstein, M. Automated classification of dopaminergic neurons in the rodent brain. In Proceedings of the International Joint Conference on Neural Networks, Atlanta, GA, USA, 14–19 June 2009; pp. 81–88.
9. Han, F.; Zeng, J. Research for neuron classification based on support vector machine. In Proceedings of the 3rd International Conference on Digital Manufacturing and Automation, Guilin, China, 31 July–2 August 2012; pp. 646–649.
10. Zhang, B. Neurons Classification Based on the Bayesian Classifier. Master's Thesis, Chongqing Jiaotong University, Chongqing, China, 2012.
11. Mihaljević, B.; Bielza, C.; Benavides-Piccione, R.; DeFelipe, J.; Larrañaga, P. Multi-dimensional classification of GABAergic interneurons with Bayesian network-modeled label uncertainty. *Front. Comput. Neur.* **2014**, *8*, 150. [CrossRef] [PubMed]
12. Mihaljević, B.; Benavides-Piccione, R.; Bielza, C.; DeFelipe, J.; Larrañaga, P. Bayesian network classifiers for categorizing cortical GABAergic interneurons. *Neuroinformatics* **2015**, *13*, 193–208. [CrossRef]
13. Hernández-Pérez, L.A.; Delgado-Castillo, D.; Martín-Pérez, R.; Orozco-Morales, R.; Lorenzo-Ginori, J.V. New features for neuron classification. *Neuroinformatics* **2019**, *17*, 5–25. [CrossRef]
14. Zhang, J.; Deng, S.H.; Guo, H.; Shen, F.; Gu, W.G.; Li, Y.W. Application of cluster analysis in morphological characteristics of neurons. *J. Zhejiang Univ. (Agric. Life Sci.)* **2011**, *37*, 493–500.
15. Yu, C.; Han, Z.; Zeng, W.; Liu, S. Morphology cluster and prediction of growth of human brain pyramidal neurons. *Neural Regen. Res.* **2012**, *7*, 36–40. [PubMed]

16. Scorcioni, R.; Polavaram, S.; Ascoli, G.A. L-Measure: A web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. *Nat. Protoc.* **2008**, *3*, 866–876. [CrossRef] [PubMed]
17. Cannon, R.C.; Turner, D.A.; Pyapali, G.K.; Wheal, H.V. An on-line archive of reconstructed hippocampal neurons. *J. Neurosci. Meth.* **1998**, *84*, 49–54. [CrossRef]
18. Yamasaki, T.; Isokawa, T.; Matsui, N.; Ikeno, H.; Kanzaki, R. Reconstruction and simulation for three-dimensional morphological structure of insect neurons. *Neurocomputing* **2006**, *69*, 1043–1047. [CrossRef]
19. Ascoli, G.A.; Donohue, D.E.; Halavi, M. NeuroMorpho.Org: A central resource for neuronal morphologies. *J. Neurosci* **2007**, *27*, 9247–9251. [CrossRef] [PubMed]
20. Cannon, R.C. Structure Editing and Conversion with CVAPP. 2000. Available online: http://www.compneuro.org/CDROM/nmorph/usage.html (accessed on 5 June 2018).
21. Myatt, D.R.; Hadlington, T.; Ascoli, G.A.; Nasuto, S.J. Neuromantic –from semi-manual to semi-automatic reconstruction of neuron morphology. *Front. Neuroinfor.* **2012**, *6*, 4. [CrossRef] [PubMed]
22. Ashburner, J. Computational neuroanatomy. *Nat. Methods* **2000**, *8*, 493–500.
23. Bjerke, I.E.; Øvsthus, M.; Papp, E.A.; Yates, S.C.; Silvestri, L.; Fiorilli, J.; Pennartz, C.M.A.; Pavone, F.S.; Puchades, M.A.; Leergaard, T.B.; et al. Data integration through brain atlasing: Human Brain Project tools and strategies. *Eur. Psychiat.* **2018**, *50*, 70–76. [CrossRef]
24. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
25. Shen, D.; Wu, G.; Suk, H.I. Deep learning in medical image analysis. *Annu. Rev. Biomed. Eng.* **2017**, *19*, 221–248. [CrossRef] [PubMed]
26. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
27. Montufar, G.F.; Pascanu, R.; Cho, K.; Bengio, Y. On the number of linear regions of deep neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 7–14 December 2014; pp. 2924–2932.
28. Erb, R.J. Introduction to backpropagation neural network computation. *Pharm. Res.* **1993**, *10*, 165–170. [CrossRef] [PubMed]
29. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, Nevada, 26 June–1 July 2016; pp. 770–778.
31. Wang, Y.; Moulin, P. Optimized feature extraction for learning-based image steganalysis. *IEEE Trans. Inf. Forensics Secur.* **2007**, *2*, 31–45. [CrossRef]
32. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2012**, *12*, 2825–2830.
33. Keras. Available online: https://github.com/keras-team/keras (accessed on 5 June 2018).
34. Dozat, T. Incorporating nesterov momentum into Adam. In Proceedings of the International Conference of Learning Representation. San Juan, Puerto Rico, 2–4 May 2016; pp. 2013–2016.
35. Keskar, N.S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; Tang, P.T.P. On large-batch training for deep learning: Generalization gap and sharp minima. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
36. Cucchiara, A. Applied logistic regression. *Technometrics* **2012**, *34*, 358–359. [CrossRef]
37. Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.* **1991**, *21*, 660–674.
38. Fukunaga, K.; Hostetler, L. Optimization of k-nearest neighbor density estimates. *IEEE Trans. Inf. Theory* **1973**, *19*, 320–326. [CrossRef]
39. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *28*, 2222–2232. [CrossRef]