Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9825851, 1 page
https://doi.org/10.1155/2023/9825851

Retraction

Retracted: Image Classification Model Based on Deep Learning in Internet of Things

Wireless Communications and Mobile Computing

Received 8 August 2023; Accepted 8 August 2023; Published 9 August 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] S. Zou, W. Chen, and H. Chen, "Image Classification Model Based on Deep Learning in Internet of Things," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 6677907, 16 pages, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 6677907, 16 pages
https://doi.org/10.1155/2020/6677907

Research Article

Image Classification Model Based on Deep Learning in Internet of Things

Songshang Zou,[1] Wenshu Chen,[2] and Hao Chen [1]

[1]*College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China*
[2]*College of Computer Science, University of Bristol, Bristol BS8 1QU, UK*

Correspondence should be addressed to Hao Chen; chenhao@hnu.edu.cn

Received 17 October 2020; Revised 12 November 2020; Accepted 25 November 2020; Published 8 December 2020

Academic Editor: Hongju Cheng

In the environment of Internet of Things, the convolutional neural network (CNN) is an important tool and method of image classification. However, the features that are extracted by each layer of CNN are all high dimensional, and the features differ among the layers. In addition, these features contain substantial amounts of redundant information. To prevent the increase in the computational burden and the decline of the model generalization performance that are caused by high dimensionality, this paper proposes an improved image classification algorithm based on deep feature fusion, which designs and builds an 8-layer CNN. In addition, it reduces the dimensionality of the features via the principal component analysis (PCA) dimensionality reduction algorithm and fuses the features that have undergone dimensionality reduction to make the obtained features more typical and differential. The experimental results demonstrate that the proposed algorithm improves the performance of the model and achieves satisfactory accuracy.

1. Introduction

In the era of Internet of Things, image classification plays an important role in multimedia information processing. The image classification accepts the given input images and produces output classification for identifying whether the disease is present or not. As artificial intelligence technology has been widely applied, image classification and recognition technology have received increasing attention and have been utilized in an increasing number of fields, such as image information retrieval, real-time target tracking, and medical image analysis. In recent years, deep learning has attracted increasing attention [1]. The previous machine learning methods have various limitations. For example, when there are few samples, it is highly difficult to represent complex functions. When using deep learning algorithms to represent complex data distributions, nonlinear network models with deep layers can be used to learn the deep features from the data in the case of few samples. Deep learning is a type of algorithm and topological structure that can be used to solve generalization problems [2]. The combination of deep hierarchical neural networks and GPU (graphics processing unit) has accelerated the execution of deep learning algorithms. Deep learning has advanced by leaps and bounds, and big data has propelled this development momentum. A convolutional neural network is a type of feed-forward neural network. Its artificial neurons can respond to surrounding units within the coverage area, and it is suitable for processing large-batch image datasets.

CNN continuously extracts and compresses image features and obtains higher level features. It condenses the original features repeatedly and obtains more reliable features [3]. Various tasks can be conducted with the features in the last layer, e.g., classification and regression. CNN has unique advantages in automatic speech recognition (ASR) and image processing due to its special structure of shared local weights and its similar layout to real biological neural networks [4]. Weight sharing lowers the network complexity; since an image with multidimensional input vectors can be directly input into the network, the complexity of data reconstruction in feature extraction and classification is avoided [5]. Through the study of current image classification and

recognition algorithms, it is discovered that various algorithms have failed to effectively fuse the multilayered deep learning features of CNN and that they have poor accuracy.

In the environment of Internet of Things, the convolutional neural network (CNN) is an important tool and method of image classification. In order to further improve the classification accuracy of the CNN model, this paper has effectively fused deep features via a cascading strategy and has increased the diversity and the expressiveness of the extracted features to enhance the classification performance of the network mode. The main contributions of this paper include the following:

(i) This paper analyzes the structure of CNN, studies the principles of activation functions, specifies the role that nonlinear activation functions play in neural networks, and shows that via facilitation by nonlinear functions; CNN has stronger feature representation performance and can realize complex image classification

(ii) To address the problems that conventional image classification algorithms that are based on deep learning cannot effectively fuse multilayered deep features and perform poorly in terms of classification accuracy, this paper proposes an improve image classification algorithm that is based on deep feature fusion and improves the diversity and expressiveness of the extracted features to improve the classification performance

(iii) By comparing the classification performances of the CNN model on the Food-101 and Places2 datasets under various activation functions, it is demonstrated that the activation function that is used in this paper can improve the classification accuracy of the model on image datasets and ensure its convergence

(iv) This paper conducts a performance analysis and evaluation of the proposed algorithm in comparison with other algorithms. The experimental results demonstrate that the algorithm that is proposed in this paper realizes higher accuracy

The remainder of this paper is organized as follows. Section 2 discusses related work. The CNN network structure and the activation function performance are analyzed in Section 3. Section 4 proposes an improved CNN image classification and recognition algorithm that is based on feature fusion. Section 5 presents the experiment results and analysis. Section 6 summarizes the conclusions of the paper and discusses future research directions.

2. Related Work

As a highly important research direction in computer vision, image classification and recognition involves knowledge from several disciplines and has been applied in multiple research fields. With the rapid development of internet technology, a substantial amount of image data are encountered in people's lives, thereby leading to increased demands for machine learning and computer vision techniques and more in-depth research [6]. According to in-depth research that has been conducted on digital image processing and deep learning, compared with other neural networks, CNN has the following strengths: the input image matches well with the topological structure of CNN. Feature extraction proceeds simultaneously with pattern classification and generation in the training process, and weight sharing can reduce the number of training parameters, thereby rendering the CNN structure simpler and more adaptive [7]. CNN is mainly used to recognize 2D images with invariance in terms of shifting, scaling, and other forms of distortion [8]. The CNN feature detection layer learns by training on the data; hence, CNN can avoid explicit feature extraction and learns implicitly from data training. In addition, because the neurons in the same feature mapping plane share the same weight, CNN can learn in parallel. This is another advantage of CNN compared to networks that have interconnected neurons. In the study of image classification, feature extraction will directly affect the classification performance of the network model. In essence, CNN is a mapping from input to output. In practical applications, it typically uses multilayer convolution and trains with a fully connected layer. The features that are learnt via one-layer convolution are typically local. In multilayer convolution, the higher the layer, the more global the learnt features are [9].

Neocognitron can be regarded as the first implementation of CNN, and it is also the first application of the receptive field in artificial neural networks. It attempts to model the visual system and enables it to complete recognition even if there is shift or slight distortion in the objects [8]. The deep learning architecture did not emerge until the last two decades. It has substantially increased the number and types of problems that can be solved by neural networks. There are 5 popular deep learning architectures: recursive neural network (RNN), long short-term memory (LSTM)/gated recurrent unit (GRU), convolutional neural network, deep brief network (DBN), and deep stacking network (DSN). CNN is a type of multilayered neural network, and it is inspired by the animal visual cortex. The first CNN was built by Yann LeCun for handwritten character recognition. As a deep network, the early layers mainly recognize features (e.g., edges) and the subsequent layers recombine these features into higher level input [10]. Therefore, deep learning can be regarded as "deep parameter adjustment." However, it also magnifies the network shortcomings. A limited training dataset easily results in overfitting. The larger the network, the more complex the computation and the more difficult the application of the network, and the deeper the network, the more easily the gradient vanishes and the more difficult the model optimization. In the study of image classification and recognition, feature extraction will directly impact the classification capacity of the network model. The main problem that is encountered with available image classification algorithms in feature extraction is that they cannot effectively utilize various deep features that are extracted by networks [11, 12].

3. Convolutional Neural Network (CNN)

A convolutional neural network is a type of feed-forward neural network that typically specializes in the processing of image data (multiarray data). The design of the CNN structure can effectively preserve the structure of the original data and generate a layered representation. A typical CNN structure includes multilevel processing layers that are ordered from left to right. CNN typically has four types of layers: convolutional, pooling, fully connected, and classification layers. Convolutional layers and pooling layers are the core layers of the design, and they are typically utilized in the first few phases.

3.1. Convolutional Layers. In CNN, the convolutional layers are the most important layers, which are typically used for feature extraction. As parts of an image may have the same statistical properties, feature learning for an image can be conducted on randomly selected parts of subimages, and the learned features will be used as a filter to scan the entire image and to obtain the feature activation values of various positions in the image to complete feature extraction [13].

In a conventional neural network, every neuron must be connected with every pixel; consequently, numerous weights will render the network difficult to train. Additionally, the number of weights of each neuron in a neural network that has a convolutional layer is equal to the size of the convolution kernel; this is equivalent to each neuron being connected with all pixels. Thus, the number of weights is substantially reduced [14].

The convolution computation consists of two steps. Step 1 is a linear operation. It processes a group of weights that are connected with the original input image or the low-level feature map, translates the convolution kernel for multiple convolutions according to the stride l, and adds the sum of the bias b_x and the results of multiple convolutions. Step 2 is a nonlinear operation. It uses the activation function $f(x)$ to obtain the output feature map C_x, namely, it performs weighted summation on one neuron through multiple input signals and outputs via the activation function [15].

The feature extractor can be replaced by a trained convolution kernel. Convolution kernels differ in terms of the structural features that they extract from an image. To extract multiple features at the same position, multiple convolution kernels can be used, and CNN will output the combination of these features from the convolutional layer. The convolutional layer has two properties that can reduce the number of parameters to be calculated: weight sharing and local perception.

(a) Weight sharing

In weight sharing, the same weight is used by all neurons in the same feature mapping. If the convolution of a group of weights and the input image yields edge features, these weights can be regarded as edge feature extractors, and they can be used directly to extract edge features of other image regions [16].

(b) Local perception

Local perception is the connection of part of the network. Similar to the human visual system, CNN's perception process of an image is from local to global, and every neuron is connected with neurons that belong to the previous layer. Therefore, the information of an entire image is perceived by neurons repeating the process of activation in a small region and translating to another region.

A convolutional layer extracts features, and the most important elements in this layer are the trained convolution kernels. Kernels can detect specified shapes, colors, and contrasts, among other features, and feature maps preserve the spatial structure after extraction; therefore, the feature maps that correspond to convolution kernels represent features of a corresponding dimension, and with the increase in the number of CNN layers, the extracted features become increasingly concrete [17].

Every node in the convolutional layer and pooling layer is connected with only some nodes in the previous layer, and the input of each node in the convolutional layer is a small block of the previous layer; the size of which is determined by the size of the window of the convolution kernel. Typically, after being processed by the convolutional layer, the node matrix will become deeper and the depth will be determined by the number of kernels [18]. Parameter sharing in the kernel enables the image contents not to be affected by the positions and can substantially reduce the number of parameters of the network model and reduce the complexity of the operation; see Figure 1 for an illustration of CNN feature extraction.

In Figure 1, the 1st convolution extracts a low-level feature, the 2nd a mid-level feature, and the 3rd a high-level feature.

3.2. Activation Function. In a neural network, every neuron node will accept the output value from the previous layer as its input value and convey this value to the next layer. The nodes in the input layer will input and transmit the property value to the next output layer. In a multilayer neural network, an activation function is used to represent the relationship between the output values of the neuron nodes in the previous layer and the input values of those in the next layer [19].

A nonlinear function is similar to an activation function, through which neural networks realize stronger representation performance and overcome the finite approximation limitation that is caused by the use of a linear function.

In the early study of neural networks, the sigmoid activation function and the tan$h$ activation function were frequently used. In the back-propagation of neural networks, the sigmoid activation function will result in gradient explosion and loss, and because the output values of the sigmoid function are not of zero mean, the convergence is slow and the training time is substantially increased in deep neural networks. In deep network learning, learning a large amount of data typically takes a long time; hence, the convergence speed of the training model is of high importance. When a deep network is trained, zero-mean data can accelerate convergence. The ReLU function is very fast in calculation, and

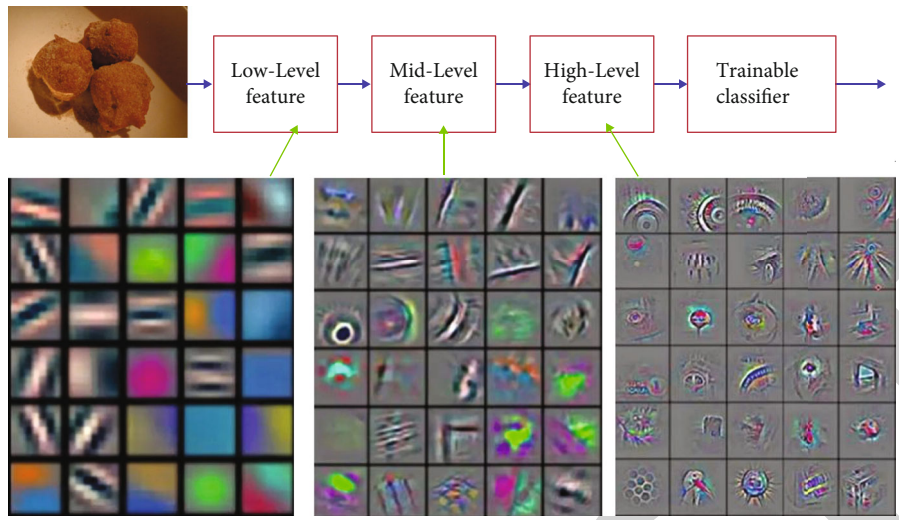


Figure 1: Feature extraction by CNN.

its convergence speed is much faster than those of the sigmoid activation function and the tan*h* activation function. It can also avoid the gradient vanishing that is caused by the sigmoid function and the tan*h* function [20, 21].

The common activation functions include the following:

(1) Sigmoid function

The sigmoid function is the most frequently used continuous and smooth activation function; it is also known as the logistic function. It is used in the output of neurons in a hidden layer, and it can map a real number to the range of (0,1) for binary classification. The formula of the sigmoid function is

$$f(x) = \frac{1}{1+e^{-x}}. \tag{1}$$

The range of Formula (1) is (0,1), and its derivative is

$$f'(x) = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) = f(x)(1-f(x)). \tag{2}$$

In Figure 2, if $x = 10$ or $x = -10$, $f'(x) \approx 0$, and if $x = 0$, $f'(x) = 0.25$.

(2) Tan*h* function

The formula of the tan*h* function is

$$f(x) = \tan h(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{3}$$

The range of Formula (3) is (-1, 1), and its derivative is

$$f'(x) = -(\tan h(x))^{2}. \tag{4}$$

In Figure 3, if $x = 10$ or $x = -10$, $f'(x) \approx 0$, and if $x = 0$, $f'(x) = 1$.

(3) ReLU function

The ReLU function, which is showed in Figure 4, is the most frequently used nonlinear function in neural networks. It is continuous but not smooth, and its formula is

$$f(x) = \max(0, x). \tag{5}$$

The range of Formula (5) is [0,+∞), and its derivative is

$$f'(x) = \begin{cases} 0, & x < 0, \\ 1, & x > 0, \\ \text{undefined}, & x = 0. \end{cases} \tag{6}$$

The ReLU function has the following properties: unilateral inhibition, relatively broad excitement boundary, and sparse activation.

(4) PReLU function

The formula of the PReLU function is

$$f(x) = \max(\alpha x, 0). \tag{7}$$

In Figure 5, parameter α in the PReLU function is not fixed and can be learned during training. Although it ensures that the output result follows a zero-mean distribution, it also activates all feature values in the negative semiaxis. Hence, noises will also be activated, and the final convergence will be affected.

(5) TReLU function

The formula of the TReLU function is

$$f(x) = \begin{cases} x, & x > 0, \\ \tan h(\alpha x), & x \le 0. \end{cases} \tag{8}$$

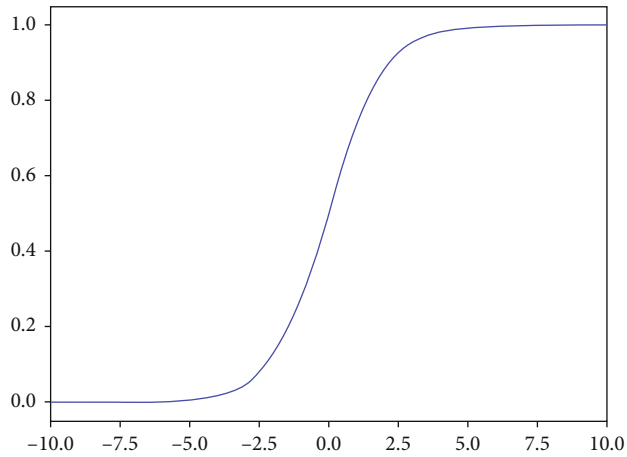


Figure 2: Sigmoid function.

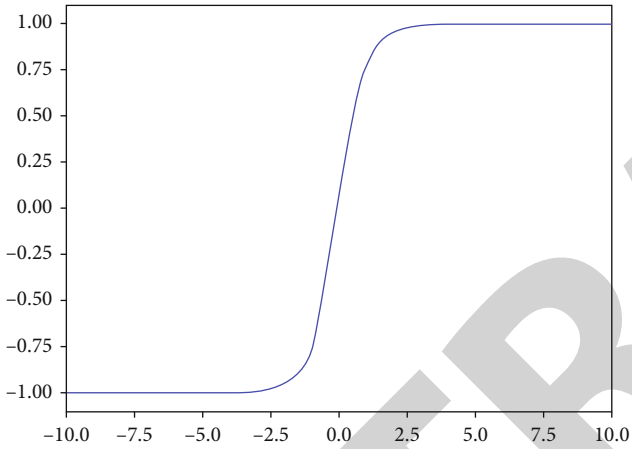


Figure 3: Tan*h* function.

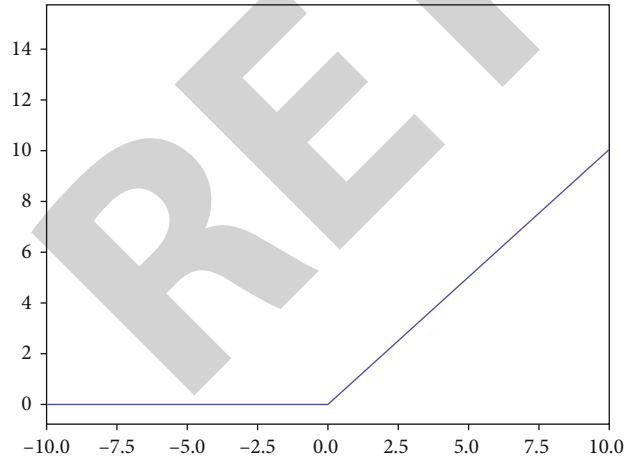


Figure 4: ReLU function.

In Figure 6, parameter α is a variable parameter, and it is used to control the unsaturated region of the function. We set the initial value as 1. The function is almost linear at the origin, and it can yield faster convergence.

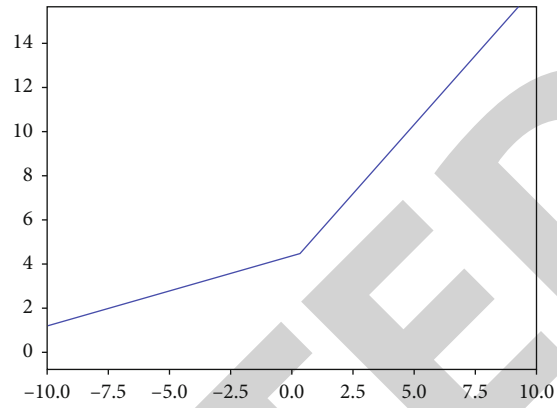


Figure 5: PReLU function ($\alpha = 0.5$).

The TReLU function overcomes the problem of gradient vanishing. As its derivative is always 1 if $x > 0$, the TReLU function is unattenuated if $x > 0$. In addition, the TReLU function preserves some gradient values in the unsaturated region of the negative semiaxis. If the activation value falls in the unsaturated region, it still can be effectively activated, and it preserves some of the effective features while more effectively activating negative value features by controlling the size of the unsaturated region with parameter α.

(6) Leak ReLU function

The formula of the Leak ReLU function, which is showed in Figure 7, is

$$f(x) = \begin{cases} ax, & x < 0, \\ x, & x > 0. \end{cases} \tag{9}$$

The range of Formula (9) is $(-\infty,+\infty)$, and its derivative is

$$f'(x) = \begin{cases} a, & x < 0, \\ 1, & x > 0, \\ \text{undefined}, & x = 0. \end{cases} \tag{10}$$

3.3. Pooling Layer. A pooling layer typically follows a convolutional layer; hence, the output from the convolutional layer is pooled in the pooling layer. The convolutional layer extracts features while the pooling layer reduces the number of parameters. The pooling layer is mainly used to reduce the dimensionality of the features by compressing the number of data and parameters, thereby reducing overfitting and improving the fault tolerance of the model. Although the pooling layer reduces the dimensions of various feature maps, it can still preserve most important information. Located between continuous convolutional layers, the pooling layer reduces the number of data and parameters and reduces overfitting. The pooling layer has no parameters, and it downsamples the result from the previous layer, which is known as data compression. In Figure 8, the

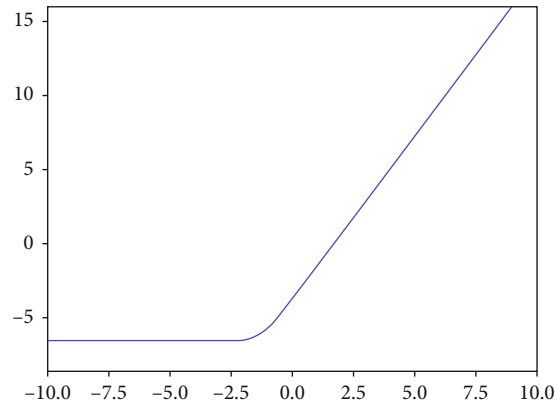


Figure 6: TReLU function ($\alpha = 1$).

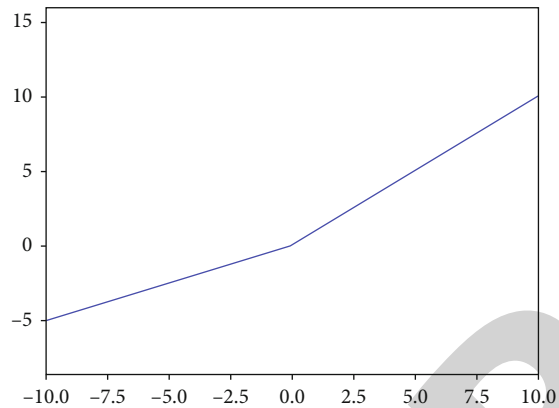


Figure 7: Leak ReLU Function ($a = 0.5$).

downsampling process includes max pooling and mean pooling operations [18, 22].

Max pooling: define a spatial neighborhood, e.g., a window of size $2 * 2$. Extract the largest element from the modified feature map within the window. It has been proven that max pooling outperforms mean pooling

Mean pooling: define a spatial neighborhood, e.g., a window of size $2 * 2$. Calculate the mean value of the modified feature map within the window

The input of every node in the pooling layer is a small block of the previous layer (which is typically a convolutional layer), and the size of this small block is determined by the size of the window of the pooling kernel. The pooling layer changes the size of the node matrix instead of its depth. For image processing, the pooling operation in this layer can be regarded as transforming a high-resolution image into a low-resolution image. After the convolutional layer and pooling layer, the number of parameters in the network model can be further reduced [23].

3.4. Fully Connected Layer. A fully connected layer has many neurons, and it is represented as a column vector (single sample). It is typically one of the latter few layers of a deep neural network in the field of computer vision, and it is used for image classification. In this layer, all neurons are connected via weights, and this layer is typically situated in the rear part of CNN. When the convolutional layers in the front part have extracted weights that are sufficient for recognizing the image, the next task is classification. In the end of CNN, typically, a cuboid is spread into a long vector and sent into the fully connected layer for classification in collaboration with the output layer [24].

A fully connected layer can be transformed into a convolutional layer and vice versa. Any convolutional layer can be converted into a fully connected layer by converting the weight into a huge matrix. In this matrix, most entries are 0, except in designated regions (local perception), and many regions share the same weight (shared weight). Any fully connected layer can also be converted into a convolutional layer.

A fully connected layer works as a "classifier" in the entire CNN. If the convolutional layer, pooling layer, and activation function layer map the original data into the feature space in a hidden layer, the fully connected layer maps the learnt "distributed feature representation" into the sample label space. In practical applications, a fully connected layer can be realized via convolutional computation: a fully connected layer that is fully connected in the previous layer can be converted into a convolution with a $1 * 1$ kernel, and a fully connected layer that has a convolutional layer as its previous layer can be transformed into a global convolution with an $h * w$ kernel, where h and w are the height and width of the convolutional result of the previous layer [25].

The core operation of full connection is to multiply the vectors in the matrix and, in essence, to linearly transform from one feature space to another. In CNN, full connection is typically found in the final few layers, and it calculates a weighted sum of the features that were designed previously. The previous convolution and pooling are similar to feature engineering, while the full connection in the tail part is equivalent to feature weighting [26]. An operation example in the fully connected layer is illustrated in Figure 9.

In Figure 9, the last two columns of small balls represent two fully connected layers. At the end of the last convolution, the final pooling operation is conducted, which outputs 20 images of size $12 * 12$, which are converted into $1 * 100$ vectors by a fully connected layer.

4. CNN Image Recognition and Classification Based on Feature Fusion

The model that is proposed in this paper includes 8 layers: 6 convolutional layers and 2 fully connected layers. The key design details are as follows.

In the feature mapping, nonlinear transformation is conducted via the ReLU activation function [27]. It activates the extracted image features and generates the corresponding output feature mapping [28]. Pooling layers are introduced behind the 1^{st}, 3^{rd}, and 6^{th} convolutional layers, and the max pooling operation is used to perform feature dimension reduction on the output feature mapping. Meanwhile, to improve the training efficiency and classification performance of the network, local normalization is conducted after the convolution operation in every convolutional layer to accelerate the convergence.

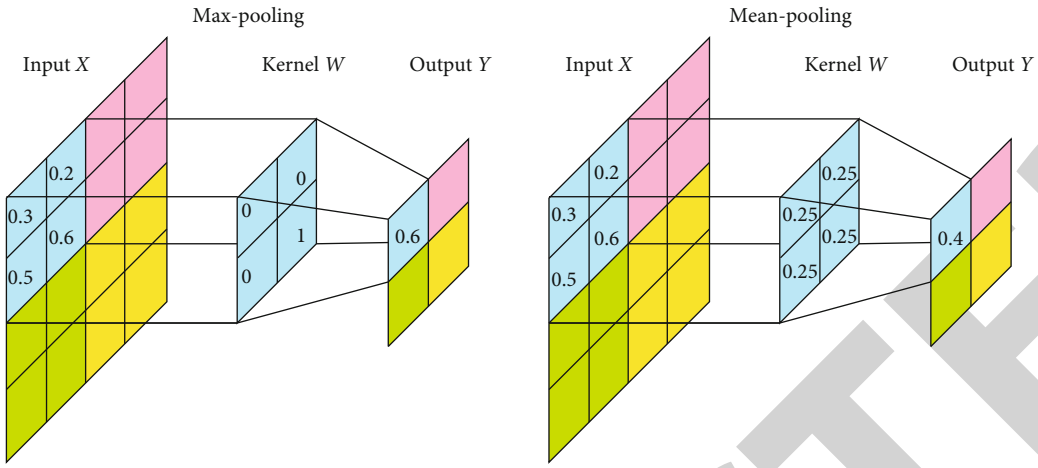


Figure 8: Max pooling and mean pooling.

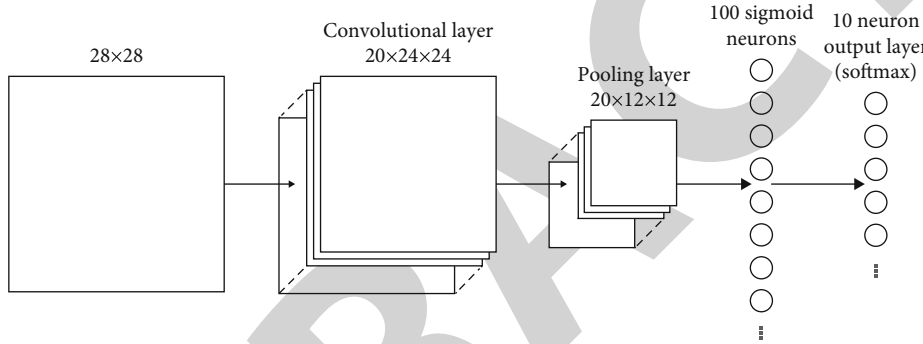


Figure 9: Operations in a fully connected layer.

The 1st convolutional layer is of size $227 * 227 * 3$. The convolution operation is performed by an $11 * 11 * 96$ convolution kernel, and the convolution kernel moves 4 pixels each time, namely, stride = 4. The size of every generated feature mapping matrix is $((227 - 11)/4 + 1)^2 = 55^2$.

Pooling utilizes the sampling operation, and the size of the sampling kernel is $3 * 3$. It slides 2 pixels in the original image each time. After sampling, the size of the generated feature mapping matrix is $((55 - 3)/2 + 1)^2 = 27^2$.

For the 2nd convolutional layer, two pixels are padded on the edges of the input feature mapping matrix, and the size of the mapping matrix becomes $31 * 31$. The convolution operation is performed by a $5 * 5 * 256$ convolution kernel, which moves 1 pixel each time, namely, stride = 1. The size of each generated feature mapping matrix is $((31 - 5)/1 + 1)^2 = 27^2$.

For the 3rd convolutional layer, one pixel is padded on the edges of the input feature mapping matrix, and the size of the mapping matrix becomes $15 * 15 * 1$. The convolution operation is conducted by 256 $3 * 3$ convolution kernels. After each convolution operation, the convolution kernel is moved 1 pixel. The size of each generated feature mapping matrix is $((12 - 3)/1 + 1)^2 = 13^2$.

Pooling utilizes the sampling operation. The size of the sampling kernel is $3 * 3$. The convolution kernel is moved 2 pixels each time after the convolution operation. After sampling, the size of the generated feature mapping matrix is $((27 - 3)/2 + 1)^2 = 13^2$.

For the 4th convolutional layer, one pixel is padded on the edges of the input feature mapping matrix, and the size of the mapping matrix becomes $15 * 15 * 1$. The convolution operation is conducted by 384 $3 * 3$ convolution kernels. After each convolution operation, the convolution kernel is moved 1 pixel. The size of each generated feature mapping matrix is $((15 - 3)/1 + 1)^2 = 13^2$.

For the 5th convolutional layer, one pixel is padded on the edges of the input feature mapping matrix, and the size of the mapping matrix becomes $15 * 15 * 1$. The convolution operation is conducted by $3 * 3 * 256$ convolution kernels. After each convolution operation, the convolution kernel is moved 1 pixel. The size of each generated feature mapping matrix is $((15 - 3)/1 + 1)^2 = 13^2$.

For the 6th convolutional layer, the convolution operation is conducted on the image by a convolution kernel of size 3×3 to realize feature extraction. The extracted image features are activated with the ReLU activation function to produce the corresponding output feature mapping.

Pooling utilizes the sampling operation [29]. The size of the sampling kernel is $3 * 3$. It slides 2 pixels in the original image each time. After sampling, the size of the generated feature mapping matrix is $((13 - 3)/2 + 1)^2 = 6^2$.

Figure 10: Sample image data of Food-101.

This paper adopts the method of cascade and performs feature fusion on the output features after pooling in the 6th layer with the 1st and the 2nd fully connected layers to make the features that are extracted by the network more diverse, expressive, and differential and to improve the classification performance of the network model.

Extract image features using the constructed CNN model are as follows:

Step 1. Denote the pooling output result of the 6th layer as p_8

Step 2. Calculate the outputs of the 1st and 2nd fully connected layers according to the formula $y_j^l = f(y^{l-1} + b_j^l)$ and denote the results as FC_1, FC_2, respectively. Here, l represents the fully connected layer, y^{l-1} is the output result of the layer before the fully connected layers, and b denotes the bias

Step 3. Select p_8, FC_1, and FC_2 as three deep features of the image dataset and prepare for the subsequent feature fusion

Principal component analysis is a multivariate statistical method that transforms scalars into several principal components [30]. These principal components can reflect most of the original information, and they are typically represented as linear combinations of the original variables [31]. To ensure that the information that is contained in these principal components is nonoverlapping, these principal components must be unrelated. Principal component analysis can effectively reduce the dimensions of data and minimize the mean square error between the extracted components and the original data. It can be used in feature extraction. The process of this algorithm is as follows:

(a) Let $X = [X_1, X_2, \cdots, X_p]^T$ be a p-dimensional random vector and let $\mu = E(x)$ and $\Sigma = D(x)$. The corresponding feature vectors to the p feature values of Σ : $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ are $t_1, t_2, \cdots, t_p$, namely,

$$\Sigma t_i = \lambda_i t_i, t_i^T t_i = 1, t_i^T t_j = 0 \ (i \neq j; i, j = 1, 2, \cdots, p). \quad (11)$$

Perform the following linear transformation:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} L_{11} & \cdots & L_{1p} \\ \vdots & \ddots & \vdots \\ L_{n1} & \cdots & L_{np} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} = \begin{bmatrix} L_1^T \\ L_2^T \\ \vdots \\ L_n^T \end{bmatrix} X \quad (n \leq p). \quad (12)$$

If $Y = [Y_1, Y_2, \cdots, Y_n]^T$ is expected to be used to describe $X = [X_1, X_2, \cdots, X_p]^T$, then Y should reflect as much information of vector X as possible, namely, the larger the variance $D(Y_i) = L_i^T \Sigma L_i$ of Y_i, the better the description. In addition, to express the original information as effectively as possible, Y_i and Y_j should not contain repeated content, namely, cov $(Y_i, Y_j) = L_i^T \Sigma L_j = 0$. It can also be proven that if $L_i = t_i$, $D(Y_i)$ has maximum value λ_i and Y_i and Y_j are orthogonal.

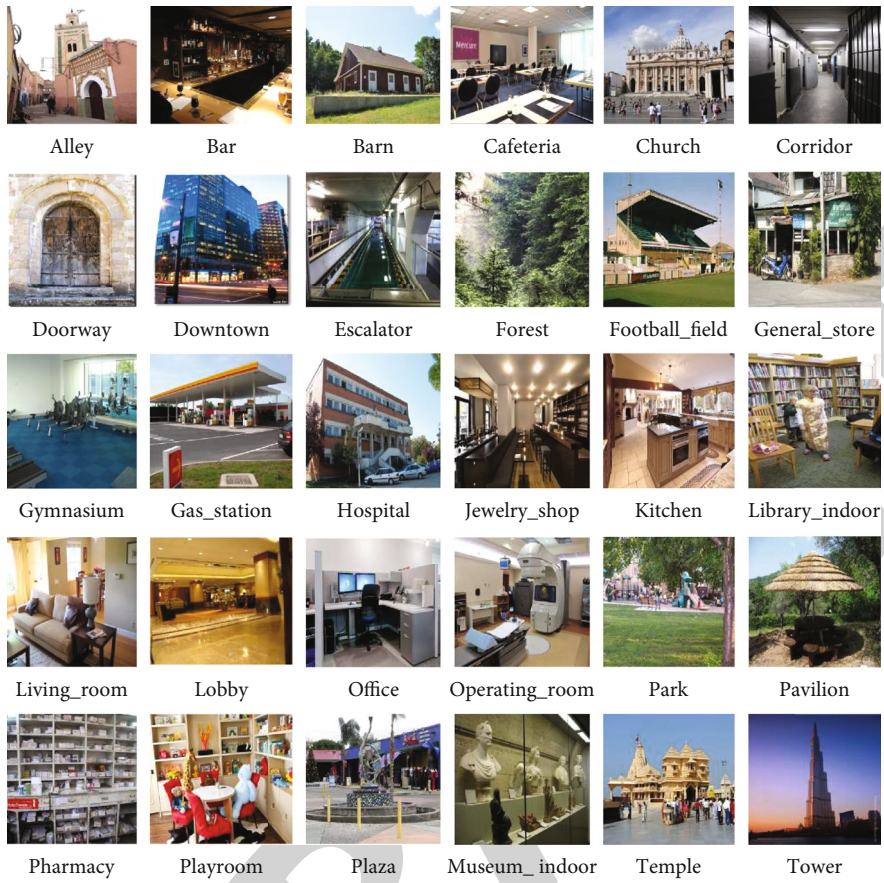


Figure 11: Sample image data of Places2.

Table 1: Accuracies of various activation functions on Food-101 and Places2.

Activation function	Accuracy rate (%)	
	Food-101	Places2
Sigmoid	67.62	45.74
Tan*h*	75.94	53.38
ReLU	81.25	60.58
PReLU	83.44	64.36
TReLU	85.16	68.23

(b) Reconstruct samples from the score matrix

In practice, the covariance matrix of global X is typically unknown and must be estimated from samples. Let $X_1, X_2, \cdots, X_n$ be samples of global X and let $X_i = [X_{i1}, X_{i2}, \cdots, X_{ip}]^T$. Then, the sample measurement matrix is

$$X = \begin{bmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_n^T \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix}. \quad (13)$$

Each row of matrix X corresponds to a sample and each column to a variable. Then, the sample covariance matrix S and the correlation coefficient matrix R are expressed as

$$S = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(X_i - \bar{X}\right)^T = \left(S_{ij}\right), \quad (14)$$

$R = (R_{ij})$,
$R_{ij} = S_{ij}/\sqrt{S_{ii}S_{jj}}$.

Define the score of the jth principal component of sample X_i as $\mathrm{SCORE}(i, j) = XTit_j$. It is expressed as follows in matrix form:

$$\mathrm{SCORE} = \begin{bmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_N^T \end{bmatrix} \left[t_1, t_2, \cdots, t_p\right] = XT, \quad (15)$$

Invert Formula (15) and reconstruct the original samples from the score matrix:

$$X = \mathrm{SCORE} \cdot T^{-1} = \mathrm{SCORE} \cdot T^T. \quad (16)$$

Typically, principal component analysis only uses the first m principal components to approximate the original samples.

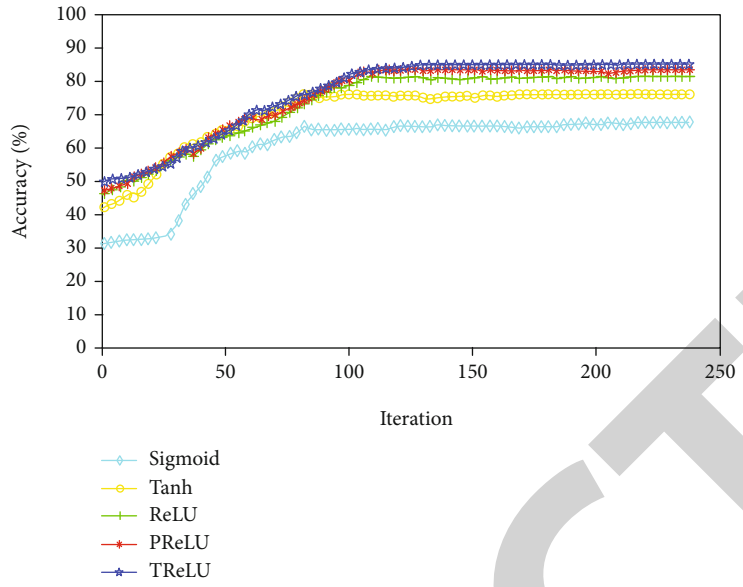


Figure 12: Comparison of five activation functions on Food-101.

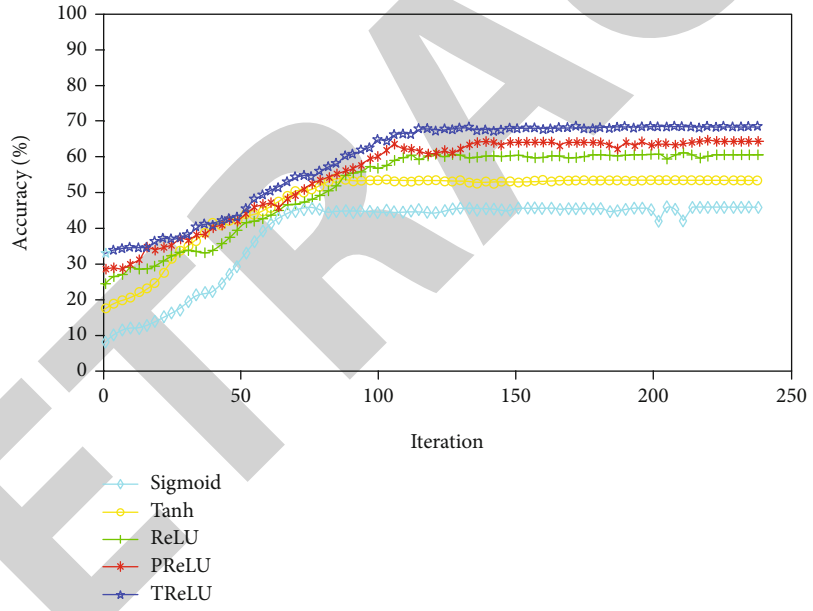


Figure 13: Comparison of five activation functions on Places2.

Table 2: Training times of five activation functions on Food-101 and Places2.

Activation function	Time (h)	
	Food-101	Places2
Sigmoid	2.6	3.5
Tan*h*	2.3	3.2
ReLU	1.9	2.5
PReLU	2.1	2.9
TReLU	2.2	3.0

Table 3: Classification accuracies on Food-101 and Places2 of four algorithms.

Method	Accuracy rate (%)	
	Food-101	Places2
NIN	85.62	66.90
DSN	88.18	69.53
DBN	88.53	70.34
Method of this paper	89.47	71.56

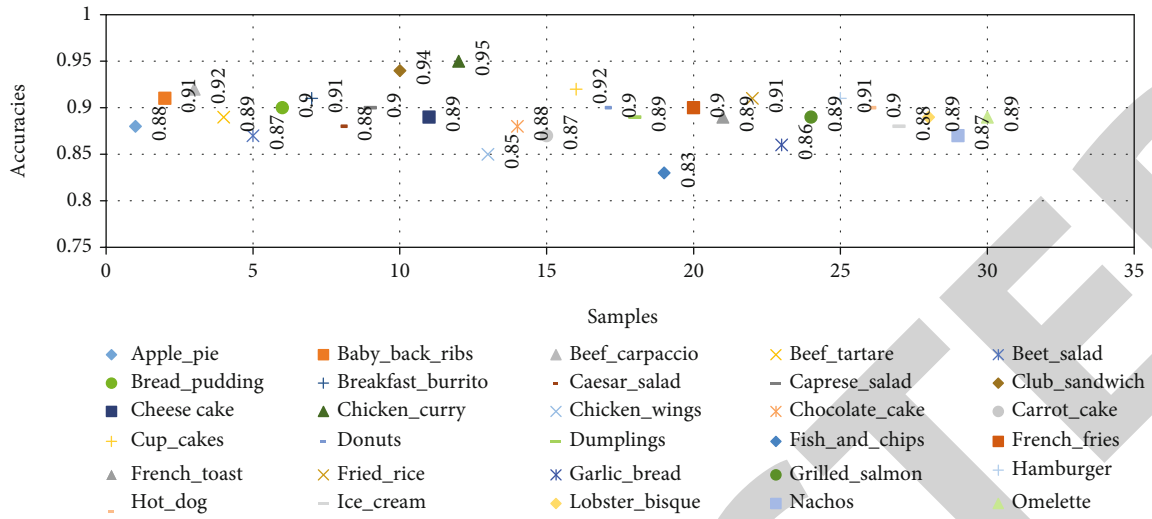


Figure 14: Accuracies on samples of the Food-101 dataset.

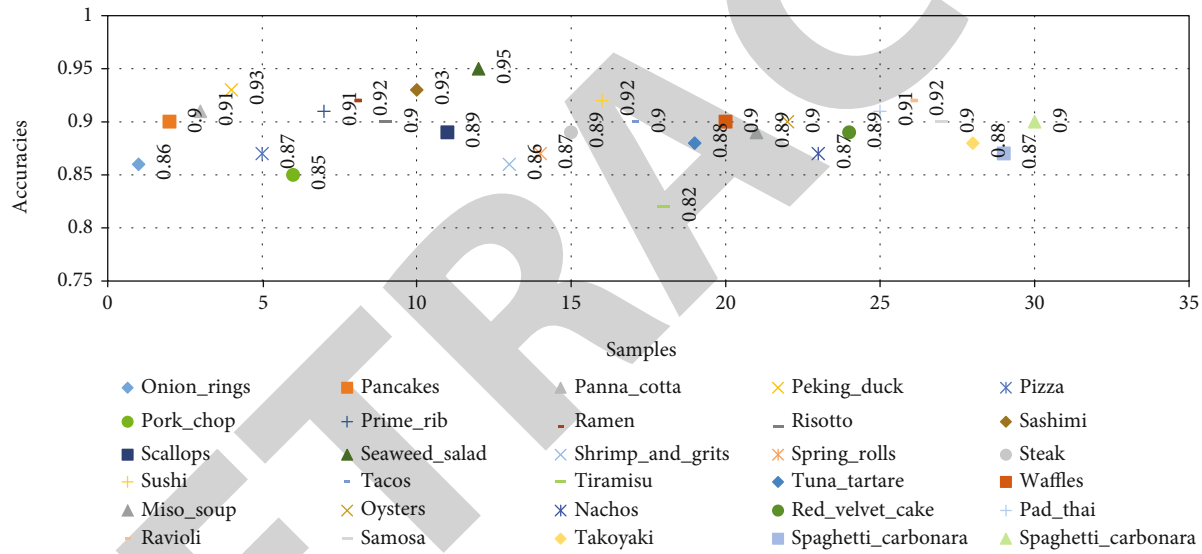


Figure 15: Accuracies on samples of the Food-101 dataset.

5. Analysis of the Experimental Result

5.1. Experimental Environment and Testing Datasets. The experimental environment that is utilized in this paper includes the following: CPU: Intel(R) Core(TM) i7-6700HQ CPU@2.60 GHz; GPU: NVIDIA GeForce GTX 1070; the physical memory (RAM): 16.0 G; and a PC with the deep learning framework of TensorFlow.

To examine the classification performance of the proposed CNN model, experiments are conducted on two image datasets: Food-101 and Places2. Food-101 is an image dataset that contains images of food. It includes 101 classes of food (western cuisine), and each class has 1000 images, which are used to automatically recognize the class of gourmet. Places2 is an image dataset of scenarios. It contains 10 million images from over 400 classes of scenarios, and it is used for visual cognition tasks with scenarios and environments as the application contents. Figures 10 and 11 show sample image data from Food-101 and Places2.

5.2. Accuracy Comparison of Activation Functions. In the study of image classification and recognition, activation functions are highly important for CNN models. Through the nonlinear mapping of an activation function, CNN can realize stronger feature representation performance for handling more complex classification problems. This paper uses the TReLU activation function to improve the classification performance of the CNN model.

To evaluate the performance of the TReLU activation function in boosting the classification performance and based on the CNN that is designed in this paper, comparison experiments are conducted on Food-101 and Places2 using the TReLU activation function and other common activation functions. Food-101 and Places2 include many classes and

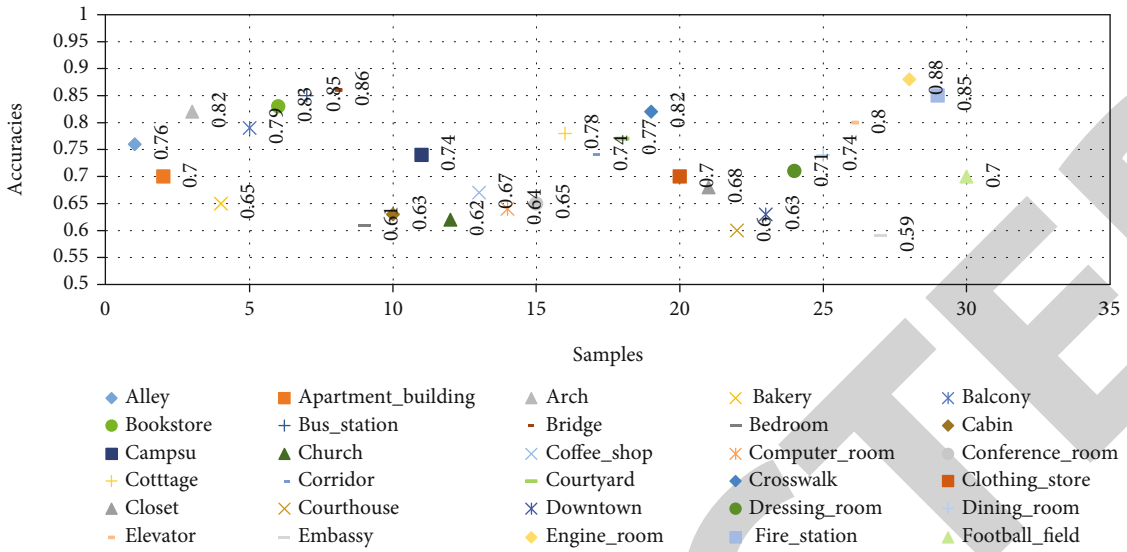


Figure 16: Accuracies on samples of the Places2 dataset.

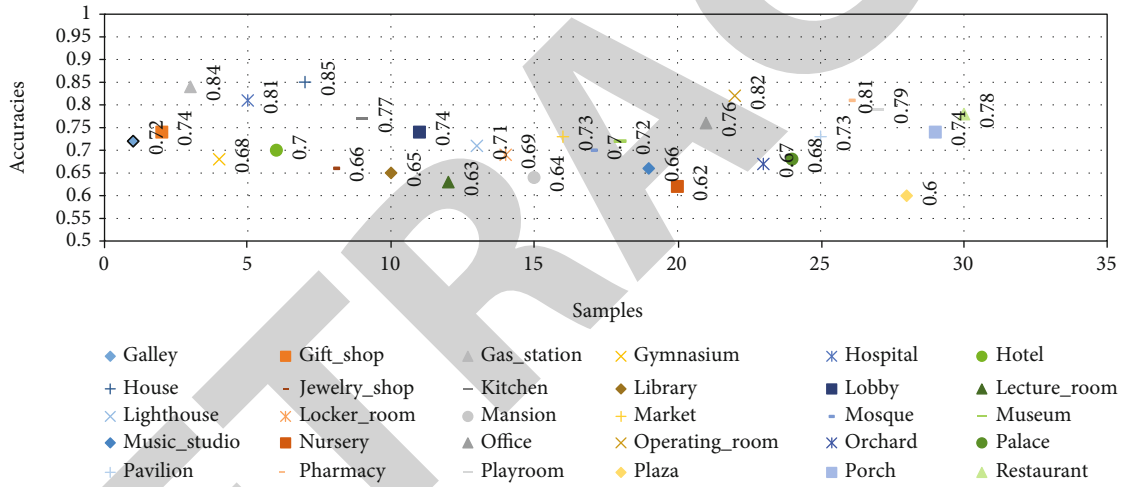


Figure 17: Accuracies on samples of the Places2 dataset.

are of high classification difficulty; see the experimental results in Table 1.

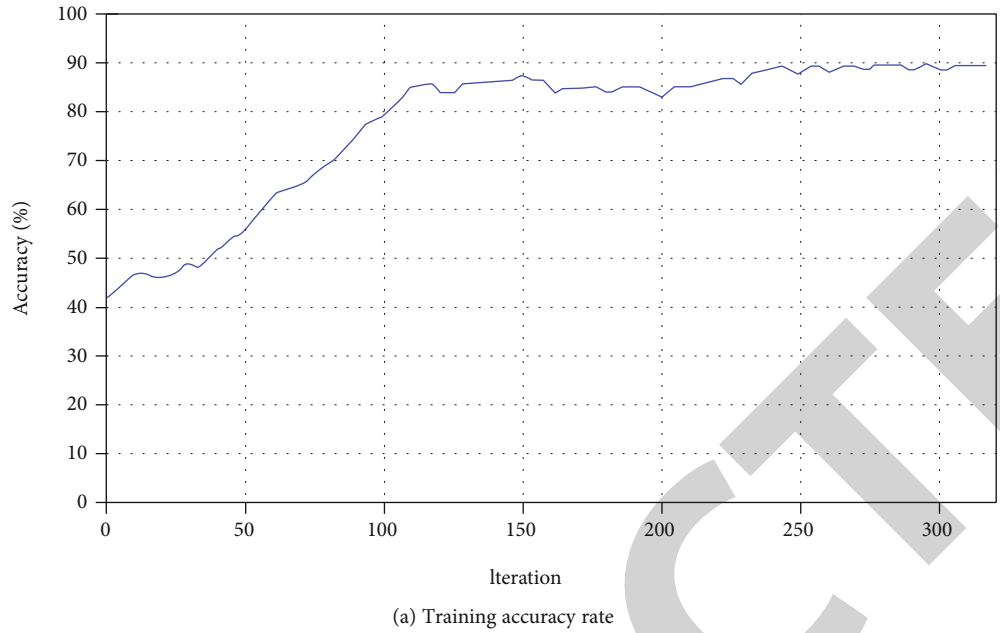
According to the experimental results in Table 1, the unsaturated nonlinear activation functions (e.g., ReLU) realize lower error rates than the saturated nonlinear activation functions (e.g., sigmoid), thereby suggesting that activation functions that are similar to biological neurons improve the classification performance.

In terms of the classification accuracy, the saturated nonlinear activation functions, namely, the sigmoid function and the tan$h$ function, are outperformed by the unsaturated nonlinear activation functions, namely, ReLU, PReLU, and TReLU; hence, the activation functions that approximate biological neurons can improve the classification accuracy. The TReLU activation function exhibits excellent classification performance on the complex datasets, namely, Food-101 and Places2, and it outperforms the other functions, which further proves that the TReLU activation function can improve the classification performance of the CNN model and yield excellent generalization performance. Figures 12 and 13 compare the classification performances of the CNN model under the five considered activation functions more vividly. The TReLU activation function not only realizes higher classification accuracy, but also has higher convergence speed than those of the other functions.

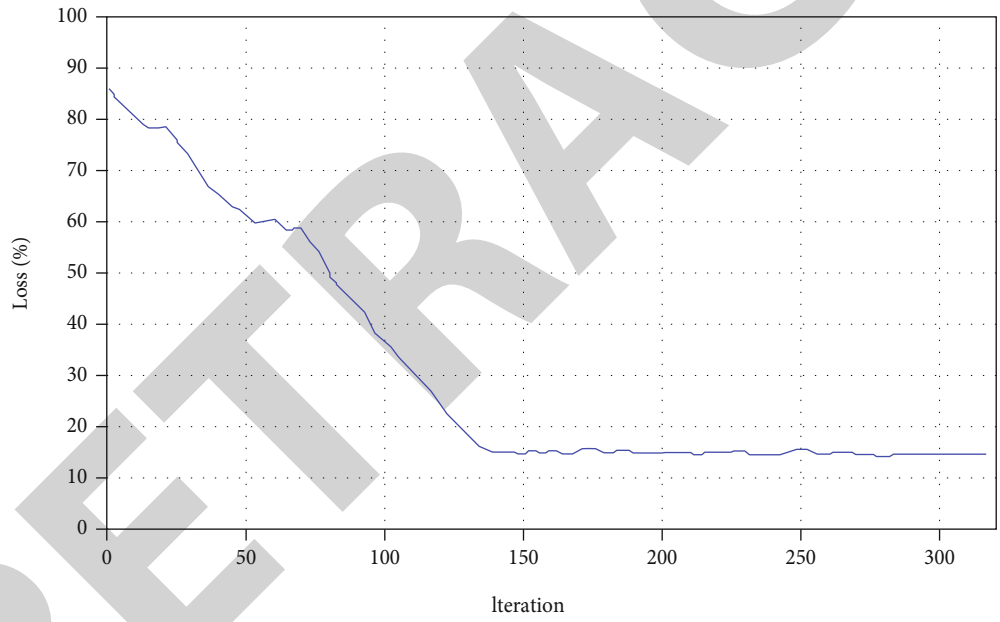
The experimental results also include the training times that were required by the five activation functions on Food-101 and Places2, which are shown in Table 2.

According to the experimental results in Table 2, the TReLU activation function requires almost the same training time as the tan$h$ function. This is acceptable. The key result is that the TReLU activation function can enhance the accuracy and further increase the convergence speed. The operation of activation function is expressed as Formula (17).

$$x_j^l = f\left(\sum_{i \in N_j} W_{ij}^l * x_i^{l-1} + b_j^l\right), \tag{17}$$



(a) Training accuracy rate



(b) Training loss rate

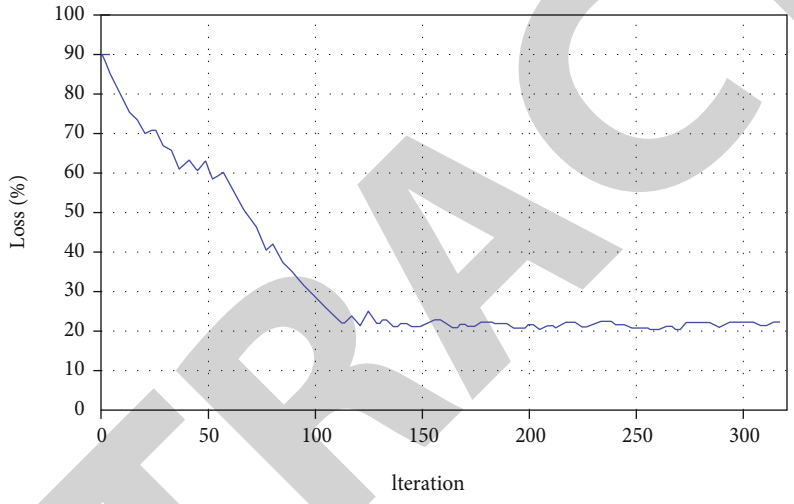
Figure 18: Accuracy and loss on Food-101 for the algorithm that is proposed in this paper.

where x_j^l represents the jth feature map of the lth layer; W_{ij}^l represents the connection weight of the ith feature map in the $l-1$th layer and the jth feature map in the lth layer; $*$ represents the convolution operation; b_j^l represents the bias; and N_j represents the total number of input feature maps.

5.3. Analysis of the Experimental Results. To evaluate the classification performance of the CNN model that is designed in this paper, which is based on deep feature fusion, experiments have been conducted on two image datasets, namely, Food-101 and Places2, and the results are compared with those of other image classification methods. Table 3 lists the classification accuracies. The recognition accuracies for each action category in the Food-101 and Places2 datasets are shown in Figures 14–17.

According to the experimental results in Table 3, the proposed method can effectively improve the classification performance of the network model, and its classification accuracy is higher than those of the other algorithms. Network in network (NIN) is the predecessor of inception. It expands the $1 * 1$ convolution kernel behind the convolutional layer and replaces the fully connected layer by a global average pooling layer to reduce the number of training parameters and to effectively avoid overfitting. DSN differs from other traditional deep learning frameworks. It contains a deep network that is a deep set of networks, each of which

(a) Training accuracy rate



(b) Training loss rate

Figure 19: Accuracy and loss on Places2 for the algorithm that is proposed in this paper.

has its own hidden layer. DSN enables isolated training of each module for training in parallel; hence, it has high efficiency. Supervised training realizes back-propagation in each module, instead of over the entire network. DBN is composed of a multilayer unsupervised restricted Boltzmann machine (RBM) network and a one-layer supervised back-propagation (BP) network, and its training includes pretraining and fine-tuning.

The proposed CNN model is based on deep feature fusion, facilitates training and optimization of the network model via of dimension reduction, utilizes a local normalization operation to expedite the network training, and improves its classification performance. Moreover, it effectively fuses the deep features and enables the network to extract as much useful feature information as possible. Via this approach, the classification performance of the model is enhanced. Figures 18 and 19 plot the training convergence of the proposed algorithm on Food-101 and Places2, followed by the training accuracy and loss curves.

It is clearly evident from the Figure 18, in the training on Food-101 by the network model that is proposed in this paper, when the training is iterated to the 150^{th} generation, the training loss stabilizes and reaches the convergence state. At this time, the accuracy is 89%. As indicated in Figure 19, when the network trains on Places2, its classification performance is still very high, even though Places2 has higher classification complexity. When the training is iterated to the 125^{th} generation, the network has arrived at the convergence state and the accuracy is 69%.

Through the above experimental results, it can be observed that the proposed method can realize satisfactory image classification performance. In Food-101, the proposed model realizes an accuracy of 89.47%, which is 3.85%, 1.29%, and 0.94% higher than those of NIN, DSN, and DBN, respectively, and its classification and recognition performances are also satisfactory. In Places2, its accuracy is 71.56%, which is 4.66%, 2.03%, and 1.22% higher than those of NIN, DSN, and DBN, respectively, and it outperforms these methods on classification and recognition.

6. Conclusions

The arrival of the Internet of Things is accompanied by a large number of multimedia data. The key problems to be

handled by image classification and recognition are to identify and classify the target objects that are contained in the image regions of interest and to make judgments. With the properties of local connection and shared weight, it has stronger robustness in its invariance to translation, rotation, and scaling of the input image data space and realizes stronger image classification and recognition performances. Facilitated by the cascade method, this paper has effectively fused the deep features of CNN, reduced the dimensions of the features using the PCN algorithm, and made the extracted features more typical and diverse to strengthen its classification performance. It has also introduced local normalization after every convolutional layer to accelerate the convergence. The experimental results demonstrate that the proposed algorithm has stabilized and expedited the network training, thereby leading to higher classification performance and accuracy.

Data Availability

The simulation experiment data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (2018YFB1402600) and the National Science Foundation of China (Grant Nos. 61772190, 61672221, and 61702173).

References

[1] J. P. CobeñaCevallos, J. M. AtienciaVillagomez, and I. S. Andryshchenko, "Convolutional neural network in the recognition of spatial images of sugarcane crops in the Troncal region of the coast of Ecuador," *Procedia Computer Science*, vol. 150, no. 10, pp. 757–763, 2019.

[2] Ş. Öztürk and B. Akdemir, "HIC-net: a deep convolutional neural network model for classification of histopathological breast images," *Computers & Electrical Engineering*, vol. 76, no. 6, pp. 299–310, 2019.

[3] H. T. Mustafa, J. Yang, and M. Zareapoor, "Multi-scale convolutional neural network for multi-focus image fusion," *Image and Vision Computing*, vol. 85, no. 5, pp. 26–35, 2019.

[4] B. A. ŞabanÖztürk, "Effects of histopathological image preprocessing on convolutional neural networks," *Procedia Computer Science*, vol. 132, no. 7, pp. 396–403, 2018.

[5] N. Sharma, V. Jain, and A. Mishra, "An analysis of convolutional neural networks for image classification," *Procedia Computer Science*, vol. 132, no. 5, pp. 377–384, 2018.

[6] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, no. 11, pp. 120–147, 2018.

[7] C. Bai, L. Huang, X. Pan, J. Zheng, and S. Chen, "Optimization of deep convolutional neural network for large scale image retrieval," *Neurocomputing*, vol. 303, no. 16, pp. 60–67, 2018.

[8] F. Zhang, N. Cai, G. Cen, F. Li, and X. Chen, "Image super-resolution via a novel cascaded convolutional neural network framework," *Signal Processing: Image Communication*, vol. 63, no. 4, pp. 9–18, 2018.

[9] H. Liang, J. Zou, K. Zuo, and K. Muhammad Junaid, "An improved genetic algorithm optimization fuzzy controller applied to the wellhead back pressure control system," *Mechanical Systems and Signal Processing*, vol. 142, article 106708, 2020.

[10] Y. Duan, F. Liu, L. Jiao, P. Zhao, and L. Zhang, "SAR image segmentation based on convolutional-wavelet neural network and Markov random field," *Pattern Recognition*, vol. 64, no. 4, pp. 255–267, 2017.

[11] S. Zagoruyko and N. Komodakis, "Deep compare: a study on using convolutional neural networks to compare image patches," *Computer Vision and Image Understanding*, vol. 164, no. 11, pp. 38–55, 2017.

[12] S. Yu, S. Jia, and C. Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing*, vol. 219, no. 5, pp. 88–98, 2017.

[13] Z. Huang, X. Xu, J. Ni, H. Zhu, and C. Wang, "Multimodal representation learning for recommendation in Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10675–10685, 2019.

[14] U. Raghavendra, H. Fujita, S. V. Bhandary, A. Gudigar, and U. R. Acharya, "Deep convolution neural network for accurate diagnosis of glaucoma using digital fundus images," *Information Sciences*, vol. 441, no. 5, pp. 41–49, 2018.

[15] Z. Liu, B. Hu, B. Huang, L. Lang, H. Guo, and Y. Zhao, "Decision optimization of low-carbon dual-channel supply chain of auto parts based on smart city architecture," *Complexity*, vol. 2020, Article ID 2145951, 14 pages, 2020.

[16] A. Sharma, X. Liu, X. Yang, and D. Shi, "A patch-based convolutional neural network for remote sensing image classification," *Neural Networks*, vol. 95, no. 11, pp. 19–28, 2017.

[17] R. DonidaLabati, A. Genovese, E. Muñoz, and F. S. Vincenzo Piuri, "A novel pore extraction method for heterogeneous fingerprint images using convolutional neural networks," *Pattern Recognition Letters*, vol. 113, no. 10, pp. 58–66, 2018.

[18] H. Kandi, D. Mishra, and S. R. K. S. Gorthi, "Exploring the learning capabilities of convolutional neural networks for robust image watermarking," *Computers & Security*, vol. 65, no. 3, pp. 247–268, 2017.

[19] Y. Zhang, R. Zhu, Z. Chen, J. Gao, and D. Xia, "Evaluating and selecting features via information theoretic lower bounds of feature inner correlations for high-dimensional data," *European Journal of Operational Research*, 2020.

[20] E. R. S. de Rezende, G. C. S. Ruppert, A. Theóphilo, E. K. Tokuda, and T. Carvalho, "Exposing computer generated images by using deep convolutional neural networks," *Signal Processing: Image Communication*, vol. 66, no. 8, pp. 113–126, 2018.

[21] A. Haidar, R. J. S. Almubarak, R. Long, S. Antani, and S. R. Frazier, "Convolutional neural network based localized classification of uterine cervical cancer digital histology images," *Procedia Computer Science*, vol. 114, no. 3, pp. 281–287, 2017.

[22] C. Barat and C. Ducottet, "String representations and distances in deep convolutional neural networks for image classification," *Pattern Recognition*, vol. 54, no. 6, pp. 104–115, 2016.

[23] N. Kumar, R. Verma, and A. Sethi, "Convolutional neural networks for wavelet domain super resolution," *Pattern Recognition Letters*, vol. 90, no. 15, pp. 65–71, 2017.

[24] H. Liang, A. Xian, M. Min Mao, P. Ni, and H. Wu, "A research on remote fracturing monitoring and decision-making method supporting smart city," *Sustainable Cities and Society*, vol. 62, article 102414, 2020.

[25] P. Witoonchart and P. Chongstitvatana, "Application of structured support vector machine backpropagation to a convolutional neural network for human pose estimation," *Neural Networks*, vol. 92, no. 8, pp. 39–46, 2017.

[26] A. Jalali, R. Mallipeddi, and M. Lee, "Sensitive deep convolutional neural network for face recognition at large standoffs with small dataset," *Expert Systems with Applications*, vol. 87, no. 30, pp. 304–315, 2017.

[27] D. Tomè, F. Monti, L. Baroffio, L. Bondi, and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," *Signal Processing: Image Communication*, vol. 47, no. 9, pp. 482–489, 2016.

[28] H. Choi and K. H. Jin, "Fast and robust segmentation of the striatum using deep convolutional neural networks," *Journal of Neuroscience Methods*, vol. 274, no. 1, pp. 146–153, 2016.

[29] H. Zheng, W. Guo, and N. Xiong, "A kernel-based compressive sensing approach for mobile data gathering in wireless sensor network systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 12, pp. 2315–2327, 2018.

[30] H. Liang, D. Zou, Z. Li, M. J. Khan, and Y. Lu, "Dynamic evaluation of drilling leakage risk based on fuzzy theory and PSO-SVR algorithm," *Future Generation Computer Systems*, vol. 95, pp. 454–466, 2019.

[31] Y. Zhou, D. Zhang, and N. Xiong, "Post-cloud computing paradigms: a survey and comparison," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 714–732, 2017.