Sensors and Materials, Vol. 35, No. 1 (2023) 183–198
MYU Tokyo

S & M 3162

Change Detection for High-resolution Remote Sensing Images Based on a UNet-like Siamese-structured Transformer Network

Chen Liang,[1,2,3] Pinxiang Chen,[1,2] Huiping Liu,[3]
Xiaokun Zhu,[1,2*] Yuanhao Geng,[1,2] and Zhenwei Zhang[4]

[1]Beijing Institute of Surveying and Mapping, 60 Nanlishi Road, Beijing 100045, China
[2]Beijing Key Laboratory of Urban Spatial Information Engineering, 60 Nanlishi Road, Beijing 100045, China
[3]School of Geography, Beijing Normal University, 19 Xinjiekouwai Road, Beijing 100875, China
[4]School of Remote Sensing and Geomatics Engineering,Nanjing University of Information Science and Technology, 219 Ningliu Road, Nanjing 210044, China

(Received October 18, 2022; accepted January 12, 2023)

Keywords: change detection, deep learning, Swin Transformer V2, UNet

Change detection using high-resolution remote sensing images provides crucial information for geospatial monitoring, which is of great importance as urbanization continues. However, current deep learning models for change detection tasks are mostly based on convolutional neural networks (CNNs), from which it is difficult to extract global information owing to the locality of convolution operations. In this paper, we propose a deep learning model, Siam-Swin-UNet (SSUNet), for remote sensing change detection. SSUNet is designed following the classic UNet-like encoder-decoder framework but has three major innovations: (1) The encoder and decoder are pure transformer-based and hierarchically structured, which avoids the locality problem of CNN but retains the capability of hierarchical representation. (2) The encoder incorporates the Siamese structure, which can process bi-temporal remote sensing images in parallel, and to which is added a fusion module to properly fuse the feature maps extracted from the Siamese structure. (3) The backbone of the SSUNet is Swin Transformer V2 blocks, which can be more stable in further applications of the model, such as transfer learning or scaling up of the model capacity. We experimented with the proposed SSUNet on the LEVIR-CD dataset, along with CNN-based models such as UNet, UNet++, FC-Siam-Conc, and FC-Siam-Diff. The results showed our model outperformed the CNN-based models by a large margin based on evaluation metrics including precision, recall, F1-score, and overall accuracy (OA). Moreover, we conducted ablation studies to further prove the effectiveness of the Siamese structure and the choice of the backbone. The proposed SSUNet has great potential for use in remote sensing change detection tasks.

1. Introduction

With the rapid development of urbanization, the types of land cover in urban areas are changing dramatically. Therefore, it is highly necessary to carry out urban geospatial monitoring.

*Corresponding author: e-mail: zhuxk@radi.ac.cn
https://doi.org/10.18494/SAM4180

ISSN 0914-4935
https://myukk.org/

Change detection is an important part of urban geospatial monitoring because it provides essential information for city planning, urban management, damage assessment, environmental protection and other factors.[1–4] Consequently, change detection has attracted more and more research interest in recent years.

In regard to change detection in urban areas, labor-heavy methods such as manual census and government sampling inspections are inefficient and can no longer meet the need for modern urban geospatial monitoring. Luckily, with the development of remote sensing technology, which has the characteristics of being useful in all types of weather, wide coverage, and fast update periods, remote sensing images can provide valuable data for geospatial monitoring. Change detection through remote sensing uses images at different periods covering the same area to study the changes that take place over time, and it has become an important means for land cover monitoring.[5] Research on change detection methods has appeared as early as the 1970s,[6] and based on a literature review, existing methods of remote sensing image change detection can be roughly divided into two categories: traditional methods and deep learning methods. The traditional detection methods, such as change vector analysis (CVA), principal component analysis (PCA), and support vector machines (SVMs), have made considerable contributions to research on remote sensing image change detection but cannot perform as well as needed on high-resolution images. In recent years, deep learning has been developed and has now achieved state-of-the-art performance on various computer vision (CV) tasks. Therefore, many scholars applied deep learning methods to change detection tasks and achieved satisfactory results because deep learning has significantly better generalization abilities than traditional methods.[7,8]

During the last few years, many neural networks have emerged and have been applied to the challenge of the detection of changes in images by remote sensing; convolutional neural networks (CNNs) are the most commonly used. Especially with the development of a fully convolutional network (FCN) that has the ability to make dense predictions, FCN and its adaptations soon became suitable choices for the task of change detection. For example, Daudt *et al.* proposed three different models based on FCNs and realized end-to-end training for the first time.[9] Liu *et al.* utilized UNet to extract high-level features from optical aerial images to accomplish the change detection task.[10] However, CNN-based models have difficulties extracting global information due to the mechanism by which convolution operates. Accordingly, many scholars have suggested different measures to address this problem, such as adding attention mechanisms and using dilated convolutions.[11]

As deep learning developed, researchers began to use methods from other fields to solve problems in CV. The transformer network was originally proposed for natural language processing (NLP) and has achieved great success in the field. Dosovitskiy *et al.* applied the transformer to the field of CV and proposed a vision transformer (ViT) network, which enabled the production of excellent results on ImageNet, CIFAR-100, VTAB, and other datasets.[12] Since then, many researchers have turned to transformer-based networks for various CV tasks and achieved comparable results with CNN-based models.[13] In 2021, Liu *et al.* proposed the Swin Transformer, which further bridges the gap between NPL and CV and has taken the CV field by storm.[14] The main idea of the Swin Transformer is to build a hierarchical representation that

adapts the multi-scale characteristics of image elements. The key innovation of the network is a scheme to calculate self-attention with shifted windows, which reduces the computational burden and helps to promote information exchange between windows, thus making it possible for a Swin Transformer to achieve state-of-the-art performance with relatively fewer computations. Subsequently, Liu *et al.* proposed another version of the Swin Transformer called Swin Transformer V2 in which several modifications were made to improve performance on large-scale models.[15] Applications of the Swin Transformer have proven its effectiveness in image classification and segmentation tasks and have shown great potential in change detection tasks.[16,17]

Motivated by the success of the Swin Transformer, we proposed a transformer-based network called the Siam-Swin-Unet (SSUnet) for remote sensing image change detection. The task of change detection distinguishes itself from other CV tasks in that its inputs are bi-temporal remote sensing images, and its outputs are dense predictions. Therefore, to accomplish the change detection task, SSUnet incorporates both the Siamese structure and the UNet structure to combine their strengths. The Siamese structure is capable of parallel processing bi-temporal images and the UNet structure is responsible for feature extraction and upsampling of change maps. Moreover, SSUNet leverages the Swin Transformer as a backbone in the UNet structure to make full use of its feature extraction ability and avoid the locality disadvantage of convolution operations. To be exact, the backbone used in our model is the Swin Transformer V2 version, which is more stable in training large models and in transfer learning; therefore, our model can have a better generalization ability when the model capacity is scaled up or is used for transfer learning. Our experiment shows that SSUNet can achieve better performance in change detection for high-resolution remote sensing images than other models. The architecture of SSUNet and our experiment with it in change detection tasks are discussed in detail in the following sections.

2 Materials and Methods

2.1 Materials

For urban geospatial monitoring, the dataset used for model training and model testing in this article is the LEVIR-CD dataset. LEVIR-CD is a large-scale remote sensing change dataset that focuses on building-related changes in 20 different regions in Texas, in the United States, with a time span from 2002 to 2018.[18] It consists of 637 bi-temporal image pairs made from very high resolution (0.5 m/pixel) Google Earth images. Those image pairs are further divided into a training dataset, a validation dataset, and a test dataset. The fully annotated LEVIR-CD contains a total of 31333 individual instances of changes in buildings.

2.2 Model architecture overview

The overall architecture of the proposed SSUNet is illustrated in Fig. 1. SSUNet consists of three parts: encoder, bottleneck, and decoder. Like UNet, the encoder is responsible for feature

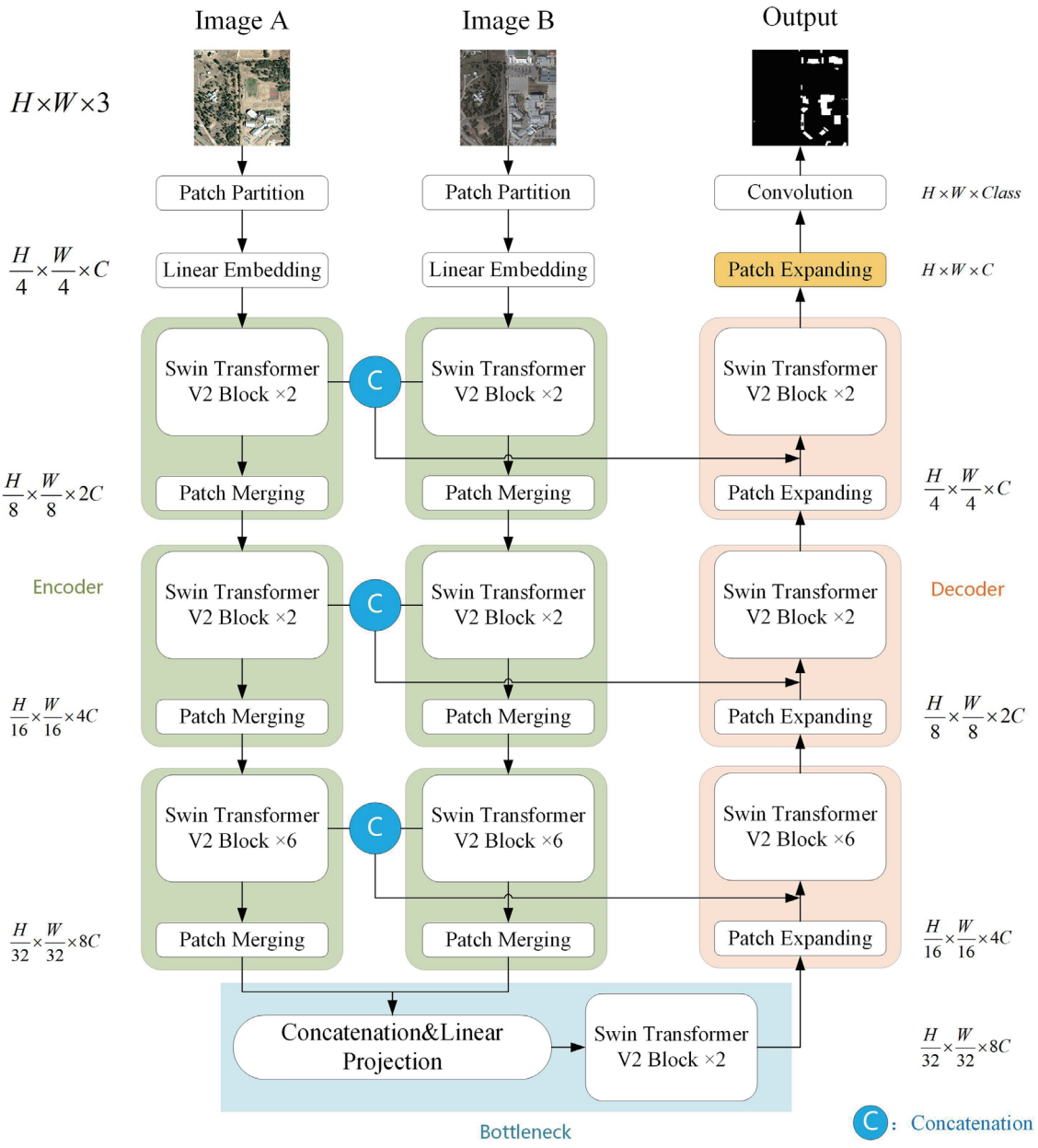

Fig. 1. (Color online) Aarchitecture of SSUNet.

extraction; the decoder, for image restoration; and the bottleneck, for feature fusion. In the encoder, a Siamese structure is incorporated to process bi-temporal remote sensing images in parallel. Each image is put through patch partition and linear embedding to convert the input into image tokens. These tokens are placed in the Swin Transformer blocks afterwards for feature extraction and then downsampled in patch merging layers. The Swin Transformer V2 version, which is an improvement over the original Swin Transformer version, is used in our model for the boost it provides in performance. The Swin Transformer blocks and patch merging

layers are repeated three times to obtain hierarchical feature maps. The bottleneck combines the features from the bi-temporal remote sensing images generated by the encoder with the help of concatenation, linear projection, and Swin Transformer V2 blocks. Following this, the fused feature map is put through the decoder to obtain dense predictions. The decoder has a design symmetric with that of the encoder, which consists of repeated Swin Transformer V2 blocks and patch expanding layers. The Swin Transformer V2 blocks are used to restore change information hierarchically, and the patch expanding layers are designed for upsampling. Moreover, in the process of gradually expanding the feature map to the original size of the input image, the feature map from the decoder is fused with the feature maps from the encoder on the same hierarchical level via skip connections to compensate for the loss of spatial information during downsampling. Lastly, the output of the decoder is put through a convolution layer to generate the change map. The SSUNet is a pure transformer-based network whose specific modules are described in the following in detail.

2.3 Swin Transformer V2 block

The Swin Transformer block serves as the backbone of the SSUNet and is known for its improvement over the multi-head self-attention (MSA) module based on shifted windows. A Swin Transformer block consists of an attention module, a two-layer MLP module with GELU nonlinearity in between, LayerNorm (LN) layers, and residual connections. There are two types of Swin Transformer blocks, and the only difference between them is how the multi-head self-attention is computed within the attention module. One type of attention module is called the window-based multi-head self-attention (W-MSA) module and the other is called the shifted window-based multi-head self-attention (SW-MSA) module. These two types of Swin Transformer blocks are usually used successively: the feature map first goes through the Swin Transformer block with the W-MSA module and then through the Swin Transformer block with the SW-MSA module.

In the original Swin Transformer block, an LN layer is applied before each (S)W-MSA module and each MLP, and a residual connection is applied after each (S)W-MSA module and each MLP. However, Liu *et al.* pointed out that in the pre-normalization configuration, the output value of each (S)W-MSA module and each MLP is directly merged back to the main branch through residual connection, leading to a significant increase in the activation values as the layers go deeper as a result of this accumulation.[15] When the model capacity is scaled up, the large discrepancy between different layers makes the model unstable for training. Therefore, some modifications were made in the Swin Transformer V2 version, one of which is a residual post-normalization (res-post-norm) approach. In this approach, the LN layer is applied after each (S)W-MSA module and each MLP so that the output is normalized before adding back to the main branch and the accumulation effect is suppressed. One of the reasons we chose the Swin Transformer block V2 version as the backbone of SSUNet is to eliminate problems if scaling up the model was needed. The differences between the original Swin Transformer blocks and the V2 version blocks are illustrated in Figs. 2(a) and 2(b). The calculations for two successive Swin Transformer V2 blocks are given by

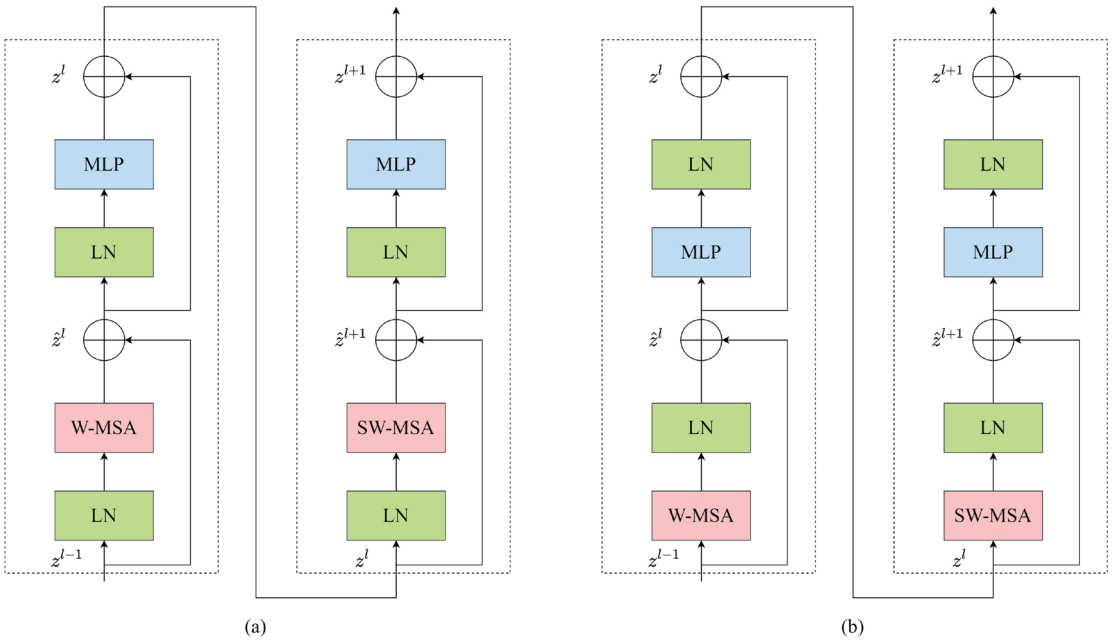

Fig. 2. (Color online) Illustration of two successive Swin Transformer blocks: (a) original Swin Transformer blocks; (b) Swin Transformer V2 blocks.

$$\begin{aligned} \hat{z}^l &= LN\left(W-MSA\left(z^{l-1}\right)\right)+z^{l-1}, \\ z^l &= LN\left(MLP\left(\hat{z}^l\right)\right)+z^l, \\ \hat{z}^{l+1} &= LN\left(SW-MSA\left(z^l\right)\right)+z^l, \\ z^{l+1} &= LN\left(MLP\left(\hat{z}^{l+1}\right)\right)+z^{l+1}, \end{aligned} \tag{1}$$

where $\hat{z}^l$ denotes the output of the (S)W-MSA module of the lth block and z^l denotes the output of MLP module of the lth block.

2.3.1 W-MSA

In the previous Transformer models used for computer vision, such as ViT, the computation of MSA is implemented globally, which means that each pixel in the feature map needs to be calculated with all pixels in the feature map. This type of global MSA module leads to huge computational burdens and is not applicable to dense prediction tasks. In the W-MSA module, the feature map is partitioned evenly into non-overlapping windows, and the MSA is computed within local windows. This window-based self-attention computing approach makes the computation complexity linear with respect to the size of the input image (when the window size is fixed), thus greatly improving model efficiency in computer vision tasks.

In the original Swin Transformer, the self-attention within each window is computed using Scaled Dot-Product Attention[19] and the formula as shown in

$$Attention\left(Q,K,V\right)=SoftMax\left(\frac{QK^{T}}{\sqrt{d}}+B\right)V, \tag{2}$$

where $Q,K,V\in\mathbb{R}^{M^2\times d}$ represent the query, key, and value matrices, respectively, and $B\in\mathbb{R}^{M^2\times M^2}$ represents the relative position bias. M represents the window height (or width), and d represents the channel dimension. However, Liu *et al.* found that for large visual models, the attention maps learned are often dominated by a small number of pixel pairs, especially after the res-post-normal approach was used.[15] Therefore, Liu *et al.* proposed scaled cosine attention to compute self-attention in the Swin Transformer V2 and the formula as shown in Eq. (3) for the attention computation of a pixel pair i and j:

$$Sim\left(\boldsymbol{q}_i,\boldsymbol{k}_j\right)=\frac{\cos\left(\boldsymbol{q}_i,\boldsymbol{k}_j\right)}{\tau}+B_{i,j}, \tag{3}$$

where $\boldsymbol{q}_i$ denotes the query vector of pixel i, and $\boldsymbol{k}_j$ represents the key vector of pixel j; $B_{i,j}$ denotes the relative position bias between pixel i and j; τ is a trainable scalar set to be larger than 0.01 and not shared between different layers and different heads. The cosine function is naturally normalized, so the attention values generated from it tend to be milder. The scaled cosine attention approach, coupled with the res-post-norm approach, makes the Swin Transformer V2 block more stable for training, especially when the model scale is large.

2.3.2 SW-MSA

In the W-MSA module, the self-attention calculation is limited in local windows and lacks information exchange between windows. Therefore, the SW-MSA module was introduced. The SW-MSA module also computes MSA in local windows, but the window partition strategy has been changed. Supposing the window size is $M \times M$, the windows in the SW-MSA module are shifted by $\left(\lfloor M/2\rfloor,\lfloor M/2\rfloor\right)$ pixels compared with those in the preceding block using the W-MSA module, as shown in Fig. 3. In this way, information exchange is possible across windows. However, the shifted window approach leads to an increased number of windows of inconsistent size. To solve this problem, Liu *et al.* proposed an efficient batch computation approach to compute self-attention in shifted window partitioning, which involves cyclic shifts and masked MSA.[14]

2.3.3 Log-spaced continuous relative position bias

During the computation of self-attention in the original Swin Transformer, relative position bias is used, as shown in Eq. (2), which has resulted in an uplift in the model performance. In

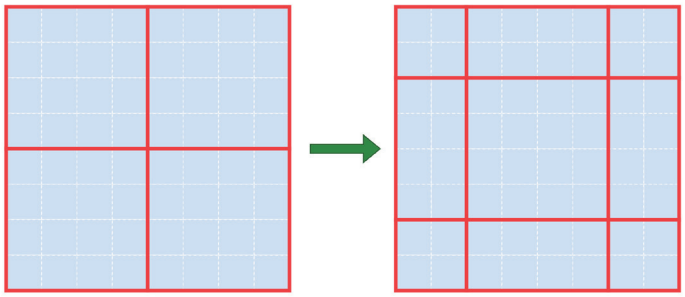

Fig. 3. (Color online) Illustration of shifted window approach in SW-MSA module.

Swin Transformer V2, a modified relative position bias method is proposed to improve the performance in transfer learning; this method is called log-spaced continuous relative position bias (Log-CPB).

Compared with the relative position bias used in the original Swin Transformer, there are primarily two differences with Log-CPB.

First, the relative position coordinates are transformed from linear space to log space because, during transfer learning, if the window size is larger than that in pre-training, a portion of the relative position range needs to be extrapolated. When the relative position coordinates are converted from linear space to log space, the extrapolation ratio is significantly smaller, which can effectively reduce the model performance drop due to varying window sizes. The log space transformation is made using Eq. (4).

$$\begin{aligned} \widehat{\Delta x} &= \operatorname{sign}(x) \cdot \log\left(1 + \Delta x\right) \\ \widehat{\Delta y} &= \operatorname{sign}(y) \cdot \log\left(1 + \Delta y\right) \end{aligned} \tag{4}$$

Here, Δx and Δy denote coordinates in linear space, and $\widehat{\Delta x}$ and $\widehat{\Delta y}$ denote coordinates in log space.

Second, instead of optimizing the parameterized biases, Log-CPB uses a meta network $\mathcal{G}$ to generate relative position biases, as shown in Eq. (5). The small network $\mathcal{G}$ can take in arbitrary relative coordinates, which makes the model more suitable for transfer learning.

$$B\left(\Delta x, \Delta y\right) = \mathcal{G}\left(\Delta x, \Delta y\right) \tag{5}$$

2.4 Encoder

To better process the change detection tasks for bi-temporal remote sensing images, the encoder in SSUNet has a Siamese structure that is capable of the parallel processing of bi-temporal images through parameter-shared layers. As shown in Fig. 1, images A and B are bi-temporal remote sensing images of the same area, which were homogenized in advance, and they have the same dimension of $H \times W \times 3$. When image A enters the encoder, it will first go through a Patch Partition layer that converts the image into tokens. The Patch Partition layer

divides the image into non-overlapping patches 4 × 4 in size, and then flattens each patch in the channel direction. Each patch is made of 16 pixels from the original image, so after flattening, there are 48 channels per patch. Therefore, the output of the Patch Partition layer has dimensions of $\frac{H}{4}\times\frac{W}{4}\times 48$. A Linear Embedding layer is then applied to map the channel dimension to a specified dimension C, and the output feature has dimensions of $\frac{H}{4}\times\frac{W}{4}\times C$. In our model, C is set to 96.

Afterward, the output of the Linear Embedding layer is put through three stages of feature extraction to obtain hierarchical representations. Each stage consists of Swin Transformer V2 blocks and a Patch Merging layer. The Swin Transformer V2 blocks are responsible for feature extraction, and the number of blocks used in each stage is 2, 2, and 6, respectively. The window size in each Swin Transformer V2 block is set to 7 for attention computation, and the same window size is used in the bottleneck and the decoder of the SSUNet. The Swin Transformer blocks do not change the dimension of the feature map, while the Patch Merging layer plays the role of downsampling. For example, the feature map from the Swin Transformer blocks in Stage 1 has dimensions of $\frac{H}{4}\times\frac{W}{4}\times C$. In the Patch Merging layer, an interval sampling is performed from the top-left of the feature map with a stride of 2 to generate four new feature maps of half the original size. After that, the four feature maps are concatenated in the channel dimension, and then a linear projection is applied to map the channel dimensions from 4C to 2C. Consequently, the output of the Patch Merging layer in Stage 1 has dimensions of $\frac{H}{8}\times\frac{W}{8}\times 2C$. As a result, if we denote the output of Stage 1 to Stage 3 for image A as $X^A_{E_1}, X^A_{E_2}, X^A_{E_3}$, their dimensions are as follows: $X^A_{E_1}\in\mathbb{R}^{\frac{H}{8}\times\frac{W}{8}\times 2C}$, $X^A_{E_2}\in\mathbb{R}^{\frac{H}{16}\times\frac{W}{16}\times 4C}$, and $X^A_{E_3}\in\mathbb{R}^{\frac{H}{32}\times\frac{W}{32}\times 8C}$. In addition, for image A, we denote the feature maps extracted from the Swin Transformer blocks in Stage 1 to Stage 3 as $X^A_{E_{1c}}\in\mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times C}$, $X^A_{E_{2c}}\in\mathbb{R}^{\frac{H}{8}\times\frac{W}{8}\times 2C}$, and $X^A_{E_{3c}}\in\mathbb{R}^{\frac{H}{16}\times\frac{W}{16}\times 4C}$, respectively, and store these feature maps in the process so that the decoder can restore the change map later. Similarly, the feature maps extracted from image B have the same dimensions as those from image A, and $X^B_{E_c}$, $X^B_{E_{2c}}$ and $X^B_{E_{3c}}$ are also stored for the decoder to use.

2.5 Bottleneck

The bottleneck of the proposed SSUNet fuses the feature maps extracted separately from image A and image B in the encoder. The bottleneck has a simpler structure that contains a concatenation layer, a linear projection layer, and two successive Swin Transformer V2 blocks. After three stages of feature extraction in the Siamese-structured encoder, the output feature maps from image A and image B are concatenated in the channel dimension through the concatenation layer, and the output dimensions are $\frac{H}{32}\times\frac{W}{32}\times 16C$. A linear projection layer is then applied to reduce the channel dimension to 8C. Finally, the feature map goes through the Swin Transformer blocks for feature fusion, and the output dimensions of the bottleneck are $\frac{H}{32}\times\frac{W}{32}\times 8C$.

2.6 Decoder

The decoder is responsible for hierarchically restoring the change information extracted from the encoder to the same size of the original input images. As illustrated in Fig. 1, the decoder consists of three stages of information restoration, one additional Patch Expanding layer called Patch Expanding 4X, a convolution layer, and several skip connections from the encoder. Each stage is composed of a Patch Expanding layer, a linear projection layer, several Swin Transformer V2 blocks, and skip connections. The Swin Transformer V2 blocks are responsible for decoding the change information, and the numbers of blocks used in each stage are 6, 2, and 2, respectively. As mentioned earlier, the Swin Transformer V2 blocks do not change the dimension of the feature map, while the Patch Expanding layer plays the role of upsampling. Skip connections are used to compensate for the loss of spatial information in the feature map, and linear projection layers are used for channel dimension adjustment.

We explain here how each decoding stage works by taking Stage 1 as an example. The input of Stage 1 is the output of the bottleneck, which is the fused feature map from the Siamese-structured encoder with the shape $\frac{H}{32}\times\frac{W}{32}\times 8C$. The input first goes through the Patch Expanding layer, which includes three steps: First, it uses a linear projection to increase the channel dimension to twice the original size. Second, it uses a rearrange operation to double both the height dimension and the width dimension while reducing the channel dimension to a quarter of that in the previous step. Finally, it uses linear normalization to generate the output. As a result, the height and width of the output feature map are both doubled, while the number of channels is halved. The output of the Patch Expanding layer in Stage 1 has dimensions of $\frac{H}{16}\times\frac{W}{16}\times 4C$, which is denoted as $X_{D_{1c}}$. A skip connection is applied to concatenate $X_{D_{1c}}$ with $X^{A}_{E_{3c}}$ and $X^{B}_{E_{3c}}$ from the encoder in the channel dimension for information merging. After concatenation, the feature map has dimensions of $\frac{H}{16}\times\frac{W}{16}\times 12C$. Then a linear projection layer is applied to reduce the channel dimension to 4C. After that, the feature map is put through successive Swin Transformer blocks to decode the change information. The output of Stage 1 in the decoder has dimensions of $\frac{H}{16}\times\frac{W}{16}\times 4C$. Similarly, the output dimensions of Stages 2 and 3 are $\frac{H}{8}\times\frac{W}{8}\times 2C$ and $\frac{H}{4}\times\frac{W}{4}\times C$, respectively.

After three stages of decoding and upsampling, the height and width of the feature map are still one-quarter of those of the original image. Therefore, an additional Patch Expanding 4X layer is applied to restore the feature map to its original size. The Patch Expanding 4X layer basically follows the same steps as the previous Patch Expanding layer but with different scales of expansion. To be exact, in Step 1, the channel dimension is increased by 16 times instead of twice, and in Step 2, the height and width are both expanded 4 times and the channel dimension is reduced to 1/16 of that in the previous step. As a result, the output of the Patch Expanding 4X layer has dimensions of $H \times W \times C$, which restores the feature map to its original size.

Finally, the decoder applies a convolution layer to transform the channel dimension from C to 2 to generate a change map with dimensions of $H \times W \times 2$. At this point, the change detection task is completed.

3. Results and Discussion

3.1 Implementation details and experiment results

The experiments were carried out under the PyTorch framework using a GPU of GeForce RTX 2080Ti with 11 GB of memory. Data augmentation techniques including random flips, random rotation, and Gaussian blur were applied to all the training samples before training. All the images were resized to 224 × 224 before they were put into SSUNet. During training, the optimizer used was the AdamW optimizer,[20] and the batch size was set at 16. In addition, we employed the "poly" learning rate policy for training, the same as the DeepLab network,[21] with an initial learning rate set at 1×10^{-3}. The epoch was set to 500 for sufficient training.

During model training and model test, we selected four indicators as the evaluation criteria: precision, recall, F1-score, and overall accuracy (OA). With TP, TN, FP, and FN denoting the number of true positives, the number of true negatives, the number of false positives, and the number of false negatives, respectively, the evaluation metrics are calculated as in Eqs. (6)–(9):

$$Precision = \frac{TP}{TP + FP}, \tag{6}$$

$$Recall = \frac{TP}{TP + FN}, \tag{7}$$

$$F1 = \frac{2 \times precision}{precision + recall}, \tag{8}$$

$$OA = \frac{TP + TN}{TP + TN + FP + FN}. \tag{9}$$

These indicators can be used to evaluate different aspects of the model, with the F1-score being the most important according to many scholars because it is a comprehensive review of precision and recall.[22] The closer each of the four evaluation indicators is to 100%, the better the performance of the model. For the LEVIR-CD dataset after 500 epochs of training, the test results showed that the proposed SSUNet reached the precision of 88.43%, the recall of 86.89%, the F1-score of 87.61%, and the OA metrics of 98.14%.

3.2 Comparative experiments

To prove the performance improvement of our proposed SSUNet, which is a pure transformer-based deep learning network, we conducted a series of comparative experiments with several CNN-based deep learning networks, including UNet, UNet++, FC-Siam-Conc, and FC-Siam-Diff.

UNet is famous for its U-shaped encoder-decoder structure, which achieved state-of-the-art performance in medical image segmentation when it was proposed.[23] However, UNet was not developed for change detection tasks, so it cannot take in bi-temporal images directly. The common practice is to use the early fusion method, which means stacking the bi-temporal images into a single image and then following the standard semantic segmentation procedure to extract the change map. The same procedure applies to UNet++ when the comparative experiment is conducted. UNet++ is an adaptation of UNet with redesigned dense skip connections and deep supervision, which has proven to be more powerful in medical image segmentation.[24] Different from these two networks, FC-Siam-Conc and FC-Siam-Diff have a Siamese structure in the encoder part, which can handle bi-temporal remote sensing images directly. The difference between FC-Siam-Conc and FC-Siam-Diff lies in how the extracted feature maps from the Siamese structure are fused: the former directly concatenates the feature maps at the same hierarchical level for decoding, and the latter concatenates the absolute value of their difference during the decoding.[9]

These four comparative models were built using the source code provided by the authors and were trained separately using the same LEVIR-CD dataset. They were then tested on the same test dataset and were evaluated through the same metrics as the SSUNet model. Some of the change maps from the test results are shown in Fig. 4. As seen in Fig. 4, the actual changes from the bi-temporal remote sensing images are irregular and often very tiny patches, which is why it is difficult to extract the change map precisely. However, it is clear that our SSUNet model provides a better representation of the ground truth with fewer false changes and fewer omissions of actual changes compared with the other four models. Quantitatively, the evaluation metrics of comparative models are shown in Table 1. From the data in Table 1, it is clear that our SSUNet model has the best results in all four-evaluation metrics. Furthermore, the models that have the

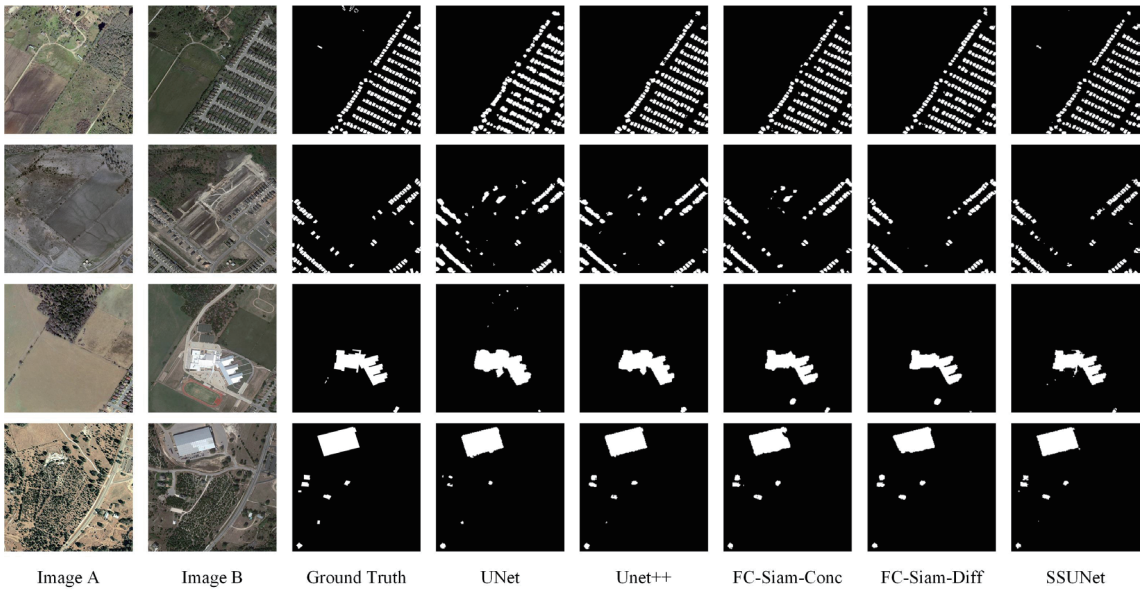

Fig. 4. (Color online) Test results of comparative experiment.

Table 1
Results of evaluation metrics of comparative experiment.

	Precision (%)	Recall (%)	**F1-score (%)**	OA (%)
U-Net	74.11	74.36	74.02	97.14
UNet++	79.47	80.25	79.69	97.43
FC-Siam-diff	85.14	74.24	78.86	97.43
FC-Siam-conc	81.81	78.51	79.73	97.57
SSUNet	88.43	86.89	87.61	98.14

Siamese structure tend to perform better than those that use the early fusion method. Among the four comparative models, FC-Siam-Diff has the best precision on the test dataset, while FC-Siam-Conc has the best F1-score, and UNet++ has the best recall rate.

3.3 Ablation study

The proposed SSUNet distinguishes itself from other models by having a Siamese structure in the encoder and using Swin Transformer V2 blocks as a backbone. To prove the effectiveness of such a design, we conducted ablation studies on these two aspects.

3.3.1 Effect of Siamese structure

Because of the unique nature of remote sensing change detection, the input requires two images at different time periods covering the same area, which is different when constructing deep learning models compared with routine semantic segmentation tasks. There are primarily two options based on previous studies: one is to stack the two images into a single image before putting them into the model, which is called the early fusion method; the other is to apply a Siamese structure to accept two inputs. In addition, when using the Siamese structure, there are two options when fusing two feature maps from two separate images: one is to concatenate the two feature maps directly in the manner of FC-Siam-Conc, and the other is to concatenate the absolute value of the differences between two feature maps in the manner of FC-Siam-Diff. In our SSUNet model, we adopted the former option for the Siamese structure: the two feature maps are concatenated directly in the channel dimension as in the FC-Siam-Conc model. To prove the effectiveness of the Siamese structure design in the proposed SSUNet, we designed two models for ablation experiments: one uses the early fusion method that stacks the bi-temporal images before feeding them into the model, which is denoted as the Swin-UNet-Early-Fusion (SUNet-EF) model; the other adopts the Siamese structure but uses the absolute value of the differences between the two feature maps for feature fusion, which is denoted as the Siam-Swin-UNet-Diff (SSUNet-Diff) model. Both models were trained and tested on the same LEVIR-CD dataset, and the test results along with the previous SSUNet model results are shown in Table 2. From Table 2, we can see that models with a Siamese structure achieved better results than the early fusion model. Of the two Siamese-structured models, SSUNet has better results on precision, recall, and F1-score than the Siam-Diff model, which proves the effectiveness of our proposed Siamese structure.

Table 2
Results of ablation experiment on Siamese structure.

	Precision (%)	Recall (%)	**F1-score (%)**	OA (%)
SUNet-EF	86.35	79.80	82.49	98.14
SSUNet-Diff	87.88	83.35	85.39	98.14
SSUNet	88.43	86.89	87.61	98.14

3.3.2 Effect of backbone

The proposed SSUNet model uses Swin Transformer V2 block as the backbone for feature extraction in the encoder and change information restoration in the decoder. The comparative experiments in Sect. 3.2 have already proven the effectiveness of using Swin Transformer as the backbone as it outperforms CNN-based models such as UNet, UNet++, FC-Siam-Conc, and FC-Siam-Diff. Furthermore, we intend to prove the effectiveness of choosing the Swin Transformer V2 block as the backbone for SSUNet instead of the original Swin Transformer block. Therefore, we conducted an ablation experiment to test a model with the original Swin Transformer block as the backbone. We built the model in the same way as the SSUNet model except for replacing the Swin Transformer V2 block with the original Swin Transformer block; we denote this model as SSUNet-V1. The SSUNet-V1 model was trained and tested on the same dataset as the SSUNet model, and the ablation test results are shown in Table 3. From Table 3 we can see that the two models have the same OA metric but that the SSUNet has better results on the precision, recall, and F1-score metrics. The ablation experiment showed that using the Swin Transformer V2 block as the backbone confers a slight advantage compared with using the original Swin Transformer block as the backbone.

Furthermore, the Swin Transformer V2 block gives the model an advantage in model applications because it is stable even when the model capacity is scaled up, and it does not suffer from degradation in performance when the model is transferred across different window sizes. As for the model capacity, because SSUNet is an integration of Swin Transformer V2 and UNet, our model is expandable by fusing different versions of Swin Transformer V2 networks with different model capacities. The SSUNet model described in Sect. 2 is based on the tiny configuration of Swin Transformer V2 to provide a balance of model performance and computational cost. In some complex cases, the model capacity needs to be scaled up to achieve better model performance; this can be easily accomplished by replacing the tiny configuration of Swin Transformer V2 with the larger configuration of Swin Transformer V2. The differences between these configurations are the number of Swin Transformer V2 blocks used in the extraction and restoration stages and the number of channels (C) in the Linear Embedding layer.[15] The res-post-normal approach and cosine attention in Swin Transformer V2 blocks can keep the model stable when the model capacity is scaled up. As for the window size in the Swin Transformer V2 blocks, it is set to 7 in our SSUNet model, but the window size can be varied during transfer learning based on the task demand. For example, it is better to adjust the window size so that the image can be divided by the window size. The window size can also be adjusted to tune the receptive fields if needed. The design of Log-CPB in Swin Transformer V2 blocks allows more flexibility and stability of the model when the window size has to be adjusted.

Table 3
Results of ablation experiment on backbone.

	Precision (%)	Recall (%)	**F1-score (%)**	OA (%)
SSUNet-V1	88.36	86.67	87.46	98.14
SSUNet	88.43	86.89	87.61	98.14

Therefore, for better generalization and transfer learning abilities, the Swin Transformer V2 block is the preferable choice for the model's backbone.

4. Conclusions

Change detection through high-resolution remote sensing images is an important tool for geospatial monitoring. However, because of the unique characteristics of high-resolution remote sensing images, automatically and accurately performing change detection tasks can be very challenging. In this study, we have proposed a deep learning model called SSUNet for the detection of remote sensing changes. It consists of three parts: an encoder, a decoder, and a bottleneck for fusion. SSUNet uses the Swin Transformer V2 block as the backbone and is a pure transformer-based network that overcomes the locality of CNN. It also incorporates the Siamese structure, which is capable of dealing with bi-temporal remote sensing images directly. In addition, when the model is used for transfer learning or the model capacity needs to be scaled up, SSUNet is capable of meeting these changes without degradation in performance because it uses the Swin Transformer V2 block as the backbone.

After training on the LEVIR-CD dataset, the SSUNet model achieved satisfactory results on evaluation metrics including precision, recall, F1-score, and OA. Comparative experiments have been carried out with CNN-based models such as UNet, Unet++, FC-Siam-Conc, and FC-Siam-Diff, and they have established that SSUNet achieved better performance than these classic models. Furthermore, ablation studies have proved the effectiveness of the Siamese structure in SSUNet and verified the choice of the backbone. Consequently, the proposed SSUNet has great potential to carry out remote sensing change detection tasks, and it can be an effective tool for geospatial monitoring.

Acknowledgments

Our research was supported by the Science and Technology Project of the Beijing Institute of Surveying and Mapping, namely the "Study on the Ecological Environment Changes of Yongding River in Beijing".

References

1 L. Ma, M. Li, T. Blaschke, X. Ma, D. Tiede, L. Cheng, Z. Chen, and D. Chen: Remote Sens. **8** (2016) 761. https://doi.org/10.3390/rs8090761
2 Y. You, J. Cao, and W. Zhou: Remote Sens. **12** (2020) 2460. https://doi.org/10.3390/rs12152460
3 Y. Kim and M.-J. Lee: Remote Sens. **12** (2020) 1978. https://doi.org/10.3390/rs12121978
4 X. Huang, L. Zhang, and T. Zhu: EEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **7** (2013) 105. https://doi.org/10.1109/jstars.2013.2252423

5 Y. Han, A. Javed, S. Jung, and S. Liu: Remote Sens. **12** (2020) 983. https://doi.org/10.3390/rs12060983
6 R. L. Lillestrand: IEEE Trans. Comput. C-**21** (1972) 654. https://doi.org/10.1109/t-c.1972.223570
7 Z. Hong, Z. Fan, R. Zhou, H. Pan, Y. Zhang, Y. Han, J. Wang, S. Yang, and Y. Jin: Sens. Mater. **34** (2022) 237. https://doi.org/10.18494/SAM3564
8 T.-Y. Lee, M.-H. Jeong, and A. Peter: Sens. Mater. **34** (2022) 251. https://doi.org/10.18494/SAM3732
9 R. C. Daudt, B. Le Saux, and A. Boulch: 2018 25th IEEE Int. Conf. Image Processing (ICIP, 2018) 4063. https://doi.org/10.1109/icip.2018.8451652
10 J. Liu, K. Chen, G. Xu, X. Sun, M. Yan, W. Diao, and H. Han: IEEE Geosci. Remote Sens. Lett. **17** (2019) 127. https://doi.org/10.1109/lgrs.2019.2916601
11 M. Zhang, G. Xu, K. Chen, M. Yan, and X. Sun: IEEE Geosci. Remote Sens. Lett. **16** (2018) 266. https://doi.org/10.1109/lgrs.2018.2869608
12 A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, and S. Gelly: arXiv preprint arXiv:2010.11929 (2020). https://doi.org/10.48550/arXiv.2010.11929
13 S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, and P. H. Torr: Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (2021) 6881. https://doi.org/10.1109/cvpr46437.2021.00681
14 Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo: arXiv preprint arXiv:2103.14030 (2021). https://doi.org/10.48550/arXiv.2103.14030
15 Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, and L. Dong: arXiv preprint arXiv:2111.09883 (2021). https://doi.org/10.48550/arXiv.2111.09883
16 H. Cao, Y. Wang, J. Chen, D. Jiang, X. Zhang, Q. Tian, and M. Wang: arXiv preprint arXiv:2105.05537 (2021). https://doi.org/10.48550/arXiv.2105.05537
17 C. Zhang, L. Wang, S. Cheng, and Y. Li: IEEE Trans. Geosci. Remote Sens. **60** (2022) 1. https://doi.org/10.1109/TGRS.2022.3160007
18 H. Chen and Z. Shi: Remote Sens. **12** (2020) 1662. https://doi.org/10.3390/rs12101662
19 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin: Advances in Neural Information Processing Systems 30 (NIPS, 2017). https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
20 I. Loshchilov and F. Hutter: arXiv preprint arXiv:1711.05101 (2017). https://doi.org/10.48550/arXiv.1711.05101
21 L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille: IEEE Trans. Pattern Anal. Mach. Intell. **40** (2017) 834. https://doi.org/10.1109/tpami.2017.2699184
22 J. Dong, W. Zhao, and S. Wang: IEEE Geosci. Remote Sens. Lett. **19** (2021) 1. https://doi.org/10.1109/lgrs.2021.3121094
23 O. Ronneberger, P. Fischer, and T. Brox: Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI, 2015) 234. https://doi.org/10.1007/978-3-319-24574-4_28
24 Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (DLMIA ML-CDS, 2018) 3. https://doi.org/10.1007/978-3-030-00889-5_1