Semantic Segmentation of Remote Sensing Images With Self-Supervised Multitask Representation Learning

Wenyuan Li, Hao Chen, and Zhenwei Shi, *Member, IEEE*

***Abstract*—Existing deep learning-based remote sensing images semantic segmentation methods require large-scale labeled datasets. However, the annotation of segmentation datasets is often too time-consuming and expensive. To ease the burden of data annotation, self-supervised representation learning methods have emerged recently. However, the semantic segmentation methods need to learn both high-level and low-level features, but most of the existing self-supervised representation learning methods usually focus on one level, which affects the performance of semantic segmentation for remote sensing images. In order to solve this problem, we propose a self-supervised multitask representation learning method to capture effective visual representations of remote sensing images. We design three different pretext tasks and a triplet Siamese network to learn the high-level and low-level image features at the same time. The network can be trained without any labeled data, and the trained model can be fine-tuned with the annotated segmentation dataset. We conduct experiments on Potsdam, Vaihingen dataset, and cloud/snow detection dataset Levir_CS to verify the effectiveness of our methods. Experimental results show that our proposed method can effectively reduce the demand of labeled datasets and improve the performance of remote sensing semantic segmentation. Compared with the recent state-of-the-art self-supervised representation learning methods and the mostly used initialization methods (such as random initialization and ImageNet pretraining), our proposed method has achieved the best results in most experiments, especially in the case of few training data. With only 10% to 50% labeled data, our method can achieve the comparable performance compared with random initialization. Codes are available at https://github.com/flyakon/SSLRemoteSensing.**

***Index Terms*—Cloud detection, remote sensing images, self-supervised representation learning, semantic segmentation.**

I. Introduction

The rapid development of remote sensing technology has greatly widened the scope of exploring the earth. Satellite images have been widely used in resource exploration, land census, natural disaster monitoring, etc. The semantic segmentation [1]–[3] (also called pixel level classification) plays a key role in the analysis of remote sensing images and the fully convolutional neural networks (FCN) [4]–[6]-based methods have brought a great breakthrough to the semantic segmentation [7]–[12] of remote sensing images.

Despite the great success, recent FCN-based semantic segmentation methods for remote sensing images still rely on training with a large number of manually annotated data. Although there are some annotated datasets available, most of remote sensing data from the Internet are not labeled that adapts to semantic segmentation task. These unlabeled data have no effect on improving the semantic segmentation of remote sensing images. The purpose of this article is to design an effective pretraining method with unlabeled data to improve the effect of remote sensing semantic segmentation.

The most commonly used pretraining paradigm is ImageNet [13] pretraining. However, it is time-consuming and laborious to construct such a large-scale remote sensing dataset like ImageNet to pretrain networks as the annotation of remote sensing data may rely heavily on professional domain knowledge. In addition, considering that remote sensing images are increasingly showing the characteristics of multisources and multiresolutions, even if there is a large-scale remote sensing dataset, it cannot meet the requirements of downstream tasks for all remote sensing images obtained from various satellites.

Self-supervised representation learning is the other recently emerged research topic that learns effective visual representations of images by taking advantage of self-supervised learning ideas [14]–[16]. It is an elegant subset of unsupervised learning, which can obtain supervision information from data itself during training. Therefore, it does not need any labeled data for training and can possibly learn from any scale of unlabeled data. In self-supervised representation learning, a set of pretext tasks are usually designed to explore the relationships between image patches or image transformations. Through the pretext tasks, the networks can be trained with unlabeled data and a pretraining model can be obtained. Then the downstream tasks such as semantic segmentation can be fine-tuned on this pretraining model to obtain better results. According to the type of supervision acquired, the pretext tasks in previous self-supervised representation learning methods can be divided into three categories: 1) Image level pretext tasks [17]–[35], 2) patch level pretext

Manuscript received April 30, 2021; revised June 2, 2021; accepted June 13, 2021. Date of publication June 18, 2021; date of current version July 14, 2021. The work was supported in part by the National Key R&D Program of China under Grant 2019YFC1510905, in part by the National Natural Science Foundation of China under Grant 61671037, and in part by the Beijing Natural Science Foundation under Grant 4192034. *(Corresponding author: Zhenwei Shi.)*

The authors are with the Image Processing Center, School of Astronautics, Beihang University, Beijing 100191, China, with the Beijing Key Laboratory of Digital Media, Beihang University, Beijing 100191, China, and also with the State Key Laboratory of Virtual Reality Technology and Systems, School of Astronautics, Beihang University, Beijing 100191, China.

Digital Object Identifier 10.1109/JSTARS.2021.3090418

tasks [36]–[40], and 3) pixel level pretext tasks [41]–[47]. However, remote sensing images have random viewing angle and no specific salient area but more complex hierarchical structure and more abundant background information. The above self-supervised methods designed for natural images do not consider the characteristics of remote sensing images and may not work properly.

Moreover, considering remote sensing images usually show more high-frequency details and hierarchical structure compared to natural images, the pretraining methods should extract the high-level and low-level features. Especially for the semantic segmentation, it requires that the networks should take these two aspects into account at the same time. However, the current methods for natural images and remote sensing images only consider one of them. In order to solve this problem, this article proposes a novel self-supervised representation learning method for remote sensing semantic segmentation. Our method is designed to focus on both high-level and low-level features. In our method, we design three different pretext tasks for pretraining, including an image inpainting task, an augmentation transform prediction (ATP) task and a contrastive learning task. We design a triplet-Siamese network with three output branches. The backbone network shares the same set of image features and network parameters. Each output branch corresponds to a different pretext task and is trained with its own loss function. The total loss for training the whole network is a multitask loss function which combines losses of the three pretext tasks.

By designing the image inpainting task, we aim to help networks to learn low-level representations. We propose a moderate approach to construct occluded areas by randomly transforming the in-box areas with image rotation, flip, color transformation, etc. By designing ATP task and contrastive learning task, we aim to help networks to learn high-level representations. For ATP task, according to the problem that remote sensing image has no obvious imaging perspective, we build the Siamese networks, and take the image and its transformation as the input to predict the type of transformation.

After the pretraining, the networks can be easily applied to semantic segmentation by fine-tuning on their labeled datasets. In the experimental part, we use the Potsdam dataset and Vaihingen dataset [48] to verify the effectiveness of our method. In addition, the cloud/snow detection task can also be regarded as a semantic segmentation task, so we select Levir_CS dataset [49] to verify our method in cloud/snow detection task. The results show that our method outperforms other recent self-supervised representation learning methods [24], [32], [33] in the semantic segmentation task. Our method achieves better results than ImageNet pretrained models, and the best results with limited training data. In addition, with only 10% to 50% labeled data, our method can achieve the comparable performance compared with random initialization, which shows that our method can effectively reduce the demand on annotated data.

The contributions of this article are summarized as follows.

1) We propose a self-supervised representation learning method for remote sensing semantic segmentation. A multitask loss function is designed to guide the networks to learn both high-level and low-level features at the same time. A large number of unlabeled remote sensing images can be effectively used to train the networks and improve the performance of the semantic segmentation task.
2) In the remote sensing semantic segmentation task, we achieve better results than models with ImageNet pretraining and other recent self-supervised pretraining methods.
3) Our method can achieve the comparable performance with only 50% labeled data on Vaihingen dataset and 20% labeled data on Potsdam dataset compared with random initialization, while only 20% labeled data for cloud detection and 10% labeled data for snow detection are needed to achieve the comparable performance.

The rest of this article is organized as follows. In Section III, we give a detailed introduction of our proposed method, including network configuration, multitask loss function, and implementation details. In Section IV, the experimental datasets and experimental results are introduced. Discussion and conclusions are drawn in Sections V and VI.

II. Related Work

A. *Self-Supervised Representation Learning*

Self-supervised representation learning is a recently emerged research topic that learns effective visual representations of images by taking advantage of self-supervised learning ideas [14]–[16]. Self-supervised representation learning obtains supervision information from data itself and train the networks without using manual annotations by designing a series of pretext tasks. The trained models can be used to improve the performance of downstream tasks. According to the type of supervision acquired, the pretext tasks in previous self-supervised representation learning methods can be divided into three categories. 1) Image level pretext tasks [17]–[35], 2) patch level pretext tasks [36]–[40], and 3) pixel level pretext tasks [41]–[47].

1) *Image Level Pretext Task*

   The image level pretext tasks in self-supervised representation learning explore the intrinsic properties or the relationship of images. For example, data augmentation can be used to generate transformed images and corresponding labels from the original image [17], [18], [22], [24], [26], [29], [35], and their correspondence can be thus explored during training. Such a group of methods can also benefit from the adversarial training [50] and the training of networks can be guided at the image level by building adversarial losses [19], [31]. In addition to augmentation methods, we can also use clustering methods to divide the images into different groups roughly. The inputs of cluster methods are features extracted from neural networks and clustering results are integrated into the loss function to guide networks training [20], [21], [23], [27], [30], [34]. Another popular way to define image level pretext tasks is to design contrastive loss functions. These methods encourage the networks to learn similar representations from similar images (typically the image and its random transformations) and learn different representations from different ones. The contrastive loss functions can also be used to help networks obtain higher robustness to image rotation and scaling, and therefore improve the generalization of their feature representations [25], [28], [32], [33].

2) *Patch Level Pretext Task*
The key to patch level pretext tasks is that if we divide an image into several patches, then we can construct self-supervised loss functions by simply exploring the location or semantic relationship between them. The networks thus can be trained to learn from the patches and their surroundings. Doersch *et al.* [36] propose to evenly divide an image into nine patches and train the networks to predict the position of a certain patch relative to the center patch. However, this method can only learn the relationship between adjacent patches, and is hard to learn the overall arrangement of the content in an image. The pretext task based on the "jigsaw" solves this problem [39], [40]. The jigsaw-based methods divide an image into patches and shuffle them. Then networks are trained to predict orders of the patches to recover the input image. During this process, the networks need to fully understand the content and relationship between each patch, and show better performance in downstream tasks [37], [38].

3) *Pixel Level Pretext Task*
The goal of pixel level pretext tasks is to make networks understand semantic information. Compared with image level tasks, pixel level pretext tasks focus more on the learning of semantic level information. However, they may also force the networks to learn too many details or shortcuts between the input and output, which is sometimes meaningless for downstream tasks [43]. Therefore, pixel level pretext tasks usually need some skills to prevent overfitting during training. Autoencoder is a group of commonly used unsupervised/self-supervised representation learning methods [41], [42], [47]. However, if autoencoder is directly used to reconstruct the input images, it will easily overfit to raw pixels rather than fully "understand" the image. Pathak *et al.* [42] propose to combine autoencoder with the image inpainting task to alleviate this problem. In their method, they train an autoencoder to recover the input image, but at the same time a region in the input image will be randomly discarded. Then they train a discriminator with the recovered images and some real ones to further improve the features. Zhang *et al.* [41] combine the ideas of autoencoder and contrastive learning. They switch the input of autoencoder to the transformed images. The split-brain autoencoder [47] modifies the one-way calculation of an autoencoder and proposes a two-way reversible autoencoder structure. Through a group of reversible operations, the self-supervised model can be trained with pair-wise losses. Besides, the tasks of image colorization [43]–[45] and image inpainting [46] are also widely used in self-supervised representation learning where a network has to learn to recognize objects and make a full understanding of the details of the images (e.g., sky is blue and trees are green) before achieving such goals.

Although self-supervised representation learning has made great progress in recent years, it still falls behind ImageNet pretraining in most downstream tasks. Self-supervised representation learning outperforms ImageNet pretrained models only on a few tasks such as object detection [51]–[53]. In addition, most of the above methods are designed for natural image tasks without considering the characteristics of remote sensing images.

B. *Self-Supervised Representation Learning for Remote Sensing Images*

Although researches of self-supervised representation learning for natural images are developing rapidly, methods for remote sensing images are relatively less. Compared with natural images, remote sensing images usually consists of more than three bands. Vincenzi *et al.* [54] propose to use high-dimensional data to reconstruct image color for pretraining, which can help networks learn image representations. But for hyperspectral image processing task [55]–[58], this method may not work well. Some self-supervised learning methods [59]–[61] are proposed for hyperspectral images, and have achieved good results.

In addition, the longitude and latitude information of remote sense images and multitemporal data [62] can also be used for self supervised learning. The SauMoCo [63] method utilize the spatial information of remote sensing images, and achieved good results. Researches [64], [65] combine images with spatial information and multitemporal images to the contrastive learning, and improve the performance of downstream tasks.

However, the above methods still follow ideas for natural images, trying to extract the supervised information from remote sensing images by designing pretext tasks just like natural images, but it does not take advantage of the characteristics of remote sensing images. In addition to the self-supervised learning methods, some early researches usually focus on the representation learning for specific tasks. In [66] and [67], a feature learning method for the scene classification task of remote sensing images is designed, which can effectively improve the effect of scene classification. In order to solve the problem of multiple remote sensing image data sources, Neumann *et al.* [68] proposes a feature learning method between different datasets. However, these methods are designed for an only single task and lack of generality.

III. Methods

Given a backbone convolutional network (e.g., VGG16 [69], ResNet50 [70]), we start by designing a triplet Siamese network on top of the backbone. The triplet Siamese network consists of three input branches, three output branches, and the backbone as a feature extraction network. In this section, we introduce the detailed configuration of our network architecture and pretext tasks.

A. *Overview of the Proposed Method*

Fig. 1 shows an overall architecture of our method. For different tasks, weights are shared among input branches and the backbone networks. Different pretext tasks are implemented by

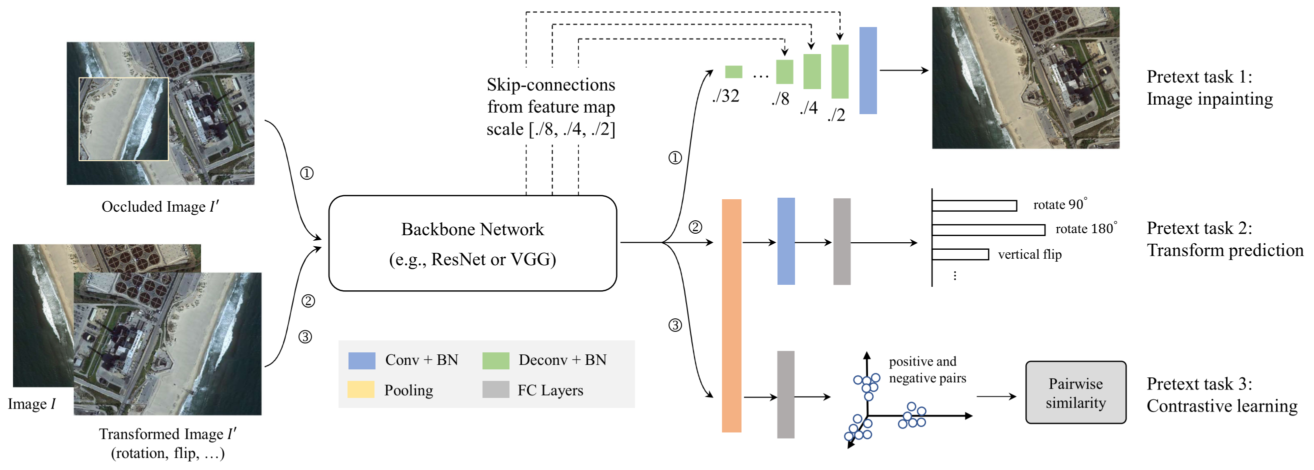
Occluded Image I'
Image I
Transformed Image I' (rotation, flip, …)
Skip-connections from feature map scale [./8, ./4, ./2]
Backbone Network (e.g., ResNet or VGG)
Conv + BN
Deconv + BN
Pooling
FC Layers
./32
./8
./4
./2
rotate 90°
rotate 180°
vertical flip
positive and negative pairs
Pairwise similarity
Pretext task 1: Image inpainting
Pretext task 2: Transform prediction
Pretext task 3: Contrastive learning

Fig. 1. Overview of the proposed method. Given a backbone convolutional neural network, we build three branches on its output: An inpainting branch, an ATP branch, and a contrastive learning branch (from top to bottom in this figure). The inpainting branch takes in a randomly occluded image and is trained to repair the occluded area. The ATP branch and the contrastive learning branch share a same pair of input images, where the former is trained to predict the transformation type and the latter ensures that the backbone produces similar features for similar input and vice versa.

adding different heads and loss functions on top of the backbone networks.

For the inpainting branch, the input is a randomly occluded image. The branch repairs the occluded area by adding several transposed convolution layers on top of the backbone networks. To increase the details of texture and edge, we also fuse the global and local information by introducing skip connections between different convolution layers and transposed convolution layers. For the ATP branch and the contrastive learning branch, their inputs are the images before and after random transformation. Their features produced by the backbone networks are concatenated along their channel dimension. We then construct two fully connected networks: One takes in the features and predicts the transformation type as the output of ATP branch, and the other one maps the features to the latent space to calculate the contrastive loss function.

B. Pretext Tasks and Loss Functions

We define three pretext tasks for self-supervised training: An inpainting task that helps the backbone networks learn low-level features, and an ATP task + a contrastive task that are responsible for learning high-level features.

1) *Inpainting Task*

The inpainting branch helps networks to learn useful features by repairing occluded areas of the input image. Suppose I represents an original input image. We randomly occlude I with an $S \times S$ pixels square box B_p and suppose I' represents the occluded image.

In conventional image inpainting tasks, the pixel values of the occluded area are set to 0 or 255. However, the strategy of filling with 0 or 255 will cause the loss of information in the occluded area, thereby increasing the difficulty and instability of network training. Current researches [43], [46] usually utilize generative adversarial networks to improve this problem. But the use of generative adversarial networks will increase the difficulty of network design and training too. Therefore, we use a more moderate approach to construct occluded areas by randomly transform the in-box areas with image rotation, flip, color transformation, etc. In this way, the backbone can use the information both inside and outside occluded area for the restoration. To further improve the generalization ability of the pretrained model, we also perform random clipping and color jittering on the input image I'.

We define the following loss function to train the backbone and the inpainting branch:

$$L_p = \|\beta(I - \widehat{I})\|_1 \tag{1}$$

where $\widehat{I}$ is outputs of the networks. $\beta = |I - I'|$ is a predefined weighting map, which guides the networks to pay more attention to the areas with bigger changes. $\|\cdot\|_1$ is the pixel-wise l-1 function.

2) *Augmentation Transform Prediction (ATP) Task*

Given I as an original input image, in the ATP branch, we define a series of image transformation operations (image rotation, flip, etc.) $T = \{t_1, t_2, \ldots, t_M\}$ and transform I by using a randomly selected one operation t from T. Suppose I' represents the transformed image in the ATP branch. We feed I' to the networks and train it to recognize which type of transformation is applied. The ATP thus can be essentially formulated as a standard classification problem. The loss function of the ATP branch is defined as follows:

$$L_a = -\sum_{m=1}^{M} \hat{A}_{(m)} \log P_{(m)} \tag{2}$$

where $\hat{A}_{(m)} = \{0, 1\}$ represents the one-hot encoding of the ground truth class label. P represents the predicted class-probability of M different transformations. The number of categories is six. The used data augmentations include, rotating 90 degrees, rotating 180 degrees, rotating 270 degrees, horizontal flip, vertical flip, and no augmentation.

3) *Contrastive Learning Task*

We follow the paper [28] to build a contrastive loss function to guide networks to learn high-level features of remote sensing images. The contrastive branch and ATP branch share the same group of input images.

Given a pair of input image I and its transformation I', we suppose $\phi(I)$ and $\phi(I')$ represent their image features produced by the backbone networks respectively. We calculate the similarity between $\phi(I)$ and $\phi(I')$ as follows:

$$S(I, I') = \frac{\phi^T(I)\phi(I')}{\|\phi(I)\|_2 \|\phi(I')\|_2} \tag{3}$$

where $\| \cdot \|_2$ represents the l-2 norm.

We further assume that a mini-batch during the training consists of N image pairs, and (I_i, I_i') represents the ith image pair. We define the contrastive loss function $l(I_i, I_i')$ of the ith input pair as follows:

$$\begin{aligned} L(I_i, I_i') &= l(I_i, I_i') + l(I_i', I_i) \\ &= -\log \frac{\exp(S(I_i, I_i'))}{\sum_{k \neq i} \exp(S(I_i, I_k))} \\ &\quad -\log \frac{\exp(S(I_i, I_i'))}{\sum_{k \neq i} \exp(S(I_i', I_k))} \end{aligned} \tag{4}$$

where (I_i, I_i') is a positive pair and (I_i, I_k) is a negative pair. Minimizing the above contrastive loss function can ensure that the feature similarity of a positive image pair is larger than any other negative combinations.

For all image pairs in a training batch, the total contrastive loss function of this branch is written as follows:

$$L_c = \sum_{i=1}^{N} L(I_i, I_i'). \tag{5}$$

The final loss function of the three pretext tasks is defined as the linear combination of their losses

$$L = \gamma_p L_p + \gamma_a L_a + \gamma_c L_c \tag{6}$$

where γ_p, γ_a, and γ_c are positive coefficients to balance the losses of the above three tasks.

C. Implementation Details

We experiment on two widely used convolutional neural networks architechtures - VGG16 [69] and Resnet50 [70], and use them as our backbone networks. To improve the training stability, we add batch normalization (BN) [71] layers to the three prediction branches after each convolution and transposed convolution layer. We also use data augmentation on input images to avoid overfitting. We augment the input images by using random image rotation ([0, 90, 180, 270] degrees), horizontal flip, and vertical flip. We add color jittering and random clipping to the transformed image, which makes the pretext tasks more difficult. The input image is converted into a gray image with a certain probability to avoid learning too much color information.

To balance the loss functions of the three tasks numerically, especially at the beginning of training, We set $\gamma_p = 20.0, \gamma_a = 1.0$, and $\gamma_c = 1.0$. For self-supervised training stage, we set batch size as 8. The network training lasts 13 epochs in total, and 242 300 steps are carried out. The input image size is 256×256. The training of the network takes approximately 27 h. We use pytorch-1.5 to build our codes. The learning rate is set to 5e-4 and is reduced to its 95% after every two training epochs. Codes are available at https://github.com/flyakon/SSLRemoteSensing.

TABLE I
DETAILED CONFIGURATION OF OUR NETWORKS (VGG16-BACKBONE AS AN EXAMPLE)

	Name	Layer	Input	Ker	S	#Ker
Backbone	C1	conv_pool	image	3x3	2	64
	C2	conv_pool	C1	3x3	2	128
	C3	conv_pool	C2	3x3	2	256
	C4	conv_pool	C3	3x3	2	512
	C5	conv_pool	C4	3x3	2	512
ATP	Cat1	concat	C5	-	-	-
	C6	conv_pool	Cat1	3x3	8	512
	FC1	MLP	C6	-	-	6
Contrast	Pool1	avg_pool	layer5	8x8	1	512
	FC2	MLP	pool1	-	-	1024
	FC3	MLP	fc1	-	-	512
Inpainting	UP1	deconv	layer5	3x3	2	512
	UP2	deconv	UP1+C4	3x3	2	256
	UP3	deconv	UP2+C3	3x3	2	128
	UP4	deconv	UP3+C2	3x3	2	64
	UP5	deconv	UP4+C1	3x3	2	32
	Pred	conv	UP5	3x3	1	3

In Table I, we take VGG16 [69] as an example to show the detailed structure of our prediction branches. The columns "Ker," "S," and "#Ker" denote the size, stride, and channel number of the convolution layers, respectively. "conv" and "deconv" denote convolution and transposed convolution operations, respectively. The pseudocode of training process is shown in Algorithm 1.

IV. RESULTS

A. Dataset for Experiments

1) Dataset for Pretraining

Since there are still few researches in remote sensing focusing on self-supervised pretraining, no such benchmark data are publicly available. Therefore, we construct a large unlabeled dataset by combining several well-known remote sensing datasets, including DIOR [72], DOTA [73], and Levir [74]. To increase the versatility of the dataset, the images are selected with different resolutions. We use the method in [75] remove some low contrast images, and the final number of images for pretraining is 186 486.

2) Datasets for Semantic Segmentation

We use three datasets: Levir_CS [49], Potsdam, and Vaihingen [48] to verify the effectiveness of our method for semantic segmentation of remote sensing images. The Potsdam and Vaihigen datasets are commonly used datasets for remote sensing semantic segmentation. Levir_CS dataset is a large-scale dataset for cloud / snow detection task that is in essence a pixel classification task. The detailed information of all the above datasets are shown in Table II. The Potsdam and Vaihingen [48] datasets

Algorithm 1 Training process of our method

```
1:  Input: Training data $X$, backbone $f$, different pretext
    tasks heads $g_p, g_a, g_c$. Transforms for tasks: $t_p, t_a, t_c$.
2:  function TrainLoop $X$
3:    for all $I \in X$ do
4:      $L_p$=INPAINTING$(I)$
5:      $L_a$=ATP$(I)$
6:      $L_c$=CONSTRASTIVE$(I)$
7:      $L = \gamma_p L_p + \gamma_a L_a + \gamma_c L_c$
8:      update networks to minimize $L$
9:    end for
10: end Function
11: function Inpainting $(I)$
12:   $I' \leftarrow t_p(I)$
13:   $h \leftarrow f(I')$
14:   $\widehat{I} \leftarrow g_p(h)$
15:   $L_p = \|\beta(I - \widehat{I})\|_1$
16:   return $L_p$
17: end Function
18: function ATP $(I)$
19:   $I', \hat{A} \leftarrow t_a(I)$
20:   $h' \leftarrow f(I')$
21:   $h \leftarrow f(I)$
22:   $P \leftarrow g_p(h', h)$
23:   $L_a = -E\{\hat{A} \log P\}$
24:   return $L_a$
25: end Function
26: function Constrastive $(I)$
27:   $I' \leftarrow t_c(I)$
28:   $h' \leftarrow f(I')$
29:   $h \leftarrow f(I)$
30:   $z' \leftarrow g_c(h', h)$
31:   $z \leftarrow g_c(h, h)$
32:   compute constrastive losss with $z, z'$
33:   return $L_c$
34: end Function
```

TABLE II
DATASETS FOR DOWNSTREAM TASKS AND THEIR STATISTICS

Dataset	#Classes	Split
Levir_CS	2	Training Set: 8,096 Validation Set: 4,176 Testing Set: 4,400
Potsdam	6	Training Set: 12,672 Validation Set: 4,608 Testing Set: 4,608
Vaihingen	6	Training Set: 1,584 Validation Set: 572 Testing Set: 682

both have six categories in each dataset. We crop the images into patches with size of 256×256, and randomly divide them into a training set (60%), a validation set (20%), and a testing set (20%). The Levir_CS dataset consists of two categories with cloud and snow. We also crop the images into patches with size of 256×256, and randomly divide them into a training set (60%), a validation set (20%), and a testing set (20%).

B. Experimental Setup

In the pretraining stage, we do not use the validation set, but used all the data for network training. Because even if we set the validation set, we cannot reasonably infer the performance of the pretrained model on the semantic segmentation task through the validation set. We infer whether the pretraining process is completed according to the loss function during the training process.

We compare four different network initialization methods: 1) From scratch (random initialization), 2) from ImageNet pretraining, 3) from our self-supervised pretraining, and 4) from ImageNet pretraining + self-supervised pretraining (by continuing to pretrain the networks with self-supervised losses on top of the ImageNet pretraining).

We compare our method with three state of the art methods for self-supervised representation learning, which are NPID [24], MoCo [32], and MoCo v2 [33]. All methods are trained and evaluated using the datasets described above.

For the semantic segmentation task, we add five transposed convolution layers on top of the backbone model with each layer followed by a BN layer (the same architecture as our inpainting branch). We compute the accuracy on validation set every 20 epoch and save the model with the highest accuracy. We stop training after 200 epochs. The learning rate is set to 0.005. We adjust the learning rate to its 90% every 10 epochs. For the cloud/snow detection task, we adopt the same network structure and training strategy with those for semantic segmentation, except that the learning rate is 0.001.

We use the intersection-over-union (IoU) as the evaluation accuracy. The IoU can be computed as follows:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{7}$$

where TP is the number of true positive pixels, and FP and FN are the number of false positive and false negative pixels. All of these values are calculated from the confusion matrix of categories. Finally, after getting the IoU of each category, we compute mIoU—the averaged accuracy of all categories as the final evaluation accuracy.

C. Semantic Segmentation Results

We verify the performance of our method on remote sensing image semantic segmentation task on Potsdam and Vaihingen datasets [48]. The results are shown in Tables III, IV, and Fig. 2. Considering humans are able to recognize novel instances with very few training examples, we also show the performance of our method with very limited training data. The columns in the

TABLE III
SEMANTIC SEGMENTATION RESULTS ON VAIHINGEN DATASET

	0.25%	0.33%	0.5%	1%	2%	5%	10%	20%	50%	100%
VGG16 Random	0.3600	0.3207	0.3540	0.3810	0.4041	0.4596	0.4770	0.5024	0.6313	0.6521
Resnet50 Random	0.3054	0.3439	0.3369	0.3846	0.3757	0.4194	0.4727	0.5106	0.6309	0.6448
VGG16 ImageNet	0.3218	0.3430	0.3777	0.3866	0.3855	0.4731	0.4807	0.5028	0.6293	0.6753
Resnet50 ImageNet	0.2974	0.3424	0.3575	0.3470	0.4050	0.4640	0.4455	0.5177	0.6611	0.7015
NPID [24]	0.3633	0.3366	**0.3967**	0.4096	0.4140	0.4737	0.5007	0.5162	0.6620	0.6878
MoCo [32]	0.3444	0.3528	0.3736	0.4143	**0.4228**	0.4551	0.5054	0.5057	0.6197	0.6584
MoCo v2 [33]	0.3534	0.3518	0.3686	0.3922	0.4136	0.4614	0.4863	0.5028	0.6496	0.6777
Ours (VGG16)	0.3489	0.3520	0.3810	0.4073	0.4019	**0.5035**	0.5276	0.5556	0.6897	0.7400
Ours (Resnet50)	0.2931	0.3487	0.3508	0.3686	0.4195	0.4687	0.4995	**0.6054**	0.6887	0.7274
Ours* (VGG16)	**0.3805**	**0.3591**	0.3888	**0.4232**	0.4157	0.4878	**0.5898**	0.5745	**0.7102**	**0.7413**
Ours* (Resnet50)	0.3102	0.3341	0.3245	0.3852	0.4088	0.4904	0.5205	0.5669	0.6833	0.7222

IoU is used as the metric. The highest scores are marked in bold. Ours* represents ImageNet pretraining + self-supervised pretraining of our method.

TABLE IV
SEMANTIC SEGMENTATION RESULTS ON POTSDAM DATASET

	0.25%	0.33%	0.5%	1%	2%	5%	10%	20%	50%	100%
VGG16 Random	0.3620	0.3026	0.3534	0.4007	0.4240	0.4864	0.5407	0.5814	0.6748	0.6810
ResNet50 Random	0.3253	0.3673	0.3495	0.4098	0.4435	0.5340	0.5364	0.5624	0.6522	0.6768
VGG16 ImageNet	0.3764	0.3303	0.4066	0.4568	0.5132	0.5414	0.6282	0.6175	0.6662	0.6656
ResNet50 ImageNet	0.3225	0.3864	0.3884	0.4211	0.4637	0.5514	0.5651	0.6088	0.6281	0.6828
NPID [24]	**0.4115**	0.3401	0.3683	0.4606	0.4929	0.4959	0.5646	0.5801	0.6478	0.6714
MoCo [32]	0.3984	0.3135	0.4210	0.4409	0.5057	0.5134	0.6155	0.5948	0.6457	0.6726
MoCo v2 [33]	0.3854	0.3305	0.3908	0.4636	0.4774	0.5034	0.5994	0.5809	0.6564	0.6696
Ours (VGG16)	0.3817	0.3803	0.4606	**0.5314**	0.5646	0.6126	0.6419	0.6618	0.6790	0.7031
Ours (Resnet50)	0.3662	**0.4104**	0.4175	0.4646	0.5099	0.6050	0.6110	0.6538	**0.6922**	0.6942
Ours* (VGG16)	0.3933	0.4004	0.4672	0.5150	**0.5696**	**0.6385**	**0.6468**	**0.6823**	0.6869	**0.7065**
Ours* (Resnet50)	0.4043	0.3555	**0.4788**	0.5187	0.5617	0.6292	0.6463	0.6697	0.6817	0.7039

IoU is used as the metric. The highest scores are marked in bold. Ours* represents ImageNet pretraining + self-supervised pretraining of our method.

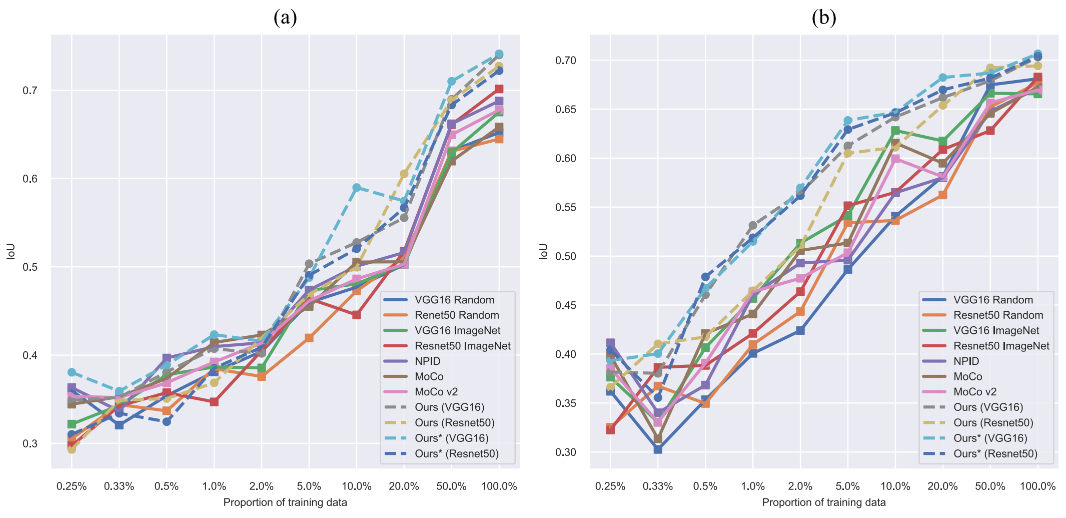

Fig. 2. Semantic segmentation results. (a) Results on Vaihingen dataset. (b) Results on Potsdam dataset. The dotted line shows the result of our method. Ours* represents ImageNet pretraining + self-supervised pretraining of our method. (a) Segmentation IoU of Vaihingen. (b) Segmentation IoU of Potsdam.

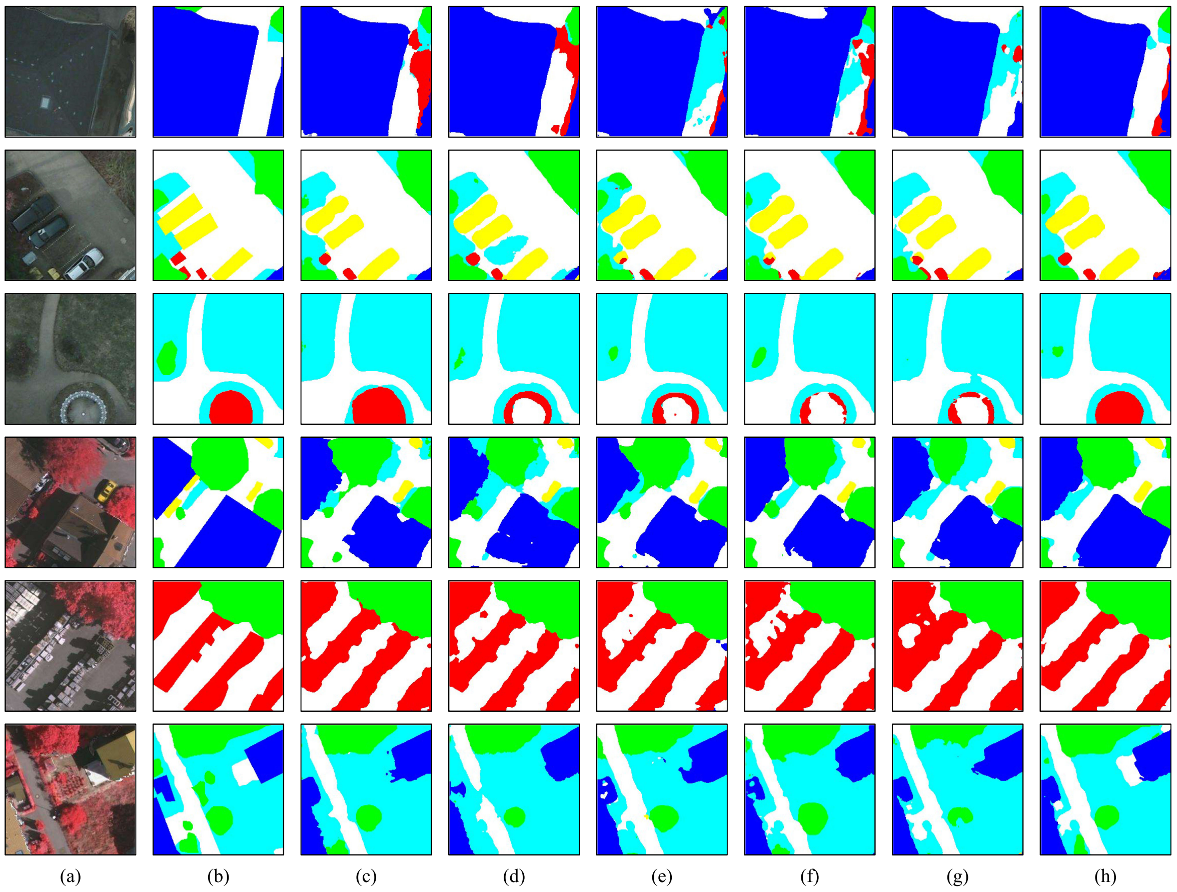

Fig. 3. (Better viewed in color) Some examples of the semantic segmentation results of comparison methods on the Potsdam (the first three rows) and Vaihingen dataset [48] (the last two rows). The first column shows the input images, and the second column shows the label image. The third to seventh columns are the results of the comparison methods. The last column is the result of our method (VGG16*). (a) Image. (b) Label. (c) Random. (d) ImageNet. (e) NPID. (f) Moco. (g) Moco v2. (h) Ours.

tables and the abscissa in the figure represent the proportion of training data in the total training set. “100%” means that all training data are used to train the segmentation network, while “0.25%” means that only 0.25% of training data are used to train the segmentation network. The number of training images represented by different proportions can be calculated as follows: $N_r = \text{floor}(N * r)$, where r represents the proportion listed in the tables and N means the total number of training data. floor represents rounding operation and N_r means the number of training data under proportion r. In addition, we have counted the number of training data for each class under different proportions to ensure that even at the ratio of 0.25% and 0.33%, each category has corresponding training data. It can be seen that our method achieves the best results in above two datasets and obtain the best segmentation results in almost every scale of training set. As semantic segmentation usually requires a large number of effective low-level features to supplement the details of outputs, the results suggest that our method can extract better low-level features than other methods. Therefore, we can use a sufficiently large unlabeled dataset which can be obtained easily to pretrain any segmentation model before fine-tuning on target datasets even with every limited labels. The results show that our method is qualified to be an alternative or even a better replacement for the ImageNet pretraining on standard remote sensing image segmentation tasks.

In addition, it can been seen from Fig. 2 that the segmentation performance has a positive correlation with the scale of training data for every method. With the increase of training data, the segmentation performance has been improved. The performance improvement of our method is more obvious when the amount of training data is limited. When the training data increases to 100%, the advantages of our method that that of other methods begin to decrease. We can reasonably speculate that if the training data are large enough, the advantages of our method will eventually be wiped out. However, due to the difficulty of segmentation data annotation in reality, we can hardly get enough training data, and we can not know what scale of labeled data is enough for network training. Therefore, our method can effectively improve the accuracy of segmentation, and reduce the workload of data annotation.

Fig. 3 shows some examples of the semantic segmentation results of comparison methods on the Potsdam (the first three rows) and Vaihingen dataset [48] (the last three rows). The first column shows the input images, and the second column shows the label image. The third to seventh columns are the results of the comparison methods. The last column shows the results of our method. All the models are trained with 100% training

TABLE V
CLOUD DETECTION RESULTS

	0.5%	1%	2%	5%	10%	20%	50%	100%
VGG16 Random	0.6546	0.6402	0.6290	0.6975	0.6705	0.6924	0.6977	0.7148
ResNet50 Random	0.6582	0.6364	0.6198	0.6905	0.6764	0.6889	0.7093	0.7320
VGG16 ImageNet	0.6683	0.6705	0.6965	0.6726	0.7325	0.7405	0.7351	0.7601
ResNet50 ImageNet	**0.6892**	0.6586	**0.7077**	0.6817	0.6618	0.7219	0.7026	0.7344
NPID [24]	0.6444	0.6418	0.5999	0.6893	0.6583	0.7047	0.7203	0.7175
MoCo [32]	0.6567	0.6318	0.6503	0.5748	0.6715	0.6850	0.6776	0.6754
MoCo v2 [33]	0.6487	0.6439	0.6567	0.6566	0.6801	0.7111	0.7114	0.7405
Ours (VGG16)	0.6748	0.6874	0.6913	0.6960	0.7034	0.7219	0.7366	0.7486
Ours (Resnet50)	0.6847	0.6651	0.6988	**0.7183**	**0.7432**	**0.7430**	0.7574	**0.7652**
Ours* (VGG16)	0.6839	**0.6906**	0.7033	0.7027	0.7117	0.7184	0.7391	0.7551
Ours* (Resnet50)	0.6728	0.6861	0.6855	0.7130	0.7377	0.7524	**0.7597**	0.7603

IoU is used as the metric. The highest scores are marked in bold. Ours* represents ImageNet pretraining + self-supervised pretraining of our method

TABLE VI
SNOW DETECTION RESULTS

	0.5%	1%	2%	5%	10%	20%	50%	100%
VGG16 Random	0.0502	0.2039	0.2518	0.3308	0.3290	0.3882	0.3996	0.4302
ResNet50 Random	0.0057	0.1578	0.2513	0.2928	0.3190	0.4154	0.4012	0.4447
VGG16 ImageNet	0.2147	0.2494	0.3447	0.3273	0.4240	0.4772	0.4662	0.5580
ResNet50 ImageNet	0.2326	0.2743	0.3425	0.2911	0.3115	0.3670	0.4060	0.4700
NPID [24]	0.2290	0.3088	0.1832	0.3146	0.2961	0.3983	0.4143	0.4248
MoCo [32]	0.2112	0.2138	0.2858	0.2125	0.3754	0.3381	0.3992	0.3112
MoCo v2 [33]	0.2045	0.2294	0.2822	0.3110	0.3377	0.4171	0.3733	0.4624
Ours (VGG16)	0.2173	0.2674	0.3282	0.3720	0.4198	0.4267	0.4857	0.5058
Ours (Resnet50)	0.2031	0.2761	0.3566	**0.4201**	0.4739	0.4720	**0.5431**	0.5390
Ours* (VGG16)	**0.2631**	0.2860	**0.3720**	0.3729	0.4046	0.3987	0.5160	0.5529
Ours* (Resnet50)	0.1732	**0.3159**	0.3181	0.4039	**0.4910**	**0.5270**	0.5311	**0.5640**

IoU is used as the metric. The highest scores are marked in bold. Ours* represents ImageNet pretraining + self-supervised pretraining of our method

data. It can be seen that our method can effectively improve the performance of semantic segmentation, and can reduce the false alarms.

D. Cloud/Snow Detection

The cloud/snow detection is in essence a pixel classification task, so it can be regarded as a special semantic segmentation problem. Cloud/snow detection task consists of two sub tasks: Cloud detection and snow detection. The difficulty of cloud/snow detection lies in the high similarity between cloud and snow. In addition, the cloud samples are widely distributed and easy to obtain, but the snow samples are limited by terrain and season so that they are relative rare. The cloud/snow detection results are shown in Tables V, VI, and Fig. 4. Considering that the snow samples are relatively rare, we start from 0.5% of the training data to verify the effect of our method on different scale of training data, rather than from 0.25% as in the semantic segmentation experiment. It can be seen that our methods have achieved the best results both on cloud and snow detection.

When the scale of training data is limited, almost all the methods can achieve a good cloud detection performance, but under the same scale of training data, the performance of snow detection is pretty low. This is because even if some clouds are difficult to distinguish, cloud detection is still a relatively easy task. Most clouds have similar texture information, and a small amount of annotation data is enough for relatively simple cloud detection. But for snow detection, most of snow samples are similar with cloud ones and the scale of snow samples is usually small, which leads to the networks tends to label snow as cloud, resulting in the performance degradation of snow detection. As can be seen from the Tables V, VI, and Fig. 4, our method is superior to other methods in cloud detection results, but the advantage is not particularly great. For snow detection, our method is significantly better than other methods, especially in the case of less labeled data.

Fig. 5 shows some examples of the cloud detection results of comparison methods on the Levir_CS [49] dataset. The first column shows the input cloud images, and the second column shows the label image. The third to seventh columns are the results of the comparison methods. The last column is the predicted cloud result of our method. The parts marked in gray correspond to the cloud in the input image, and the parts marked in black and white correspond background and snow separately. For the

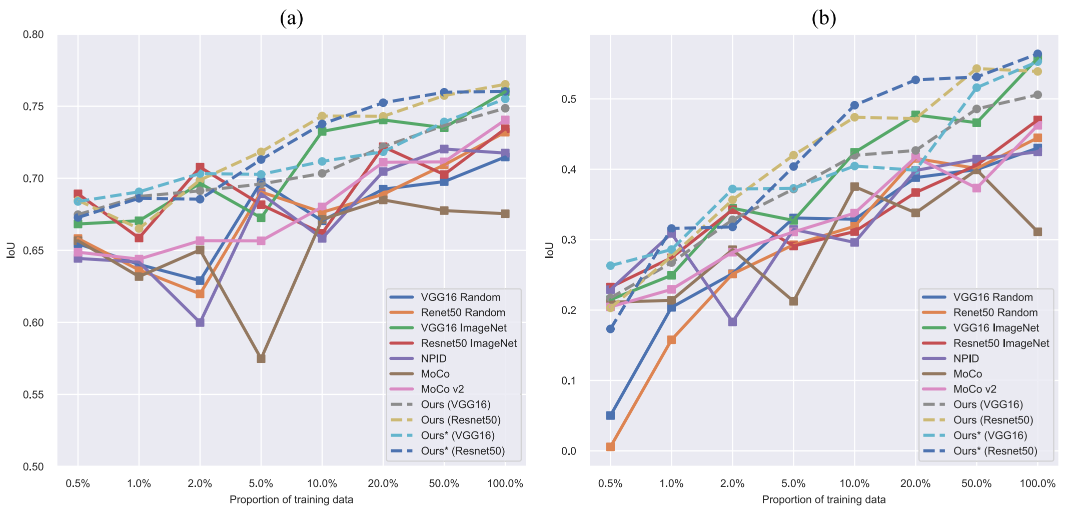

Fig. 4. Cloud/snow detection results. (a) Cloud detection results. (b) Snow detection results. The dotted line shows the result of our method. Ours* represents ImageNet pretraining + self-supervised pretraining of our method.

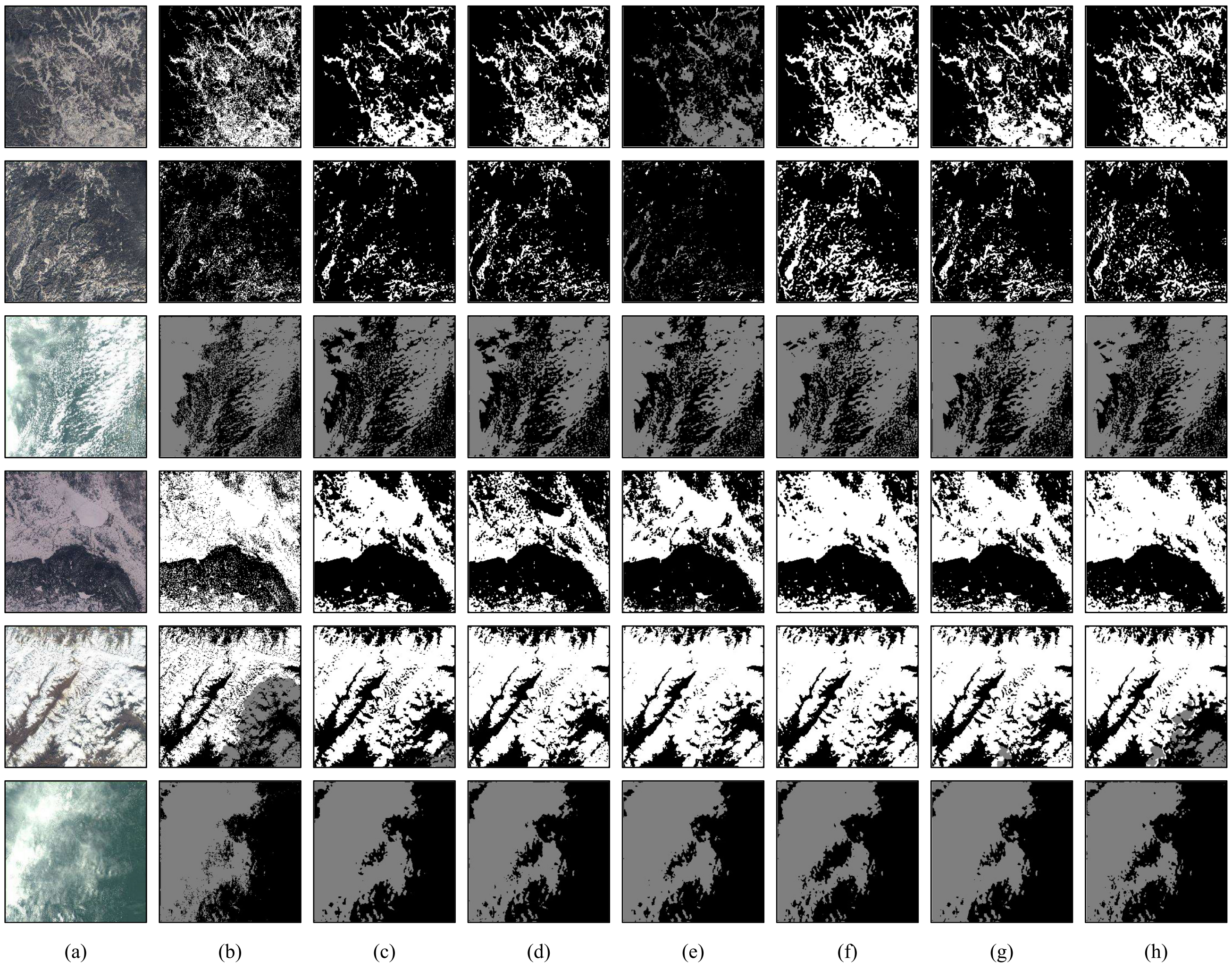

Fig. 5. (Better viewed in color) Some examples of the cloud / snow detection results of comparison methods on the Levir_CS [49] dataset. The first column shows the input cloud images, and the second column shows the label image. The third to seventh columns are the results of the comparison methods. The last column is the predicted cloud result of our method (VGG16*). The parts marked in gray correspond to the cloud in the input image, and the parts marked in black and white correspond background and snow separately. (a) Image. (b) Label. (c) Random. (d) ImageNet. (e) NPID. (f) Moco. (g) Moco v2. (h) Ours.

TABLE VII
ABLATION STUDIES OF MULTITASK SELF-SUPERVISED REPRESENTATION LEARNING. ABLATIONS ARE PERFORMED ON 1) ATP, 2) CONTRASTIVE, AND 3) INPAINTING

ATP	Contrastive	Inpainting	Resnet50 Vaihingen	VGG16 Vaihingen
×	×	×	0.6448	0.6521
✓	×	×	0.6631	0.6672
✓	✓	×	0.6803	0.7092
✓	✓	✓	0.7274	0.7400

cloud detection, the performance of our method is comparable with other methods, but for the snow detection, we can obviously see that our method has achieved better snow detection results.

E. Ablation Studies

We design the following ablation analysis to analyze the importance of each pretext task in our method, including 1) the inpainting task, 2) the ATP task, and 3) the contrastive learning task. We first start from a baseline approach where we directly train the networks on downstream tasks from scratch. Then the above pretext tasks are added one by one to pretrain the networks with self-supervised losses. Finally we fine-tune the pretrained networks on the Vaihingen dataset and their mIoUs are recorded. Results are shown in Table VII. The results show that each task achieves a noticeable improvement on the semantic segmentation task, where the "ATP" and "Contrasitive" tasks improve the segmentation accuracy by about 2%, while the "Inpainting" task further improves the segmentation accuracy by 4%. As the inpainting task pays more attention to the low-level features, it improves the semantic segmentation more significantly. In addition, although ATP and contrastive tasks are both for learning high-level features, they can continue to improve the accuracy of segmentation from the results. The features they focus on and the effects they produce are not exactly the same. The ATP task may make the network pay more attention to the changes in the texture and position of objects, while the contrastive learning task may help networks pay attention to the semantic information of images.

F. Experimental Results Analysis

In this part, we analysis the performance of our method in the face of different task with various difficulty and training data scale. In terms of the scale of training data, our method can significantly reduce the demand for training data, which is manifested in two aspects. On the one hand, it can be seen from Tables III and IV that the performance of our method is limited when the training data is extremely small (0.25%, 0.33%). But with the increase of training data, our method first shows a leap in performance. Compared with our method with the most commonly used ImageNet pretraining method, our method can save almost half of the training data, i.e., our method with only half of the training data can achieve the performance that ImageNet pretraining method can achieve with all data. compared with random initialization, our method can achieve the comparable performance with only 50% labeled data on Vaihingen dataset and 20% labeled data on Potsdam dataset. On the other hand, when using all the training data, our method can still improve the segmentation mIoU by 4%. Without changing the network structure, the most effective way to improve the performance is to increase the training data. But the segmentation data annotation is very time-consuming and laborious, our method provides a new way to continue to improve the performance.

In addition, the experiment results of cloud/snow detection shows performance of different methods for segmentation tasks with various difficulty. From Fig. 4, we can see that the curve of cloud detection performance is relatively flat. Although our method is still better than other methods in most cases, the improvement of cloud detection is not particularly great. But for the snow detection task, our method has brought great performance improvement to the snow detection. The reason for the difference between cloud detection and snow detection is that cloud detection is a relatively simple task. In most cases, cloud and ground objects are easy to distinguish. However, snow and cloud have similar characteristics, and usually the cloud samples are more than twice as large as the snow samples, which leads to the network tends to label snow as cloud and makes it difficult to improve the performance of snow detection. The experimental results on snow detection shows that our method is more effective in the face of complex tasks. Compared with random initialization, only 20% labeled data for cloud detection and 10% labeled data for snow detection are needed to achieve the comparable performance.

V. DISCUSSION AND FUTURE WORK

The self-supervised representation learning method provides an effective way to utilize large amount of unlabeled data. Up to now, most deep learning methods rely on a large number of labeled data, but for remote sensing images, the vast majority of available data are not labeled. How to use these remote sensing images effectively is a great challenge to be solved. In order to improve the utilization efficiency of large-scale unlabeled remote sensing data via self-supervised representation learning method, the following three issues need to be considered in the future.

1) Since self-supervised representation learning needs to cooperate with large-scale datasets to give full play to its advantages, future work will consider building a large-scale remote sensing representation learning dataset. The dataset needs to fully consider the characteristics of multisource and multiresolution of remote sensing images, and try to cover the main data sources of remote sensing images.
2) As the great difference between remote sensing images and natural images, the method which performs well for natural images may not be effective for remote sensing images. Therefore, we will systematically study and compare the differences of different methods in these two images in the future, so as to provide reference that help self supervised representation learning to play a greater role in the field of remote sensing.

3) In addition to the image itself, remote sensing images also contains a lot of geographic information. We will consider how to apply this geographic information into the self-supervised representation learning method of remote sensing images, so as to greatly improve the performance of networks for remote sensing images.

VI. Conclusion

This article proposes a self-supervised representation learning method for remote sensing semantic segmentation. Considering the characteristics of remote sensing images, we design multiple pretext tasks (inpainting, augmentation transform prediction, and contrastive learning) to guide networks to learn both low-level and high-level features at the same time. The pretrained models can be applied to various downstream tasks as an alternative of the ImageNet pretrained models. The experimental results show that our method outperforms random initialization, ImageNet pretraining, and other self-supervised methods in remote sensing the semantic segmentation task. Our method has achieved better results especially with limited training data. This proves that the model trained by our methods can be considered as an effective initialization for various remote sensing image semantic segmentation tasks and can be also used to improve the performance of semantic segmentation for remote sensing images.

References

[1] W. Sun and R. Wang, “Fully convolutional networks for semantic segmentation of very high resolution remotely sensed images combined with DSM,” *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 3, pp. 474–478, Mar. 2018.

[2] B. Yu, L. Yang, and F. Chen, “Semantic segmentation for high spatial resolution remote sensing images based on convolution neural network and pyramid pooling module,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 9, pp. 3252–3261, Sep. 2018.

[3] L. Ding, J. Zhang, and L. Bruzzone, “Semantic segmentation of large-size VHR remote sensing images using a two-stage multiscale training architecture,” *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 8, pp. 5367–5376, Aug. 2020.

[4] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 3431–3440.

[5] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervention*, 2015, pp. 234–241.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[7] Z. Zou, W. Li, T. Shi, Z. Shi, and J. Ye, “Generative adversarial training for weakly supervised cloud matting,” in *Proc. IEEE/CVF Int. Conf. Comput. Vision*, 2019, pp. 201–210.

[8] Z. Zou, T. Shi, W. Li, Z. Zhang, and Z. Shi, “Do game data generalize well for remote sensing image segmentation?” *Remote Sens.*, vol. 12, no. 2, 2020, Art. no. 275.

[9] W. Li, Z. Zou, and Z. Shi, “Deep matting for cloud detection in remote sensing images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 12, pp. 8490–8502, Dec. 2020.

[10] J. Zhang, M. Zhang, B. Pan, and Z. Shi, “Semisupervised center loss for remote sensing image scene classification,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 1362–1373, 2020.

[11] F. Xie, M. Shi, Z. Shi, J. Yin, and D. Zhao, “Multilevel cloud detection in remote sensing images based on deep learning,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 8, pp. 3631–3640, Aug. 2017.

[12] Z. An and Z. Shi, “Scene learning for cloud detection on remote-sensing images,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 8, pp. 4206–4222, Aug. 2015.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2009, pp. 248–255.

[14] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, “S4l: Self-supervised semi-supervised learning,” *Proc. IEEE/CVF Int. Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 1476–1485.

[15] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash, “Boosting self-supervised learning via knowledge transfer,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 9359–9367.

[16] P. Liu, M. Lyu, I. King, and J. Xu, “Selflow: Self-supervised learning of optical flow,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 4571–4580.

[17] Z. Feng, C. Xu, and D. Tao, “Self-supervised representation learning by rotation feature decoupling,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 10364–10374.

[18] D. Li, W.-C. Hung, J.-B. Huang, S. Wang, N. Ahuja, and M.-H. Yang, “Unsupervised visual representation learning by graph-based consistent constraints,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 678–694.

[19] J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial feature learning,” 2016, *arXiv:1605.09782*.

[20] J. Yang, D. Parikh, and D. Batra, “Joint unsupervised learning of deep representations and image clusters,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 5147–5156.

[21] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.

[22] M. Noroozi, H. Pirsiavash, and P. Favaro, “Representation learning by learning to count,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 5898–5906.

[23] O. Kilinc and I. Uysal, “Learning latent representations in neural networks for clustering through pseudo supervision and graph-based activity regularization,” 2018, *arXiv:1802.03063*.

[24] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, “Unsupervised feature learning via non-parametric instance discrimination,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 3733–3742.

[25] I. Misra and L. v. d. Maaten, “Self-supervised learning of pretext-invariant representations,” in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2020, pp. 6707–6717.

[26] H. Lee, S. J. Hwang, and J. Shin, “Rethinking data augmentation: Self-supervision and self-distillation,” 2019, *arXiv:1910.05872*.

[27] X. Yan, I. Misra, A. Gupta, D. Ghadiyaram, and D. Mahajan, “Clusterfit: Improving generalization of visual representations,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6509–6518.

[28] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” 2020, *arXiv:2002.05709*.

[29] P. Bojanowski and A. Joulin, “Unsupervised learning by predicting noise,” 2017, *arXiv:1704.05310*.

[30] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 132–149.

[31] J. Donahue and K. Simonyan, “Large scale adversarial representation learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 10542–10552.

[32] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2020, pp. 9729–9738.

[33] X. Chen, H. Fan, R. Girshick, and K. He, “Improved baselines with momentum contrastive learning,” 2020, *arXiv:2003.04297*.

[34] J. Huang, Q. Dong, S. Gong, and X. Zhu, “Unsupervised deep learning by neighbourhood discovery,” 2019, *arXiv:1904.11567*.

[35] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” 2018, *arXiv:1803.07728*.

[36] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 1422–1430.

[37] R. Santa Cruz, B. Fernando, A. Cherian, and S. Gould, “DeepPermNet: Visual permutation learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3949–3957.

[38] T. H. Trinh, M.-T. Luong, and Q. V. Le, “Selfie: Self-supervised pretraining for image embedding,” 2019, *arXiv:1906.02940*.

[39] D. Kim, D. Cho, D. Yoo, and I. S. Kweon, “Learning image representations by completing damaged jigsaw puzzles,” in *Proc. IEEE Winter Conf. Appl. Comput. Vision*, 2018, pp. 793–802.

[40] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *Proc. Eur. Conf. Comput. Vision*, 2016, pp. 69–84.
[41] L. Zhang, G.-J. Qi, L. Wang, and J. Luo, “Aet vs. AED: Unsupervised representation learning by auto-encoding transformations rather than data,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 2547–2555.
[42] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 2536–2544.
[43] M. Minderer, O. Bachem, N. Houlsby, and M. Tschannen, “Automatic shortcut removal for self-supervised representation learning,” 2020, *arXiv:2002.08822*.
[44] G. Larsson, M. Maire, and G. Shakhnarovich, “Learning representations for automatic colorization,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 577–593.
[45] G. Larsson, M. Maire, and G. Shakhnarovich, “Colorization as a proxy task for visual understanding,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6874–6883.
[46] S. Jenni and P. Favaro, “Self-supervised feature learning by learning to spot artifacts,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 2733–2742.
[47] R. Zhang, P. Isola, and A. A. Efros, “Split-brain autoencoders: Unsupervised learning by cross-channel prediction,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 1058–1067.
[48] F. Rottensteiner *et al.*, “The ISPRS benchmark on urban object classification and 3D building reconstruction,” *ISPRS Ann. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 1, no. 1, pp. 293–298, 2012.
[49] X. Wu, Z. Shi, and Z. Zou, “A geographic information-driven method and a new large scale dataset for remote sensing cloud/snow detection,” *ISPRS J. Photogrammetry Remote Sens.*, vol. 174, pp. 87–104, 2021.
[50] I. Goodfellow *et al.*, “Generative adversarial nets,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[51] P. Goyal, D. Mahajan, A. Gupta, and I. Misra, “Scaling and benchmarking self-supervised visual representation learning,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2019, pp. 6391–6400.
[52] A. Newell and J. Deng, “How useful is self-supervised pretraining for visual tasks?” in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2020, pp. 7345–7354.
[53] A. Kolesnikov, X. Zhai, and L. Beyer, “Revisiting self-supervised visual representation learning,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 1920–1929.
[54] S. Vincenzi *et al.*, “The color out of space: Learning self-supervised representations for earth observation imagery,” 2020, *arXiv:2006.12119*.
[55] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, “Graph convolutional networks for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, to be published, doi: 10.1109/TGRS.2020.3015157.
[56] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, “An augmented linear mixing model to address spectral variability for hyperspectral unmixing,” *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1923–1938, Apr. 2019.
[57] B. Rasti *et al.*, “Feature extraction for hyperspectral imagery: The evolution from shallow to deep: Overview and toolbox,” *IEEE Geosci. Remote Sens. Mag.*, vol. 8, no. 4, pp. 60–88, Dec. 2020.
[58] D. Hong, X. Wu, P. Ghamisi, J. Chanussot, N. Yokoya, and X. X. Zhu, “Invariant attribute profiles: A spatial-frequency joint feature extractor for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 6, pp. 3791–3808, Jun. 2020.
[59] W. Chen, X. Zheng, and X. Lu, “Hyperspectral image super-resolution with self-supervised spectral-spatial residual network,” *Remote Sens.*, vol. 13, no. 7, 2021, Art. no. 1260.
[60] K. Li, Y. Qin, Q. Ling, Y. Wang, Z. Lin, and W. An, “Self-supervised deep subspace clustering for hyperspectral images with adaptive self-expressive coefficient matrix initialization,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 3215–3227, 2021.
[61] Y. Wang *et al.*, “Self-supervised low-rank representation (SSLRR) for hyperspectral image classification,” IEEE Trans. Geosci. Remote Sens., vol. 56, no. 10, pp. 5658–5672, Oct. 2018.
[62] D. Hong *et al.*, “More diverse means better: Multimodal deep learning meets remote-sensing imagery classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 5, pp. 4340–4354, May 2021.
[63] J. Kang, R. Fernandez-Beltran, P. Duan, S. Liu, and A. J. Plaza, “Deep unsupervised embedding for remotely sensed images based on spatially augmented momentum contrast,” *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 3, pp. 2598–2610, Mar. 2021.
[64] K. Ayush *et al.*, “Geography-aware self-supervised learning,” 2020, *arXiv:2011.09980*.
[65] O. Mañas, A. Lacoste, X. Giro-i Nieto, D. Vazquez, and P. Rodriguez, “Seasonal contrast: Unsupervised pre-training from uncurated remote sensing data,” 2021, *arXiv:2103.16607*.
[66] A. Romero, C. Gatta, and G. Camps-Valls, “Unsupervised deep feature extraction for remote sensing image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1349–1362, Mar. 2016.
[67] X. Lu, X. Zheng, and Y. Yuan, “Remote sensing scene classification by unsupervised representation learning,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 9, pp. 5148–5157, Sep. 2017.
[68] M. Neumann, A. S. Pinto, X. Zhai, and N. Houlsby, “In-domain representation learning for remote sensing,” 2019, *arXiv:1911.06721*.
[69] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, *arXiv:1409.1556*.
[70] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 770–778.
[71] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015, *arXiv:1502.03167*.
[72] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, “Object detection in optical remote sensing images: A survey and a new benchmark,” *ISPRS J. Photogrammetry Remote Sens.*, vol. 159, pp. 296–307, 2020.
[73] G.-S. Xia *et al.*, “Dota: A large-scale dataset for object detection in aerial images,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 3974–3983.
[74] Z. Zou and Z. Shi, “Random access memories: A new paradigm for target detection in high resolution aerial remote sensing images,” *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1100–1111, Mar. 2017.
[75] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

Wenyuan Li received the B.S. degree from North China Electric Power University, Beijing, China, in 2017. He is currently working toward the doctorate degree with the Image Processing Center, School of Astronautics, Beihang University, Beijing, China.

His research interests include deep learning, image processing, and pattern recognition.

Hao Chen received the B.S. degree from the Image Processing Center School of Astronautics, Beihang University, Beijing, China, in 2017, where he is currently working toward the doctorate degree in the Image Processing Center, School of Astronautics.

His research interests include machine learning, deep learning, and semantic segmentation.

Zhenwei Shi (Member, IEEE) received the Ph.D. degree in mathematics from Dalian University of Technology, Dalian, China, in 2005.

He was a Postdoctoral Researcher with the Department of Automation, Tsinghua University, Beijing, China, from 2005 to 2007. He was Visiting Scholar with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA, from 2013 to 2014. He is currently a Professor and the Dean of the Image Processing Center, School of Astronautics, Beihang University. He has authored or coauthored over 100 scientific papers in refereed journals and proceedings, including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON NEURAL NETWORKS, the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, and the IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. His personal website is http://levir.buaa.edu.cn/. His current research interests include remote sensing image processing and analysis, computer vision, pattern recognition, and machine learning.

Dr. Shi serves as an Associate Editor for the *Infrared Physics and Technology*.