Journal of Computer and Communications, 2025, 13(3), 176-194
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Combination Method of Stacked Convolutional Auto-Encoder and Selective Kernel Attention Mechanism for Image Classification

Shuangshuang Chen[1], Na Jin[2], Ning Li[2], Wei Guo[1]

[1]College of Information Engineering, Xinchuang Software Industry Base, Yancheng Teachers University, Yancheng, China
[2]Yancheng Agricultural College, Yancheng, China
Email: chenss@yctu.edu.cn

How to cite this paper: Chen, S.S., Jin, N., Li N. and Guo, W. (2025) A Combination Method of Stacked Convolutional Auto-Encoder and Selective Kernel Attention Mechanism for Image Classification. *Journal of Computer and Communications*, **13**, 176-194.
https://doi.org/10.4236/jcc.2025.133012

Received: February 28, 2025
Accepted: March 24, 2025
Published: March 28, 2025

Abstract

Convolutional auto-encoders have shown their remarkable performance in stacking deep convolutional neural networks for classifying image data during the past several years. However, they are unable to construct the state-of-the-art convolutional neural networks due to their intrinsic architectures. In this paper, we have proposed an effective stacked convolutional auto-encoder that integrates a selective kernel attention mechanism for image classification. This model is based on a fully convolutional auto-encoder and can be trained end-to-end. It consists of two parts: encoder and decoder. The encoder and decoder are composed of convolutional layer chain and deconvolution layer chain, respectively. The proposed method consists of three main modifications. First, a selection kernel (SK) convolution module and a selection kernel deconvolution module are constructed to form convolutional layer chain and deconvolution layer chain. Second, to solve the problem of network degradation, the idea of residual networks is cited to add skip layer connections between the SK convolution module and the symmetrically connected SK deconvolution module. Third, to alleviate the overfitting of the model, a method of adding noise in data augmentation is used to improve the generalization ability of the model. The experimental results show that this method can effectively integrate the channel attention module and the fully convolutional autoencoder. Although it is an unsupervised feature learning model, it can still achieve good classification results.

Keywords

Image Classification, Unsupervised Feature Learning, Auto-Encoder

1. Introduction

In recent years, Deep Neural Network (DNN) has been widely used in the field of computer vision due to their powerful feature extraction capabilities. Among them, Convolutional Neural Network (CNN) is the classic model in DNN. CNN possesses the following advantages [1]: 1) Local connections; 2) Weight sharing; 3) Down-sampling dimension reduction. Deep CNN can significantly improve the accuracy of image classification. However, deep CNN usually requires significant computational resources during training, as well as training a large number of parameters and labeled data. In contrast, unlabeled data is not only easy to obtain but also typically larger in scale. How to better utilize them to improve the performance of feature learning algorithms is still a topic worth studying. Hence, many scholars began to study unsupervised DNN, which attempts to learn deep feature representations from the data itself without knowing the data label. A typical unsupervised DNN is stacked Auto-Encoder (AE), which is formed by stacking shallow AE modules. The input layer, hidden layer, and output layer form the AE module, where the number of neurons in the input layer is the same as the number of neurons in the output layer. Usually, the conversion from the input layer to the hidden layer is called an encoder, and the conversion from the hidden layer to the output layer is called a decoder. The encoder extracts features from the input data, while the decoder reconstructs the input data from the features. We can train an AE module by minimizing the difference between input data and reconstructed data. Stacked AE is composed of multiple trained AE modules stacked together, which can be used to learn hierarchical feature representations. Although stacked AE can learn hierarchical features from complex image data, it is necessary to convert the input image into vector form in advance, which will change the inherent structure of the image and thus destroy the neighboring relationships between image pixels [2] [3]. To address this problem, the paper [4] proposed a CAE (Convolution Auto-encoder) structure that can input 2D image data. In this literature, the encoder consists of a convolutional layer and a pooling layer, while the decoder consists of only one deconvolution layer. Compared with stacked AE, CAE can retain more image structural information. In this CAE structure, convolutional layers are used to encode image content and can be used as encoders; The deconvolution layer is used to restore abstract image features into concrete image content details and can be used as a decoder. Another type of CAE architecture operates by first extracting small patches from the input image or the feature map output from the previous layer, then training a basic AE (such as DAE, SAE) using a layer-wise training rule, and subsequently converting the learned weights into convolutional kernels [5]. Finally, the learned and trained convolution kernels are convolved with the input image to obtain image features. This CAE structure can also preserve the local correlation of the image. However, when using layer by layer training rules to achieve feature transformation in this CAE structure, it can be time-consuming and laborious when dealing with neural

networks with deeper layers. To address this issue, this paper adopts a fully convolutional auto encoder (FCAE) structure [6], which can be trained end-to-end. Compared with traditional stacked AE, FCAE utilizes the two-dimensional local structure of the input image to reduce parameter redundancy. In addition, unlike traditional stacked AE, its weights are shared by all positions in each feature map to maintain spatial locality.

Some researchers have found that as the number of layers in the FCAE network increases, two problems may arise: 1) as the number of convolutional layers increases, a large amount of image details may be lost or damaged. Furthermore, it remains to be determined whether the details of an abstract image content can be restored through a decoder; 2) In terms of optimization, deep networks often encounter the problem of vanishing gradients, which makes train more difficult. Inspired by the deep residual network [7], the authors [8] use a skip layer structure to connect two corresponding convolutional and deconvolution layers. These skip layer structures can divide the FCAE network into several modules, which can backpropagate gradients to the bottom layer and transmit specific details of the image to the top layer. These two features make the end-to-end training of the network from input image to output image easier and more effective, resulting in improved performance while increasing the number of layers in the FCAE network. However, this work only applies this skip layer structure of FCAE to low-level image processing tasks for image restoration, and has not been applied to high-level image processing tasks such as image classification.

In recent years, attention mechanisms have been widely used in the construction of deep learning models in various application fields such as image processing [9], speech recognition [10] and natural language processing [11]. Essentially, similar to human selective visual attention mechanisms, the key objective of attention mechanisms in deep learning is to select information closely related to the current task from a large amount of information. In the field of image classification, the authors [12] successfully combined deep CNN with visual attention mechanism for the first time, achieving better classification results. The paper [13] points out that in the field of neuroscience, the size of stimuli will affect the adjustment of the receptive field size of visual cortex neurons. In the entire process of building neural networks, people should also set the size of convolution kernels according to different stimuli. However, when building CNNs, only one type of convolution kernel is usually used in the same layer, and the effectiveness of using multiple convolution kernels is rarely considered. Therefore, this literature elucidates that each neuron can adaptively adjust the size of its receptive field (convolution kernel) based on input information by introducing attention mechanisms, and constructs a module called Selective Kernel (SK) convolution. On the basis of references [12] [13], we have further expanded and formed the SK deconvolution module, which combines the SK convolution module with FCAE to propose a stacked convolutional autoencoder that integrates the selection kernel attention mechanism.

2. The Overall Framework

The stacked convolutional autoencoder with fusion selection kernel attention mechanism is an unsupervised deep network that generates advanced feature representations. Figure 1 illustrates the overall structure and process of the method proposed in this paper (for ease of illustration, the encoder and decoder in the figure are illustrated using a 4-layer convolution module and deconvolution module as examples. In addition, for the sake of simplicity in illustration, the max pooling layer after the con-volution module in the encoder has been omitted. "LeRU + dropout" indicates that the hidden layer uses LeRU activation function and dropout regularization. "Softmax" indicates the classification layer).

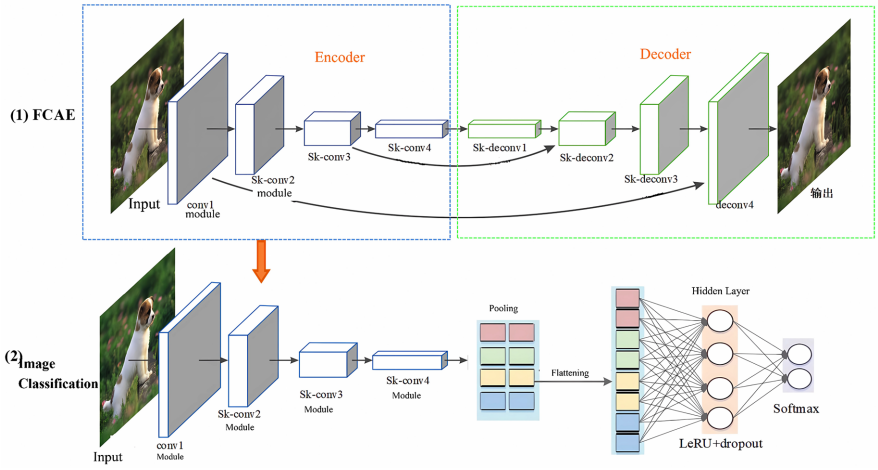

Figure 1. The structural flowchart of the method proposed in this paper.

3. Stacked Convolutional Autoencoder with Fusion Selection Kernel Attention Mechanism

3.1. Construction of Network Structure

In this paper, the stacked convolutional autoencoder network structure constructed with fusion selection kernel attention mechanism is based on FCAE, which consists of an encoder and a decoder. It can be shown in Figure 2(a). The encoder part is composed of convolutional layer chains, using SK convolution to form the convolutional layer. Using an encoder as a feature extractor can preserve the main components of objects in an image while also eliminating noise in the image. During this process, subtle details of the image content may be lost. The decoder part is used to restore the details of the image content, and the output image is a reconstructed version of the input image. The decoder is composed of an anti-convolutional layer corresponding to the convolutional layer. We have added skip layer connections between the SK convolution module and the symmetric connected SK deconvolution module, summed the deconvolution feature map with the passed convolution feature map by element, and then input it into the next layer of the network. Add batch normalization layer and Leaky ReLU layer to each convolutional layer and deconvolution layer, as shown in Figure

2(b). The input image and output image of this network have the same size.

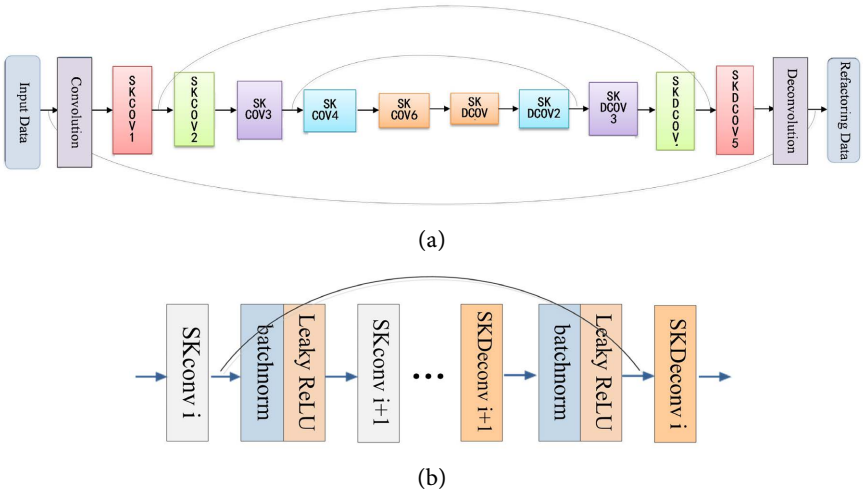

Figure 2. (a) Network structure diagram; (b) Convolution module and deconvolution module.

Encoder: In the network structure constructed in this paper, the encoder uses SK convolution instead of traditional convolution operations. In traditional CNN networks, the receptive field size of each layer of neurons is the same. In neuroscience, the stimulation mechanism determines the size of the receptive field of visual neurons, but this factor is rarely considered in CNN [13]. However, SK convolution can enable neurons to adaptively adjust the size of their receptive field for input information of different sizes. In order to adjust the receptive field to different sizes, SK convolution uses softmax to fuse features from different kernels. SK convolution includes three operations: Split, Fuse, and Select [13].

Split: For any given feature map $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}$, by default we first conduct two transformations $\tilde{F} : \mathbf{X} \to \tilde{\mathbf{U}} \in \mathbb{R}^{H \times W \times C}$ and $\hat{F} : \mathbf{X} \to \hat{\mathbf{U}} \in \mathbb{R}^{H \times W \times C}$ with kernel sizes k_1 and k_2 respectively. The height, width, and number of channels of the feature map after convolution transformation are H, W, C. In the process of transformation, both $\tilde{F}$ and $\hat{F}$ are composed of efficient grouped convolutions, Batch Normalization and LReLU function in sequence.

Fusion: The fundamental principle of this operation is to control the information flow from multiple branches through a Gate mechanism. Information at different scales is carried by these branches to the neurons in the next layer. Fusing the information from these branches requires the implementation of a gate mechanism.

This operation first performs a simple pixel-level addition fusion, as shown in Equation (1):

$$\mathbf{U} = \tilde{\mathbf{U}} + \hat{\mathbf{U}} \tag{1}$$

Then, global average pooling is used to encode global information, thereby deriving statistical analysis information for each channel. Among them, the c-th element Sc in s is obtained through compression calculation on the $H \times W$ dimen-

sion of **U**, and the calculation method is shown in Equation (2):

$$s_c = F_{gp}\left(\mathbf{U}_c\right) = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}\mathbf{U}_c\left(i, j\right) \tag{2}$$

Furthermore, to achieve precise control and automatic selection, a fully connected layer is used to generate a compact feature that enables precise and adaptive feature selection, reducing dimensionality while improving efficiency. This feature, denoted as **z**, is a vector used to determine the final weights, and its calculation method is shown in formula (3):

$$\mathbf{z} = F_{fc}\left(\mathbf{s}\right) = \delta\left(B\left(\mathbf{Ws}\right)\right) \tag{3}$$

where δ represents the Leaky ReLU function, B denotes the batch normalization operation, and $\mathbf{W} \in \mathbb{R}^{d \times C}$. In the paper [13], the **Ws** operation is implemented by setting the convolution kernel to 1 × 1. In this paper, it is implemented through a fully connected layer with d is set to $\frac{C}{2}$. The fusion operation can combine information from multiple channels to obtain a global representation for weight selection.

Under the guidance of the compact feature descriptor **z**, the cross-channel soft attention mechanism is used to adaptively select information from different spatial scales. Furthermore, the softmax operation is employed to compute the weights for each channel, as shown in Equation (4):

$$a_c = \frac{e^{\mathbf{A}_c\mathbf{z}}}{e^{\mathbf{A}_c\mathbf{z}} + e^{\mathbf{B}_c\mathbf{z}}}, b_c = \frac{e^{\mathbf{B}_c\mathbf{z}}}{e^{\mathbf{A}_c\mathbf{z}} + e^{\mathbf{B}_c\mathbf{z}}} \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{C \times d}$, $\mathbf{B} \in \mathbb{R}^{C \times d}$, a and b represent the attention vectors of $\tilde{\mathbf{U}}$ and $\hat{\mathbf{U}}$ respectively. $\mathbf{A}_c \in \mathbb{R}^{1 \times d}$ denotes the c-th row of $\boldsymbol{A}$, and a represents the c-th element value of a; the same applies to vector b. The final feature map **V** is computed through the attention weights on each convolution kernel. The calculation method for each channel's feature map $\mathbf{V}_c$ is shown in Equation (5):

$$\mathbf{V}_c = a_c \cdot \tilde{\mathbf{U}}_c + b_c \cdot \hat{\mathbf{U}}_c \tag{5}$$

According to formula (5), it can be inferred that $a_c + b_c = 1$. The final result of the feature map **V** is $\mathbf{V} = \left[\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_C\right]$ and $\mathbf{V}_c \in \mathbb{R}^{H \times W}$. The purpose of the selection operation is to fuse feature maps of different convolution kernel sizes based on the calculated weights.

Decoder: In recent years, networks combining convolutional and deconvolution layers have been mainly applied in semantic segmentation of images [14]. In the convolutional layer, multiple input activation values in the filtering window are fused together to output a single activation value. On the contrary, a single input activation value is associated with multiple outputs through a deconvolution layer. The convolutional layers in the encoder and the deconvolution layers in the decoder are organized symmetrically. The decoder will reconstruct the input image using the features learned by the encoder. In this chapter, the SK convolution module is further expanded and the SK deconvolution module is constructed as

the decoding unit. Compared with the SK convolution module, the SK deconvolution module also has three steps: splitting, fusion, and selection. The difference is that in the splitting stage, deconvolution is used instead of convolution in the SK convolution module. In addition, group convolution was not used in the SK deconvolution module.

Cross layer connection: As the FCAE network gets deeper, the increase in the number of convolutional layers in the network can result in the loss or damage of some image details that contain rich information. This makes it difficult for the deconvolution layer to recover the image. In addition, deep networks often become difficult to train due to gradient vanishing. Inspired by the literature [8], we have added skip connections between the convolutional module and the corresponding deconvolution module. These skip connections can propagate image details to the top layer and backpropagate gradients to the bottom layer. This makes it easier and more effective to train the network end-to-end from the input image to the output image, and improve the performance of the FCAE network as the number of layers is increased.

3.2. Data Augmentation of Training Data

In practical image classification problems, data augmentation can be used to increase the diversity of the dataset and expand the size of the data set, thereby effectively alleviating model overfitting. In addition, it can improve the model's generalization ability [15]. Data augmentation is a method of adjusting the original data information and increasing it to the training set based on some image processing methods [16]. There are many specific methods, and most of them are simulated by manual methods. In addition to the methods provided by various training frameworks (such as Tensorflow and PyTorch), there are many third-party libraries. **Figure 3** summarizes the commonly used data augmentation methods. Based on the aforementioned method, data augmentation was carried out for the two datasets, CIFAR-10 and CIFAR-100. In this paper, two data augmentation methods, namely random cropping and flipping, were adopted. Additionally, two data augmentation methods (pixel-level Gaussian noise and block masking noise) for adding noise were also employed. **Figure 4** shows the noisy image, the reconstructed image obtained by the method proposed in this paper, and the original dataset image.

3.3. Training Process

Unlike the layer-by-layer training rule of traditional AEs, this paper adopts an end-to-end approach to train FCAE. Given a certain image, a copy H is obtained through random cropping and horizontal flipping. On this basis, the input to the network is its noisy version, and the reconstructed image output by the network is denoted as $\tilde{H}$. The network is trained end-to-end based on the minimum mean square error between H and $\tilde{H}$. Once the FCAE network is trained, labeled training samples can be input into the network to obtain the image feature representation learned by the encoder, which is then fed into the classifier to train the classifier.

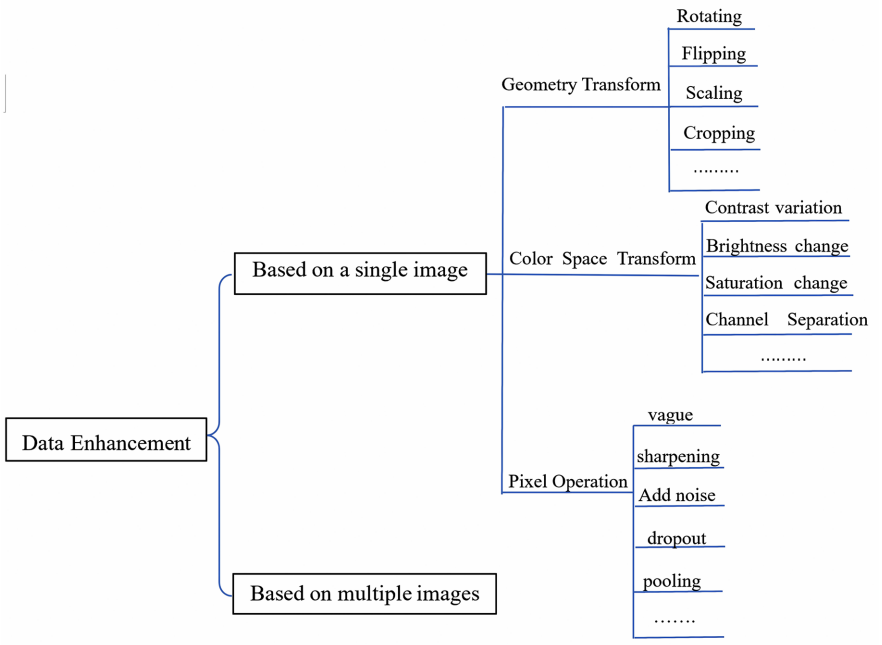

Figure 3. The commonly used data augmentation methods.

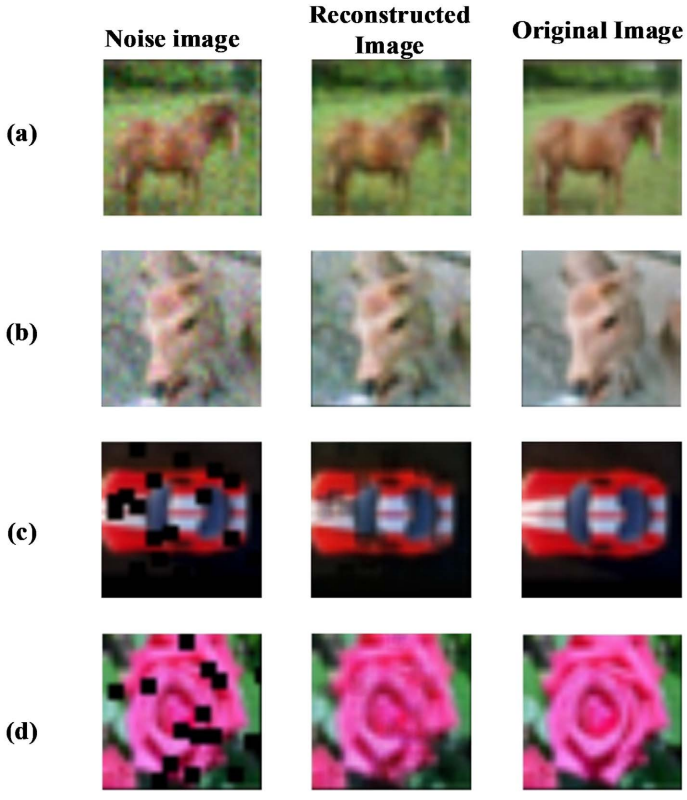

Figure 4. Images with noise, reconstructed images and original images (a); (b) are the cases of adding Gaussian noise (c) and (d) are the cases of adding block noise.

4. Experimental Results and Analysis

Given the limitations of computational resources, this paper selects the CIFAR-10

and CIFAR-100 datasets for experiments. The purpose is to first explore the impact of introducing SK convolution module and SK deconvolution module into the encoder and decoder on the classification performance of the model; Secondly, the impact of randomly adding noise and adding two different types of noise on the classification performance of the model has been analyzed. Finally, by comparing the classification results with existing network models, corresponding analysis is provided.

4.1. Experimental Dataset

The CIFAR-10 dataset was provided by Krizhevsky et al. from Hinton's team, consisting of 50,000 training images and 10,000 test images, each with a size of 32 × 32. The dataset includes 10 classes, with each class containing 5000 training images and 1000 test images. Figure 5 shows example images from the CIFAR-10 dataset [17].

Figure 5. Samples of CIFAR-10 dataset.

Similar to the CIFAR-10 dataset, the CIFAR-100 dataset also contains 60,000 RGB three-channel images with a size of 32 × 32. Unlike CIFAR-10, the CIFAR-100 dataset includes 100 classes, with each class containing 600 images in total (500 training images and 100 test images). Additionally, the 100 classes in CIFAR-100 are divided into 20 superclasses. Each image is labeled with a "fine" label (the category it belongs to) and a "coarse" label (the superclass it belongs to). The detailed category information is shown in Table 1.

Table 1. The categories about the CIFAR-100 dataset.

Superclass	Category
fish	aquarium fish, flounder, shark, trout
aquatic mammal	beaver, dolphin, seal, whale, otter
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	Bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

4.2. Model Parameter Configuration

The basic environment adopted in our experiments is as follows: the operating system is Windows 10; the Python version is Anaconda 4.5.11; the deep learning framework is TensorFlow 2.0; the single GPU model is GTX 1660. To improve GPU computational efficiency, CUDA and cuDNN are used to optimize its architecture.

In the experiment, the FCAE model optimization algorithm (Optimizer) used Adam in the system architecture stage (1), with an initial learning rate set to 0.001. The learning rate was set in a heuristic manner. The specific setting method is to reduce the current learning rate to 1/10 of the previous learning rate when the model validation set reconstruction loss value no longer decreases. In addition, the step size for cross layer connections between the encoder and decoder is 2. Set the batch size of each training batch to 64. On both two datasets, the number of epochs for model optimization in stage 1) is set to 300. In stage 2), the optimization algorithm used for the classifier is a small batch SGD with momentum coefficients, and the batch size is set to 128. The learning rate is set in annealing mode, with an annealing step size of 10, a final learning rate of 0.00001, and an initial

learning rate of 0.01. When the previous epoch matches the set stride, the learning rate will be adjusted. The number of epochs for training the classifier is set to 1300, and the sample size for each training batch is set to 128. Group convolution is introduced in the SK convolution module of the encoder, with the number of groups set to 4. In the SK deconvolution module of the decoder, grouped convolution is not used.

In terms of data augmentation for dataset preprocessing: First, each image boundary is extended by 4 pixels with a value of 0, forming an image size of 40 × 40. Then, images are processed through random sampling and horizontal flipping, and the mean RGB value is subtracted from each pixel, resulting in an image size of 32 × 32. Finally, the image size is adjusted to 256 × 256 using the tf.image.resize_images function within the TensorFlow framework. Additionally, two noise addition methods are used for data augmentation: randomly adding pixel-level Gaussian noise or block masking noise to the training images in the dataset, with a random probability θ set to 0.33. For pixel-level Gaussian noise, each pixel in the image uniformly adds random Gaussian noise with a mean of 0 and a standard deviation of σ, where σ is set to 0.1. For block masking noise, the block size is set to 3, and the number of randomly added blocks in the image ranges between [18].

4.3. Analysis of Experimental Results

In the construction of the network structure, both the encoder and decoder networks have 6 layers. The first layer of the encoder network uses a standard convolution operation, while the second to sixth layers use SK convolution modules; the first to fifth layers of the decoder network use SK deconvolution modules, and the sixth layer uses a standard deconvolution operation. Both the SK convolution module and SK deconvolution module have 2 branches. To avoid overfitting, a dropout layer with a probability of 0.5 is introduced after the fully connected layer in phase (2). This dropout layer is only used during the entire training process, not during the testing process. The specific configuration (1) of the network structure in phase (1) is shown in **Table 2**. In this table, "—" indicates that the parameter does not exist, and "SAME" and "VALID" in boundary padding represent two modes for convolution, deconvolution, and pooling; each branch in the SK convolution and SK deconvolution modules uses the same parameters.

This section takes CIFAR-10 as an example to explore the impact of embedding SK convolution module and SK deconvolution module on network performance. This section considers three scenarios: 1) Using a regular FCAE model, without embedding SK convolution modules and SK deconvolution modules in the encoder and decoder; 2) Only embedding the SK convolution module into the encoder, and not embedding the SK deconvolution module into the decoder; 3) Embed the SK convolution module and SK deconvolution module into the encoder and decoder respectively. The experimental results of these three scenarios were analyzed and compared, as shown in **Table 3**. According to **Table 3**, both the SK

convolution module and the SK deconvolution module can effectively improve the classification accuracy of the model.

Table 2. Specific configuration of network structure (1).

Name	Input			Number of convolution kernels	Size of convolution kernels	Step	Boundary filling	Output		
	Width	Height	Depth					Width	Height	Depth
Input	—	—	—	—	—	—	—	256	256	3
conv1	256	256	3	8	3	1	SAME	256	256	8
Maxpool1	256	256	8	—	2	2	VALID	128	128	8
Sk-conv2	128	128	8	16 16	5 3	1 1	SAME	128	128	16
Maxpool2	128	128	16	—	2	2	VALID	64	64	16
Sk-conv3	64	64	16	32 32	5 3	1 1	SAME	64	64	32
Maxpool3	64	64	32	—	2	2	VALID	32	32	32
Sk-conv4	32	32	32	64 64	5 3	1 1	SAME	32	32	64
Maxpool4	32	32	64	—	2	2	VALID	16	16	64
Sk-conv5	16	16	64	128 128	5 3	1 1	SAME	16	16	128
Maxpool5	16	16	128	—	2	2	VALID	8	8	128
Sk-conv6	8	8	128	256 256	5 3	1 1	SAME	8	8	256
Maxpool6	8	8	256	—	2	2	VALID	4	4	256
Sk-deconv1	4	4	256	128 128	5 3	2 2	SAME	8	8	128
Sk-deconv2	8	8	128	64 64	5 3	2 2	SAME	16	16	64
Sk-deconv3	16	16	64	32 32	5 3	2 2	SAME	32	32	32
Sk-deconv4	32	32	32	16 16	5 3	2 2	SAME	64	64	16
Sk-deconv5	64	64	16	8 8	5 3	2 2	SAME	128	128	8
deconv6	128	128	8	3	3	2	SAME	256	256	3

Table 3. Impact of SK convolution module and SK deconvolution module on model performance (CIFAR-10).

Model	Scenario (1)	Scenario (2)	Scenario (3)
Classification accuracy (%)	90.02	90.28	90.45

In addition, this section also analyzed the classification results of the model under three scenarios: no noise added, random Gaussian noise added, and block noise added in the CIFAR-10 training dataset, as shown in Table 4. According to the experimental results, adding noise can make the trained model more robust to disturbances, thereby improving the classification performance of the model. In addition, randomly adding block noise can improve the accuracy of model classification more than randomly adding Gaussian noise. At the same time, this section also tested the classification results of the CIFAR-10 training dataset after random sampling and horizontal flipping data augmentation operations, and adjusted the image size to 128 × 128 using the tf.image.resice_images function. The network structure configuration with input image size adjusted to 128 × 128 is shown in Table 5. In this table, "-" indicates that the parameter does not exist. In boundary filling, "SAME" and "VALID" represent two modes: convolution, deconvolution, and pooling; The convolution branches in each SK convolution module and SK deconvolution module use the same parameters.

Table 4. The effect of adding noise on model performance (CIFAR-10).

Types of Noise	Without adding noise	Gaussian Noise	Block Noise
Classification accuracy (%)	86.55	88.77	90.45

Table 5. Network structure configuration (2).

Name	Input			Number of convolution kernels	Size of convolution kernels	Step	Boundary filling	Output		
	Width	Height	Depth					Width	Height	Depth
Input	—	—	—	—	—	—	—	128	128	3
conv1	128	128	3	8	3	1	SAME	128	128	8
Maxpool1	128	128	8	—	2	2	VALID	64	64	8
Sk-conv2	64	64	8	16	5	1	SAME	64	64	16
				16	3	1				
Maxpool2	64	64	16	—	2	2	VALID	32	32	16
Sk-conv3	32	32	16	32	5	1	SAME	32	32	32
				32	3	1				
Maxpool3	32	32	32	—	2	2	VALID	16	16	32
Sk-conv4	16	16	32	64	5	1	SAME	16	16	64
				64	3	1				
Maxpool4	16	16	64	—	2	2	VALID	8	8	64

Continued

Sk-conv5	8	8	64	128 128	5 3	1 1	SAME	8	8	128
Maxpool5	8	8	128	—	2	2	VALID	4	4	128
Sk-conv6	4	4	128	256 256	5 3	1 1	SAME	4	4	256
Maxpool6	4	4	256	—	2	2	VALID	2	2	256
Sk-deconv1	2	2	256	128 128	5 3	2 2	SAME	4	4	128
Sk-deconv2	4	4	128	64 64	5 3	2 2	SAME	8	8	64
Sk-deconv3	8	8	64	32 32	5 3	2 2	SAME	16	16	32
Sk-deconv4	16	16	32	16 16	5 3	2 2	SAME	32	32	16
Sk-deconv5	32	32	16	8 8	5 3	2 2	SAME	64	64	8
deconv6	64	64	8	3	3	2	SAME	128	128	3

With the basic structure and parameters of the model remaining unchanged, this section analyzes and compares two types of network structures with input image sizes of 128 × 128 and 256 × 256. The comparison results are shown in Table 6. In this experimental analysis, to reduce the training time of the classifier, the number of epochs was set to 650, while the other training parameters remained unchanged. As can be seen from Table 6, when the input image size of the network is 128 × 128, the classification performance of the model is relatively inferior. The main reason is that the size of the input image affects the dimension of the feature map output by the encoder. An increase in the dimension of the feature map leads to an increase in the number of network parameters and complexity, which can improve the classification accuracy of the model to a certain extent.

Table 6. Impact of input image size on model performance (CIFAR-10).

The size of input Image	128 × 128	256 × 256
Classification accuracy (%)	86.45	89.65

For the CIFAR-100 dataset, the same network structure settings and parameter configurations were used in this section. Similarly, SK convolution module and SK deconvolution module were added to the encoder and decoder; Block noise was introduced into the training dataset to improve the robustness of the model. The only difference is that in stage (2), the number of categories in the final soft-

max layer is set to 100. In addition, the methods proposed in this chapter were compared with some advanced algorithms, and the comparison results are shown in Table 7. Among them, "-" indicates that no relevant experiments or descriptions have been conducted in the literature. In Table 7, SCDAE-2 has two convolutional layers, which are stacked in a convolutional manner through layer by layer optimization. The feature maps in each layer are generated by convolutional kernels learned from the underlying features and DAE. SFCAE-2 consists of an encoder and a decoder, with the encoder containing convolutional and pooling layers. The SFCAE-2 encoder contains two basic modules. SCDAE-2 and SFCAE-2 are both belong to unsupervised feature learning networks. The network constructed in this paper belongs to the CAE series and is also an unsupervised deep network. However, compared to these two models, the method proposed in this paper has significant performance advantages.

Table 7. Classification accuracy comparison on CIFAR-10 and CIFAR-100.

Frame type	Model	CIFAR-10 (%)	CIFAR-100 (%)
Unsupervised Learning	SCDAE-2 [5]	80.4	-
Unsupervised Learning	SFCAE-2 [3]	83.5	-
Supervised learning	Maxout ($k = 2$) [19]	90.62	61.43
Supervised learning	NIN [20]	91.19	64.32
Supervised learning	ResNet [7]	93.57	74.84
Supervised learning	Pre-activation ResNet [21]	95.38	77.29
Unsupervised Learning	Ours	90.45	66.42

In addition, this section also compared four types of deep networks with supervised learning. Compared with the classic NIN network and Maxout network, the method proposed in this chapter is slightly inferior on the CIFAR-10 dataset. However, the proposed method showed better classification results than NIN and Maxout on the CIFAR-100 dataset. At the same time, this section also compared the proposed method with ResNet and Pre-activation ResNet which have some recent high-performance. Although the proposed model has slightly worse research results than these two networks, ResNet and Pre-activation ResNet use up to 110 and 1001 layers of network layers, respectively.

Compared to this, the methods proposed in this paper still have significant room for improvement in terms of network depth. This method consists of two parts: the encoder part includes one convolutional layer and five SK convolution modules in the system architecture Stage (1). In each SK convolution module, there are 2 convolution operations, a fully connected layer FC1 for compressing feature z, and a fully connected layer FC2 before the softmax operation. One deconvolution layer and five SK deconvolution modules constitute the decoder. Each SK deconvolution module also includes two deconvolution operations and a fully connected layer FC1 for compressing feature z, as well as a fully connected

layer FC2 before the softmax operation. In Stage (2), there is a fully connected layer FC and a softmax classification layer. Therefore, the network layers of the model proposed in this chapter are $(1 + 5 \times (2 + 1 + 1) + 1 + 5 \times (2 + 1 + 1) + 1 + 1) = 44$ layers. In summary, although the classification results of the model proposed in this paper are slightly inferior to ResNet and Pre-activation ResNet, the method structure in this paper is simple and has fewer network layers. More noteworthy is that this method is an unsupervised feature learning approach.

4.4. Feature Visualization

To better explain the internal mechanisms of the method proposed in this chapter, this section will visualize the output features after Maxpool1~Maxpool6 layers in the encoder part of the network structure (see Table 2 for detailed configuration). For ease of discussion, this section uses the training model on the CIFAR-10 dataset. In this model, the encoder and decoder are respectively equipped with SK convolution module and SK deconvolution module, and two scenarios are considered: without noise and with block noise. Figure 6 and Figure 7 show the summed results of all output channel feature maps after Maxpool1~Maxpool6 layers, with test cases selected from the test dataset. As the network layers gradually increase, the resolution of each layer's output feature map decreases, and the number of feature channels increases. As shown in Figure 6 and Figure 7, feature maps at different levels describe the target and its surroundings from various perspectives. Shallow-layer features tend to focus more on detecting image textures and details, providing comprehensive detection content while also extracting key information. As the network deepens, it extracts image contours, shapes, and strongest features while ignoring many secondary details. The higher stages of the network learn increasingly abstract semantic information, resulting in lower resolution feature maps with more discriminative features.

Figure 6 shows the test images without added noise and the output feature maps after each pooling layer. Figure 7 shows the images with block noise added and the output feature maps after each pooling layer. From these figures, it can be

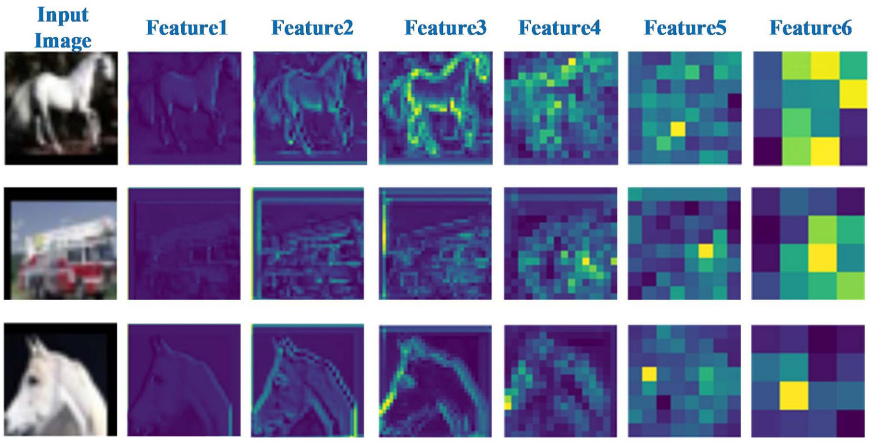

Figure 6. Images without noise and corresponding feature maps.

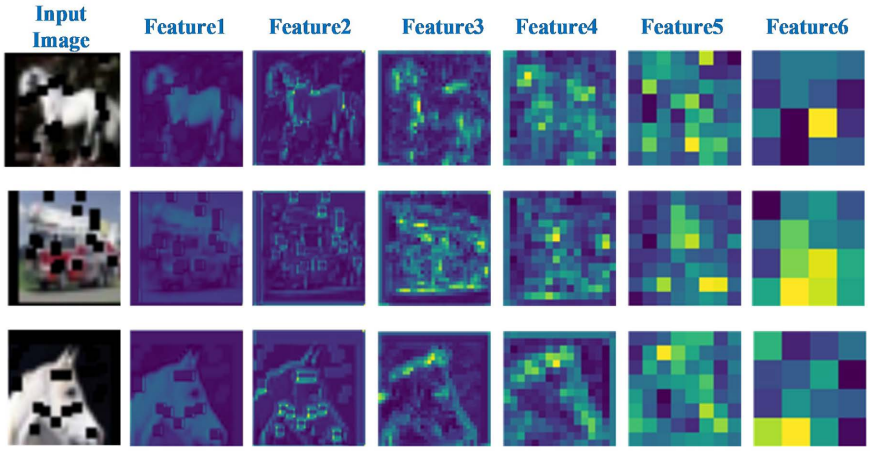

Figure 7. Images with block masking noise and corresponding feature maps.

observed that when the input image is occluded, the learned low-level feature representations (such as feature maps 1 - 3) are affected to some extent; However, as the network layers deepen, the high-level feature representations (such as feature maps 4 - 6) are almost unaffected. This also demonstrates that the method proposed in this paper is robust to noise.

5. Conclusion

This paper has proposed a stacked convolutional autoencoder that integrates the selection kernel attention mechanism. This method is based on FCAE and can be trained in an end-to-end manner. Among them, the convolutional layer chain composed of SK convolutional modules forms the encoder part. Correspondingly, the decoder is composed of a deconvolution layer chain using SK deconvolution modules. In addition, we have drawn on the idea of residual networks. In order to better handle the degradation problem caused by the increase of network depth, skip layer connections are added between the SK convolution module and the SK deconvolution module symmetrically connected to it. Furthermore, in order to effectively alleviate model overfitting, the method of adding Gaussian noise and block noise in data augmentation was used to increase the difficulty of model training, thereby further improving the model's adaptability to new samples. In the experimental results and analysis, this method was not only compared with unsupervised feature learning models based on AE, but it was also compared and analyzed with some supervised feature learning methods with better performance. In addition, in order to better understand the internal mechanism of the model, the output feature maps of each pooling layer in the encoder were visualized.

Acknowledgements

This work was supported by the Natural Science Foundation of Yancheng City, grant number YCBK2023059 and YCBK2024038.

Conflicts of Interest

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

References

[1] Li, Z., Yang, W., Peng, S., *et al.* (2020) A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects.

[2] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017) Imagenet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, **60**, 84-90. https://doi.org/10.1145/3065386

[3] Sun, Y., Xue, B., Zhang, M. and Yen, G.G. (2019) A Particle Swarm Optimization-Based Flexible Convolutional Autoencoder for Image Classification. *IEEE Transactions on Neural Networks and Learning Systems*, **30**, 2295-2309. https://doi.org/10.1109/tnnls.2018.2881143

[4] Masci, J., Meier, U., Cireşan, D. and Schmidhuber, J. (2011) Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In: *Proceedings of International Conference on Artificial Neural Networks*, Springer, 52-59. https://doi.org/10.1007/978-3-642-21735-7_7

[5] Du, B., Xiong, W., Wu, J., Zhang, L., Zhang, L. and Tao, D. (2017) Stacked Convolutional Denoising Auto-Encoders for Feature Representation. *IEEE Transactions on Cybernetics*, **47**, 1017-1027. https://doi.org/10.1109/tcyb.2016.2536638

[6] Li, F., Qiao, H. and Zhang, B. (2018) Discriminatively Boosted Image Clustering with Fully Convolutional Auto-encoders. *Pattern Recognition*, **83**, 161-173. https://doi.org/10.1016/j.patcog.2018.05.019

[7] He, K., Zhang, X., Ren, S. and Sun, J. (2016) Deep Residual Learning for Image Recognition. 2016 *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), Las Vegas, 27-30 June 2016, 770-778. https://doi.org/10.1109/cvpr.2016.90

[8] Mao, X.J., Shen, C. and Yang, Y.B. (2016) Image Restoration Using Convolutional Auto-Encoders with Symmetric Skip Connections.

[9] Cai, W.W. and Wei, Z.G. (2020) Remote Sensing Image Classification Based on a Cross-Attention Mechanism and Graph Convolution. *IEEE Geoscience and Remote Sensing Letters*, **19**, Article ID: 8002005.

[10] Vaswani, A., *et al.* (2017) Attention Is All You Need. *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 6000-6010.

[11] Galassi, A., Lippi, M. and Torroni, P. (2019) Attention, Please! A Critical Review of Neural Attention Models in Natural Language Processing.

[12] Wang, F., Jiang, M., Qian, C., *et al.* (2017) Residual Attention Network for Image Classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, 21-26 July 2017, 3156-3164.

[13] Li, X., Wang, W., Hu, X. and Yang, J. (2019) Selective Kernel Networks. 2019 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Long Beach, 15-20 June 2019, 510-519. https://doi.org/10.1109/cvpr.2019.00060

[14] Fu, J., Liu, J., Li, Y., Bao, Y., Yan, W., Fang, Z., *et al.* (2020) Contextual Deconvolution Network for Semantic Segmentation. *Pattern Recognition*, **101**, Article ID: 107152. https://doi.org/10.1016/j.patcog.2019.107152

[15] Mounsaveng, S., Laradji, I., Ayed, I.B., Vazquez, D. and Pedersoli, M. (2021) Learning Data Augmentation with Online Bilevel Optimization for Image Classification. 2021

IEEE Winter Conference on Applications of Computer Vision (*WACV*), Waikoloa, 3-8 January 2021, 1691-1700. https://doi.org/10.1109/wacv48630.2021.00173

[16] Zoph, B., Cubuk, E.D., Ghiasi, G., Lin, T., Shlens, J. and Le, Q.V. (2020) Learning Data Augmentation Strategies for Object Detection. In: *European Conference on Computer Vision*, Springer International Publishing, 566-583. https://doi.org/10.1007/978-3-030-58583-9_34

[17] Giuste, F.O. and Vizcarra, J.C. (2020) CIFAR-10 Image Classification Using Feature Ensembles.

[18] Alahmadi, A., Hussain, M., Aboalsamh, H.A. and Zuair, M. (2019) Pcapool: Unsupervised Feature Learning for Face Recognition Using PCA, LBP, and Pyramid Pooling. *Pattern Analysis and Applications*, **23**, 673-682. https://doi.org/10.1007/s10044-019-00818-y

[19] Goodfellow, I., Warde-Farley, D., Mirza, M., *et al*. (2013) Maxout Networks. *International Conference on Machine Learning*, Atlanta, 16-21 June 2013, 1319-1327.

[20] Lin, M., Chen, Q. and Yan, S. (2014) Network in Network.

[21] He, K., Zhang, X., Ren, S. and Sun, J. (2016) Identity Mappings in Deep Residual Networks. In: *European Conference on Computer Vision*, Springer International Publishing, 630-645. https://doi.org/10.1007/978-3-319-46493-0_38.