Simplified Object-Based Deep Neural Network for Very High Resolution Remote Sensing Image Classification

Xin Pan[1,2,*], Ce Zhang[3,4], Jun Xu[2], Jian Zhao[1,2]

1. School of Computer Technology and Engineering, Changchun Institute of Technology, Changchun 130012, China

2. Jilin Provincial Key Laboratory of Changbai Historical Culture and VR Reconstruction Technology, Changchun 130012, China

3. Lancaster Environment Centre, Lancaster University, LA1 4YQ, U.K.

4. UK Centre for Ecology & Hydrology, Lancaster, LA1 4AP, U.K.

Abstract: For the object-based classification of high resolution remote sensing images, many people expect that introducing deep learning methods can improve then classification accuracy. Unfortunately, the input shape for deep neural networks (DNNs) is usually rectangular, whereas the shapes of the segments output by segmentation methods are usually according to the corresponding ground objects; this inconsistency can lead to confusion among different types of heterogeneous content when a DNN processes a segment. Currently, most object-based methods utilizing convolutional neural networks (CNNs) adopt additional models to overcome the detrimental influence of such heterogeneous content; however, these heterogeneity suppression mechanisms introduce additional complexity into the whole classification process, and these methods are usually unstable and difficult to use in real applications. To address the above problems, this paper proposes a simplified object-based deep neural network (SO-DNN) for very high resolution remote sensing image classification. In SO-DNN, a new segment category label inference method is introduced, in which a deep semantic segmentation neural network (DSSNN) is used as the classification model instead of a traditional CNN. Since the DSSNN can obtain a category label for each pixel in the input image patch, different types of content are not mixed together; therefore, SO-DNN does not require an additional heterogeneity suppression mechanism. Moreover, SO-DNN includes a sample information optimization method that allows the DSSNN model to be trained using only pixel-based training samples. Because only a single model is used and only a pixel-based training set is needed, the whole classification process of SO-DNN is relatively simple and direct. In experiments, we use very high-resolution aerial images from Vaihingen and Potsdam from the ISPRS WG II/4 dataset as test data and compare SO-DNN with 6 traditional methods: O-MLP, O+CNN, OHSF-CNN, 2-CNN, JDL and U-Net. Compared with the best-performing method among these traditional methods, the classification accuracy of SO-DNN is improved by up to 7.71% and 10.78% for single images from Vaihingen and Potsdam, respectively, and the average classification accuracy is improved by 2.46% and 2.91% for the Vaihingen and Potsdam images, respectively. SO-DNN relies on fewer models and easier-to-obtain samples than traditional methods, and its stable performance makes SO-DNN

more valuable for practical applications.

Keywords: CNN; very high resolution; semantic segmentation; classification; OBIA

1. Introduction

As an increasing number of high-resolution sensors are deployed on space-borne and airborne platforms, large amounts of high-resolution remote sensing image data are becoming available, and more users are able to access finer-resolution ground information from high-resolution remote sensing images (Li et al., 2016; Lu et al., 2016; Han et al., 2018). Classification methods can be used to automatically extract land use (LU) and land cover (LC) information and efficiently provide critical data for urban planning, socioeconomic management, and various agricultural and environmental applications (Zhu et al., 2019; Li et al., 2020a). However, because of the complexity of the information contained in high-resolution remote sensing images, pixel-based shallow classification models are usually inefficient in obtaining acceptable classification results (Chen et al., 2012; Ma et al. 2017). In object-based image analysis (OBIA), image segmentation methods are adopted to partition images into segments, and classification models are then used to classify each segment (Blaschke, 2010). In OBIA, image segments serve as the base units instead of individual pixels; since each segment has relatively homogeneous feature values; traditional classification models can obtain better results than is possible with pixel-based methods (Blaschke et al., 2014; Hossain et al., 2019; Shen et al., 2019).

For the shallow classification models adopted in OBIA, the input is usually a statistical summary of all pixels in a segment. However, intra-object spectral heterogeneity and inter-object homogeneity both increase with increasing image resolution, and object-level summaries will inevitably exaggerate these characteristics, ultimately leading to misrepresentation or misclassification (Sridharan et al., 2013; Gao et al., 2020). To overcome this problem, it is necessary to introduce additional morphological, boundary, and texture information of the segments into the classification process (Lv et al., 2019; Tang et al., 2020). In recent years, deep learning technology has achieved great success in the field of computer vision (LeCun et al., 2015). In particular, convolutional neural networks (CNNs) can extract high-level features from image patches and exhibit excellent representation/classification capabilities for the morphological, shape, texture and context information of objects (Alshehhi et al., 2017; Hu et al., 2018; Medley et al.; 2019; Osuna-Coutiño et al., 2020). Therefore, it is necessary to apply deep learning methods in OBIA to improve the classification accuracy.

For object-based classification using a CNN, it is necessary to cut the original image into patches based on the locations of segments and then input these rectangular image patches into the CNN; however, since the shapes of segments are usually not rectangular, the use of such image patches may introduce additional heterogeneous content into the CNN and decrease the segment classification accuracy. Currently, the common strategy is to introduce additional models to suppress the problems caused by such heterogeneous content. This strategy requires comprehensive consideration of the scale, the parameter relationships between the models, and the sample distribution; as a result, methods of this kind are complicated and difficult to design or use. To provide a simpler and more direct deep learning OBIA method, this paper proposes a simplified object-based deep neural network (SO-DNN)

for very high resolution remote sensing image classification. SO-DNN adopts the simple linear iterative clustering (SLIC) algorithm to segment remote sensing images and uses a deep semantic segmentation neural network (DSSNN) as a segment classification model. In addition, a fuzzy representation and optimization algorithm (FROA) is used to establish a bridge from pixel-based samples to patch-based samples; by virtue of this algorithm, through a process of iterative sample set improvement and progressive training, the DSSNN model of SO-DNN can be trained using a pixel-based training set. Because the DSSNN does not require an additional model to suppress heterogeneity, there is no need to consider any interactions among multiple models in SO-DNN, thus simplifying its implementation and usage. In experiments, SO-DNN has been compared with six other object-based shallow and deep methods, and the experimental results show that it is superior in accuracy to the other methods and less sensitive to the input scale. In summary, our work offers the following contributions:

1) The reason why the traditional structured CNNs cannot be directly used for object-based remote sensing classification is analyzed.

2) A DSSNN model is used to perform object-based classification, eliminating the need for additional auxiliary models and simplifying the process of heterogeneity suppression.

3) A training process based on iterative improvement is proposed, enabling the use of a pixel-based sample set for DSSNN model training and thus making the training set easier to obtain.

4) The proposed method is simpler and easy to use, and it uses fewer models and achieves greater accuracy and stability than traditional object-based remote sensing classification methods.

The remainder of this paper is organized as follows. Section 2 contains a review of related work. Section 3 analyzes the reason why a traditional CNN cannot be directly used for object-based segmentation. The method proposed in this paper is described in section 3. Section 4 presents experimental results. Conclusions are given in section 6.

2. Related Work

2.1 Remote sensing classification based on traditional structured CNN

For the processing of remote sensing images using CNNs or other deep learning models, scene classification is the most direct application. CNNs can extract high-level features of remote sensing images and achieve accuracies much higher than those of traditional shallow models (Xia et al., 2017; Zhao et al., 2020). By modifying the structures or training mechanisms of traditional CNNs, more discriminative features of scenes can be discovered, and the accuracy can be further improved (He et al., 2019; Li et al., 2020a). CNNs can achieve very high accuracy in scene classification. However, scene classification cannot yield pixel-wise classification results and consequently is not suitable for many LU/LC applications.

With suitable adjustment of the processing mechanism, traditionally structured CNNs can be used to perform per-pixel feature extraction or classification for remote sensing images. By improving a CNN's feature representation capabilities at object boundaries, more useful pixel-based features can be obtained, and the classification accuracy can be improved (Kumar et al., 2020; You et al., 2019). Through fusion with the pixel category decisions of shallow

models, CNNs can perform pixel-wise classification and achieve higher classification accuracy (Zhang et al., 2018a). However, the CNN-based per-pixel process will incur a large computational or I/O burden and lead to very slow speeds, even when processing small images; therefore, it is necessary to introduce superpixel- or object-based strategies to address these shortcomings (Zhang et al., 2018b).

2.2 Deep semantic segmentation

DSSNNs are widely used CNN models for obtaining pixel-wise classification results. DSSNNs possess excellent end-to-end characteristics and exhibit high accuracy. The majority of DSSNN models in the field of remote sensing semantic segmentation are based on fully convolutional networks (FCNs), SegNet or U-Net architecture. An FCN consists of a "fully convolutional" structure following traditional feature extraction layers and can produce arbitrarily/correspondingly sized semantic segmentation output (Long et al., 2015; Xia et al., 2019). By virtue of the introduction of decoder and encoder structures, SegNet can retain high-frequency details and obtain smoother object borders (Badrinarayanan et al., 2017; Jiang et al., 2020). With the U-Net structure, spatial information can be hierarchically introduced when performing pixel-wise segmentation, which allows DSSNNs to achieve better segmentation effects for objects with certain geometric shapes, such as buildings and roads (Ronneberger et al., 2015; Yang et al., 2019; Hao et al., 2020; Shao et al., 2021).

By virtue of their excellent end-to-end characteristics and high accuracy, when the training data are sufficient in number and quality, DSSNNs can outperform almost all other strategies (including the object-based strategy discussed in this article). Unfortunately, the following difficulties are still encountered when applying DSSNNs for remote sensing classification: DSSNNs require a large number of patch-based samples for training, and samples of this type are usually cut from ground truth images. Obtaining ground truth-images requires manual image interpretation, which is expensive and time consuming. At the same time, due to the regional characteristics of the ground content, samples from one region usually cannot be reused to train models intended to be applied in other regions. This problem makes it uneconomical to use DSSNNs for small LU/LC applications. Thus, DSSNNs are not suitable for all mapping application scenarios.

The training of DSSNNs strongly depends on massive quantities of patch-based samples, which is expensive and time consuming (Li et al., 2021). To alleviate the sample requirements, low-cost annotations such as points, scribbles or image-level labels can be used to improve the pixel-level spatial information of a sample set (Hua et al., 2021). The methods for generating image-level labels usually rely on weakly supervised strategies, including class activation maps, saliency-aided methods, object erasing techniques and region growing approaches to obtain pixel-level annotations (Luo et al., 2021; Zhou et al., 2016; Wang et al., 2018; Wei et al.; 2017). Points are then assigned to objects in a consistent and predictable way to obtain the objects' outlines, and through point-level supervision, more annotations can be obtained (Bearman et al., 2016). Through semi-supervised learning, pseudo-labels can be generated from unlabeled images or sparse ground truth labels (Tarvainen et al., 2017; Alonso et al., 2017). The above methods have achieved success in the field of computer vision, but most of these methods are intended to find potential clues for identifying objects (foreground) that are distinct from the background; however, when processing large remote sensing images, it is not easy to define the distinction between foreground and background. In addition,

semi-supervised or unsupervised samples can be used obtain the locations of objects, but it is unlikely that the accuracy of the object boundaries can be guaranteed; hence, the use of generated pseudo-samples may cause a DSSNN to produce more errors at boundaries. Therefore, the current semantic sample set improvement methods still need further adjustments to meet the needs of remote sensing classification.

2.3 Object-based classification based on CNNs

As described in section 2.2, DSSNNs are very powerful but not suitable for all applications, therefore, at present, strategies that are usually adopted for use with traditional shallow models are more practical. Specifically, pixel-based samples are manually selected from the image and used to train a classification model, which is then used to automatically classify the whole image; through this process, users can easily obtain a training set and results with relatively few hyperparameters to tune. The typical way to achieve this objective is to perform object-based classification

When performing object-based classification of high-resolution remote sensing images, even beginners will intuitively expect that the use of deep learning can improve the classification accuracy of OBIA. However, the actual situation is more complicated than what may be imagined; traditional CNNs cannot be directly used for object-based classification due to the heterogeneity of the input content (the reason for this problem will be discussed in detail in section 3), and therefore most "object-based + deep learning" methods need to use a combination of multiple models to suppress the influence of this heterogeneity. Through filtration and correction using shallow models, a CNN's object-based classification performance at boundaries can be improved (Pan et al., 2019; Zhang et al., 2020b). By integrating multi-scale information or multi-scale CNNs, deep learning methods can achieve high accuracy in the classification of specific objects (Zhao et al.; 2017; Zhang et al., 2018b; Martins et al, 2020; Chen et al., 2020). Using a combination of a multilayer perceptron (MLP) and a CNN, the results of LU and LC classification can be iteratively improved (Zhang et al., 2019; Zhang et al., 2020a).

Although the above methods can achieve high object-based classification accuracy, they have a common problem: they require the introduction of additional models, scales and parameters, which greatly increases the complexity of the entire classification process. This makes these methods difficult for ordinary users to control and challenging to use in practical mapping applications.

3. Obstacles to Object-Based CNN Classification

A large number of existing studies have shown that CNNs can extract high-level features from images and exhibit recognition capabilities that are significantly superior to those of shallow models; thus, intuitively, they should be able to improve the classification accuracy for object-based segments. However, traditionally structured CNNs cannot directly perform object-based segment classification. The reason is illustrated in Figure 1.

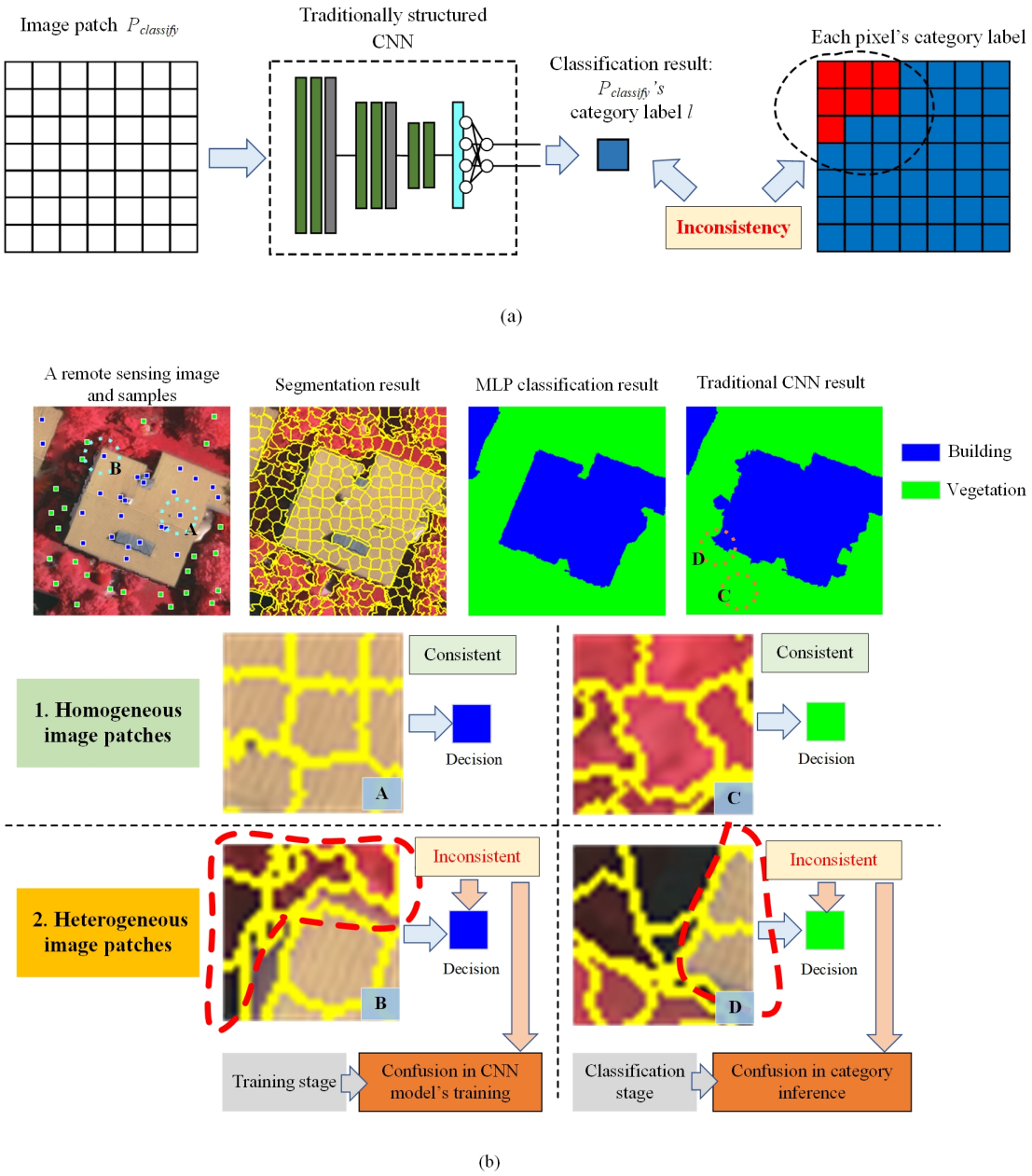

Figure 1. The reason why a traditional CNN cannot be directly used for object-based segmentation. (a) Inconsistency between a single category label and the category labels of individual pixels. (b) Examples of classification problems when a CNN processes heterogeneous image patches.

As shown in Figure 1(a), the input to a traditional CNN is an image patch $P_{classify}$, and the output is a category label l; this structure assumes that all pixels of $P_{classify}$ belong to the same category. However, the actual objects on the ground are usually not distributed strictly consistently with a pattern of square image patches, especially at the object boundaries, and consequently, the categories of some pixels may be inconsistent with l. This inconsistency will lead to classification problems, as shown in Figure 1(b).

As an example, we introduce a small high-resolution remote sensing image that contains image patches belonging to two categories: building and vegetation. Each category contains

25 samples. Since the two categories are markedly different in "color" (band value), this should be a very easy classification task. We perform object-based segmentation and then use two methods to classify the segments: an MLP, whose input is the average band value of a segment and whose output is a category label, and a traditional CNN, whose input is an image patch and whose output is a category label.

It can be seen from the results in Figure 1(b) that the MLP can correctly classify the majority of the segments and that the shapes or borders of the buildings are close to those of the real objects. For the traditional CNN, the expectation that deep learning will outperform the shallow model is not met; in contrast, there are many errors at the building boundaries, and the shape of the central building is completely destroyed, affected by a "crab-shape" phenomenon. This problem can be explained by the heterogeneity of the image patches. The image patches input into a CNN can be divided into two types:

(1) Homogeneous image patches: As illustrated in Figure 1(b), at location *A*, all of the image content belongs to the building category; similarly, at location *C*, all segments belong to the vegetation category. In this situation, the entire content of the image patch is consistent with the CNN's decision; thus, the image patch can provide comprehensive information to the CNN in either the training or classification stage, and the CNN's recognition ability can be improved.

(2) Heterogeneous image patches: At location *B*, although the sample category label is building, most of the content in the corresponding image patch is vegetation. Similarly, we expect location *D* to be classified as vegetation, but part of the content in the image patch belongs to the building category, and the brightness of the building pixels is higher than that of the vegetation pixels; consequently, the building content of the image patch is more obvious, which misleads the final decision of the CNN, resulting in an obvious error in the building's border. Under these conditions, part of the content in the image patch is inconsistent with the final decision. Such inconsistency has multiple detrimental effects: in the training stage, it will cause confusion in the classification behavior of the CNN and reduce the original classification ability, while in the classification stage; more obvious objects will sometimes mislead the model's decisions, resulting in incorrect classification.

Thus, heterogeneous image patches can lead a CNN to make incorrect decisions for segments, especially at boundaries. Consequently, if a traditional CNN is directly used to classify segments, some roads and buildings with straight boundaries will be misclassified in the resulting classification image, and the "crab-shape" phenomenon will appear.

For the above reasons, the existing object-based CNN methods mostly focus on suppressing heterogeneity: introducing a shallow model before the image patch is input into the CNN, filtering and replacing possible inconsistent contents, or using additional models or iterative processes to correct the CNN's classification results at object borders (Pan et al., 2019; Zhang 2019). Although the existing methods can yield reasonably good results in experiments, they face two significant challenges:

(1) High complexity of application: Current deep learning methods for object-based classification assume that it is possible to find one or more shallow models that can recognize heterogeneous image patches at boundaries and correct the CNN's results. However, different samples and different parameters will strongly affect this recognition capability. Therefore, methods of this type have very restrictive requirements for users; users need to understand the

relationships among the model parameters, carefully design the sample locations accordingly (either too many or too few boundary samples will result in unacceptable performance), and perform many trial-and-error experiments. This makes it very difficult for ordinary users to use such methods.

(2) The scale dilemma: High-resolution remote sensing images often have highly similar band values among different categories. Therefore, the shape and neighborhood information of objects within a broad range is needed to improve the classification ability. Unfortunately, the use of larger image patches will also lead to greater content heterogeneity, making it increasingly difficult to identify and process this heterogeneity; therefore, the scale of the image patches that can be processed using the existing methods is usually limited. At the same time, due to the introduction of multiple convolutional and pooling layers, deep neural networks (DNNs) usually exhibit good scale/size adaptability; according to the results of previous research in the computer vision field, a DNN can correctly recognize objects even when they exhibit variations in size or distance, within a certain range. Thus, it should be the case that deep learning methods are somewhat related to but not highly sensitive to the image scale; however, the results of papers on object-based remote sensing classification using CNNs always show that the input image patch scale/size is closely related to accuracy, with variations in scale leading to enormous accuracy differences. These problems are all caused by the "imperfect" capabilities of the additional models introduced to deal with content heterogeneity.

The above two problems make the existing object-based CNN classification methods difficult to use in practical applications. Therefore, there is a need to propose a more robust and simpler method that uses as few models and parameters as possible to achieve object-based classification and attempt to simplify the experimental process to allow deep learning technology to reach its full potential in object-based classification.

4. Methodology

4.1 Overall idea of the proposed method

4.1.1 A deep classification method that does not require heterogeneity suppression

As seen from the analysis in section 2, a traditional CNN attempts to assign a single label to all pixels in a single input image patch, which will inevitably lead to problems in the training and classification stages when performing object-based remote sensing classification. It would be extremely difficult to obtain a "perfect" model that is simultaneously powerful enough to correctly process complex heterogeneous content and simple enough to not add additional complexity to the entire classification process; therefore, our paper does not attempt to propose such a "perfect" auxiliary model.

Instead, DSSNNs offer us new inspiration. Given an input image patch, a DSSNN assigns each pixel its own category label, meaning that the inference results for different types of content will not be mixed with each other. Because of this characteristic, no additional models are required to solve the problem of heterogeneous content when using a DSSNN. Our concept for object-based classification based on a DSSNN is illustrated in Figure 2.

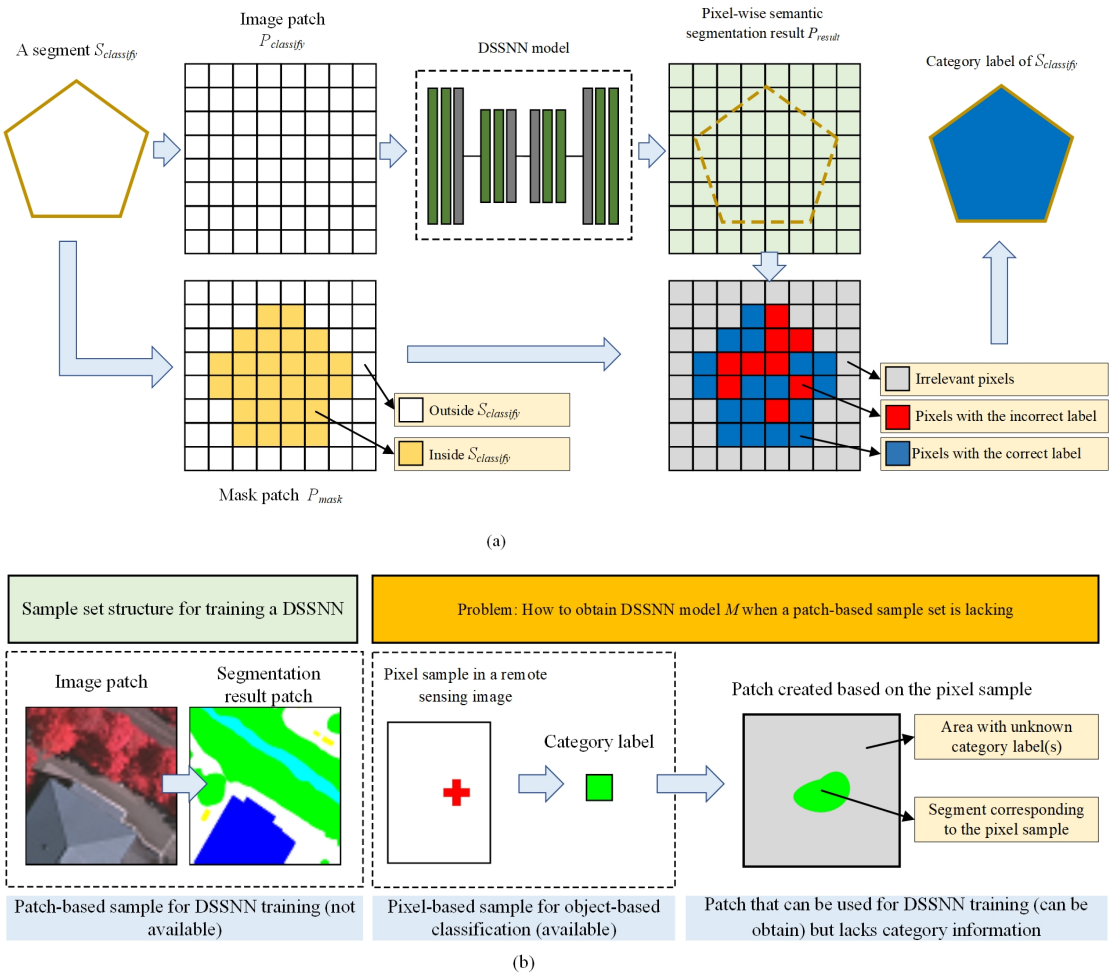

Figure 2. The concept of the proposed method. (a) Classification of a segment using a DSSNN. (b) The problem of how to train the DSSNN.

As shown in Figure 2(a), for a segment $S_{classify}$ with its center point at *location*, based on *location*, we cut an image patch $P_{classify}$ from the remote sensing image. $P_{classify}$ can be classified by a DSSNN model M to obtain a pixel-wise semantic segmentation result P_{result}. Unlike in Figure 1(a), each pixel has a corresponding category label in P_{result}, so P_{result} does not have the inconsistency problem described in section 2. This allows us to directly handle content heterogeneity without introducing additional auxiliary models; thus, the complexity issues associated with such models also do not arise.

In contrast to the relatively strict requirements for semantic segmentation tasks, we do not require P_{result} to be a "perfect" segmentation result. Based on the pixel locations in $S_{classify}$, we can obtain a mask patch P_{mask} that divides the image patch into two parts: the part inside $S_{classify}$ (consisting of the pixels located inside $S_{classify}$) and the part outside $S_{classify}$ (consisting of the pixels located outside $S_{classify}$). Then, using P_{mask}, the pixels in P_{result} can be further divided into three groups:

(1) Irrelevant pixels: Pixels that are outside $S_{classify}$, meaning that their category labels are unrelated to the category of $S_{classify}$.

(2) Pixels with the incorrect category label: Pixels that are inside $S_{classify}$, but their category labels are incorrect.

(3) Pixels with the correct category label: Pixels that are inside $S_{classify}$ and their category labels are correct.

The category label of $S_{classify}$ can be determined from the dominant label among the pixels that are inside $S_{classify}$. If the number of pixels in group (3) is greater than that in group (2), then the correct category label will be obtained for $S_{classify}$. This reduces the requirements for P_{result} and the DSSNN model M, as we do not need M's output to be correct for every pixel; as long as mostly correct results can be obtained for the central area of the image patch, this will be sufficient to obtain the correct category label for $S_{classify}$.

4.1.2 Overall process of the proposed method

According to the analysis in the above section, as long as we can obtain a suitable DSSNN model M, we can perform a new deep learning task for the object-based classification of remote sensing images that does not require a heterogeneity suppression mechanism. However, to obtain M, we must confront the following challenges:

(1) **No training set of ground truth image patches**: Training a DSSNN model requires samples with the structure shown in Figure 2(b): an input image patch and a corresponding segmentation result patch. To obtain such patches, both the remote sensing image and the corresponding manually interpreted ground truth image are needed. However, for the object-based classification task, only pixel-based samples are available, which cannot be used to directly train a DSSNN.

(2) **Samples contain relatively limited information**: Pixel-based samples cannot be directly converted into patch-based samples. Even if we assign the corresponding category label to each segment that contains a pixel sample, the categories of most areas of the image patches will still be unknown, and since all pixels in an image patch contribute to the final decision of the DSSNN, this unknown information may have a detrimental influence on the DSSNN's output.

However, even if no real patch-based training samples are available, according to the previous description of the process of object-based classification using a DSSNN, we do not necessarily require a high-precision DSSNN model that can correctly classify all pixels; we need only to obtain a model that is sufficiently close to the correct model to meet the requirements of the process illustrated in Figure 2(a).

These non-strict requirements for M make it possible for us to train a DSSNN model using pixel-based samples. Based on the above objectives, we propose our SO-DNN method for high-resolution remote sensing image classification; the overall process of SO-DNN is illustrated in Figure 3:

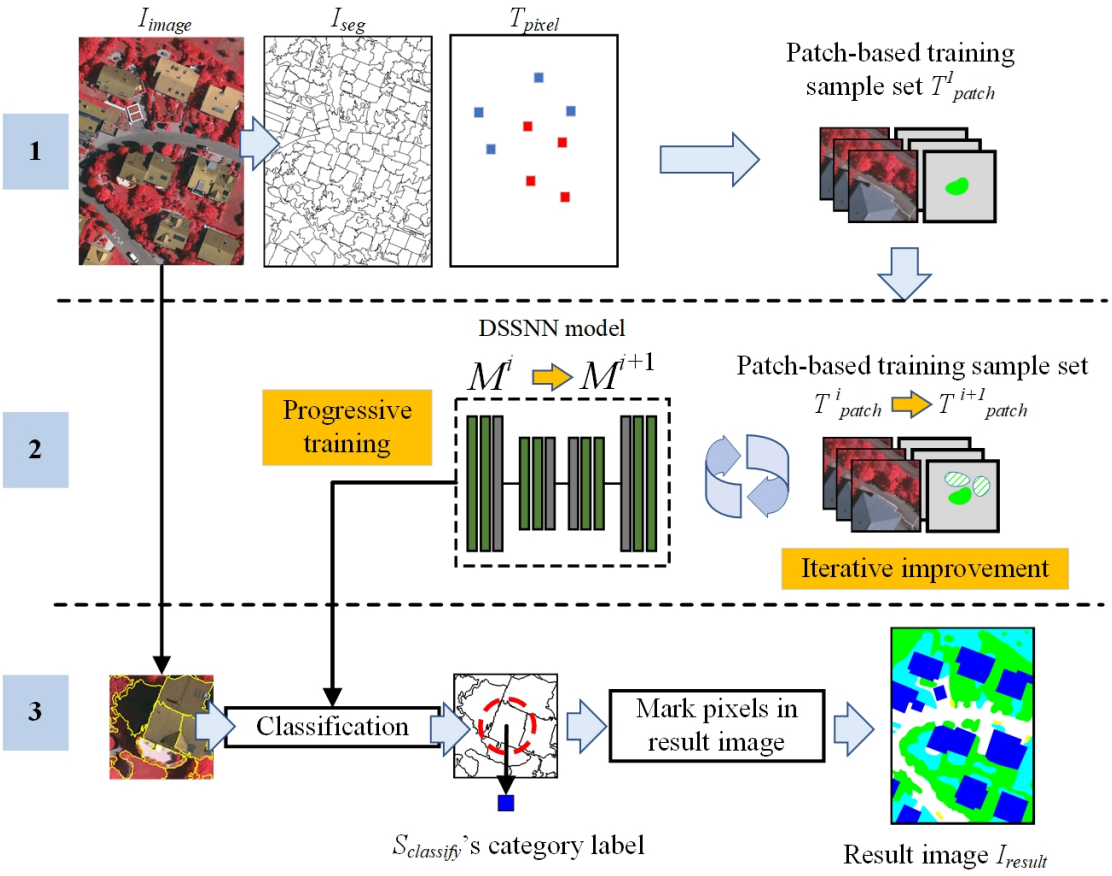

Figure 3. Overall process of SO-DNN.

As shown in Figure 3, the SO-DNN method involves three steps:

(1) Image preprocessing and initial patch-based sample set construction: The input to this step consists of a remote sensing image I_{image} and a pixel-based training sample set T_{pixel} (we have no ground truth image). First, I_{image} is segmented to obtain the segmentation result image I_{seg}; then, based on the locations of the samples in T_{pixel}, I_{image} is cut into image patches to construct the initial patch-based sample set T^{1}_{patch}.

For SO-DNN, we require that the image be over-segmented: ideally, there should be as few segments that cross the boundaries of different objects as possible, and each segment should unambiguously belong to a single object. Based on this standard, we adopt SLIC as an object-based segmentation algorithm that performs quickly and efficiently at object boundaries (Achanta et al., 2012).

(2) Iterative sample set improvement and progressive DSSNN model training: The DSSNN model M is established and trained using T^{1}_{patch}. Through an iterative improvement mechanism, M is used to improve the category information of the patch-based sample set, and the improved sample set is used to further train M. With this mechanism, the ability of M to determine the categories of objects is gradually enhanced.

(3) Segment classification: For each segment in I_{seg}, a corresponding image patch is cut from the image and classified using M. Based on the dominant category label in each segment, the category label of the segment is determined and written into the result image I_{result}.

For the above three steps, the input consists of a remote sensing image and pixel-based

training samples, and the final output consists of the classification results. This is consistent with the requirements of traditional shallow remote sensing classification methods. Moreover, only one classification model M is created in this process, and there is no need for any additional auxiliary correction process or the manual interpretation a ground truth image. This is consistent with the design target of a simple, direct and low-cost deep learning method for remote sensing image classification.

4.2 Image preprocessing and initial patch-based sample set construction

For SO-DNN, the input consists of a remote sensing image I_{image} and a pixel-based training sample set T_{pixel}. I_{image} contains N_{band} bands. The pixel-based training set is expressed as $T_{pixel} = \{t_1, t_2, \ldots, t_{Npixel}\}$, where N_{pixel} is the number of training samples and $N_{category}$ is the number of categories; for a sample $t_i = \{location_i, label_i, v_i\}$, $location_i$ is the sample's location in I_{image}, the value of $label_i$ ranges from 1 to $N_{Category}$ depending on the category of the sample, and v_i is a vector consisting of the band values of the pixel.

First, SLIC is applied as an object-based segmentation algorithm to obtain the segmentation result image I_{seg} (Achanta et al., 2012). I_{seg} is a single-band image of the same size as I_{image}; each pixel in I_{seg} has a segment ID value that indicates the segment to which the corresponding pixel in I_{image} belongs. For either a pixel or a segment, we introduce a category representation vector $d=\{d_1, d_2, \ldots, d_{Ncategory}\}$ to describe its category information. There are two possible cases for the vector d:

(1) **The category label is known:** There exists at least one training sample t located in segment S, therefore, the representation of S or the pixels in S can be determined based on t. In this case, the d_i in d can be represented as shown in formula 1:

$$d_i = \begin{cases} 1 & t.label = i \\ 0 & t.label \neq i \end{cases} \quad (1)$$

Formula 1 specifies a crisp representation. The values in d are either 0 or 1, where 0 means that the segment does not belong to the corresponding category and 1 means that it does belong to the corresponding category.

(2) **The category label is unknown:** There are no training samples located in S, and the category of S or the pixels in S is unknown. In this case, the d_i in d can be represented as shown in formula 2:

$$d_i=0 \quad (2)$$

In formula 2, all values in d are 0, which means that the corresponding category is unknown.

Based on formulas 1 and 2, we can represent the category information of each segment and create an initial patch-based sample set. The process is described in Algorithm 1.

Algorithm 1: Initial patch-based sample set construction (IPSSC)

Input: I_{image}, I_{seg}, T_{pixel}, W

Output: T^l_{patch}

Begin

$T^l_{patch}= \emptyset$;

foreach pixel-based sample t_i in T_{pixel}:

$l1$= the center point of the segment that contains t_i;

$SubI_y$=create a patch with a size of $W \times W$ and a number of bands equal to $N_{category}$;

$SubI_x$=cut a $W \times W$ image patch from I_{image}, centered at $l1$;

SubI2=cut a $W \times W$ image patch from I_{seg}, centered at *l1*;
foreach segment *s* in *SubI2*:
if there exists at least one sample located in *s*:
d=category vector represented as shown in formula 1;
else:
d= category vector represented as shown in formula 2;
ls= the *locations* of the pixels in *s*.
$SubI_y$ [*ls*]= *d*;
$T^l_{patch} \leftarrow [SubI_x, SubI_y]$;
return T^l_{patch};
End

Based on each sample in T_{pixel}, the IPSSC algorithm creates a group of patch-based samples and constructs an initial patch-based sample set T^l_{patch}. In each sample in T^l_{patch}, $SubI_x$ is a remote sensing image patch with a size of $W \times W$, and $SubI_y$ is the patch-based label corresponding to $SubI_x$. Although $SubI_y$ contains less category information because only a small proportion of the segments have labels, T^l_{patch} is consistent with the input/output form of a DSSNN and can be used for preliminary DSSNN training.

4.3 Iterative sample set improvement and progressive DSSNN model training

4.3.1 Structure of the deep network model for SO-DNN

In the SO-DNN method, the structure of the DNN *M* is derived from the standard U-Net model (a typical DSSNN model). The structure of *M* is shown in Figure 4.

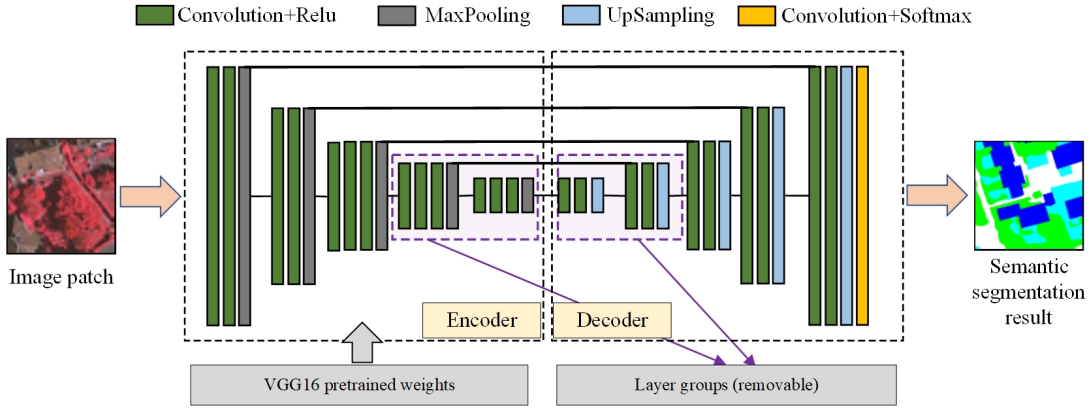

Figure 4. Structure of the DNN model for SO-DNN.

As shown in Figure 4, the model *M* has two components: an encoder and a decoder. The encoder component extracts spatial features from the input image patch in a hierarchical manner, while the decoder obtains the semantic segmentation result for the image patch. The layers in the encoder and decoder that correspond to the same scale group are directly connected to transfer spatial location information. The details of the layers in *M* are listed in Table 1.

Table 1. Details of the layers in *M*

Layer type	Layer details
Convolution+ReLU	2D convolutional layer, kernel size=3×3 and padding="same", activation function is the rectified linear unit (ReLU)

MaxPooling	2D maximum pooling layer, pooling size=2×2; through this layer, the feature map size will be halved
UpSampling	2D upsampling layer, size=(2, 2); through this layer, the feature map size will be doubled
Convolution+Softmax	2D convolutional layer, kernel size=1×1, number of output features=$N_{category}$, activation function is softmax

Based on pairs consisting of a MaxPooling layer (which reduces the feature map size) and an UpSampling layer (which increases the feature map size), all layers in M form groups with the following characteristics:

(1) VGG16-structured encoder: By default, as shown in Figure 4, the encoder contains 5 groups of layers with the same structure as that of the encoder in the VGG16 network model. This allows us to load the network with pretrained VGG16 weights, which can significantly improve the recognition ability of the encoder when there are few input image patches available.

(2) Relationship between group number and input image patch size: When the default 5 groups of layers are adopted in the encoder, since it contains 5 MaxPooling layers, the input patch size must be an integral multiple of 32 ($32=2^5$): 32, 64, 128 or 256. For an input size of less than 32, layer groups can be deleted from the encoder and decoder of M in pairs to achieve an input size of 8 (by deleting two groups) or 16 (by deleting one group). The model M is trained on T^i_{patch}, and it takes an image patch as input and outputs pixel-level semantic segmentation results.

4.3.2 Progressive improvement process

In the initial stage of SO-DNN training, we have only T^l_{patch}, generated by the IPSSC algorithm, as training data; we cannot expect the model M trained only on T^l_{patch} to have very good classification ability. Accordingly, SO-DNN faces several problems, as shown in Figure 5.

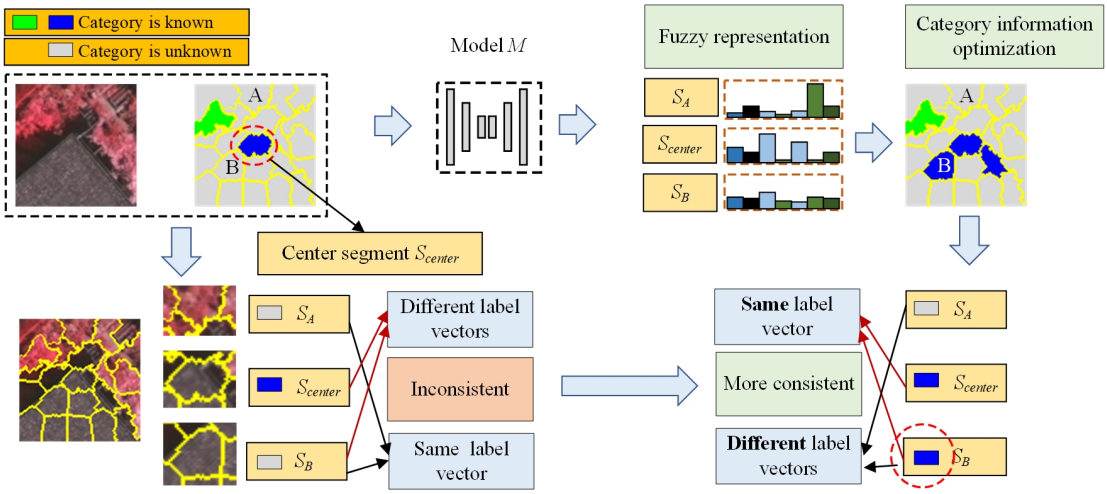

Figure 5. Limited category information in T^l_{patch} and sample optimization.

As shown in Figure 5, a training sample t_i consists of image patches $SubI_x$ and $SubI_y$; based on the segment labels in I_{seg}, the pixels in $SubI_x$ and $SubI_y$ can be grouped into segments. In $SubI_y$, the segments may have two types of labels: the category of a segment may be known (for

segments that contain pixel-based samples), or the category may be unknown. Based on the IPSSC algorithm, there will be a segment S_{center} located in the center of the patch, and the location and size of S_{center} determine the position of $SubI_x$ and $SubI_y$ in the whole remote sensing image. As mentioned in section 2.2, the model M needs to focus on S_{center} and attempt to classify it correctly. At this point, although S_{center} already has a category label, t_i is far from being a good-quality training sample; in particular, the relationships between S_{center} and the surrounding segments exhibit obvious conflicts. For example, in Figure 5, there are two segments, S_A and S_B. Clearly, S_A belongs to the tree category, and S_B belongs to the building category. The contents of S_B and S_{center} are very similar, but their label vectors are different, and the contents of S_A and S_B are very different, but their label vectors are the same; this conflict will have a detrimental impact on the training of M.

Although M cannot perform the classification task well when trained only on T^l_{patch}, M can at least be used to obtain a representation of the categorical nature of each segment. For a segment S, a fuzzy representation of its categorical nature can be expressed as the mean of the output of M:

$$fuzzyvector(S) = mean(\text{softmax layer output of } M \text{ for all pixels in } S) \quad (3)$$

Meanwhile, the difference between two fuzzy representations can be represented as follows:

$$diff(v_1, v_2) = l2norm(v_1, v_2) \quad (4)$$

If *diff* is large, this means that the two segments corresponding to v_1 and v_2 are unlikely to belong to the same category; otherwise, they may belong to the same category. We can introduce a category threshold value a to judge whether *diff* is large or small. Accordingly, we present Algorithm 2 to optimize the content of $SubI_y$.

Algorithm 2: Fuzzy representation and optimization algorithm (FROA)
Input: $SubI_x$, $SubI_y$, M, a
Output: Optimized $SubI_y$
Begin
$SubI_{Predict}$=softmax layer output of M for $SubI_x$;
$SubI_{seg}$= $SubI_x$'s corresponding segment IDs;
$S_{center\text{-}predict}$ = the center segment in $SubI_{Predict}$;
v_{center}= $fuzzyvector(S_{center\text{-}predict})$;
$d1$=category vector of S_{center} represented as shown in formula 1;
$d2$=category vector represented as shown in formula 2 (all zero);
foreach segment S in $SubI_{seg}$:
    if S contains a pixel-based sample in T_{pixel}:
        continue;
    $S_{predict}$= segment in $SubI_{Predict}$
    $v_{predict}$= $fuzzyvector$ ($S_{predict}$);
    $difference$=$diff$($v_{predict}$, v_{center});
    if $difference$<a:
        $SubI_y$[locations of the pixels in S]=$d1$;
    else:

` $SubI_y$[locations of the pixels in S]=$d2$;

return $SubI_y$;

End

For an image patch sample, FROA calculates the difference between each segment and S_{center} and uses a category threshold value a to judge this difference; when the difference is less than a, the corresponding segment is marked with the same category information as S_{center}, whereas otherwise, formula 2 is used to mark it. The value of FROA is that it alleviates obvious conflicts between S_{center} and surrounding segments, thereby improving the inference ability of M for S_{center}.

Because T^1_{patch} is created directly from pixel-based samples, we also cannot expect M trained on T^1_{patch} to have perfect representation capabilities, consequently, it is difficult to optimize the samples and M in just one step. Instead, we use an iterative approach, as shown in Figure 6.

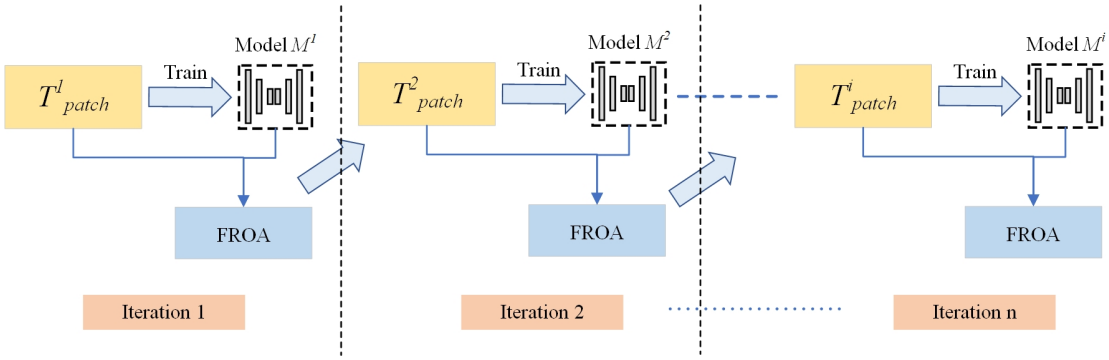

Figure 6. Progressive improvement process of the IISM algorithm.

As shown in Figure 6, using FROA as a bridge, we can execute an iterative and progressive improvement process: FROA is used to improve the training samples, and the improved samples are then used to retrain M, yielding an improved M that further enhances FROA's ability. The corresponding process is described in Algorithm 3.

Algorithm 3: Iterative sample set improvement and progressive DSSNN model training (IISM)

Input: T^1_{patch}, M, $N_{iteration}$

Output: Optimized model M

Begin

M^0=M;

for i in range(1, $N_{iteration}$):

M^i=use the training set T^i_{patch} to train model M^{i-1};

T^{i+1}_{patch}= ø;

foreach [$SubI_x$, $SubI_y$] in T^i_{patch}

$SubI_y$= $FROA$($SubI_x$, $SubI_y$, M^i, a);

T^{i+1}_{patch}←[$SubI_x$, $SubI_y$];

M=$M^{Niteration}$;

End

The IISM algorithm executes an iterative improvement process, using FROA to improve the category information in the patch-based sample set and further using the improved sample set to retrain M. In this way, the ability of M's to make decisions on the categories of objects is gradually enhanced.

4.4 Segment classification

The IISM algorithm introduced in section 3.3 solves a key problem—how to obtain the model M. With M, based on the idea presented in section 3.1.1 and Figure 2(a), we can proceed to classify the whole remote sensing image. For a segment $S_{classify}$ in I_{seg}, based on the locations of the pixels in $S_{classify}$, we can obtain a mask patch P_{mask} and apply Algorithm 4 to classify the remote sensing image.

Algorithm 4: Segment classification algorithm (SCA)
Input: I_{image}, I_{seg}, M, W
Output: Result image I_{result}
Begin
I_{result}= empty classification result image;
foreach segment $S_{classify}$ in I_{seg}
$P_{classify}$= cut a patch from I_{image} with the *location* and width W of $S_{classify}$;
P_{mask} = cut a patch from I_{seg} with the *location* and width W of $S_{classify}$;
P_{result}= predict $P_{classify}$ with model M;
$ls1$= locations of the pixels in P_{mask} for which the segment ID is equal to the segment ID of $S_{classify}$;
focusedpixels= $P_{result}[ls1]$;
l= dominant category label in *focusedpixels*;
$ls2$= locations of the pixels in I_{seg} for which the segment ID is equal to the segment ID of $S_{classify}$;
$I_{result}[ls2]=l$;
return I_{result};
End

With SCA, on the one hand, any heterogeneous content that may exist is effectively insulated during the classification process, and the category labels of these contents (regardless of their correctness) will not influence the final decision; on the other hand, the requirements for M are reduced. The model M does not need to produce perfect pixel-wise results; it needs only to correctly classify the majority of the pixels inside $S_{classify}$, which is a training goal that is easier to achieve. In this way, the category labels are written into I_{result}, and the whole classification process is completed.

5. Experiments and Results

5.1 Implementation of SO-DNN and methods for comparison

We adopted Python 3.7 to implement the SO-DNN method. For the deep learning; we adopted the Keras package with the TensorFlow backend; for image manipulation and access, we adopted the scikit-image package. All experiments were performed on a computer with an

Intel Core i7-10700F CPU, a GeForce RTX 1070 8 GB GPU and 32 GB of memory.

We considered 7 representative methods for comparison in the experiments:

(1) Object-based MLP (O-MLP): An MLP is a typical shallow classification model that takes the average band value of all pixels in a segment as input and outputs the category label of the segment. The MLP model contains three layers: an input layer, a middle layer and an output layer. The input layer contains 128 neurons and uses the ReLU activation function, the middle layer consists of 32 neurons and uses the ReLU activation function, and the output layer has a number of neurons equal to the number of classification categories and adopts the softmax activation function.

(2) Object-based + CNN (O+CNN): A traditionally structured CNN is used to classify the segments' corresponding image patches and obtain their category labels. The CNN model consists of three components: a feature extraction component, a Flatten component and an MLP component. The feature extraction component contains multiple groups of layers, each with the following structure: 2 Convolution layers (kernel size=3×3, padding="same" and activation function="ReLU") and 1 MaxPooling layer (pooling size=2×2). The number of groups= floor($\log_2$(size of input image patch/4); the maximum number of groups is 5, and the minimum number is 1. The Flatten component uses a Flatten layer to convert feature maps into vectors. The MLP component makes decisions based on the Flatten output, and its structure is the same as that of the MLP described in (1).

(3) Object-based and heterogeneous segment filter CNN (OHSF-CNN): OHSF-CNN is a typical method in which an additional model is used to preprocess inconsistent contents in image patches (Pan et al., 2019). OHSF-CNN consists of two models: a filter and a CNN. The filter uses a fuzzy neighborhood method with the neighbor threshold parameter equal to (number of samples)/10; for the CNN of OHSF-CNN, the same model described in (2) is adopted.

(4) Integration of two CNN models (2-CNN): This model uses two CNNs with different input sizes and can effectively solve the problem that a CNN tends to misclassify objects with certain specific shapes (Zhang et al., 2018b). 2-CNN consists of two models: a large-input-window CNN (LIW-CNN) and a small-input-window CNN (SIW-CNN). The LIW-CNN's input segments' corresponding image patch, with the input scale being equal to the size of an image patch; the LIW-CNN adopts the same structure described in (2). The SIW-CNN's input consists of smaller patches surrounding an ordinary image patch; the SIW-CNN's input scale can be selected in the range from 4 to (LIW-CNN's scale)/2, and the corresponding result with the best accuracy is selected as the final output result of the model. The SIW-CNN also adopts the same structure described in (2).

(5) Joint deep learning model (JDL): JDL is a typical method in which an additional model and iterative process are adopted to correct the final output (Zhang et al.., 2019). JDL uses an MLP and a CNN as classification models; the structure of the MLP is the same as that of the MLP described in (1), and for the CNN, the same model described in (2) is adopted. JDL simulates the iterative Markov updating process by alternately taking the MLP and CNN results as the input to the model, and the number of iterations is set to 10.

(6) Object-based + U-Net (U-Net): A U-Net is adopted as the classification model (described in section 4.3.1). Only the initial patch-based sample set T^{l}_{patch} generated by the IPSSC algorithm is used to train the model, and SCA is used as the classification algorithm.

(7) SO-DNN: SO-DNN is the method proposed in this paper, with the category threshold value set to a=0.5 and the number of iterations of the IISM algorithm set to $N_{iteration}$=3.

For methods (2) to (7) above, models with scales of 8, 16, 32, 64, 96 and 128 are adopted, and the scale that offers the highest accuracy is selected to obtain the final output.

5.2 Segmentation and experimental data

We selected two images from the semantic labeling contest dataset of the International Society for Photogrammetry and Remote Sensing (ISPRS) WG II/4(Rottensteiner et al., 2012). The experimental data can be seen in Figure 7.

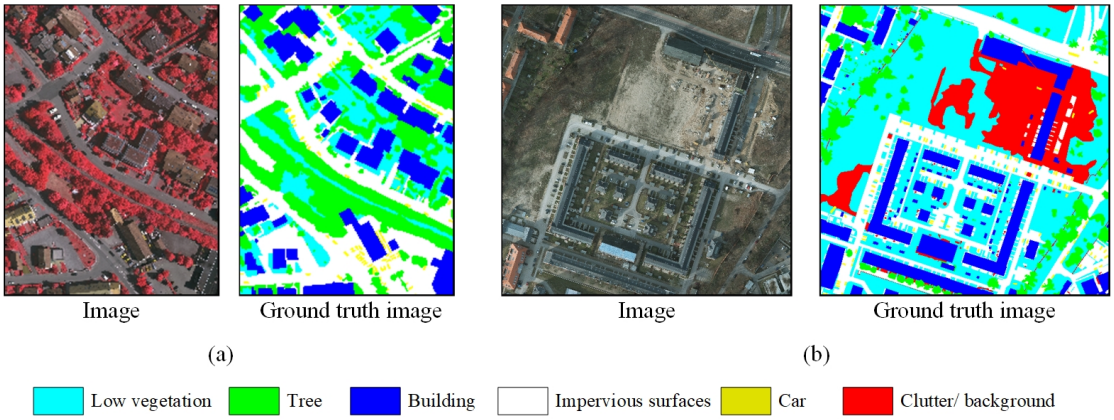

Figure 7. Experimental data. (a) test image and corresponding ground truth image from Vaihingen. (b) test image and corresponding ground truth image from Potsdam

As seen in Figure 7(a) and (b), test image 1 is from Vaihingen, and test image 2 is from Potsdam. The ground truth images contain 6 categories: low vegetation (LV), trees (T), buildings (B), impervious surfaces (IS), cars (C), and clutter/background (CB) (test image 1 does not contain clutter/background). We performed experiments with the following settings:

(1) Parameters of SLIC.

For test image 1, the segment number parameter of the SLIC algorithm was set to 20,000, and the compactness parameter was specified as 10; for test image 2, the number of segments was set to 25,000, and the compactness parameter was again specified as 10.

(2) Evaluation based on the ground truth image.

To evaluate the classification methods, we used the ground truth image as the test data, and all pixels in the result image that are not consistent with the corresponding pixels in the ground truth image are counted as incorrect pixels. The ground truth image can provide a pixel-level perspective when evaluating the accuracy of classification methods.

(3) Training sample set.

For each experimental image, we manually selected 200 pixels from each category to form the pixel-based sample set T_{pixel}. Therefore, the sample set for test image 1 consisted of 5×200=1,000 samples, and the sample set for test image 2 consisted of 6×200=1,200 samples.

(4) Test set and accuracy assessment.

To focus on the comparison of segment classification capability and further place strict criteria on the evaluation of the pixel-wise accuracy, we used the whole ground truth image as the test data for each test image. The overall accuracy (OA) of the results was calculated as follows:

$$Accuracy_{OA} = \frac{N_{correct}}{N_{image}} \quad (5)$$

where N_{image} is the total number of pixels in the ground truth image and $N_{correct}$ is the number of correctly classified pixels. For per-category evaluation, the accuracy for a particular category was calculated as follows:

$$Accuracy_{\mathrm{i}} = \frac{N_{correct\text{-}i}}{N_{image\text{-}i}} \quad (6)$$

where $N_{image\text{-}i}$ is the number of pixels of category i in the ground truth image and $N_{correct\text{-}i}$ is the number of correctly classified pixels of category i.

5.3 Iterative process and output of SO-DNN

5.3.1 Sample generation results in each iteration

To obtain the patch-based samples to train the DSSNN, SO-DNN adopts an iterative sample generation and training mechanism (the IPSSC and IISM algorithms). Examples of the generated samples for the two test images are shown in Figure 8.

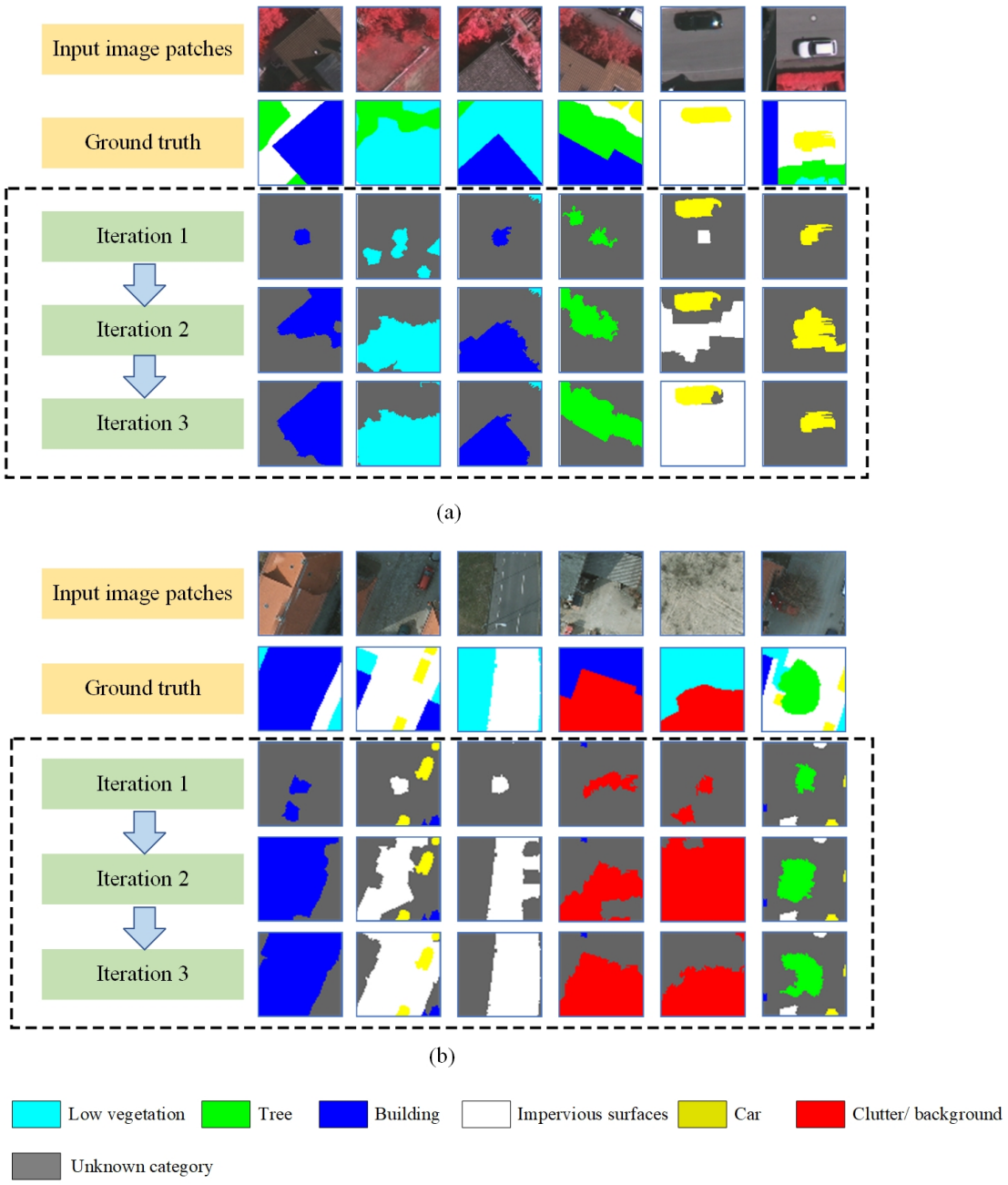

Figure 8. Details of the sample generation process. (a) generation process for test image 1. (b) generation process for test image 2.

As seen in Figure 8(a) and (b), SO-DNN took 3 iterations to generate the final patch-based training samples for both test images. All generated patches contain two types of content: (1) segments with known category labels, corresponding to six categories, which are used to drive the DSSNN to recognize these categories in image patches, and (2) segments with unknown category labels, marked in dark gray in the image patches, which are mainly used to assist the DSSNN in distinguishing between segments to be recognized and heterogeneous contents.

Throughout the 3 iterations, the category information for the contents of the generated patches gradually improved. For each image patch, there is a center segment that corresponds

to a specific category. In the 1st iteration, the category labels were derived only from the input pixel-based samples; we can see that for each image patch, only the center segment and segments containing pixel samples had category labels, and the other parts of the image patch were all assigned unknown labels. In the 2nd iteration, more areas similar to or consistent with the content of the center segment were marked as the same category; these areas were similar in size to the actual objects, but there were certain deviations in the boundary details. In the 3rd iteration, the boundary of the area consistent with the center segment was further refined, and its content approached that of the real ground truth image patch.

5.3.2 Classification results in each iteration

After each iteration i of sample generation in the IISM algorithm, we trained a model M^i and used SCA to classify the remote sensing image. The results are shown in Figure 9.

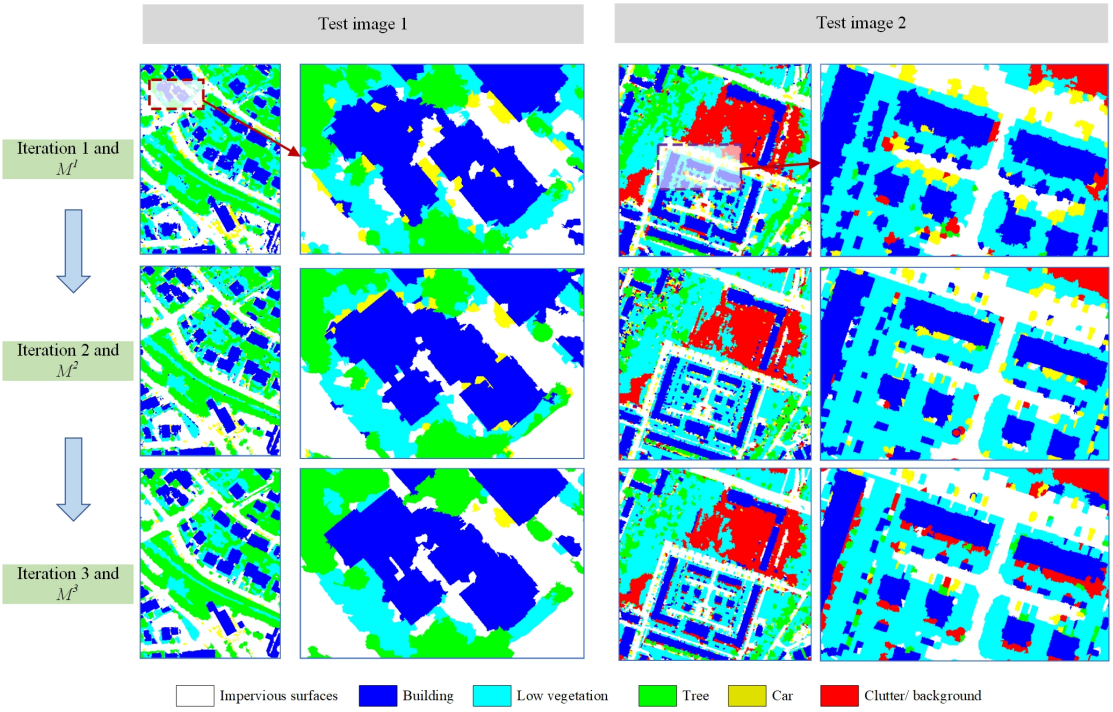

Figure 9. Classification results of SO-DNN in 3 iterations.

Figure 9 shows the classification results from all 3 iterations. For the test images, it can be seen that in the 1st iteration, although most objects were correctly classified, there was still obvious error blocks at the object boundaries, indicating that the classification ability of the DSSNN was not sufficient in this iteration. In the 2nd iteration, the classification results were significantly improved, especially at the object boundaries; in the 3rd iteration, the boundaries of the objects were further refined, some smaller objects could also be recognized, and better classification results were obtained. Three iterations of SO-DNN are executed by default, but more iteration can also be performed. The classification accuracy after 5 iterations is shown in Table 2.

Table 2. Classification accuracy after 5 iterations

Test image	OA in each of 5 iterations (%)				
	1	2	3	4	5

1	79.41	84.26	**88.14**	88.14	88.14
2	75.59	83.69	**87.03**	86.81	86.81

It can be seen from Table 2 that initially, as the iterations progressed, the performance on both test images gradually improved. After the third iteration, the SO-DNN method had achieved classification accuracies of OA=88.14% and 87.03% for test images 1 and 2 respectively. As the number of iterations continued to increase, however, the classification accuracy on the two images did not increase accordingly; for test image 1, the accuracy in iterations 4 to 5 remained unchanged, indicating that the classification model *M* and the samples were in a stable state, whereas for test image 2, the accuracy showed a slight decrease in iteration 4 and then remained unchanged. After 5 iterations, even if 6 or more iterations are performed, since the SO-DNN model is in a relatively stable state, the classification accuracy will always remain unchanged. These findings show that although the gradual improvement of the generated samples through the iterative process of SO-DNN improves the classification ability of the DSSNN, this improvement is concentrated in the first few iterations, and many iterations are not necessary.

5.3.3 Analysis of the iterative process of SO-DNN

As discussed in section 3, the category information (generated by the IPSSC algorithm) contained in the pixel-based samples is far less than that contained in the ground truth image, and using only this information to train the DSSNN will not be sufficient to endow the model *M* with the necessary classification capabilities.

As seen in Figure 9, the classification results obtained by our method in iteration 1 with model M^1 exhibit obvious errors at the boundaries, which shows that the model trained at this time does not perform sufficiently well in correctly recognizing segments. Nevertheless, we can observe that the model M^1 has some basic representation capabilities and will assign similar category labels to similar segments (regardless of whether these category labels are correct); this allows us to use M^1 in FROA to improve the contents of the training image patches and generate improved samples in iteration 2.

In iteration 2, we can see that in the generated training patches, segments consistent with the center segment are recognized by *M* and marked with consistent category labels, supporting higher recognition ability for heterogeneous content. The value of these samples can be seen from Figure 9: the boundaries of objects are clearly refined, and this refinement proves that the recognition and discrimination capabilities of M^2 are improved. The improvement of M^2 in iteration 2 is further manifested in the samples generated in iteration 3; the label information of these samples is closer to that of the ground-truth patches, and this improvement in the generated samples further improves the capabilities of the final model M^3, allowing the final classification results to reach a high accuracy.

In iterations 4 to 5, however, the accuracy does not continue to improve. This is mainly due to two reasons: 1) In our experiment, the model *M* is a DSSNN model that contains more than 23 million trainable weight parameters. This massive number of weights can easily "remember" (or be fit to) all of the details of the samples when the number of samples is small, consequently, the output of *M* becomes more prone to be simply a repeat of the output of the

previous iteration, causing the output of FROA to remain almost unchanged; As a result, the SO-DNN output converges and remains stable in more iterations. 2) The higher the classification accuracy is, the more difficult it is to improve. For these two reasons, it is not very valuable to perform too many iterations of the SO-DNN method, which is why we perform only three iterations by default.

The results in Figures 8 and 9 demonstrate that the iterative process of SO-DNN can gradually improve the information contained in the generated patch-based samples and the performance of the DSSNN model *M*. Although only a pixel-based sample set is initially available, through continuous iteration, better segment classification results can ultimately be obtained.

5.4 Comparison of methods

5.4.1 Comparison of classification results

For all of the compared deep learning methods, input scales of 8, 16, 32, 64, 96 and 128 were tested, and we selected the results with the highest accuracy as the final classification results. For test image 1, the selected scales for the various methods were as follows: 16×16 for O+CNN, 32×32 for OHSF-CNN and 2-CNN, 64×64 for JDL, and 96×96 for U-Net and SO-DNN. For test image 2, the selected scales were as follows: 64×64 for O+CNN, 32×32 for OHSF-CNN and 2-CNN, 64×64 for JDL, and 96×96 for U-Net and SO-DNN. The classification results are shown in Figure 10.

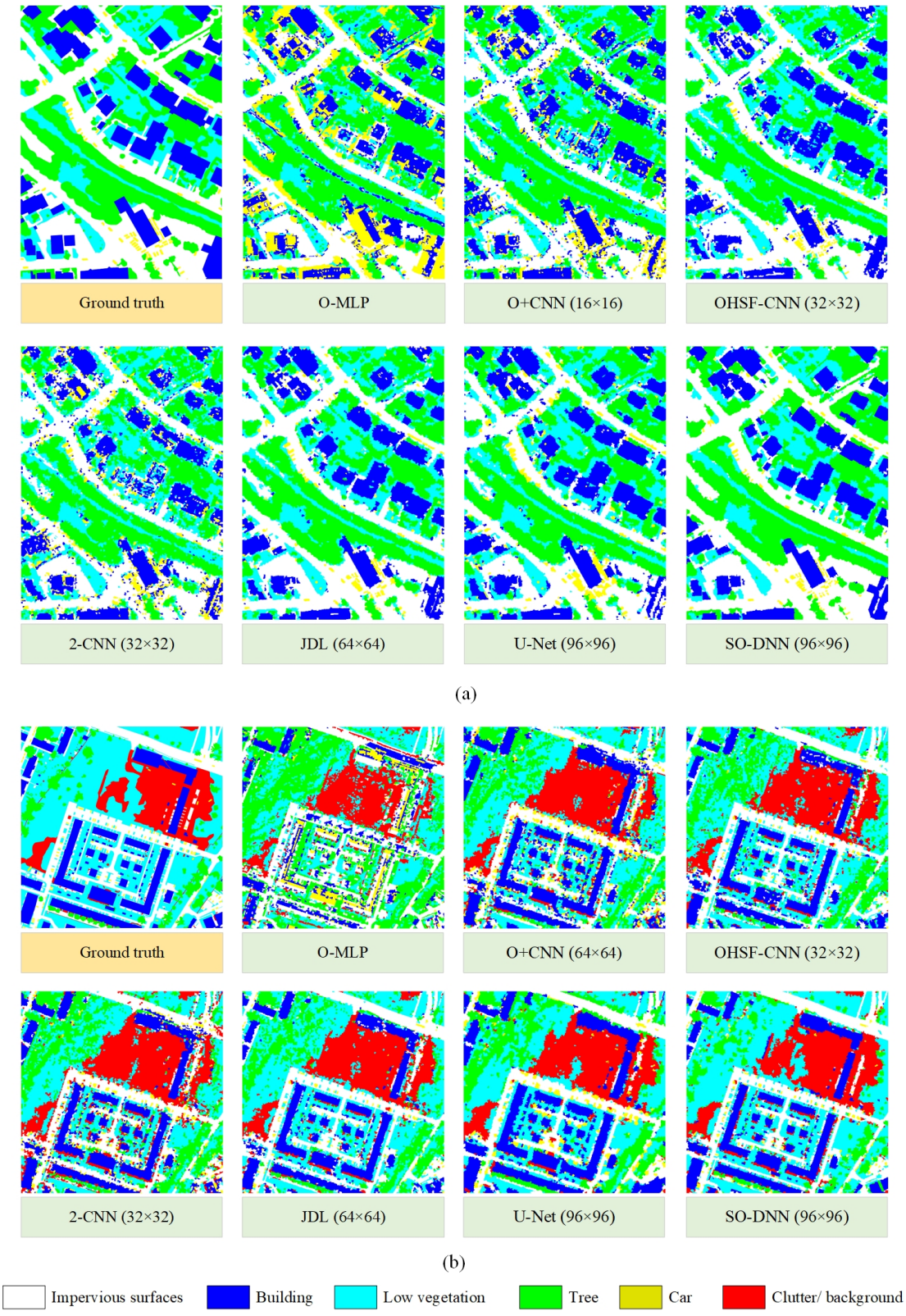

Figure 10. Comparison of the classification results of all the methods. (a) classification results for test image 1. (b) classification results for test image 2.

It can be seen from Figure 10 that for test image 1, the O-MLP result images contain obvious mistakes, with ground objects exhibiting obvious fragmentation. Due to the diversity of the

colors of cars and the lack of morphological information, many impervious surfaces and buildings are misclassified as cars. For O+CNN, because a relatively small input scale was selected for test image 1, the CNN does not effectively solve the problem of fragmentation. For test image 2, 64×64 was selected as the input scale for O+CNN, and consequently, the continuity of the results is better than that of the O-MLP; however, the expansion and deformation of some objects seriously affect its accuracy. By comparison, we can see from the result images for OHSF-CNN, 2-CNN, JDL, U-Net and SO-DNN, that a larger input scale can lead to better continuity; in particular, the SO-DNN results are closest to the ground truth.

For each test image, we also selected 3 representative locations to present more detailed comparisons of all methods, as shown in Figure 11.

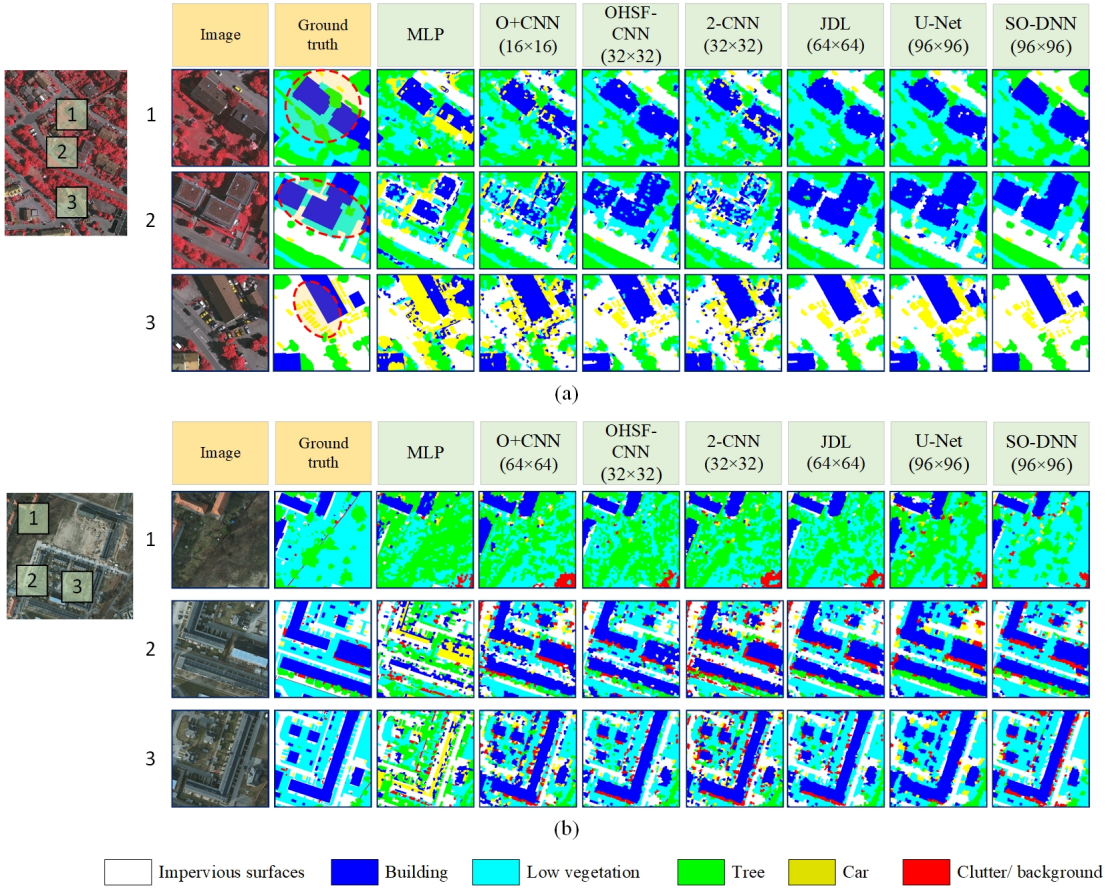

Figure 11. Detailed comparisons of all methods. (a) 3 representative locations in test image 1. (b) 3 representative locations in test image 2.

As seen from Figure 11(a), at location 1, although the color of the buildings is different from that of the surrounding objects, the building boundaries in the O+CNN results are not even as good as those in the O-MLP results; in the results of OHSF-CNN, 2-CNN, and JDL, shadow and roads are confused with buildings, which shows that these methods still have shortcomings in solving the problem of heterogeneity. At location 2, the color of the buildings is very similar to that of the low vegetation. To recognize these buildings, their shapes must be considered on a larger scale; consequently, we can observe that JDL, U-Net and SO-DNN, which can use larger-scale inputs, obtain better results. At location 3, a large number of cars of different colors affect the recognition of roads and buildings. At location, the classification

results of O-MLP are poor, O+CNN, OHSF-CNN, and 2-CNN show an insufficient ability to handle heterogeneity, leading to fragmentation of the results, whereas JDL, U-Net and SO-DNN can achieve a better balance between boundary classification and heterogeneity segmentation. From Figure 11(b), it can be seen that at location 1, U-Net and SO-DNN perform better than the other methods in recognizing trees and low vegetation. At locations 2 and 3, since the buildings are similar in color to some of the cars, the O-MLP recognition results show close to complete failure, while the deep learning methods can correctly recognize the buildings and SO-DNN achieves the best recognition results. For all of the above locations in Figure 11(a) and (b), SO-DNN performs more stably and correctly than the other six methods.

5.4.2 Analysis of classification results

When classifying high-resolution remote sensing images, it is clearly infeasible to perform classification based only on the segments' color (band) values; such a shallow classification model—O-MLP—cannot obtain good classification results. Therefore, it is necessary to introduce deep learning models to extract higher-level features from the segments to determine their category labels.

The typical dilemma when using a traditional CNN model to classify segments is as follows: on the one hand, it is necessary to adopt a larger input scale to obtain morphological and structural information about ground objects; on the other hand, adopting a larger scale directly increases the probability of heterogeneity, which will introduce new detrimental effects on the CNN model's inference results. Because O+CNN has no mechanism for addressing heterogeneity, its scale selection exhibits large fluctuations, and the results are not good; in some cases, it performs even worse than O-MLP. Both OHSF-CNN and 2-CNN rely on a shallow model or small-scale CNN for heterogeneity processing. Although they perform better than O-MLP and O+CNN, it can be seen from the results that their ability to handle heterogeneity is not strong. They reach their maximum classification accuracy at an input scale of 32×32, which directly leads to unsatisfactory performance at some locations. The iterative heterogeneity processing mechanism introduced in JDL seeks to gradually solve the problems introduced by heterogeneity; this iterative approach is superior to single-shot processing, and consequently, this method can yield better results than OHSF-CNN or 2-CNN.

In the U-Net model introduced in this paper, each pixel of an input image patch can be assigned its own category label, which naturally endows the model with the ability to separate heterogeneous content; therefore, the U-Net method does not need to rely on additional models for heterogeneity processing and can consequently handle much larger scales than the other methods. Although the U-Net does not include FROA or the IISM algorithm and uses only the most primitive pixel sample information, the quality of its results is close to that of JDL, thus providing evidence that the approach proposed in section 3.1 is feasible in practice.

On the basis of U-Net, SO-DNN further includes FROA and the IISM algorithm to gradually refine the information contained in the training samples such that the segmentation accuracy of the DSSNN model is also gradually enhanced; this allows SO-DNN to achieve a better balance between acquiring information at a larger scale and coping with heterogeneity, and consequently, it achieves the best results among the seven methods.

5.4.3 Comparison of classification accuracy and input scale

The classification accuracy of the seven methods is shown in Table 3.

Table 3. Classification accuracy comparison of the seven methods

Test image	Method	Best scale	OA (%)	Accuracy on each category (%)					
				LV	T	B	IS	C	CB
1	O-MLP	/	65.47	51.68	81.92	63.07	58.53	70.23	/
	O+CNN	16	71.46	60.34	86.36	72.56	62.67	74.89	/
	OHSF-CNN	32	80.20	73.38	83.32	83.07	79.05	82.90	/
	2-CNN	32	74.72	73.51	84.62	72.16	67.74	74.81	/
	JDL	64	82.20	77.97	84.22	86.39	79.99	85.60	/
	U-Net	96	79.41	77.62	83.19	87.65	71.87	**87.56**	/
	SO-DNN	96	**88.14**	**85.69**	**90.66**	**91.65**	**85.16**	86.50	/
2	O-MLP	/	51.14	40.10	69.06	34.91	71.27	36.32	63.87
	O+CNN	64	58.67	53.54	62.58	79.06	42.99	93.78	71.21
	OHSF-CNN	32	79.77	74.11	81.53	88.94	80.48	93.89	84.15
	2-CNN	32	75.28	70.84	76.13	86.22	71.00	94.70	81.58
	JDL	64	80.16	73.58	79.61	88.84	83.34	**95.75**	**85.38**
	U-Net	96	75.59	68.04	88.05	88.77	70.31	94.10	84.25
	SO-DNN	96	**87.03**	**86.26**	**94.81**	**90.52**	**85.07**	95.47	83.72

It can be seen from Table 3 that because O-MLP cannot adapt to high-resolution image data, its classification accuracy is very low; for test images 1 and 2, its accuracy is only 65.47% and 51.14%, respectively. For O+CNN, when 16×16 is selected as the input scale for test image 1, the accuracy is only 71.46%; when 64×64 is selected as the input scale for test image 2, the accuracy is only 58.67%. OHSF-CNN and 2-CNN are more accurate than O+CNN due to their mechanisms for processing heterogeneous content. JDL is superior in scale selection and accuracy. U-Net can tolerate an even larger scale while achieving classification accuracy close to that of JDL. Finally, SO-DNN achieves the highest classification accuracy: 88.14% on test image 1 and 87.03% on test image 2.

The input scale is a key factor in determining whether a deep learning method can recognize objects. The accuracy of each method for input scales of 8, 16, 32, 64, 96 and 128 is shown in Table 4.

Table 4. Accuracy of each method for input scales of 8, 16, 32, 64, 96 and 128

Test image	Method	Accuracy at each scale (%)					
		8	16	32	64	96	128
1	O+CNN	65.18	71.46	68.38	70.09	67.40	62.63
	OHSF-CNN	66.23	73.43	80.20	79.08	72.93	72.09
	2-CNN	65.27	72.12	74.72	72.65	67.38	66.32
	JDL	67.28	73.54	79.42	82.20	80.36	75.21
	U-Net	65.39	72.97	77.50	77.52	79.41	78.93
	SO-DNN	65.70	73.81	82.58	87.79	**88.14**	86.79
2	O+CNN	51.07	52.22	57.00	58.67	56.82	50.49
	OHSF-CNN	51.49	77.19	79.77	74.09	69.19	64.23
	2-CNN	51.81	73.25	75.28	72.08	74.54	62.01
	JDL	57.49	77.74	79.66	80.16	79.89	77.12
	U-Net	52.09	72.39	75.09	74.82	75.59	75.21
	SO-DNN	52.51	75.13	82.07	86.49	**87.03**	84.44

Corresponding to Table 4, a comparison between accuracy and scale is shown for each method in Figure 12.

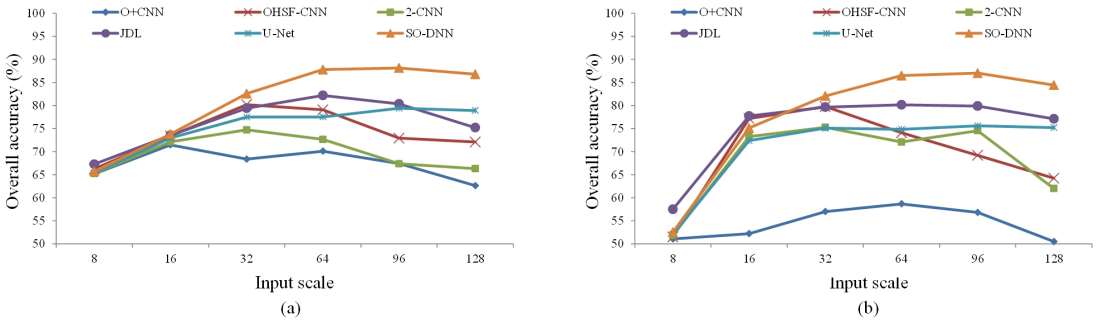

Figure 12. Comparison between accuracy and scale for each method. (a) comparison on test image 1. (b) comparison on test image 2.

It can be seen from Figure 12 that when the input scale is 8×8, none of the methods can take sufficient advantage of the capabilities of deep learning, so their accuracy is close to that of O-MLP. Because the size of the input segments is larger than the input convolution scale, the innovative processes in SO-DNN are ineffective, so U-Net and SO-DNN achieve the same accuracy at an input scale of 8×8. As the scale increases, the increase in accuracy for O+CNN is unstable, with the accuracy at 128×128 being even lower than that at 8×8. Because of the heterogeneity processing methods adopted in OHSF-CNN and 2-CNN, their accuracy shows an initial increase at small input scales; their best accuracy is reached at 32×32, and the accuracy subsequently decreases with any further increase in the input scale. JDL reaches its best accuracy at an input scale 64×64, making it superior to OHSF-CNN and 2-CNN. U-Net and SO-DNN reach their highest classification accuracy at 96×96 and show generally similar trends; at input scales of 64, 96 and 128, these two methods show no significant changes in classification accuracy, indicating that they are more adaptable to different scales.

5.4.4 Analysis of classification accuracy and input scale

In the process of remote sensing image classification using a traditional CNN, the use of a larger input scale can allow the model to extract higher-level features from a wider range of shapes, textures and neighborhoods; however, it will also lead to the introduction of higher heterogeneity, which will hinder training and classification.

For O+CNN, because it does not address heterogeneity, its accuracy fluctuates greatly. On the test images, there is no consistent pattern between scale and accuracy; at 128×128, due to the introduction of too many surrounding objects, the accuracy is lower than it is at 8×8.

Because of the heterogeneity processing methods adopted in OHSF-CNN and 2-CNN, their accuracy initially increases with an increasing input scale. However, their heterogeneity processing ability relies on another classifier, which introduces the additional difficulty of requiring collaboration between different models. When the input scale is larger than 32×32, the accuracy of both methods declines rapidly, indicating that when the heterogeneity exceeds a certain range, such a mechanism that relies on other classification models will fail; consequently, their improvement in accuracy is very limited.

JDL similarly uses an MLP for heterogeneity processing and additionally introduces an iterative mechanism to improve the heterogeneity processing capabilities; therefore, the scale at which JDL reaches its maximum accuracy is larger than that of OHSF-CNN and 2-CNN,

and its classification accuracy is also significantly better than that of O+CNN, OHSF-CNN and 2-CNN. However, as the scale increases to 128×128, its accuracy inevitably decreases significantly.

U-Net and SO-DNN use a different mechanism: they use the "semantic segmentation" network architecture of U-Net to naturally separate pixels into different categories, and consequently they do not need additional models to handle heterogeneity. This strategy simplifies the entire training and classification process, and the highest accuracy of these models is not reached until an input scale of 96×96; furthermore, at large scales (64, 96 and 128), the accuracy of these two methods does not strongly vary, indicating that this strategy is more suitable for large-scale input. U-Net still achieves high accuracy even when the segmentation information in the training data is insufficient, indicating that adopting DSSNS for object-based classification is indeed feasible. Furthermore, FROA and the IISM algorithm are additionally introduced in SO-DNN to optimize the segmentation information contained in the training data. This allows SO-DNN to achieve higher and more stable classification accuracy.

5.4.5 Comparison of classification accuracy for noisy or incomplete samples

To test the adaptability of the methods to noisy or incomplete samples, we further added incorrect category labels into the training sample set. From the original training sample set, we selected 5%, 10% and 15% of the samples in each category and changed their category labels to other random labels. The original overall classification accuracy (OA) and the decrease in accuracy (DA) compared with the original training sample set are listed in Table 5.

Table 5. Comparison of OA and DA for noisy or incomplete samples

Test image	Method	Noise=5%		Noise=10%		Noise=15%	
		OA(%)	DC(%)	OA(%)	DC(%)	OA(%)	DC(%)
1	O-MLP	65.73	-0.26	64.11	1.36	61.02	4.45
	O+CNN	71.02	0.44	69.41	2.05	66.96	4.50
	OHSF-CNN	79.37	0.83	77.09	3.11	73.85	6.35
	2-CNN	74.21	0.51	71.60	3.12	67.84	6.88
	JDL	81.48	0.72	80.21	1.99	**77.63**	4.57
	U-Net	78.88	0.53	76.93	2.48	74.29	5.12
	SO-DNN	**86.38**	**1.76**	**83.34**	**4.80**	77.54	**10.60**
2	O-MLP	51.01	0.13	49.39	1.75	46.92	4.22
	O+CNN	58.25	0.42	56.66	2.01	51.90	6.77
	OHSF-CNN	79.11	0.66	76.24	3.53	73.27	6.50
	2-CNN	74.53	0.75	71.82	3.46	67.67	7.61
	JDL	79.22	0.94	76.92	3.24	73.14	7.02
	U-Net	74.74	0.85	71.93	3.66	68.23	7.36
	SO-DNN	**85.05**	**1.98**	**81.64**	**5.39**	**75.45**	**11.58**

Table 5 shows that the accuracy of all methods decreases with the addition of noise. At the 5% noise level, although SO-DNN can still achieve the highest classification accuracy, its decrease in accuracy is the largest. The decreases in accuracy of O+CNN, OHSF-CNN and 2-CNN are all smaller than those of SO-DNN. Among the algorithms, O-MLP and O+CNN show the smallest decrease in accuracy. At 15% noise level, for test image 1, the accuracy of SO-DNN is lower than that of JDL, indicating that the optimization process of FROA has an

adverse effect in this case.

With the addition of noise, the classification accuracy of all models will inevitably decrease. O-MLP, O+CNN and U-Net each use a single classification model, and these single classification models themselves have the ability to tolerate errors from the perspective of the entire sample set; since O-MLP's classification accuracy itself is very low (a large number of categories are confused), at the 5% noise level, the classification accuracy even appears to increase (the samples causing confusion have changed in proportion). OHSF-CNN, 2-CNN and JDL all use multiple models, and the integration of more model decisions helps correct the errors in a single model.

Unfortunately, SO-DNN achieves poorer results in this respect. The reason is that the intention of FROA in SO-DNN is to expand the available information based on the known samples in an image patch. This makes FROA better able to adapt to the local characteristics of objects; however, when an incorrectly labeled sample is input into FROA, the starting point of the method is incorrect, and this error will be further propagated in subsequent iterations, introducing more errors into the subsequent U-Net training stage. In future research, we will focus on mitigating this problem by finding a better balance between adaptation to local and global characteristics to improve the noise adaptability of SO-DNN.

5.5 Impact of the segment number and compactness parameters of SLIC on SO-DNN

The segment number and compactness parameters of SLIC can greatly influence the segmentation results. To test the impact of these two parameters on SO-DNN, we set the segment number parameter to values of {5000, 10000, 15000, 20000, 25000, 30000, 35000}, and the compactness parameter to values of {1, 5, 10, 50, 100}. The influence of these parameters on the segmentation results is shown in Figure 13.

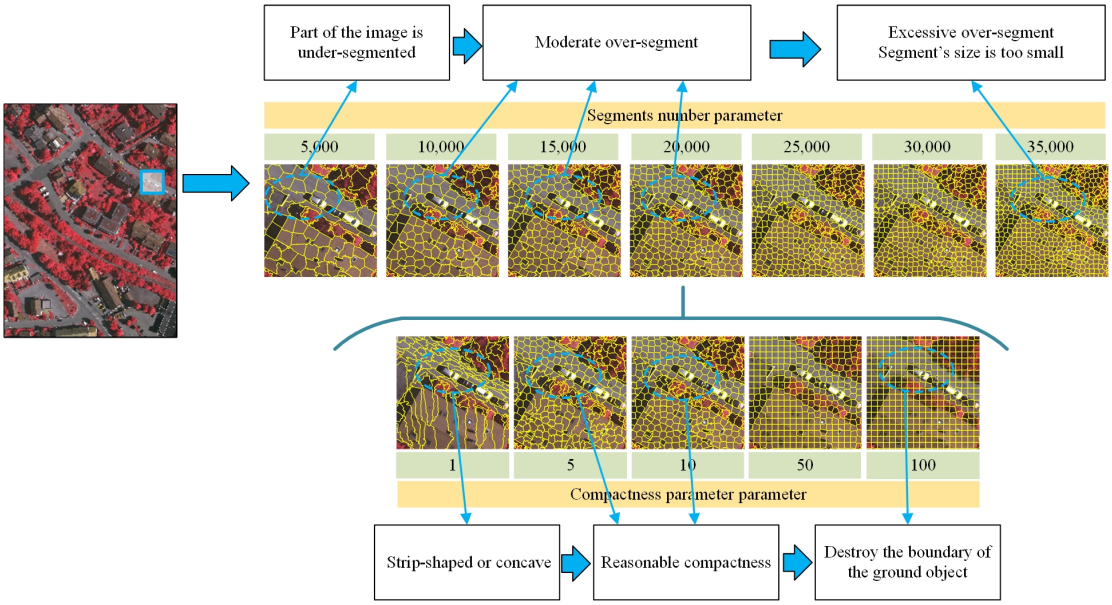

Figure 13. Influence of different SLIC parameter values on the segmentation results.

As shown in Figure 13, we first set compactness=10 and varied the number of segments, leading to different degrees of separation. With segment number=5000, some cross-boundary

segments appeared, and part of the image was under-segmented; with segment number=35000, the segments were markedly smaller, and the image was over-segmented. Then, we set the number of segments to 20000 and varied the compactness. With compactness=1, the shape of each segment had a higher degree of freedom, with some segments being strip-shaped or concave; with compactness=100, the shape of each segment was close to square and this shape was more likely to be inconsistent with the corresponding object boundary.

For test images 1 and 2, the relationship between accuracy of the SO-DNN results and the SLIC parameter values is shown in Table 6.

Table 6. Relationship between SO-DNN accuracy and SLIC parameters

Test image	Segment number parameter	OA for different compactness parameters (%)				
		1	**5**	**10**	**50**	**100**
1	5,000	69.85	82.53	80.37	76.67	75.37
	10,000	73.25	84.11	83.45	79.45	79.05
	15,000	71.33	85.40	86.25	81.72	80.21
	20,000	72.13	87.42	**88.14**	85.33	82.51
	25,000	70.23	87.67	87.31	86.26	83.47
	30,000	71.56	85.23	86.62	86.15	83.34
	35,000	68.24	81.05	83.32	82.01	81.30
2	5,000	66.33	80.71	80.94	74.94	73.23
	10,000	70.78	82.79	82.43	78.56	77.67
	15,000	71.23	83.35	84.33	81.89	81.32
	20,000	72.67	86.48	86.10	84.32	83.59
	25,000	73.22	86.32	**87.03**	85.75	85.01
	30,000	72.34	86.05	86.56	85.38	84.75
	35,000	70.45	85.21	86.31	84.22	84.13

Table 6 shows that both the number of segments and the compactness exert an influence on the SO-DNN results:

(1) Segment number parameter

According to the overall trend observed in Table 6, the classification accuracy is the lowest when the segment number parameter is set to 5000. As number of segments increases, the classification accuracy initially improves; however, an excessively large segment number parameter will reduce the classification accuracy. For test image 1, the accuracy of SO-DNN begins to decrease when the segment number parameter is greater than 20000, and for test image 2, this decrease begins to be seen when the segment number parameter is greater than 25000.

(2) Compactness parameter

The lowest accuracies for test images 1 and 2 appear in the column corresponding to compactness=1. With an increase in the compactness parameter, the accuracy initially shows significant improvement; when the compactness parameter is set to 5 or 10, the corresponding accuracy values are higher than in other columns, and the classification accuracy then decreases when the compactness parameter is further increased to 100.

A smaller segment number parameter tends to lead to a larger segment size, which significantly reduces the processing complexity of FROA but also causes under-segmentation in some areas, leading to the emergence of a large number of cross-boundary segments. Regardless of which category label SO-DNN assigns to such cross-boundary segments, some

of the pixels in them will inevitably be classified incorrectly. In contrast, a larger segment number parameter tends to result in over-segmentation, which reduces the number of cross-boundary segments and thus can significantly reduce the errors associated with the segments themselves; however, a segment size that is too small will also cause the basic processing units in SO-DNN to become too fragmented, making FROA difficult to control and compromising its inference performance.

The compactness parameter determines the shapes of the segments. A lower compactness will make the segments fit more closely to the object boundaries, thereby making the boundaries more accurate. However, a low compactness will also make the segments more prone to be strip-shaped or concave, in which case the center point of a segment may not lie inside the segment this will lead to an irregular distribution of the labeled pixels in T^{l}_{patch}, which will not be concentrated in the center of the image patch, directly affecting the results of the 1st iteration of SO-DNN. Consequently, the accuracy achieved with compactness=1 is lower than with other values of this parameter. On the other hand, a higher compactness will make the shapes of the segments more consistent and convex, making it easier for SO-DNN to concentrate the known samples in the center of the image patch during the training process; however, excessively compact segment shapes will destroy the boundaries of the ground objects and directly reduce the classification quality.

From the data in Table 6, it can be seen that SO-DNN is most successful with a compactness parameter of 5 or 10; in this situation, the locations of the labeled pixels tend to be concentrated in the center of each image patch, consistent with the expectations of SCA and the concept introduced in Figure 2(a). Regarding the segment number parameter, SO-DNN needs moderate over-segmentation to prevent an excessive negative influence of cross-boundary segments on the classification accuracy.

5.6 Experiments on larger datasets

To more extensively evaluate the classification ability of the methods, we selected 20 test images from the Vaihingen and Potsdam of the ISPRS dataset. Detailed information on these images is listed in Table 7.

Table 7. Twenty test images from the Vaihingen and Potsdam datasets

Dataset	Image	Filename	Size
Vaihingen	1	top_mosaic_09cm_area1.tif	1919×2569
	2	top_mosaic_09cm_area2.tif	2428×2767
	3	top_mosaic_09cm_area3.tif	2006×3007
	4	top_mosaic_09cm_area4.tif	1887×2557
	5	top_mosaic_09cm_area5.tif	1887×2557
	6	top_mosaic_09cm_area6.tif	1887×2557
	7	top_mosaic_09cm_area7.tif	1887×2557
	8	top_mosaic_09cm_area8.tif	1887×2557
	9	top_mosaic_09cm_area10.tif	1887×2557
	10	top_mosaic_09cm_area12.tif	1922×2575
Potsdam	1	top_potsdam_2_11_label.tif	6000×6000
	2	top_potsdam_2_12_label.tif	6000×6000
	3	top_potsdam_3_10_label.tif	6000×6000
	4	top_potsdam_3_11_label.tif	6000×6000
	5	top_potsdam_3_12_label.tif	6000×6000
	6	top_potsdam_4_10_label.tif	6000×6000

	7	top_potsdam_4_11_label.tif	6000×6000
	8	top_potsdam_4_12_label.tif	6000×6000
	9	top_potsdam_5_10_label.tif	6000×6000
	10	top_potsdam_5_11_label.tif	6000×6000

The file names listed in Table 7 are not consecutive; the main reasons are that some images in the original datasets do not have corresponding ground truth images and the two images used in the previous section are excluded. Notably, although most Vaihingen images do not contain clutter/background, "top_mosaic_09cm_area2.tif" does have a small area of clutter/background; therefore, to maintain consistency in the comparisons on images from Vaihingen, this area of this image was excluded in evaluation. For SLIC, the compactness parameter was set to 10; to achieve moderate oversegmentation, the segment number parameter was set to 20,000 for the Vaihingen images and to 25,000 for the Potsdam images. For each category in each image, 200 pixels were manually selected as training samples. The classification accuracy of all methods is shown in Table 8.

Table 8. Classification accuracy of all methods on the 20 test images.

Dataset	Image	OA (%)						
		O-MLP	O+CNN	OHSF-CNN	2-CNN	JDL	U-Net	SO-DNN
Vaihingen	1	67.75	72.65	80.03	74.01	80.25	78.24	**84.21**
	2	69.38	73.67	79.86	75.61	83.56	82.98	**85.47**
	3	68.17	75.52	83.2	78.62	85.84	77.57	**86.77**
	4	67.06	69.41	75.62	74.96	78.01	73.62	**79.56**
	5	64.77	70.82	73.39	72.35	78.12	76.37	**80.44**
	6	64.05	71.92	80.72	76.90	87.13	80.97	**89.24**
	7	68.46	73.10	78.17	75.74	80.50	79.79	**88.21**
	8	66.84	71.12	79.67	73.81	85.28	77.89	**89.31**
	9	65.89	76.01	78.81	78.37	85.93	81.90	**87.39**
	10	64.38	72.15	80.83	78.03	**87.97**	80.32	86.64
	Average	66.68	72.64	79.03	75.84	83.26	78.97	**85.72**
Potsdam	1	55.89	64.15	79.44	72.03	80.03	75.66	**82.51**
	2	50.78	67.79	76.77	75.8	80.69	79.21	**81.77**
	3	54.55	62.49	78.86	74.82	**81.03**	73.69	80.44
	4	53.45	64.51	83.42	73.20	80.87	74.87	**83.37**
	5	52.88	63.98	81.22	76.94	81.44	75.98	**84.31**
	6	54.11	62.22	77.62	74.42	**83.06**	78.39	82.25
	7	51.65	57.79	79.94	74.79	78.20	75.32	**88.98**
	8	52.03	63.64	82.87	74.93	82.57	75.94	**86.72**
	9	50.34	63.91	78.91	73.34	83.19	74.96	**87.81**
	10	54.71	60.44	79.72	75.64	82.26	75.69	**84.19**
	Average	53.04	63.09	79.88	74.59	81.33	75.97	**84.24**

It can be seen from Table 8 that O-MLP achieves the lowest accuracy on all images, indicating this object-based + shallow model cannot adapt to the classification task on these two datasets. O+CNN achieves a classification accuracy that is higher than that of O-MLP but still low. The results obtained by OHSF-CNN, 2-CNN and JDL are better than those of O-MLP and O+CNN, indicating that the use of heterogeneity suppression strategies can improve the classification accuracy. U-Net also shows improvement over O+CNN; for images 2 and 7 of Vaihingen and image 2 of Potsdam, the accuracy of U-Net is close to is close to

that of JDL. SO-DNN achieves the highest average classification accuracies of 85.72% for Vaihingen and 84.24% for Potsdam, 2.46% and 2.91% higher than those of JDL, respectively. Except for image 10 of Vaihingen and images 3 and 6 of Potsdam, SO-DNN achieves the highest classification accuracies on single images among all the methods; compared with JDL, SO-DNN achieves maximum accuracy improvements of 7.71% for image 7 of Vaihingen and 10.78% for image 7 of Potsdam. The above results show that compared to the other evaluated methods, SO-DNN can better adapt to classification tasks for very high-resolution remote sensing images and can achieve higher classification accuracy more stably.

6. Conclusion

Methods that rely on additional network structures or combinations of multiple models essentially increase the overall number of trainable parameters in a classification system. Although these methods show improved adaptability and classification accuracy for certain datasets, this strategy inevitably increases the complexity of the entire classification process. For object-based classification based on CNNs, because suppression of the heterogeneous content in the input image patches is needed, many existing methods require multiple models to closely cooperate with each other, which not only increases the difficulty of using these methods but also leads to greater instability.

This paper proposes a simplified object-based DNN classification method (SO-DNN) for very high resolution remote sensing images. This method is "simplified" from three perspectives:

(1) Simplified model

The entire classification process of SO-DNN requires the use of only one DSSNN model and does not require cooperation among multiple models. SO-DNN uses a DSSNN model for object-based classification. Since a DSSNN can directly obtain pixel-wise classification results, SO-DNN does not require multiple models for heterogeneity suppression. When using SO-DNN, there is no need to understand the meaning of the parameters of multiple models or the relationship among them, thereby reducing the knowledge requirements for users.

(2) Simplified sample collection and training

On the one hand, SO-DNN uses SCA to reduce the accuracy requirements of the DSSNN, allowing an "imperfect" DSSNN model to be used for object-based classification; on the other hand, FROA and the IISM algorithm are proposed to establish a bridge from pixel-based samples to patch-based samples and realize progressive DSSNN training starting from pixel-based samples. This makes the samples needed for SO-DNN easier to obtain, especially for applications in which only a small number of images need to be processed.

(3) Simplified usage

The process of using SO-DNN is almost the same as that of a traditional shallow method: select pixel-based samples from an image, train a model and then classify the entire image. This process is simpler than those of the existing object-based + CNN methods, allowing ordinary users to easily obtain a training set and results.

The simplification of the model and the heterogeneity processing mechanism allows SO-DNN to use larger input image patches; this, in turn, allows SO-DNN to determine the label of each segment based on contextual information drawn from a larger area.

Experimental results show that compared with other deep and shallow methods, SO-DNN can achieve higher classification accuracy and more acceptable results. Moreover, SO-DNN has fewer parameters and is less sensitive to the input scale, which reduces the difficulty of use and the burden of gaining trial-and-error experience. By virtue of the above advantages, it is expected that SO-DNN can be widely used in practical applications.

Acknowledgements:

This research was jointly supported by the National Natural Science Foundation of China (41871236; 41971193), the Foundation of the Jilin Provincial Science & Technology Department (20200403174SF; 20200403187SF), and the Foundation of the Jilin Province Education Department (JJKH20210667KJ);

References:

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., 2012. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence 34, 2274-2282.

Alonso, I., Cambra, A., Munoz, A., Treibitz, T., Murillo, A.C., 2017. Coral-segmentation: Training dense labeling models with sparse ground truth, Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 2874-2882.

Alshehhi, R., Marpu, P.R., Woon, W.L., Dalla Mura, M., 2017. Simultaneous extraction of roads and buildings in remote sensing imagery with convolutional neural networks. ISPRS Journal of Photogrammetry and Remote Sensing 130, 139-149.

Badrinarayanan, V., Kendall, A., Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence 39, 2481-2495.

Bearman, A., Russakovsky, O., Ferrari, V., Fei-Fei, L., 2016. What's the point: Semantic segmentation with point supervision, European conference on computer vision. Springer, pp. 549-565.

Blaschke, T., 2010. Object based image analysis for remote sensing. ISPRS journal of photogrammetry and remote sensing 65, 2-16.

Blaschke, T., Hay, G.J., Kelly, M., Lang, S., Hofmann, P., Addink, E., Feitosa, R.Q., Van der Meer, F., Van der Werff, H., Van Coillie, F., 2014. Geographic object-based image analysis–towards a new paradigm. ISPRS journal of photogrammetry and remote sensing 87, 180-191.

Chen, G., Hay, G.J., Carvalho, L.M., Wulder, M.A., 2012. Object-based change detection. International Journal of Remote Sensing 33, 4434-4457.

Chen, Y., Tang, L., Yang, X., Bilal, M., Li, Q., 2020. Object-based multi-modal convolution neural networks for building extraction using panchromatic and multispectral imagery. Neurocomputing 386, 136-146.

Gao, H., Wang, C., Wang, G., Li, Q., Zhu, J., 2020. A new crop classification method based on the time-varying feature curves of time series dual-polarization Sentinel-1 data sets. IEEE Geoscience and Remote Sensing Letters 17, 1183-1187.

Han, W., Feng, R., Wang, L., Cheng, Y., 2018. A semi-supervised generative framework with deep learning features for high-resolution remote sensing image scene classification. ISPRS Journal of Photogrammetry and Remote Sensing 145, 23-43.

Hao, S., Wang, W., Salzmann, M., 2020. Geometry-Aware Deep Recurrent Neural Networks for Hyperspectral Image Classification. IEEE Transactions on Geoscience and Remote Sensing 59. 2448 - 2460.

He, C., Shi, Z., Qu, T., Wang, D., Liao, M., 2019. Lifting scheme-based deep neural network for remote sensing scene classification. Remote Sensing 11, 2648.

Hossain, M.D., Chen, D., 2019. Segmentation for Object-Based Image Analysis (OBIA): A review of algorithms and challenges from remote sensing perspective. ISPRS Journal of Photogrammetry and Remote Sensing 150, 115-134.

Hu, Y., Zhang, Q., Zhang, Y., Yan, H., 2018. A deep convolution neural network method for land cover mapping: A case study of Qinhuangdao, China. Remote Sensing 10, 2053.

Hua, Y., Marcos, D., Mou, L., Zhu, X.X., Tuia, D., 2021. Semantic Segmentation of Remote Sensing Images with Sparse Annotations. arXiv preprint arXiv:2101.03492.

Jiang, J., Lyu, C., Liu, S., He, Y., Hao, X., 2020. RWSNet: a semantic segmentation network based on SegNet combined with random walk for remote sensing. International Journal of Remote Sensing 41, 487-505.

Kumar, B., Dikshit, O., Gupta, A., Singh, M.K., 2020. Feature extraction for hyperspectral image classification: a review. International Journal of Remote Sensing 41, 6248-6287.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. nature 521, 436-444.

Li, J., Lin, D., Wang, Y., Xu, G., Zhang, Y., Ding, C., Zhou, Y., 2020a. Deep discriminative representation learning with attention map for scene classification. Remote Sensing 12, 1366.

Li, M., Stein, A., 2020b. Mapping land use from high resolution satellite images by exploiting the spatial arrangement of land cover objects. Remote sensing 12, 4158.

Li, M., Stein, A., Bijker, W., Zhan, Q., 2016. Urban land use extraction from Very High Resolution remote sensing imagery using a Bayesian network. ISPRS journal of photogrammetry and remote sensing 122, 192-205.

Li, Y., Shi, T., Zhang, Y., Chen, W., Wang, Z., Li, H., 2021. Learning deep semantic segmentation network under multiple weakly-supervised constraints for cross-domain remote sensing image semantic segmentation. ISPRS Journal of Photogrammetry and Remote Sensing 175, 20-33.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation, Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431-3440.

Lu, X., Yuan, Y., Zheng, X., 2016. Joint dictionary learning for multispectral change detection. IEEE transactions on cybernetics 47, 884-897.

Luo, W., Yang, M., Zheng, W., 2021. Weakly-supervised semantic segmentation with saliency and incremental supervision updating. Pattern Recognition 115, 107858.

Lv, X., Ming, D., Chen, Y., Wang, M., 2019. Very high resolution remote sensing image classification with SEEDS-CNN and scale effect analysis for superpixel CNN classification. International Journal of Remote Sensing 40, 506-531.

Ma, L., Li, M., Ma, X., Cheng, L., Du, P., Liu, Y., 2017. A review of supervised object-based

land-cover image classification. ISPRS Journal of Photogrammetry and Remote Sensing 130, 277-293.

Martins, V.S., Kaleita, A.L., Gelder, B.K., da Silveira, H.L., Abe, C.A., 2020. Exploring multiscale object-based convolutional neural network (multi-OCNN) for remote sensing image classification at high spatial resolution. ISPRS Journal of Photogrammetry and Remote Sensing 168, 56-73.

Medley, D.O., Santiago, C., Nascimento, J.C., 2019. Deep Active Shape Model for Robust Object Fitting. IEEE Transactions on Image Processing 29, 2380-2394.

Osuna-Coutiño, J.d.J., Martinez-Carranza, J., 2020. Structure extraction in urbanized aerial images from a single view using a CNN-based approach. International Journal of Remote Sensing 41, 8256-8280.

Pan, X., Zhao, J., Xu, J., 2019. An object-based and heterogeneous segment filter convolutional neural network for high-resolution remote sensing image classication. International Journal of Remote Sensing 40, 5892-5916.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, International Conference on Medical image computing and computer-assisted intervention. Springer, pp. 234-241.

Rottensteiner, F., Sohn, G., Jung, J., Gerke, M., Baillard, C., Benitez, S., Breitkopf, U., 2012. The ISPRS benchmark on urban object classification and 3D building reconstruction. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences I-3 (2012), Nr. 1 1, 293-298.

Shao, Y., Cooner, A.J., Walsh, S.J., 2021. Assessing Deep Convolutional Neural Networks and Assisted Machine Perception for Urban Mapping. Remote Sensing 13, 1523.

Shen, Y., Chen, J., Xiao, L., Pan, D., 2019. Optimizing multiscale segmentation with local spectral heterogeneity measure for high resolution remote sensing images. ISPRS Journal of Photogrammetry and Remote Sensing 157, 13-25.

Sridharan, H., Qiu, F., 2013. Developing an object-based hyperspatial image classifier with a case study using WorldView-2 data. Photogrammetric Engineering & Remote Sensing 79, 1027-1036.

Tang, Y., Qiu, F., Jing, L., Shi, F., Li, X., 2020. Integrating spectral variability and spatial distribution for object-based image analysis using curve matching approaches. ISPRS Journal of Photogrammetry and Remote Sensing 169, 320-336.

Tarvainen, A., Valpola, H., 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. arXiv preprint arXiv:1703.01780.

Wang, X., You, S., Li, X., Ma, H., 2018. Weakly-supervised semantic segmentation by iteratively mining common object features, Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1354-1362.

Wei, Y., Feng, J., Liang, X., Cheng, M.-M., Zhao, Y., Yan, S., 2017. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach, Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1568-1576.

Xia, G.-S., Hu, J., Hu, F., Shi, B., Bai, X., Zhong, Y., Zhang, L., Lu, X., 2017. AID: A benchmark data set for performance evaluation of aerial scene classification. IEEE

Transactions on Geoscience and Remote Sensing 55, 3965-3981.

Xia, X., Persello, C., Koeva, M., 2019. Deep fully convolutional networks for cadastral boundary detection from UAV images. Remote sensing 11, 1725.

Yang, X., Li, X., Ye, Y., Lau, R.Y., Zhang, X., Huang, X., 2019. Road detection and centerline extraction via deep recurrent convolutional neural network U-Net. IEEE Transactions on Geoscience and Remote Sensing 57, 7209-7220.

You, H., Tian, S., Yu, L., Lv, Y., 2019. Pixel-level remote sensing image recognition based on bidirectional word vectors. IEEE Transactions on Geoscience and Remote Sensing 58, 1281-1293.

Zhang, C., Harrison, P.A., Pan, X., Li, H., Sargent, I., Atkinson, P.M., 2020a. Scale Sequence Joint Deep Learning (SS-JDL) for land use and land cover classification. Remote Sensing of Environment 237, 111593.

Zhang, C., Pan, X., Li, H., Gardiner, A., Sargent, I., Hare, J., Atkinson, P.M., 2018a. A hybrid MLP-CNN classifier for very fine resolution remotely sensed image classification. ISPRS Journal of Photogrammetry and Remote Sensing 140, 133-144.

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., Atkinson, P.M., 2018b. An object-based convolutional neural network (OCNN) for urban land use classification. Remote sensing of environment 216, 57-70.

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., Atkinson, P.M., 2019. Joint Deep Learning for land cover and land use classification. Remote sensing of environment 221, 173-187.

Zhang, C., Yue, P., Tapete, D., Shangguan, B., Wang, M., Wu, Z., 2020b. A multi-level context-guided classification method with object-based convolutional neural network for land cover classification using very high resolution remote sensing images. International Journal of Applied Earth Observation and Geoinformation 88, 102086.

Zhao, W., Du, S., Emery, W.J., 2017. Object-based convolutional neural network for high-resolution imagery classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 10, 3386-3396.

Zhao, X., Zhang, J., Tian, J., Zhuo, L., Zhang, J., 2020. Residual dense network based on channel-spatial attention for the scene classification of a high-resolution remote sensing image. Remote Sensing 12, 1887.

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

Zhu, Z., Zhou, Y., Seto, K.C., Stokes, E.C., Deng, C., Pickett, S.T., Taubenböck, H., 2019. Understanding an urbanizing planet: Strategic directions for remote sensing. Remote Sensing of Environment 228, 164-182.