Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 16(5): 3345–3367.
DOI: 10.3934/mbe.2019167
Received: 07 January 2019
Accepted: 22 March 2019
Published: 18 April 2019

Research article

A full convolutional network based on DenseNet for remote sensing scene classification

Jianming Zhang [1,2], Chaoquan Lu [1,2], Xudong Li [1,2], Hye-Jin Kim [3] and Jin Wang [1,2,*]

[1] Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha 410114, China
[2] School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China
[3] Business Administration Research Institute, Sungshin W. University, Seoul, 02844, Republic of Korea

* **Correspondence:** Email: jinwang@csust.edu.cn; Tel: + 86-18774086968; Fax: + 86-731-85603438.

Abstract: The convolutional neural networks (CNN) applied in remote sensing scene classification have two common problems. One is that these models have large number of parameters, which causes over-fitting easily. The other is that the network is not deep enough, thus more abstract semantic information cannot be extracted. To solve these two problems, we propose a simple and efficient full convolutional network based on DenseNet for remote sensing scene classification. We construct a small number of convolutional kernels to generate a large number of reusable feature maps by dense connections, which makes the network deeper, but does not increase the number of parameters significantly. Our network is so deep that it has more than 100 layers. However, it has only about 7 million parameters, which is far less than the number of VGG's parameters. Then we incorporate an adaptive average 3D pooling operation in our network. This operation fixes feature maps of size 7 × 7 from the last DenseBlock to 1 × 1 and decreases the number of channels from 1024 to 512, thus the whole network can accept input images with different sizes. Furthermore, we design the convolutional layer instead of the fully connected layer that is used as a classifier usually, so that the output features of the network can be classified without flattening operation, which simplifies the classification operation. Finally, a good model is trained by exploiting pre-trained weights and data augmentation technology. Compared with several state-of-the-art algorithms, our algorithm improves classification performance significantly on UCM dataset, AID dataset, OPTIMAL-31 dataset and NWPU-RESISC45 dataset.

Keywords: CNN; DenseNet; full convolutional network; remote sensing scene classification; data augmentation

1. Introduction

Remote sensing images has been applied to a broad range of fields such as land use classification [1,2], geographic image retrieval [3,4] and remote sensing scene classification. Remote sensing scene classification is an important work, which can provide some decision-making basis for the follow-up work of urban planning and environmental monitoring. Remote sensing scene classification divides remote sensing images into various categories such as forests, cities and rivers according to the semantic information of remote sensing images. Remote sensing images we focus on are obtained by aerial photography. These remote sensing images have a high spatial resolution, highly complex geometric structures and spatial patterns. However, there is high similarity between categories of remote sensing while images in the same scene category may have great differences. This makes it difficult to classify remote sensing scenes.

Multimedia information processing and communication are popular in our daily lives, and security and privacy protection techniques are becoming more and more important [5]. Remote sensing images are almost saved in the digital form, therefore they are possible to be tampered with. It should be noted that for some remote sensing scene images, modifying a small number of pixels can affect the classification results greatly, which may cause some risks. This has become another challenge in the classification of remote sensing scenes. Because of these challenges mentioned above, the classification models should be supposed to extract the semantic information from the image very well to achieve the better robustness.

Researchers pay more attention to the design of handcrafted features that are used for remote sensing scene classification before the development of deep learning. Those low-level features designed by the human is hard to represent semantic information of remote sensing scenes. The classification accuracy is unsatisfactory in the methods using artificial features. With the development of deep learning, deep learning has been applied to many research field [6–9]. The convolution neural network (CNN), as the most popular technology of deep learning, is introduced to remote sensing scene classification naturally. In general, the neural networks can be trained very well with a large amount of training data while they are often over-fitting and poor at generalization with a small amount of training data. To avoid or alleviate this situation, on the one hand, data enhancement technology can be used to expand the training data [10,11], on the other hand, a specially designed network also can make the training more efficient [12].

Some neural network architectures are applied to remote sensing scene classification, such as CaffeNet, GoogleNet, and VGGNet. Although many methods based on those networks improve classification accuracy significantly, they still have a bottle-neck. For these networks, we observe that:

1) These networks that are trained with small amounts of data have a large number of parameters. Note that this often results in over-fitting in convolutional neural networks.

2) These networks are shallow. Because of the complexity of remote sensing images, to better classify remote sensing scenes, classification network is supposed to extract high-level semantic features of images well. It is worth noting that deeper networks often can extract more abstract

semantic information.

In view of the above observations, we propose a deeper network with fewer parameters to improve the classification accuracy of remote sensing scene. Based on the DenseNet, we built a network with deeper layers but fewer parameters than those networks mentioned above. Table 1 shows the comparison of parameters and layers of different networks. To increase the diversity of training data, we use data augmentation technology on training dataset, such as brightness, color, contrast, sharpness and rotation when training the network. All of the parameters of those augmentation transformations are generated randomly. In addition, we utilize pre-training parameters trained on ImageNet [13], which can accelerate training and overcome over-fitting. Moreover, we fix the length of the final feature vector that is used for classification by an adaptive average 3D pooling operation. Thus, the network can accept images of various sizes. To avoid the flattening operation and make the network operation more concise, we design a convolutional layer instead of fully connected layer to classify the features. Unlike many methods that use Support Vector Machines (SVM) as a classifier, we use the softmax as the probability mapping function, so our network is an end-to-end process without additional training on SVM.

Table 1. Comparison of different network parameters and layers.

Network	Parameters (millions)	Convolution layers	Fully connected layers
CaffeNet	61.1	5	3
GoogleNet	6.7	22	1
VGG16	138.3	13	3
VGG19	143.6	16	3
ours	7.5	121	0

2. Related works

The methods of remote sensing scene classification are mainly based on three aspects at present: low-level visual features, middle-level visual features and high-level visual features [14].

Low-level visual features are used in many tasks, such as object tracking [15]. There are some frequently-used low-level visual features in the field of classification of remote sensing scenes, such as spectral feature, texture feature and structure feature. Those methods based on low-level visual features either extract feature vector from these low-level visual features or use a combination of feature vectors extracted from different low-level visual features to describe remote sensing images [16–19]. This means that such methods are based on the premise that remote sensing images can be described by a feature vector or a combination of different kinds of feature vectors. For some images whose scene are simple, one kind of feature can be used as a descriptor of remote sensing images [16,20]. Unlike using only one kind of feature, literatures [17–19] show that the combinations of different kind of features can improve the results. Specifically, [17] combine six different kinds of feature to form a multiple-feature representation, which describes the scene better. However, low-level visual features are only applicable to simple scenarios that have no complex structure. For complex scenarios, their expressive ability is limited [21].

To overcome the limitations of low-level visual features, many methods based on middle-level visual features are proposed. The main idea of such methods is to encode the extracted features such as SIFT, LBP and color histograms. Then further extract the more discriminant feature representation.

Bag of Word (BoVW) model [22] and its related variants [23–27] are typical algorithms based on the middle-level visual features. The BoVW regards an image as a collection of visual words, and to construct new features that are used for image classification with the frequency of visual words. Specific steps are as follows: it first extracts SIFT features from all images, next constructs a dictionary using a clustering algorithm, and then counts the occurrence frequency of each word of the dictionary in the image. The features used as a descriptor of an image for classification is composed of these occurrence frequencies of each word of the dictionary. The features obtained by such methods are a further abstraction of low-level visual features, lacking flexibility and adaptability for different scenarios [14].

The third category of methods are based on high-level visual features. Such methods achieve the best accuracy comparing with the above two kinds methods, even far than algorithms based on low-level visual features [28–35]. As mentioned above, neither low-level visual features nor middle-level visual features can represent remote sensing images well and flexibly, which means that the features obtained by these two methods lack abstract semantic information. As we all know, CNN can extract abstract semantic information from images. Thus, those features extracted by CNN are made to be a good descriptor of remote sensing images. In addition, CNN can learn how to extract the high-level semantic information of an image by itself, which is very flexible and avoids complex design of artificial features.

At present, CNN models used to the classification of remote sensing scene mainly include CaffeNet, GoogleNet and VGGNet [33,36–39]. Usually, these networks are shallow with no more than 30 layers, and the number of their parameters is large, such as VGG16 that has about 138 million parameters. Chaib et al. [37] use a VGGNet to extract the features from remote sensing images, then the features from different layers are transformed by discriminant correlation analysis (DCA), the transformed features are fused to construct a new feature that is classified by SVM that is used as a classifier in other fields [40]. The main idea of literature [39] is also features fusion. Different from [37], Yu et al. [39] proposed two schemes. One is that using the original RGB network and the mapped LBP coded network extracts the features from RGB images and LBP feature maps respectively. The features from different network are fused by concatenation layer, and then the obtained features are classified after being processed by the fully connected layers. The steps of the other one are similar. The difference is that a saliency coded network is used instead of mapped LBP coded network. Wang Q et al. [38] introduced the attention mechanism into the classification of remote sensing scene by constructing a recurrent attention structure with recurrent neural networks (RNN). Different from the above methods, Cheng G et al. [36] focus on the loss function, and propose a new loss function by applying metric learning to CNN features. By combining proposed metric learning loss with cross-entropy loss, features extracted from the same scene class are as close as possible, while features extracted from different categories are as far as possible.

3. Method description

As mentioned above, the CNN model generally used in remote sensing scene classification have huge number of parameters, or the depth of network layer is too shallow, which is easy to cause over-fitting of the model and inadequate feature extraction ability of the model. Aiming at the problem of network applied in remote sensing scene classification, we apply the DenseNet [41] to our architecture and construct a full convolution network for remote sensing scene classification. Based on cross-layer connectivity, our full convolutional network is a deep network architecture with only a few parameters. In addition, we introduce an adaptive average 3D pooling operation, which fixes the sizes of the output feature maps from 7 × 7 to 1 × 1 and decrease the number of channels from 1024 to 512. Our method can accept images that are different sizes. Furthermore, our network exploits a convolutional layer instead of full connection layer as a classification layer. Thus, features can be classified without flattening, which makes the network simpler and more efficient.

3.1. The main idea of DenseNet

In 2017, Huang et al. [41] proposed DenseNet derived from [42]. In general, the network structures are progressively hierarchical. In such a network structure, the feature maps from the $(i-1)$ th layer are the input of the ith layer. Experiments [42] have shown that the input of the ith layer can be not only the output of the $(i-1)$ th layer, but also the $(i-2)$ th layer, or the $(i-n)$ th layer (n less than the number of layers). This may lead to a more generalizable network. The basic idea of DenseNet is that each layer in the network is directly connected to the front layers. The connection strategy in DenseNet is shown in Figure 1. It is worth noting that in order to ensure that the feature maps can be concatenated, the sizes of feature maps need to be consistent. This means that the outputs of the convolutional layer are of the same size as the input.

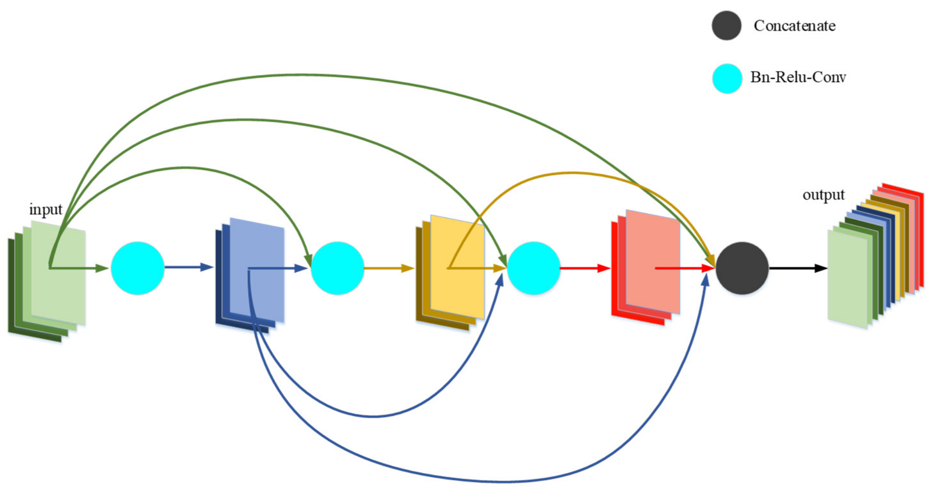

Figure 1. Densely concatenated convolution. The input of each layer comes from the output of all previous layers, and the final output of the network fuses all feature maps produced by front layers.

The input of each layer is the output of all front layers in the densely concatenated convolutions:

$$X_l = H_l([X_0, X_1, ..., X_{l-1}]) \tag{1}$$

H_l represents the lth layer, X_l represents the output of the lth layer. In DenseNet, Batch Normalizing (BN) [43] is used to normalize the input of layer, which reduces the absolute difference between data and considers the relative difference more. The BN algorithm is described as the Algorithm 1.

```
Algorithm 1. Batch Normalizing Transform
Input: Values of $x$ over a batch: $B = \{x_{1...m}\}$;
Parameters to be learned: $\gamma$, $\beta$
Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$
(1) $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
(2) $\sigma^2{}_B \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$
(3) $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma^2{}_B + \varepsilon}}$
(4) $y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$
```

The more details about BN you can get from [43]. In order to increase the nonlinearity of network, ReLU is used as the activation function, which is described as:

$$ReLU(x) = \begin{cases} x & x > 0 \\ 0 & x \le 0 \end{cases} \tag{2}$$

Compared with other networks, the network composed of Densenet has the following advantages:

1) Fewer parameters. The input of current layer of the network is composed of the feature maps from front layers, which results in the continuous accumulation of the number of feature maps. Thus, many reusable feature maps can be learned with very few convolutional kernels.

2) Strong ability to prevent over-fitting. Short paths are created from early to late layers because of the dense connection. Each layer receives additional supervision from loss function, which greatly reduces the gradient disappearance. Therefore, dense connection has a very good ability to resist over-fitting, which makes it especially suitable for applications where training data is relatively scarce.

3) Deeper layers. The network can be designed very deep by the dense connection, which only have a few parameters. For example, a network with 96 convolution layers can be designed while it only has 5.34 million parameters.

Experiments on ImageNet [13] demonstrate the efficiency of the dense connection [41]. Therefore, densely concatenated convolution can extract high-level features of remote sensing images. Based on this idea, we construct a full convolution network based on DenseNet [41] for remote sensing scene classification.

3.2. Our network architecture

Our full convolutional network based on DenseNet is shown in Figure 2. In the whole network, it is divided into an initial layer, three transition layers and four densely connected blocks (DenseBlock). After the last DenseBlock, we exploit an adaptive average 3D pooling operation to fix number and size of feature maps and then classify them by a convolution with kernel size of 1 × 1. We describe our architecture in detail in session 3.3.

As shown in Figure 2, the whole network architecture does not strictly follow the rule of dense connection, it mainly consists of densely connected blocks instead. The reason for this is that the whole network has several down-sampling operations on feature maps, which results in the inconsistency of sizes between feature maps in different stages, so that feature maps with different sizes cannot be spliced in the dimension of channel. Moreover, concatenation operation requires feature maps to be kept in memory. With the increase of network layers, feature maps kept in memory are superimposed, which causes huge memory consumption. If every layer of the network strictly follows the dense connection theory, the network will consume many resources, which makes the network stop working.

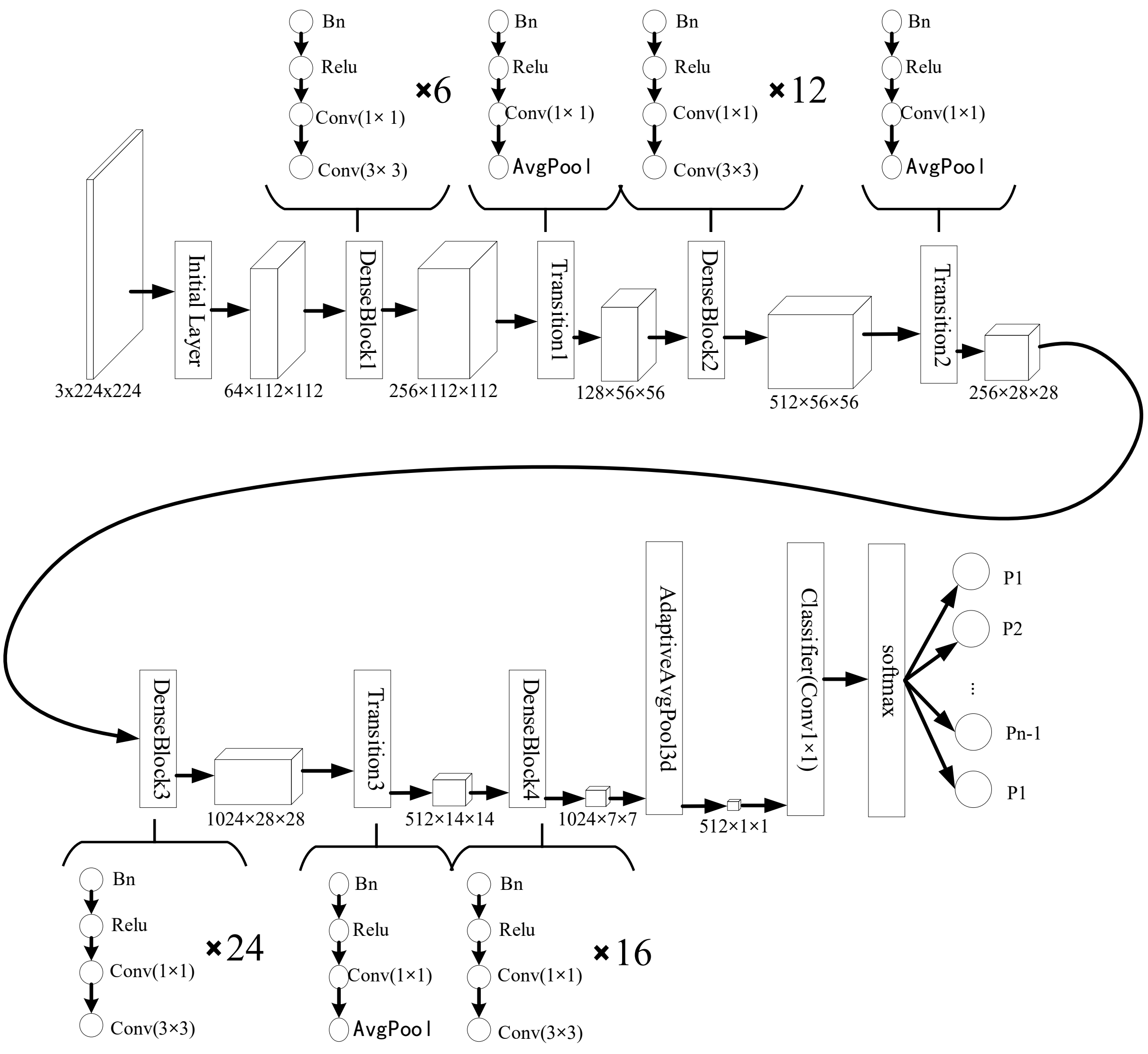

Figure 2. Our full convolutional network for remote sensing scene classification. In our network, each densely connected block (DenseBlock) consists of several Tuple Operations (TO). Each TO is composed of Batch Normalizing (BN), ReLu, Conv(1 × 1) and Conv(3 × 3). Conv(1 × 1) represents a convolutional operation with kernel size of 1 × 1 while Conv(3 × 3) represents a convolutional operation with kernel size of 3 × 3. DenseBlock1 has 6 TOs, DenseBlock2 has 12 TOs, DenseBlock3 has 24 TOs and DenseBlock4 has 16 TOs.

3.3. Architecture detail

In this section, we detail the various parts of our network architecture.

Initial layer. We first extract 64 features using 64 convolutional kernels with kernel size of 7 × 7. We set the convolutional stride as 2, which reduces the sizes of the feature maps and reduces the computational complexity of subsequent operations. After the convolution, the operations of BN, Relu and MaxPool are performed in turn. For example, when the size of the input image is 224 × 224, the size of the image after padding is 230 × 230 and convolution kernels of size 7 × 7 are used for convolution operation, then we will get feature maps of size 112 × 112. Subsequently, these feature maps will go through the operations of BN, Relu and MaxPool in turn. Finally, the obtained feature maps are size of 56 × 56. The generated feature maps are used as input of the subsequent module of the network.

Densely connected block. Our full convolutional network mainly consists of densely connected blocks (DenseBlock). Every DenseBlock acts as one stage for extracting features and consists of several Tuple Operations (TO). As shown in the Figure 2, a TO is composed of Batch Normalizing (BN), ReLU, Conv(1 × 1) and Conv(3 × 3). In particular, in all densely connected blocks, the stride of Conv(1 × 1) is 1 and the number of the convolutional kernels is 128. Moreover, the stride of Conv(3 × 3) is 1 and the number of convolutional kernels is 32. To ensure the same size of the output and input, the padding of Conv(3 × 3) is set to 1. It is worth mentioning that the dense connection operations are strictly enforced between every TO in DenseBlock. The structure of dense connection is shown as Figure 1.

Transition layer. It is observed that, this layer consists of BN, Relu, Conv(1 × 1) and AvgPool. Feature maps pass through these operations in turn. After a DenseBlock, the output of each layer in this DenseBlock is kept in memory, resulting in a large number of feature maps. Note that if the number of these feature maps does not decrease, the network will be difficult to deepen and may stop working due to too much computation. Therefore we reduce the number of the feature maps to half by using a convolutional layer with a convolution kernel size of 1 × 1 after a DenseBlock, which can not only reduce the number of feature maps but also increase the depth and improves fitting ability of the network.

Adaptive average 3D pooling. After 3 transition layers and 4 densely connected blocks, the obtained feature map size is 1/32 of the original image, and the number of feature maps is 1024. To further reduce the computational consumption, we need to reduce the number and size of the feature map. Through the adaptive average 3D pooling operation, the number of feature maps is reduced to 512 and the size becomes 1 × 1 without increasing the network parameters. As shown in Figure 2. In addition, this operation ensures that the size of output feature maps is 1 × 1 and the number is 512, which provides a fixed size feature map for the later convolutional classifier and makes the network accept images with different sizes. As shown in Figure 3.

Convolution classifier. After adaptive average 3D pooling, a 512-dim feature representation is obtained. Then we exploit a 1 × 1 convolution to classify the 512-dim features and get the final category feature representation. Finally, we exploit the softmax function to perform the probability mapping.

Softmax function. This function maps the final category feature representation into a probability for every class. The function is defined as:

$$P(c \mid v) = \frac{exp(X_c)}{\sum_{j=1}^{n} exp(X_j)} \tag{3}$$

$X_c \in v$, X_c represents the c_{th} number of vector v that comes from convolution classifier.

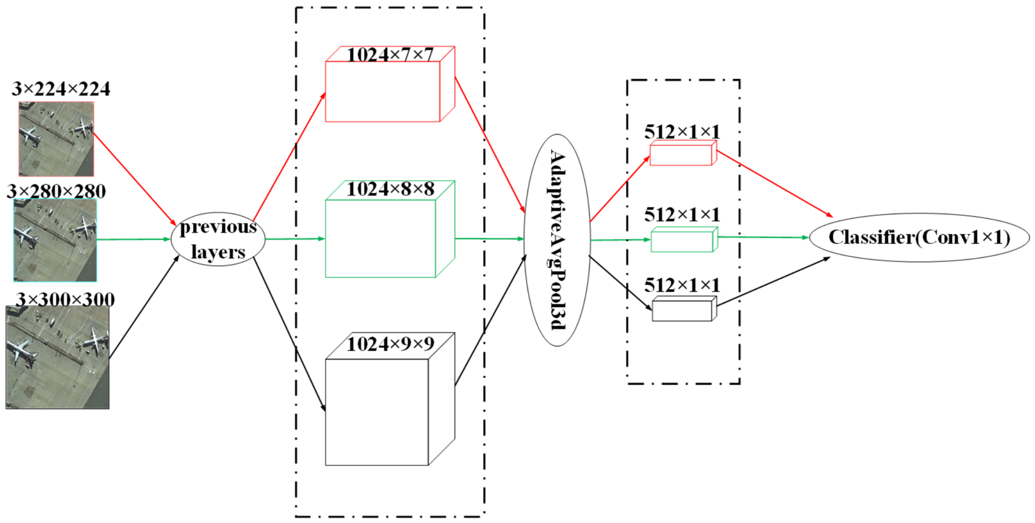

Figure 3. Our network can accept images of different sizes. After extracting the features from input images of sizes 3 × 224 × 224, 3 × 280 × 280, 3 × 300 × 300 by previous layers, the obtained feature map sizes are 1024 × 7 × 7, 1024 × 8 × 8, 1024 × 9 × 9 respectively. The sizes of the feature maps will be 512 × 1 × 1 after the adaptive average 3D pooling operation. Adaptive average 3D pooling operation produces a fixed size feature map for the later convolution classifier.

3.4. Loss function

In many classification tasks, the loss function is designed as:

$$L(x;\theta) = J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} log(P(c \mid h_\theta(x_i))) \tag{4}$$

In equation (4), h_θ is defined as mapping function of the network with parameters θ, $P(c \mid h_\theta(x_i))$ indicates the probability of real category for the ith sample, and n is the number of samples used for one epoch. In order to prevent over-fitting, we normalize the network parameters by L2-norm.

$$\text{L2-norm}(\theta) = \frac{1}{2n_\theta}\lambda\sum_{i}^{n}\theta_i^2 \tag{5}$$

In equation (5), n_θ represents the number of parameters θ of the network. Our ultimate loss function with L2-norm becomes:

$$L(x;\theta)=J(\theta)=-\frac{1}{n}\sum_{i=1}^{n}log(P(c\,|\,h_{\theta}(x_i)))+\frac{1}{2n_{\theta}}\lambda\sum_{i}^{n}\theta_i^2 \tag{6}$$

In equation (6), λ is defined as weight decay rate. In our experiment, we set λ to 5×10^{-4}.

3.5. Parameter optimization strategy

There are many algorithms that can be used to optimize the parameters of neural network, such as genetic algorithm [44] and gradient descent algorithm. It is worth noting that the gradient descent algorithm, the most commonly used algorithm in deep learning, is used to optimize the parameters of the network in our experiments. For gradient descent algorithm, there are various optimizers can be selected, such as Adam and SGD. According to our experience, SGD usually takes much more time to train, but the results are more reliable in the case of a good initialization and learning rate scheduling scheme. Because we use pre-training parameters, which satisfies the condition having a good initialization, so we use SGD as the optimization strategy for training network. For general SGD, the parameters θ of the network are updated as follows:

$$\theta=\theta\text{-}\eta\nabla_{\theta}J(\theta) \tag{7}$$

In equation (7), the gradient $\nabla_{\theta}J(\theta)$ is defined as:

$$\nabla_{\theta}J(\theta)=\frac{\partial J(\theta)}{\partial\theta}=\frac{\partial L(x,\theta)}{\partial\theta} \tag{8}$$

In equation (7), η is defined as the learning rate. One disadvantage of this type of SGD is that its update is very unstable because its update direction is completely dependent on the current batch. A simple way to solve this problem is to introduce the momentum. Momentum simulates the inertia of an object when it is moving, that is, it retains the direction of the previous update to a certain extent when updating, and uses the gradient of the current batch to fine tune the final update direction. In this way, stability can be increased to a certain extent, so that the learning is faster, and there is a certain ability to get rid of the local optimum. For SGD with momentum, the parameters θ of the network are updated as follows:

$$v'\leftarrow m\times v'+\nabla_{\theta}J(\theta) \tag{9}$$

$$\theta\leftarrow\theta-\eta v' \tag{10}$$

In equation (8) and (9), m and v' denote momentum and velocity respectively. In our experiment, we set m to 0.9.

4. Evaluation

4.1. Evaluation indicators

We adopt confusion matrix and overall classification accuracy, which are widely used in classification tasks to evaluate the effectiveness of our algorithm.

4.1.1. Confusion matrix

In the confusion matrix, each row represents the actual category, and each column represents the predicted category. The number of classes correctly classified and the number of classes misclassified can be clearly seen by the confusion matrix.

4.1.2. Overall accuracy

Overall accuracy is the ratio of the number of models correctly predicted on all test sets to the total number of datasets. The overall accuracy can generally characterize the classification accuracy of model.

$$OA = \frac{TP + TN}{TP + FN + FP + TN} \tag{11}$$

TP represents true samples that are correctly classified by the model. FN represents positive samples that are misclassified by the model. FP represents negative samples that are misclassified by the model. TN represents negative samples that are correctly classified by the model.

4.2. Evaluation dataset

In order to verify the effectiveness of our algorithm, we have done a lot of experiments. We experiment on four data sets: UC Merced Land Use Dataset [22], Aerial Image Dataset [14], OPTIMAL-31 Dataset [38] and NWPU-RESISC45 [45] Dataset. These datasets contain a lot of images and cover a lot of categories. In addition, images of these datasets are pretty complex and have some similarities among different scenes, which make these datasets challenging. Some pictures of these datasets are shown in Figure 4.

4.2.1. UC Merced Land-Use Dataset [22]

UC Merced Land-Use Dataset (UCM) is a scene classification dataset containing 21 categories. This dataset includes agricultural, golf course, medium density residential, airplane, baseball diamond, beach, buildings, chaparral, storage tanks, dense residential, overpass, freeway, harbor, intersection, mobile home park, parking lot, river, runway, forest, sparse residential, and tennis courts. There are 100 RGB color images of the size 256 × 256 and 30 cm spatial resolution for each category.

4.2.2. Aerial Image Dataset [14]

Aerial Image Dataset (AID) includes bare land, dense residential, baseball field, bridge, center, church, railway station, commercial, viaduct, desert, storage tanks, farmland, forest, industrial, meadow, medium residential, beach, mountain, park, parking, playground, pond, port, resort, river, school, square, stadium, sparse residential and airport. This data set has 30 categories and 10000 images in all. Pixel resolution of each picture is 600 × 600.

4.2.3. OPTIMAL-31 Dataset

A remote sensing scene classification dataset named OPTIMAL-31 constructed by Qi Wang et al [38]. The dataset consists of 31 classes, each of which consists of 60 images with 256 × 256 pixels. It has 1860 pictures in total. The categories of the whole dataset are playground, airplane, airport, crossroads, runway ,basketball court, bridge, parking lot, lake, bushes, church, beach, mountain, round farmland, business district, roundabout, dense houses, desert, forest, freeway, golf field, harbor, factory, island, meadow, medium houses, mobile house area, overpass, railway, square farmland, and baseball field. The whole dataset contains many categories, but the number of each category is small, so the dataset is more challenging.

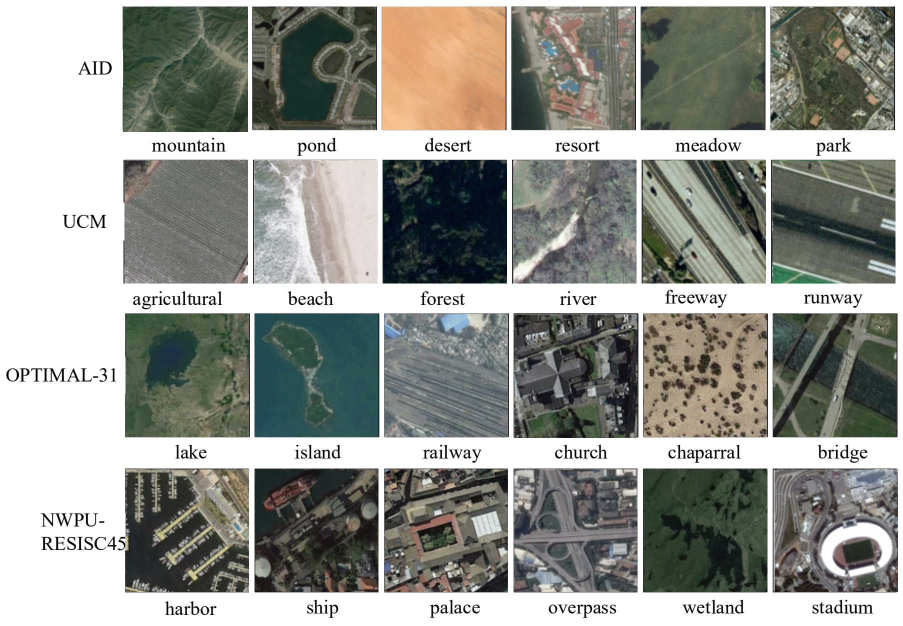

Figure 4. Some samples of AID, UCM, OPTIMAL-31, NWPU-RESISC45 datasets. From these samples, we can see that remote sensing scene image scene is complex and contains a lot of interference information, so remote sensing scene classification is a challenging task.

4.2.4. NWPU-RESISC45 Dataset [45]

The NWPU-RESISC45 dataset covers 45 scene classes, and each category has 700 images. The 45 scene classes include chaparral, airplane, airport, baseball, diamond, golf course, basketball court, beach, freeway, bridge, church, circular farmland, cloud, commercial area, rectangular farmland, river, dense residential, roundabout, desert, forest, palace, ground track field, harbor, sparse residential, industrial area, thermal power station, intersection, sea ice, island, lake, meadow, medium residential, mobile home park, storage tank, mountain, overpass, parking lot, railway, railway station, runway, ship, snowberg, stadium, tennis court, terrace and wetland.

5. Experiments

In order to evaluate the effectiveness of our network, we train and test our method on UCM, AID, OPTIMAL-31 and NWPU-RESISC45. Each dataset is divided into two parts in our experiments, one for training and the other for testing. We design seven experiments in all to evaluate our network. The ratios of training and testing on each data set are as shown in Table 2. In each experiment, we randomly generate training set and testing set to ensure the validity of the experiment. Note that the training set is only used for training, while the testing set is only used for testing.

Table 2. The setting of training and testing scales on each data.

Dataset	Train	Test
UCM	80%	20%
	50%	50%
AID	20%	80%
	50%	50%
NWPU-RESISC45	10%	90%
	20%	80%
OPTIMAL-31	80%	20%

5.1. The hyper parameter setting and the environment in experiments

In our experiments, we set train epoch to 400, each epoch will train all images of a training set. The batch size of each epoch is set to 80. After each two epochs, we will evaluate our network on the testing set once. To reduce memory usage, we resize the images used for training to the size of 224 × 224. Ones more, we initialize our network with pre-trained parameters trained on the ImageNet. Further, we use data augmentation technology to train a better model. All experiments are performed on a GPU server. The configurations of server are as follow: CPU Intel (R) Xeon (R) E5-2670 v3, 128 GB of memory, GeForce GTX TITAN X, 12GB of display memory. The experimental framework is pytorch0.4.1.

5.2. Reports of confusion matrixes

The following four obfuscation matrices are obtained by testing our network on AID, UCM, OPTIMAL and NWPU-RESISC45, respectively.

From Figure 5–8, we can see that most images can be correctly classified in our experiments. Especially in UCM dataset, only two images are misclassified. Among these experiments, the overall accuracy of NWPU-RESISC is the worst, only 94.98%. This is because there are many categories in the dataset while samples for training are few. For the case of misclassification, there are three main reasons.

1) The appearance is similar. The appearance of the two images shown in Figure 9(a) is very similar.

2) Texture is similar. As shown in Figure 9(b), two images are divided by roads, which causes a similar texture structure.

3) One category may contain information of other categories. As shown in Figure 9(c), there is a lake-like area in the wetland, it is disturbed when wetland image is classified.

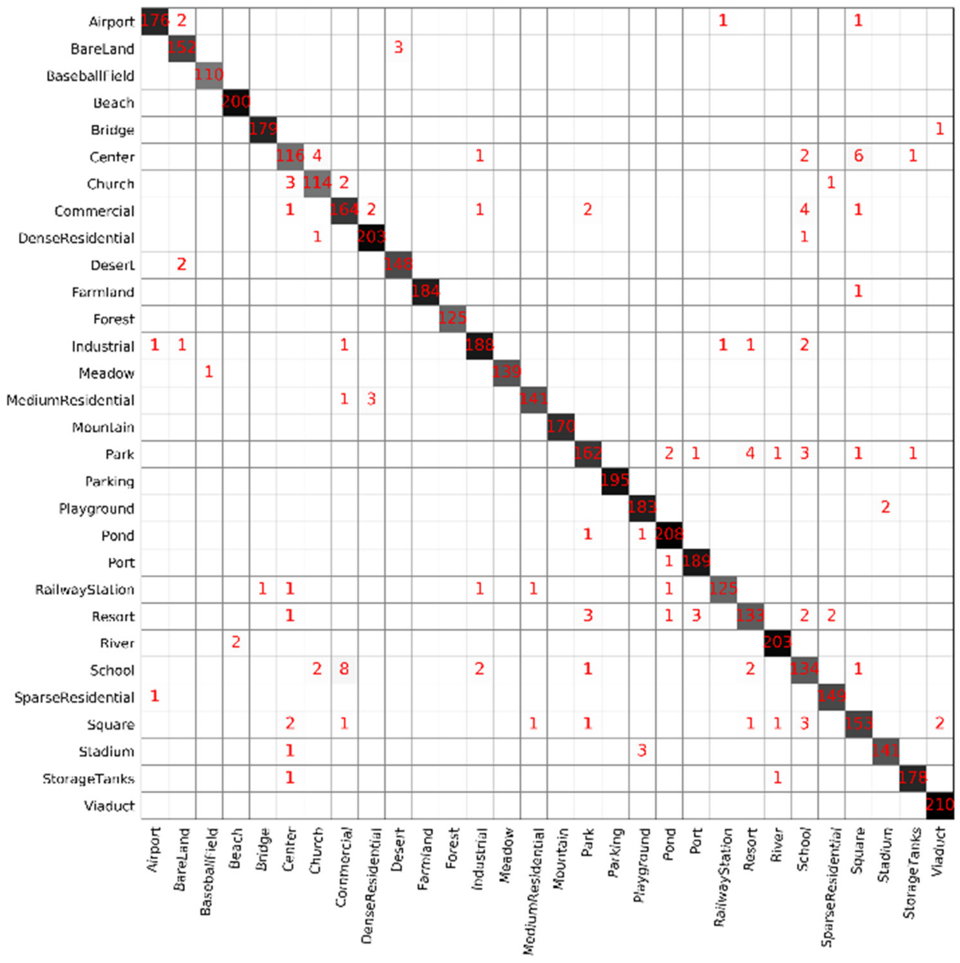

Figure 5. The confusion matrix with AID, 50% of dataset for training and 50% of the dataset for testing. The overall accuracy is 97.44%.

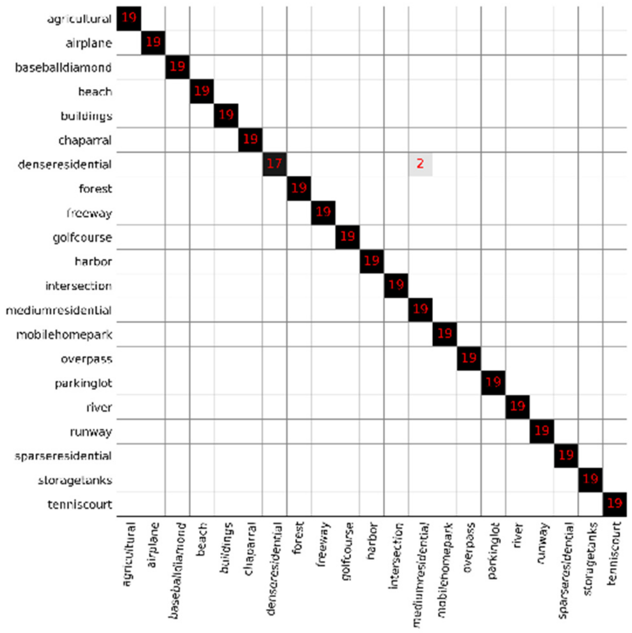

Figure 6. The confusion matrix with UCM, 80% of the dataset for training and 20% of the dataset for testing. The overall accuracy is 99.50%.

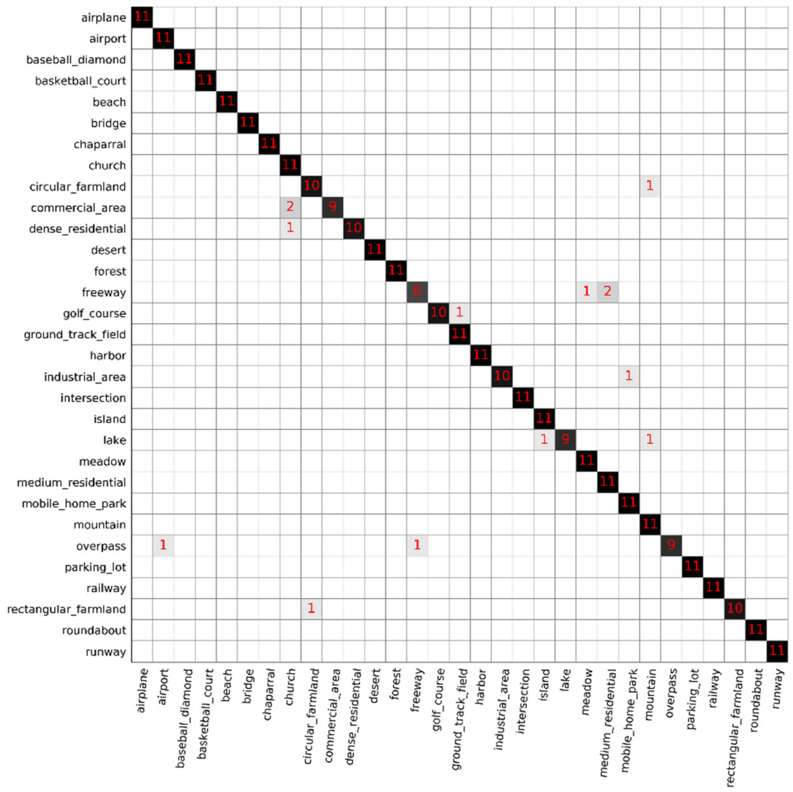

Figure 7. The confusion matrix with OPTIMAL, 80% of the dataset for training and 20% of the dataset for testing. The overall accuracy is 95.89%.

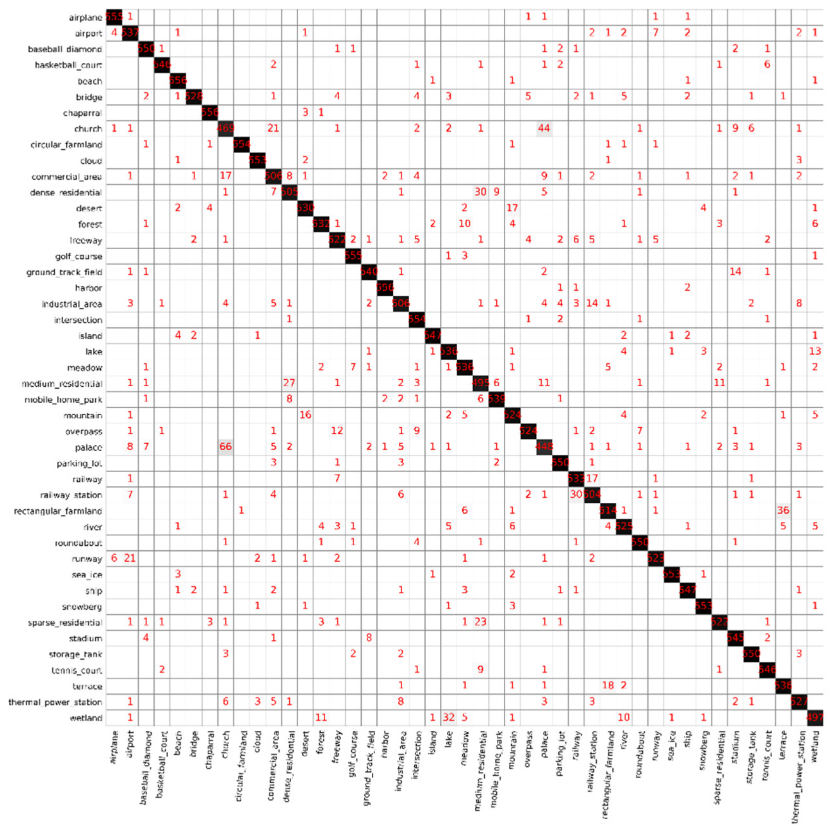

Figure 8. The confusion matrix with NWPU-RESISC45, 20% of the dataset for training and 80% of the dataset for testing. The overall accuracy is 94.98%.

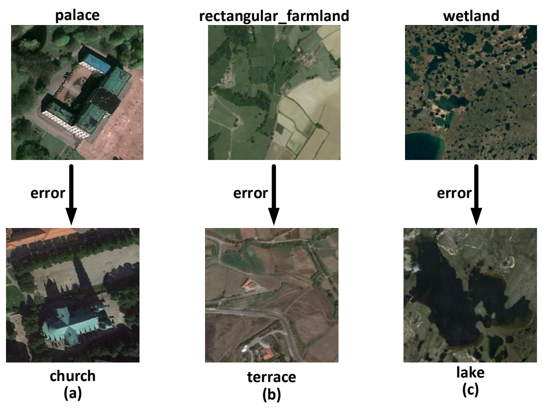

Figure 9. The cases of misclassification. In our experiment, the palace is misclassified into church easily, rectangular farmland is misclassified into terrace and wetland is misclassified.

5.3. Reports of experimental comparisons

To evaluate the performance of our algorithm, we compare it with some algorithms in recent years. From Table 3 to 6, we report the comparison results on UCM, AID, OPTIMAL-31 and NWPU-RESISC45 respectively.

From the Table 3–6, we can get the following conclusions.

1) CNN regard as an advanced feature extractor is suitable for classification of remote sensing scenes, and it has achieved a good accuracy. As shown in Table 3, with 80% of dataset for training, the accuracy of ARCNet-VGG16 [38] is 99.12 ± 0.40, while our algorithm achieves 99.50 ± 0.25, which improves by about 0.38. It is worth noting that the accuracy of those methods based on CNN are very close to 100% in many datasets. This fully demonstrates the efficiency of convolutional neural networks.

2) In contrast to various algorithms, our algorithm gets a state-of-art recognition rate. As shown in Table 3, we are 0.88 and 0.38 higher than the best algorithms at the training ratios 50% and 80% respectively. As shown in Table 4, we are 1.28 and 0.3 higher than the best algorithm at the training ratios 20% and 50% respectively. As shown in Table 5, we are 2.71 higher than the best algorithm at the training ratio 80%. As shown in Table 6, we are 3.68 and 3.06 higher than the best algorithm at the training ratios 10% and 20% respectively. This fully demonstrates the superiority of our algorithm and the excellent ability to extract features by dense connections.

3) For CNN, the more data used for training, the better results for test. The results are reported in Table 3 show that whether it is our method or other deep learning models, the training ratio used is different, and the accuracy of the test is different. It is worth noting that as the training ratio increases, the test accuracy is also improved.

Table 3. Overall accuracies and standard deviations (%) of different feature fusion methods on the UCM dataset.

Method	Training Ratios (%)	
	50%	80%
MS-CLBP + FV [46]	88.76 ± 0.76	93.00 ± 1.20
GoogLeNet [14]	92.70 ± 0.60	94.31 ± 0.89
CaffeNet [14]	93.98 ± 0.67	95.02 ± 0.81
VGG-VD-16 [14]	94.14 ± 0.69	95.21 ± 1.20
SalM3LBPCLM [47]	94.21 ± 0.75	95.75 ± 0.80
TEX-TS-Net(feature fusion model) [39]	97.55 ± 0.46	98.40 ± 0.76
SAL-TS-Net(feature fusion model) [39]	97.79 ± 0.56	98.90 ± 0.95
ARCNet-VGG16 [38]	96.81 ± 0.14	99.12 ± 0.40
ours	98.67 ± 0.42	99.50 ± 0.25

Table 4. Overall accuracies and standard deviations (%) of different feature fusion methods on the AID dataset.

Method	Training Ratios (%)	
	20%	50%
CaffeNet [14]	86.86 ± 0.47	89.53 ± 0.31
VGG-VD-16 [14]	86.59 ± 0.29	89.64 ± 0.36
GoogLeNet [14]	83.44 ± 0.40	86.39 ± 0.55
ARCNet-VGG16 [38]	88.75 ± 0.40	93.10 ± 0.55
D-CNN with GoogLeNet [36]	88.79 ± 0.10	96.22 ± 0.10
D-CNN with VGGNet-16 [36]	90.82 ± 0.16	96.89 ± 0.10
TEX-TS-Net(feature fusion model) [39]	93.31 ± 0.11	95.17 ± 0.21
SAL-TS-Net(feature fusion model) [39]	94.09 ± 0.34	95.99 ± 0.35
ours	95.37 ± 0.32	97.19 ± 0.23

Table 5. Overall accuracies and standard deviations (%) of different feature fusion methods on the OPTIMAL-31 dataset.

Method	Training Ratios (%)
	80%
Fine-tuning AlexNet [38]	81.22 ± 0.19
Fine-tuning GoogLeNet [38]	82.57 ± 0.12
Fine-tuning VGGNet16 [38]	87.45 ± 0.45
VGG-VD-16 [14]	89.12 ± 0.35
ARCNet-Alexnet [38]	85.75 ± 0.35
ARCNet-ResNet34 [38]	91.28 ± 0.45
ARCNet-VGGNet16 [38]	92.70 ± 0.35
ours	95.41 ± 0.61

Table 6. Overall accuracies and standard deviations (%) of different feature fusion methods on the NWPU-RESISC45 dataset.

Method	Training Ratios (%)	
	10%	20%
TEX-TS-Net(feature fusion model) [39]	84.77 ± 0.24	86.36 ± 0.19
SAL-TS-Net(feature fusion model) [39]	85.02 ± 0.25	87.01 ± 0.19
D-CNN with AlexNet [36]	85.56 ± 0.20	87.24 ± 0.12
D-CNN with GoogLeNet [36]	86.89 ± 0.10	90.49 ± 0.15
D-CNN with VGGNet-16 [36]	89.22 ± 0.50	91.89 ± 0.22
ours	92.90 ± 0.17	94.95 ± 0.04

5.4. Experimental skills

Literature [33] shows that making full use of the pre-training model can improve the classification accuracy of the network. In our work, one of the keys to improve the classification accuracy of our network is to initialize network with the pre-training parameters trained on ImageNet [13]. In our experiments, the parameters of the whole network (except the parameters of the convolution classifier) are trained on ImageNet first, and then are further fine-tuned on the remote sensing dataset. The other key point for our network to classify remote sensing images well is using data augmentation technology. As we all know, big data is the key to the success of deep learning. However, in the field of remote sensing, the number of images in the dataset is often small, because the images are difficult to obtain, and it is time-consuming to manually mark the corresponding scene categories. To solve this problem, data augmentation technology is introduced. In our experiments, we performed color transformation, brightness transformation, contrast transformation, and sharpness transformation on images. Besides those transformations, we rotated the image at random angles. The effect of data augmentation is shown in Figure 11. To demonstrate the effectiveness of pre-training models and data augmentation technique, we conduct experiments on various datasets. The report is shown in Figure 10. Note that when the network does not be initialized with pre-training parameters and does not use the data augmentation technology the classification accuracy of the network is very poor. But the accuracy will be greatly improved when using the pre-training parameters and the data augmentation technology. Thus, using pre-training parameters and data augmentation techniques is effective for classification and may lead to improved classification accuracy.

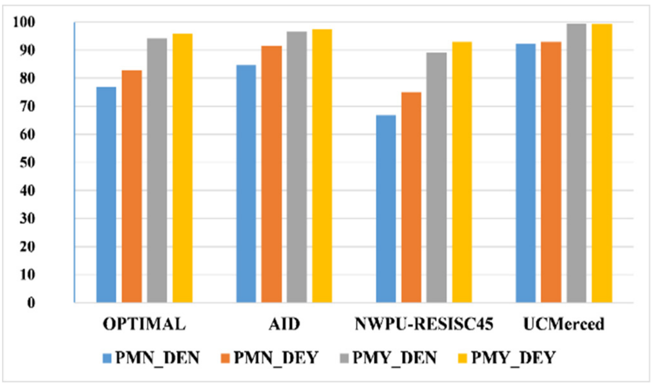

Figure 10. The experimental results of whether to use the pre-training parameters and data augmentation technology on dataset. PMN_DEN denotes that pre-training parameters and data augmentation technique are not both used. PMN_DEY denotes that pre-training parameters is not used but a data augmentation technique is used. PMY_DEN denotes that pre-training parameters is used but a data augmentation technique is not used. PMY_DEY denotes that pre-training parameters and data augmentation technique are both used. The datasets are split as follows: For OPTIMAL, 80% of dataset are treated as training set, while 20% are treated as testing set. For AID 50% of dataset are treated as training set, while the 50% are treated as testing set. For NWPU-RESISC45, 10% of dataset are treated as training set, while the 90% are treated as testing set. For UCM, 80% of dataset are treated as training set, while the 20% are treated as testing set.

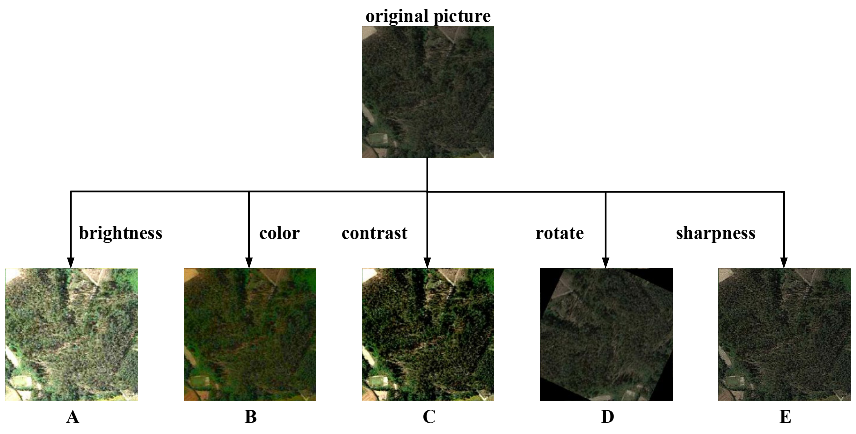

Figure 11. The category of the original picture is a forest. The pictures A, B, C, D and E obtained are changed by brightness, color, contrast, rotate and sharpness, respectively.

6. Conclusion

In this paper, we exploit the advantages of densely concatenated convolution to propose a full convolutional network for remote sensing scene classification. Our network is deep, but the parameters are few. As shown in Figure 2, these TOs directly contribute to network depth. The numbers of convolutional kernels of Conv(3 × 3) and Conv(1 × 1) are 32 and 128 respectively in DenseBlock, which produces few parameters, this is the reason that our network has few parameters. We use an adaptive average 3D pooling operation to down-sample the feature maps of the network output. The number of channel decrease to 512, which reduces the calculated consumption for the latter operations. And the size of the feature map is fixed, so that network can accept images with different sizes.

The experiment results on the several common datasets show that our fully convolutional network gets a state-of-art recognition rate.

Acknowledgments

The research work was funded by the National Natural Science Foundation of China grants nos. 61772454, 61811540410 and 61811530332, the Scientific Research Fund of Hunan Provincial Education Department Grant no. 16A008, the Industry-University Cooperation and Collaborative Education Project of Department of Higher Education of Ministry of Education Grant no. 201702137008, the Postgraduate Scientific Research Innovation Fund of Hunan Province Grant no. CX2018B565, the Undergraduate Inquiry Learning and Innovative Experimental Fund of CSUST Grant no. 2018-6-119, and the Postgraduate Training Innovation Base Construction Project of Hunan Province Grant no. 2017-451-30.

Conflict of interest

The authors declare no conflict of interest.

References

1. X. W. Yao, J. W. Han, G. Cheng, et al., Semantic annotation of high-resolution satellite images via weakly supervised learning, *IEEE Trans. Geosci. Remote Sens.*, **54** (2016), 3660–3671.
2. S. Y. Cui and M. H. Datcu, Comparison of approximation methods to Kullback–Leibler divergence between Gaussian mixture models for satellite image retrieval, *Remote Sens. Lett.*, **7** (2016), 651–660.
3. Y. B. Wang, L. Q. Zhang, X. H. Tong, et al., A three-layered graph-based learning approach for remote sensing image retrieval, *IEEE Trans. Geosci. Remote Sens.*, **54** (2016), 6020–6034.
4. J. Muñoz-Marí, F. Bovolo, L. Gómez-Chova, et al., Semisupervised one-class support vector machines for classification of remote sensing data, *IEEE Trans. Geosci. Remote Sens.*, **48** (2010), 3188–3197.
5. L. Y. Xiang, Y. Li, W. Hao, et al., Reversible natural language watermarking using synonym substitution and arithmetic coding, *Comput. Mater. Continua*, **55** (2018), 541–559.
6. Y. Tu, Y. Lin, J. Wang, et al., Semisupervised learning with generative adversarial networks on digital signal modulation classification. *Comput. Mater. Continua*, **55** (2018), 243–254.
7. D. J. Zeng, Y. Dai, F. Li, et al., Adversarial learning for distant supervised relation extraction, *Comput. Mater. Continua*, **55** (2018), 121–136.

8. J. M. Zhang, X. K. Jin, J. Sun, et al., Spatial and semantic convolutional features for robust visual object tracking, *Multimedia Tools Appl.*, Forthcoming 2018. Available at https://doi.org/10.1007/s11042-018-6562-8.
9. S. R. Zhou, W. L. Liang, J. G. Li, et al., Improved VGG model for road traffic sign recognition, *Comput. Mater. Continua*, **57** (2018), 11–24.
10. S. Karimpouli and P. Tahmasebi, Image-based velocity estimation of rock using convolutional neural networks, *Neural Netw.*, **111** (2019), 89–97.
11. S. Karimpouli and P. Tahmesbi, Segmentation of digital rock images using deep convolutional autoencoder networks, *Comput. Geosci-UK*, **126** (2019), 142–150.
12. P. Tahmasebi and A. Hezarkhani, Application of a modular feedforward neural network for grade estimation, *Nat. Resour. Res.*, **20** (2011), 25–32.
13. O. Russakovsky, J. Deng, H. Su, et al., ImageNet large scale visual recognition challenge, *Int. J. Comput. Vision*, **115** (2015), 211–252.
14. G. S. Xia, J. W. Hu, F. Hu, et al., AID: a benchmark data set for performance evaluation of aerial scene classification, *IEEE Trans. Geosci. Remote Sens.*, **55** (2017), 3965–3981.
15. J. M. Zhang, Y. Wu, X. K. Jin, et al., A fast object tracker based on integrated multiple features and dynamic learning rate, *Math. Probl. Eng.*, 2018 (2018), Article ID 5986062, 14 pages.
16. Y. Yang and N. Shawn, Comparing sift descriptors and gabor texture features for classification of remote sensed imagery, *15th IEEE International Conference on Image Processing*, (2008), 1852–1855.
17. B. Luo, S. J. Jiang and L. P. Zhang, Indexing of remote sensing images with different resolutions by multiple features, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, **6** (2013), 1899–1912.
18. A. Avramović and V. Risojević, Block-based semantic classification of high-resolution multispectral aerial images, *Signal Image Video Process.*, **10** (2016), 75–84.
19. X. Chen, T. Fang, H. Huo, et al., Measuring the effectiveness of various features for thematic information extraction from very high resolution remote sensing imagery, *IEEE Trans. Geosci. Remote Sens.*, **53** (2015), 4837–4851.
20. J. A. dos Santos, O. A. B. Penatti and R. da Silva Torres, Evaluating the potential of texture and color descriptors for remote sensing image retrieval and classification, *5th International Conference on Computer Vision Theory and Applications*, (2010), 203–208.
21. Y. Yang and N. Shawn, Geographic image retrieval using local invariant features, *IEEE Trans. Geosci. Remote Sens.*, **5** (2013), 818–832.
22. Y. Yang and N. Shawn, Bag-of-visual-words and spatial extensions for land-use classification, *18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, (2010), 270–279.
23. Y. Yang and N. Shawn, Spatial pyramid co-occurrence for image classification, *IEEE International Conference on Computer Vision*, (2011), 1465–1472.
24. W. Shao, W. Yang, G. S. Xia, et al., A hierarchical scheme of multiple feature fusion for high-resolution satellite scene categorization, *IEEE International Conference on Computer Vision Systems*, (2013), 324–333.
25. W. Shao, W. Yang and G. S. Xia, Extreme value theory-based calibration for the fusion of multiple features in high-resolution satellite scene classification, *Int. J. Remote Sens.*, **34** (2013), 8588–8602.

26. N. Romain, P. David and G. Philippe-Henri, Evaluation of second-order visual features for land-use classification, *12th International Workshop on Content-Based Multimedia Indexing*, (2014), 1–5.
27. L. J. Chen, W. Yang, K. Xu, et al., Evaluation of local features for scene classification using VHR satellite images, *2011 Joint Urban Remote Sensing Event*, (2011), 385–388.
28. F. Hu, G. S. Xia, J. W. Hu, et al., Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery, *Remote Sens.*, **7** (2015), 14680–14707.
29. M. Castelluccio, G. Poggi, C. Sansone, et al., Land use classification in remote sensing images by convolutional neural networks, preprint, arXiv:1508.00092.
30. O. A. B. Penatti, K. Nogueira and J. A. dos Santos, Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?, *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, (2015), 44–51.
31. F. P. S. Luus, B. P. Salmon, F. Van den Bergh, et al., Multiview deep learning for land-use classification, *IEEE Geosci. Remote Sens. Lett.*, **12** (2015), 2448–2452.
32. F. Zhang, B. Du and L. P. Zhang, Scene classification via a gradient boosting random convolutional network framework, *IEEE Trans. Geosci. Remote Sens.*, **54** (2016), 1793–1802.
33. K. Nogueira, O. A. B. Penatti and J. A. dos Santos, Towards better exploiting convolutional neural networks for remote sensing scene classification, *Pattern Recogn.*, **61** (2017), 539–556.
34. G. Cheng, P. C. Zhou and J. W. Han, Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images, *IEEE Trans. Geosci. Remote Sens.*, **54** (2016), 7405–7415.
35. X. W. Yao, J. W. Han, G. Cheng, et al., Semantic annotation of high-resolution satellite images via weakly supervised learning, *IEEE Trans. Geosci. Remote Sens.*, **54** (2016), 3660–3671.
36. G. Cheng, C. Y. Yang, X. W. Yao, et al., When deep learning meets metric learning: remote sensing image scene classification via learning discriminative CNNs, *IEEE Trans. Geosci. Remote Sens.*, **56** (2018), 2811–2821.
37. S. Chaib, H. Liu, Y. F. Gu, et al., Deep feature fusion for VHR remote sensing scene classification, *IEEE Trans. Geosci. Remote Sens.*, **55** (2017), 4775–4784.
38. Q. Wang, S. T. Liu, J. Chanussot, et al., Scene classification with recurrent attention of VHR remote sensing images, *IEEE Trans. Geosci. Remote Sens.*, **99** (2018), 1–13.
39. Y. L. Yu and F. X. Liu, Dense connectivity based two-stream deep feature fusion framework for aerial scene classification, *Remote Sens.*, **10** (2018), 1158.
40. Y. T. Chen, W. H. Xu, J. W. Zuo, et al., The fire recognition algorithm using dynamic feature fusion and IV-SVM classifier, *Cluster Comput.*, Forthcoming 2018. Available at https://doi.org/10.1007/s10586-018-2368-8.
41. G. Huang, Z. Liu, L. van der Maaten, et al., Densely connected convolutional networks, *IEEE Conference on Computer Vision and Pattern Recognition*, (2017), 4700–4708.
42. G. Huang, Y. Sun, Z. Liu, et al., Deep networks with stochastic depth, *European Conference on Computer Vision*, (2016), 646–661.
43. S. loffe and C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, *32nd International Conference on Machine Learning*, (2015), 448–456.
44. P. Tahmasebi, F. Javadpour and M. Sahimi, Data mining and machine learning for identifying sweet spots in shale reservoirs, *Expert Sys. Appl.*, **88** (2017), 435–447.

45. G. Cheng, J. W. Han and X. Q. Lu, Remote sensing image scene classification: benchmark and state of the art, *Proc. IEEE*, **105** (2017), 1865–1883.
46. L. H. Huang, C. Chen, W. Li, et al., Remote sensing image scene classification using multi-scale completed local binary patterns and fisher vectors, *Remote Sens.*, **8** (2016), 483.
47. X. Y. Bian, C. Chen, L. Tian, et al., Fusing local and global features for high-resolution scene classification, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, **10** (2017), 2889–2901.