UCDNet: A Deep Learning Model for Urban Change Detection from Bi-temporal Multispectral Sentinel-2 Satellite Images

Basavaraju K S, Sravya N, Shyam Lal, *Senior Member, IEEE,* J Nalini, Chintala Sudhakar Reddy, Fabio Dell'Acqua *Senior Member, IEEE*

***Abstract*—Change detection from satellite images has become an inevitable process in Earth observation. Methods for detecting changes in multi-temporal satellite images are very useful tools when characterization and monitoring of urban growth patterns is concerned. Increasing, world-wide availability of multispectral images with a high revisit frequency opened up more possibilities in the study of urban change detection. Even though there exists several deep learning methods for change detection, most of these available methods fail to predict the edges and to preserve the shape of the changed area from multispectral images. This paper introduces a deep learning model called Urban Change Detection Network (UCDNet) for urban change detection from bi-temporal multispectral Sentinel-2 satellite images. The model is based on an encoder-decoder architecture which utilizes modified residual connections and the new spatial pyramid pooling (NSPP) block, giving better predictions while preserving the shape of changed areas. The modified residual connections help to locate the changes correctly and the NSPP block can extract multiscale features and will give awareness about global context. The UCDNet utilizes a proposed loss function which is combination of weighted class categorical cross-entropy (wcce) and modified Kappa loss. The Onera Satellite Change Detection (OSCD) dataset is used to train, evaluate and compare the proposed model with the benchmark models. The UCDNet is giving better results from the reference models used here for the comparison. It is giving accuracy - 99.3%, F1 score - 89.21%, Kappa coefficient - 88.85% and Jaccard index of 80.53% on OSCD dataset.**

***Index Terms*—Deep Learning, Change Detection, Multispectral satellite images, Spatial Pyramid Pooling**

I. Introduction

The task of identifying differences in the state of an object or phenomenon by observing it at different times is known as change detection (CD) [1]. CD is used in many applications like urban planning to give solutions for a more effective use of the infrastructure [2]; Land Use and Land cover (LULC) maps of an area to understand the current landscape [3]; underwater sensing [4], disaster assessment [5] and deforestation monitoring [6].

The planet's structure keeps on changing due to artificial and natural phenomena creating effective monitoring methods. Currently, more than 50% of the global population lives in cities, and, with the ongoing trend, by year 2050 approximately 70% of the global people will reside in cities [7]. With such a rapid urbanization trend, to manage the urban land cover, the administrative authorities need accurate information from time to time. To take precautions and plan good city infrastructure, systematic observation of urban spread using high- and very-high spatial resolution remote sensing images is crucial. Easy- and sometimes even free- access to Earth observation images acquired from a range of satellites [8] have paved the way for space-based urban change detection. Initially, manual methods were adopted to identify differences among remotely sensed datasets but the major drawback of those methods was time-consuming. Today, many supervised and unsupervised CD approaches exist in the literature, like graphical models [9], [10], principal component analysis [11], Markov Random Fields [12], and kernels [13]. Due to advancements in Machine Learning [ML], in last few years many techniques based on neural networks [14]-[17] have emerged that can effectively tackle CD problems. Amid ML approaches, due to the state of the art architecture, deep learning techniques have come into the focus for computer vision applications [18]-[21], including CD in remote sensing [22]-[25], [26]-[31]. Indeed, notwithstanding fruitful results in the remote sensing domain, the lack of adequate data sets specific multimodal information for training hinders further development of deep neural networks. Developing a robust deep learning framework is still a dynamic investigation area, particularly for models that ultimately exploit multitemporal data, despite many supervised and unsupervised models. Considering this, one can recognize that changes can happen until deep networks provide fully operational and reliable tools for remote sensing applications [32]. In spite of these impediments, investigation can still be carried out with the accessible assets, enhancing the existing information on subjects, like semantic segmentation and CD.

Numerous researchers have endeavored to present profound deep learning strategies to the CD work, particularly the techniques described through convolutional neural networks

This research work was supported by RESPOND scheme of Indian Space Research Organization (ISRO), Govt. of India under Grant No. ISRO/RES/4/683/19-20, December 30, 2019.

Basavaraju K S, Sravya N, and Shyam Lal are with the Department of Electronics and Communication Engineering, National Institute of Technology Karnataka, Mangalore 575025, India (e-mail: ksbasavaraju@gmail.com; sravyanedungatt@gmail.com; shyam.mtec@gmail.com).

J Nalini is with the Aerial Services & Digital Mapping of National Remote Sensing Centre, Indian Space Research Organisation, Balanagar, Hyderabad, India (email: nalini_j@nrsc.gov.in).

Chintala Sudhakar Reddy is with the Forest Biodiversity and Ecology Division, National Remote Sensing Centre, Indian Space Research Organisation, Balanagar, Hyderabad, India (email: drsudhakarreddy@gmail.com).

Fabio Dell'Acqua is with the Department of Electrical, Computer and Biomedical Engineering, University of Pavia, I-27010 Pavia, Italy (e-mail: fabio.dellacqua@unipv.it).

Manuscript received December, 2021

(CNNs) because of their excellent feature learning from satellite images [33]. In CNN's, utilization of the pooling layer weakens numerous features, and the subsequent feature maps and predictions cannot accomplish the specified pixel-based image analysis. Fully Convolutional Network (FCN) [34] outperforms CNNs because of fully convolutional layers, which gives better efficiency. A typical system of FCN comprises two components: encoders and decoders. Encoders are derived from [35] and can be thought of as CNNs that extract feature maps, whereas decoders change these feature maps, with size same as the input images. FCNs use transpose convolutional layers to upsample the deep feature maps into labeling results, which improves classification performance resulting in good predictions. Still, there are disadvantages: i) transpose convolution layers are computationally expensive because of demanding memory requirements [36] and ii) spatial boundaries of class-homogeneous areas are poorly identified [37]. To solve the boundary blur problem researchers have come up with different models. In [38], fully-convolutional Siamese difference (FC-Siam-diff), fully-convolutional Siamese concatenation (FC-Siam-conc), and fully-convolutional Early fusion (FC-EF) are introduced. FC-EF is derived from U-net architecture [39], which use skip connections to combine high- and low-level features for joint learning. FC-Siam-conc use Siamese neural network into FC-EF to learn similar features between the original bi-temporal images. FC-Siam-conc calculates the difference of the same layer features of the Siamese network to increase the difference features. To guide the network to learn the difference characteristics between the bitemporal images, FC-Siam-diff uses skip connections. These methods are not exceptionally viable when detecting detailed features. To concatenate and combine the feature maps, in Deeplabv3 [40], [41] Augmented Atrous Spatial Pyramid Pool (AASPP) is used. In [42], reused pooling indices, and in [43], semantic segmentation is combined with semantically informed edge detection to make boundaries explicit. In all these methods, it is challenging to meet the requirements of multiscale change objects because of single feature extraction in each skip connection. Therefore in UNet++ [44], dense skip connection is used to get multi-scale feature extraction and identify pseudo-changes caused by scale variance. BiDateNet [45] integrated Long-short-term memories into the skip connection to get more temporally differentiable features and take full advantage of temporal dependence between bi-temporal images. To get more discriminative spatial and temporal features, in STANet [46] a spatial and temporal attention mechanism is used. In all these methods, the change map shows poor morphology due to a lack of semantic information. FDCNN [47] uses CNN to get features from the remote sensing images and transfer learning to build a two-channel network with shared weights to achieve a multiscale and multi-depth feature difference map for CD. FDCNN, however, offers poor prediction quality. In ADS-Net [48], spatial and channel features are combined by an adaptive attention mechanism to grab the relationship of different scale changes, to highlight the characteristics of change. In urban change detection, however, the performance of ADS-Net is not satisfactory.

To overcome the problems faced in the methods mentioned above, this paper proposes a robust deep learning model for urban change detection from bi-temporal multispectral satellite images, called UCDNet. This paper makes the following contributions to the literature as briefed below:

- A new CD network, UCDNet, is proposed; in the encoder of the UCDNet, features are extracted at different levels using modified residual connections.
- A new spatial pyramid pooling (NSPP) block is introduced which takes the features extracted by the encoder as an input. It extracts the features at various ranges to give awareness about the global context, preserve the shape of the changed area, and predict the boundary pixels more accurately.
- An improved loss function is proposed, which is a combination of weighted class categorical cross-entropy and modified kappa loss, is used to train the proposed UCDNet model. The proposed UCDNet model gives state-of-the-art performance on the OSCD dataset with an overall accuracy of 99.3%, F1 score of 89.21%, Kappa coefficient of 88.85% and a Jaccard index of 80.53%.

The rest of this paper is formulated as follows: Section II focuses on the detailed outline of the proposed UCDNet architecture. Section III focuses on the Training and Implementation details of the proposed network. The Ablation study is given in Section IV. In Section V, the Experimental results of our model are discussed. The conclusion of our work is presented in Section VI.

II. The Proposed Architecture

This section describes the designed architecture of UCDNet for urban change detection from bi-temporal multispectral images. This model is inspired by the Fully Convolutional Siamese concatenation model (FC-Siam-conc) [38], and follows its encoder-decoder block scheme as shown in Fig.1.

A. Encoder of the UCDNet

The encoder unit is divided into two streams with identical structures sharing weights as in [38]. Each input image is given to one of these identical structures. Like the FC-Siam-conc network, the encoder part consists of convolutional and pooling layers. In each stream, 3 pooling layers are used. Since, the main objectives of the CD techniques are for identifying the changes from two images, the difference of the features learnt is taken as the input of the modified residual connection in each stage of the encoder part. In modified redsidual connections, the difference of the features from the two streams just after the first convolution is calculated and convoluted point-wise. This resulting feature is then concatenated channel-wise with the output feature of the corresponding stage before passing it to the max-pooling layer. In Fig.1, $13 \rightarrow 16$, means 13 is the number of channels (bands) given as input, and 16 is the number of feature maps at that layer after the convolution operation. Similarly, at all other layers, the first number indicates the number of input feature maps, and the last number indicates the number of output feature maps at that layer after the convolution operation. For example, $320 \rightarrow 64 \rightarrow 64 \rightarrow 32$ means 320 feature maps

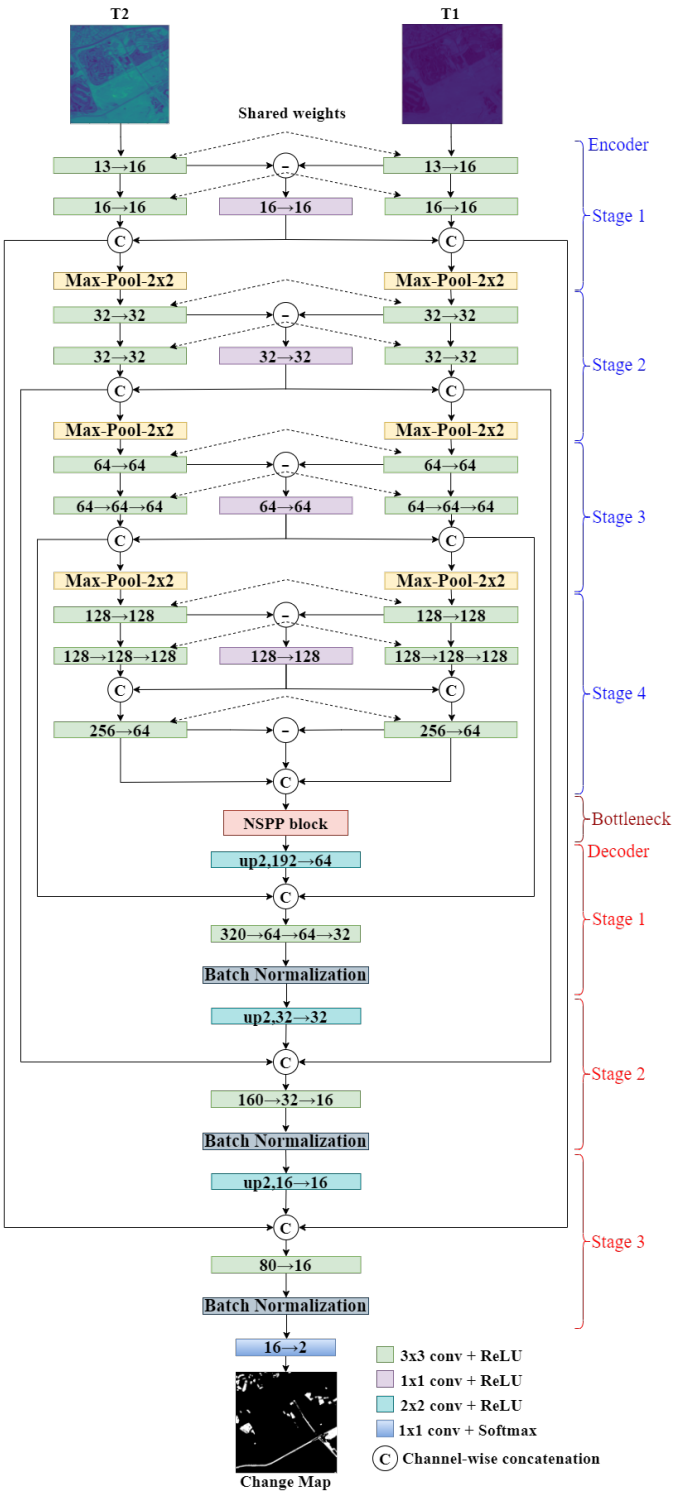


Fig. 1. Architecture of proposed UCDNet

from the previous layer are given as input, and 32 is the output feature maps at that layer after the convolution operation. If the i^{th} stage consists of m convolutions, then the processes included in this stage are described by the Equations (1) to (3).

$$\left(c_{i_j}\right)_{enc1,enc2} = \varphi\left\{\left(x_{i_j}\right)_{enc1,enc2} * w_j^{3\times3}\right\}, \quad j = 1,..,m \tag{1}$$

$$c_{r\_i} = \varphi\left\{\left((c_{i_1})_{enc1} - (c_{i_1})_{enc2}\right) * w^{1\times1}\right\} \tag{2}$$

$$(c_{i_{out}})_{enc1,enc2} = \left((c_{i_m})_{enc1,enc2} \copyright\, c_{r\_i}\right)/2 \tag{3}$$

In Equation (1), $\left(c_{i_j}\right)_{enc1,enc2}$ is the feature obtained after the j^{th} convolution from the i^{th} encoder stage, $\left(x_{i_j}\right)_{enc1,enc2}$ is the input to the j^{th} convolution, $w_j^{3\times3}$ is the 3×3 kernel for j^{th} convolution, $*$ represents the convolutional operation and φ is the ReLU activation function. In Equation (2), c_{r_i} is the feature obtained from the modified residual connection and $w_r^{1\times1}$ is the kernel of the 1×1 convolution. In Equation (3), $(c_{i_{out}})_{enc1,enc2}$ is the output of the i^{th} encoder stage, © represents channel-wise concatenation and $/2$ represents max-pooling operation.

In the 4th stage, after this concatenation, the resulting feature is convoluted again instead of doing max-pooling to improve the feature extraction at the encoder stage. Then the difference of these features, from both the streams is concatenated with the features themselves, giving a single encoder output.

B. New Spatial Pyramid Pooling (NSPP) block

The input to the NSPP block is composed of the learned features from the encoder part. It helps to extract the features at various ranges and will give awareness about global context. Fig.2 shows the diagram of the NSPP block. The input feature to this block goes through four parallel paths. Instead of having average pooling, batch normalization, ReLU activation and resizing operation in each parallel path of SPP block [49], the parallel paths of NSPP consist of a pooling block and a global pooling block. The function of the pooling block is the same as the SPP block, but it extracts features in a more effective way. The pooling blocks create features at four different scales through these four parallel paths.

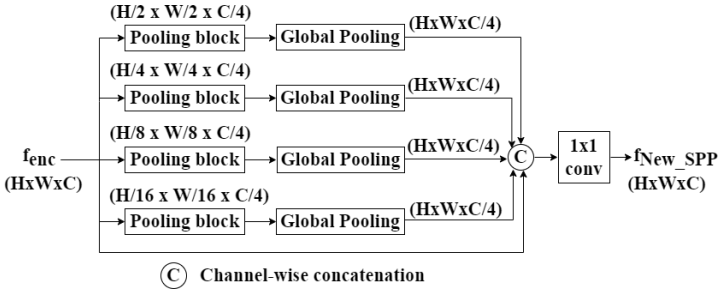


Fig. 2. New Spatial Pyramid Pooling (NSPP) block diagram

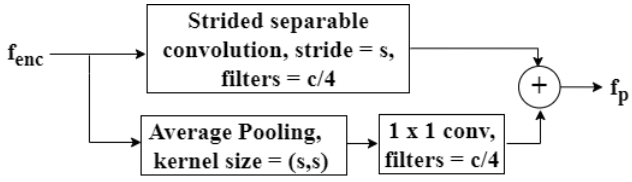


Fig. 3. Block diagram of Pooling block

Fig.3 shows the diagram of the pooling block. The input to this block undergoes average pooling and a strided convolution

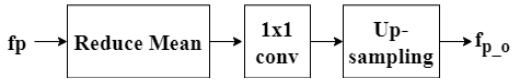


Fig. 4. Block diagram of Global Pooling block

in parallel to combine the effects of both. It is possible to get multiscale features using pooling and strided convolution. In both cases, some information will be lost; by taking the combined result, however, the loss can be minimized. Moreover, since the learning process also uses strided convolution, better results can be achieved from the pooling block than from the old SPP. Point-wise convolution is carried out after average pooling in order to make the number of channels of the output from the pooling block one by fourth of the input signal of the NSPP block. These two extracted features, from average pooling and strided convolution, are added to get the pooling block's output. The extracted feature from the pooling block, $f_p \in R^{H/s \times W/s \times C/4}$ can be represented as in Equation (4).

$$f_p = \varphi\left(f_{enc} * w_p^{3\times3}\right)_{stride=(s,s)} + \{avgpool_{s\times s}\left(f_{enc}\right)\} * w_p^{1\times1} \tag{4}$$

In Equation (4), $f_{enc} \in R^{H\times W\times C}$ is the input to the NSPP block, i.e. the encoder output and $w_p^{3\times3}$ is the 3×3 kernel for strided convolution. $avg\_pool_{s\times s}\left(f_{enc}\right)$ is the output obtained after taking average pooling of input signal with a $s\times s$ kernel and $w_p^{1\times1}$ is the kernel of point-wise convolution.

In addition to extracting multiscale features in SPP blocks, the NSPP includes global pooling blocks. Global pooling is used to improve the awareness of global information and to reduce degradation problems. Global context feature information will improve change detection performance by classifying pixels with greater accuracy. The feature from the pooling block is given to the global pooling block. Here, global pooling is carried out based on the method presented in [50]. The feature from the pooling block is reduced by taking the mean along the x and y dimensions. It will result in only one response for each feature map. This intermediate feature is then convoluted point-wise and it is up-sampled using transposed convolution to scale into the size of the NSPP's input. The operations in this block are given in the Equations (5) and (6).

$$f_{g\_o} = \{reduce\_mean\left(f_p\right)\} * w_g^{1\times1} \tag{5}$$

$$f_{p\_o} = \varphi\left\{\left(f_{g\_o} * w_t^{3\times3}\right)_{(stride=(H,W))}\right\} \tag{6}$$

In Equation (5), $f_{g\_o} \in R^{1\times1\times C/4}$ is the feature obtained after global average pooling, $f_p \in R^{H/s\times W/s\times C/4}$ is the input to the global pooling block and $w_g^{1\times1}$ is the point-wise convolutional kernel. In Equation (6), $f_{p\_o} \in R^{H\times W\times C/4}$ is output feature from the global pooling block and $w_t^{3\times3}$ is the 3×3 kernel of transposed convolution.

The outputs from these four parallel paths of NSPP block are concatenated channel-wise along with the input feature and then convoluted point-wise as a final layer of this module to make the number of channels the same as the input feature of this block. Equation (9) shows the output feature from the NSPP block.

$$f_{new\_SPP} = \{f_{p\_o1} © f_{p\_o2} © f_{p\_o3} © f_{p\_o4} © f_{enc}\} * w^{1\times1} \tag{7}$$

In equation (9), f_{p_o1}, f_{p_o2}, f_{p_o3} and f_{p_o4} are the features from the four parallel paths and $w^{1\times1}$ is the kernel of point-wise convolution.

C. Decoder of the UCDNet

The decoder part is used to project the features learned onto the pixel space. The feature from the NSPP block is given as the input to the decoder part. As in the FC-Siam-conc model [38], the channel-wise concatenation of the features from the corresponding stages in the encoder part is used here. Three up-sampling layers are included in the decoder part. The input signal is first up-sampled by a factor of 2 and then passed through a carried out 2×2 convolution step. This signal is then concatenated with the features from corresponding stages of both encoder streams. The subsequent feature map is passed through a series of 3×3 convolutions and accompanied by a batch normalization layer. These operations are repeated after each up-sampling layer but the number of 3×3 convolutions used after first, second and third up-sampling layers are 3, 2 and 1 respectively. These operations can be represented as in the Equations (8) to (10).

$$up_i = \varphi\left\{\left(x_{up_i} * 2\right) * w_u^{2\times2}\right\} \tag{8}$$

$$C_c = \left(c_{i_m}\right)_{enc1} © up_i © \left(c_{i_m}\right)_{enc2} \tag{9}$$

$$f_m = \varphi\left\{C_{in} * w^{3\times3}\right\} \tag{10}$$

In Equation (8), up_i represents output of i^{th} up-sampling layer, $*2$ represents up-sampling operation by a factor 2 and $w_u^{2\times2}$ is the 2×2 convolutional kernel. In Equation (9), $\left(c_{i_m}\right)_{enc1}$ and $\left(c_{i_m}\right)_{enc2}$ are the features from the corresponding encoder streams and C_c is the concatenated output. In Equation (10), f_m and C_{in} are the output and input of m^{th} 3×3 convolutional layer respectively. For $m=1$, C_{in} will be C_c.

The resulting feature after these operations is convoluted point-wise and then Softmax activation function as a final layer of the model. If f_d is the feature map obtained from the decoder part, then the change map, X_{out}, given by the model can be represented as in equation (11).

$$X_{out} = \sigma\left\{f_d * w^{1\times1}\right\} \tag{11}$$

The σ used in Equation (11) is the softmax activation function and it can be mathematically represented as:

$$\sigma\left(\vec{z}\right)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{12}$$

In Equation (12), $\vec{z}$ is the input vector and k is the number of classes in the multi-class classifier.

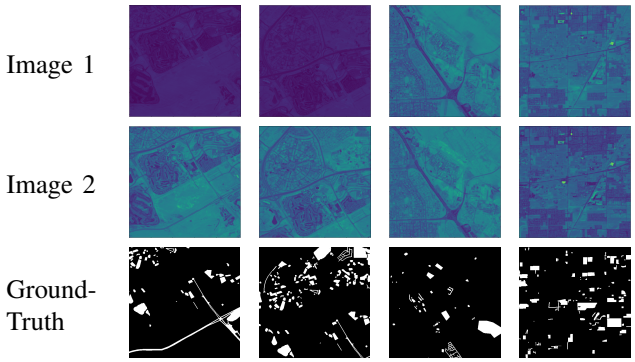
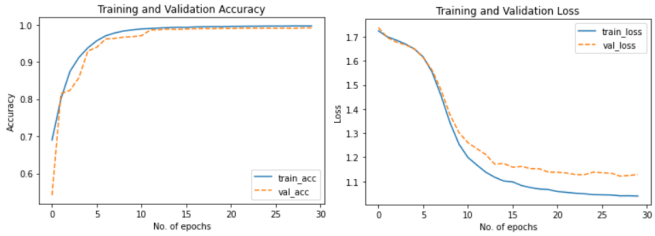


Fig. 5. OSCD dataset



(a) Training and Validation accuracy graph

(b) Training and Validation loss graph

Fig. 6. Learning graph of training and validation datasets

III. Training & Implementation

A. Dataset

The performance of the proposed and the benchmark models on urban change detection is evaluated on the OSCD dataset [51]; a few images of the OSCD dataset are as displayed in Fig.5. The dataset was created from Sentinel-2 satellite images, and focuses on urban growth and changes, ignoring natural changes. From these satellite images with various resolutions between 10m and 60m, images of 24 regions worldwide where urbanization was evident were selected. These images are sized approximately 600×600 pixels at 10m resolution, and were cropped based on geographical coordinates resulting in image pairs with 13 spectral bands. From these images, patches of spatial dimension 512×512 were created in a way that adjacent patches had an overlapping of 64 pixels. From the resulting images, 200, 57 and 30 images were selected randomly in order to create training, validation and test sets respectively.

B. Proposed Loss Function

The loss function used here is the combined function of weighted class categorical cross-entropy (*wcce* loss) and *modified kappa* loss as in Equation (13).

$$loss = L_{wcce} + k_{weight} \times L_{modified\_kappa} \tag{13}$$

The first part of the Equation (13), i.e. L_{wcce}, is the *WCCE* loss. For an image with $d1 \times d2$ pixels and K classes, it can be defined as in Equation (14).

$$L_{wcce} = -\frac{1}{K}\left(\sum_{i,j,k=1}^{d1,d2,K} w_k Y_{ijk} \log p_{ijk}\right) \tag{14}$$

where $p, Y \in \{0,1\}^{d1 \times d2 \times K}$

In Equation (14), p_{ijk} is the predicted probability that the pixel (i,j) pertains to class k. The value of Y_{ijk} will be 1 if pixel (i,j) belongs to class k, 0 otherwise.

In the second part, the $L_{modified\_kappa}$ is the modified kappa loss. In order to reduce the loss, more attention is needed on FN (False Negative) and FP (False Positive) cases. For this, reduced the effects of TP (True Positive) and TN (True Negative) by adding weights, α and β respectively, where the $0 \leq \alpha, \beta \leq 1$. The $L_{modified\_kappa}$ can be defined mathematically as in Equation (15). The Log-Cosh function is used here for smoothing the curve [55]. The $L^{'}_{kappa}$ can be calculated using the Equation (16) where the $ka^{'}$ is the kappa coefficient (Ka) [52] calculated using αTP, βTN, FP, and FN. The k_{weight} is used here to reduce the effects of Kappa loss in the loss function as training progresses and it is defined in the Equation (17).

$$L_{modified\_kappa} = Log\_Cosh\left(L^{'}_{kappa}\right) \tag{15}$$

$$L^{'}_{kappa} = 1 - ka^{'} = 1 - ka\left(\alpha TP, \beta TN, FP, FN\right) \tag{16}$$

$$k_{weight} = 1 + \frac{L^{'}_{kappa}}{ka^{'}} \tag{17}$$

C. Training Setup

The proposed model is implemented on the NVIDIA Quadro RTX4000 GPU with 8GB onboard memory. Tensorflow 2.7 with Keras API framework is used to implement the model. The Adam optimizer with a learning rate of 0.0001 is used. The proposed and the benchmark models used in this study are trained for 30 epochs with a batch size of 1. All these models are trained for 5 times on OSCD dataset with 13 bands and average of these values are reported here.

Learning curves on training and validation datasets can give information about how well a model is learned. The Fig.6(a) is the graph proposed model training and validation accuracy . After ten epochs, the training and validation accuracy became almost equal. The graph of training and validation loss of the proposed UCDNet is given in Fig.6(b). Validation and training losses decrease as training progresses, but, after ten epochs, validation loss reduces at a slower rate than training loss. **The reproducible python code of UCDNet model will be available at : https://github.com/shyamfec/UCDNET**

D. Evaluation Metrics

The performance of the proposed and the benchmark models used for comparison is evaluated using accuracy, *Jaccard* index (JI), precision (Pr), $F1-$score (F1), *kappa* coefficient (Ka), and recall (Re) [52] [53]. The $F1$ is the harmonic mean of precision and recall. An excellent $F1$ indicates that there are few false positives and false negatives [53]. The *Ka* is

TABLE I
ABLATION STUDY OF PROPOSED LOSS FUNCTION WITH FC-SIAM-CONC

	Ka loss	Modified Ka loss	WCCE loss	Proposed loss
F1	0.6335	0.6476	0.8276	0.8366
Ka	0.6216	0.6350	0.8214	0.8311

a measure of consistency between predicted and ground-truth classes [52]. The *JI* is the measure of similarity between finite sample sets [52].

IV. ABLATION STUDY

A. Ablation study of proposed loss function

The proposed loss function is a combination of WCCE) and modified Kappa loss. The effectiveness of the loss function with the base model FC-siam-conc on the OSCD (13 bands) is discussed in the section. The results are tabulated in table I. The $F1$ and *Ka* values given by the FC-siam-conc model with are 0.6335 and 0.6216 respectively. These values are increased by 1.41% and 1.34% respectively by using modified kappa loss. With the *wcce*, FC-siam-conc is giving a $F1$ of 0.8276 and *Ka* of 0.8214. By using the proposed loss function, these values are improved by 0.9% and 0.97% respectively from FC-siam-conc with *wcce* loss function.

B. Ablation study of proposed model

This proposed UCDNet model is developed by the experimentation of different techniques on the FC-Siam model. The UCDNet consists of modified residual connections in the encoder part, a NSPP block that uses a combined loss function of *wcce* loss and *modified kappa* loss. In order to analyze the effectiveness of the model, each intermediate model in the development of UCDNet is explained in this section. Table I shows the results of these models on the OSCD dataset. All these models are trained using the proposed loss function. The FC-Siamese-conc model is the base model of the UCDNet and it gives an accuracy of 98.93%. The values of $F1$, *Ka*, and *JI* are 0.8366, 0.8311, 0.7192 respectively.

1) Intermediate model 1: At every stage in the encoder part of the FC-Siam-conc model, a modified residual connection, i.e. similar to the encoder part of the proposed model, was added and the first stage in the decoder part was removed. This intermediate model gives an accuracy of 99.13%. The $F1$ given by the model is 0.8654, i.e. an improvement of 2.88% from the FC-Siam-conc model. The *Ka* and the *JI* given by this model are 0.8609 and 0.7628 respectively. These values showed an improvement of 2.98% and 4.36% respectively.

2) Intermediate model 2: In the FC-Siam-conc architecture, after the fourth stage of the encoder streams, the difference of the features from the two streams is concatenated with the features themselves giving a single encoder feature which is then fed into the NSPP block. This intermediate model also uses only three stages in the decoder part. This intermediate model gives an accuracy of 99.17%. The $F1$, *Ka* and *JI* values are 0.8731, 0.8688, and 0.7752 respectively. These values are giving an improvement of 3.65%, 3.77% and 5.6% from the FC-Siam-conc model.

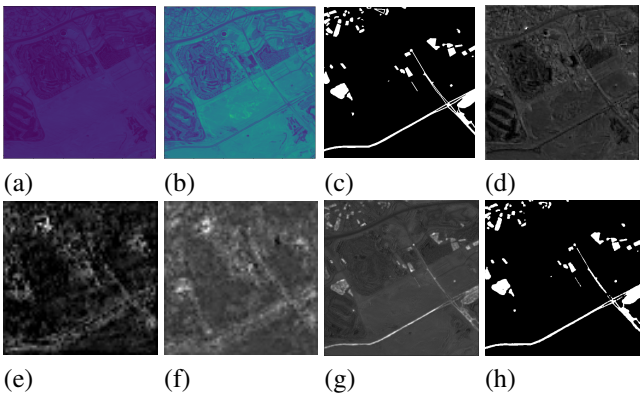


Fig. 7. Visualization of feature maps (a) Image 1 (b) Image 2 (c) Ground-truth (d) After 2^{nd} stage of encoder (e) After encoder (f) After NSPP (g) Before Final layer of the model (h) Predicted image

To better understand the influence of each stage in the proposed UCDNet, a visualization of feature maps is included in Fig.7. Fig.7(d) shows the feature map obtained after the second stage of the encoder, and as the reader may note it is difficult to understand the changed area from this image. From the images of the feature map after the encoder part and NSPP block, it is clear that the changed area is brighter after the incorporation of the NSPP block. From the feature map obtained before the final layer of the model, i.e. point-wise convolution and softmax function, the changed areas became more predictable. Fig.7(h) shows the final predicted output of the proposed UCDNet.

V. EXPERIMENTAL RESULTS

A. Comparison with benchmark models

The performance of the proposed model is compared with the benchmark models U-Net++ (2019) [44], FCNPP (2019) [54], FDCNN (2020) [47], ADSNet (2021) [48] and AGCDetNet (2021) [52] using the evaluation metrics accuracy, Pr, Re, $F1$, *ka* and *JI*. All these models were evaluated on the OSCD dataset and results are shown in Table II. The AGCDetNet model achieves best results whereas the FDCNN shows poor performance from the benchmark models. The accuracy given by the proposed UCDNet is 99.30% whereas the AGCDetNet model gives 98.58%. The $F1$, *Ka* and *JI* values of the proposed model are 0.8921, 0.8885, and 0.8053 respectively. These values given by the AGCDetNet model and FDCNN are (0.7816, 0.7743, 0.6424) and (0.3948, 0.3758, 0.2461) respectively. From all the models, the best value of *Pr* is given by the proposed UCDNet and the least value is given by the FDCNN, which are 0.9253 and 0.4372 respectively. From the benchmark models, the best *Pr* is given by ADSNet which is 0.9098. The *Re* is high for FCNPP which is 0.9210 and this value for proposed UCDNet is 0.8616. The least value of *Re* is 0.3640 which is given by the model FDCNN.

Fig.10 shows the prediction results of the proposed model and the reference models on the OSCD dataset. To evaluate the performance of these models, some portions of these images are highlighted by the yellow colored boxes. The proposed

TABLE II
ABLATION STUDY OF PROPOSED MODEL ON OSCD DATASET(13 BANDS)

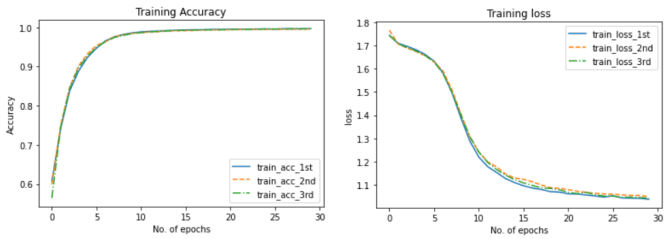
Model	Pr	Re	F1	Ka	JI	Acc
FC-Siam-conc (2018)	0.8583	0.8183	0.8366	0.8311	0.7192	0.9893
Intermediate model1	0.8991	0.8345	0.8654	0.8609	0.7628	0.9913
Intermediate model2	0.8969	0.8518	0.8731	0.8688	0.7752	0.9917
Proposed UCDNet (Final model)	**0.9253**	**0.8616**	**0.8921**	**0.8885**	**0.8053**	**0.9930**

TABLE III
AVERAGE QUALITY METRICS OF MODELS ON OSCD DATASET (13 BANDS)

Model	Pr	Re	F1	Ka	JI	Acc	Parameters	FLOPs (Millions)
U-Net++ (2019) [44]	0.6501	0.8381	0.7410	0.7308	0.5927	0.9797	4,880,418	169588
FCNPP (2019) [54]	0.6631	**0.9210**	0.7704	0.7611	0.6273	0.9815	3,111,666	**30308**
FDCNN (2020) [47]	0.4372	0.3640	0.3948	0.3758	0.2461	0.9628	**14,630**	44410
ADSNet (2021) [48]	0.9098	0.5182	0.6591	0.6506	0.5757	0.9821	2,577,752	31827
AGCDetNet (2021) [52]	0.8079	0.7584	0.7816	0.7743	0.6424	0.9858	60,205,927	652646
Proposed UCDNet	**0.9253**	0.8616	**0.8921**	**0.8885**	**0.8053**	**0.9930**	1,349,842	50671

TABLE IV
3-FOLD CROSS VALIDATION AVERAGE QUALITY METRICS OF MODELS ON OSCD DATASET(13 BANDS)

Model	Pr	Re	F1	Ka	JI	Acc	Parameters	FLOPs (Millions)
U-Net++ (2019) [44]	0.6952	0.7887	0.7299	0.7209	0.5750	0.9821	4,880,418	169588
FCNPP (2019) [54]	0.7788	**0.8997**	0.8341	0.8304	0.7155	0.9892	3,111,666	**30308**
FDCNN (2020) [47]	0.2083	0.2051	0.2040	0.1801	0.1144	0.9530	**14,630**	44410
ADSNet (2021) [48]	0.9034	0.4518	0.6005	0.5923	0.4315	0.9819	2,577,752	31827
AGCDetNet (2021) [52]	0.7935	0.6371	0.7025	0.6943	0.6046	0.9836	60,205,927	652646
Proposed UCDNet	**0.9079**	0.8723	**0.8895**	**0.8861**	**0.8011**	**0.9934**	1,349,842	50671



(a) Training accuracy (b) Training loss

Fig. 8. Learning curves of UCDNet on 3-fold cross validation

model predicts more accurately than the other models. From the highlighted portions, it is visible that the proposed model could retain the shape and boundaries of the changed areas better than the reference models. The poorest predictions were given by the FDCNN model. From the reference models, UNet++ and AGCDetNet are giving good predictions.

B. 3-fold cross validation

The ability of the proposed model to predict changes was tested by using 3-fold cross validation approach, which permits to analyse how the model is working irrespective of the specific dataset. To do so, from the total dataset, 3 folds (subset) of 95 images were created randomly to carry out 3-fold cross validation. For the first run, the first two folds are taken as training set and the third as test set. For the second run, second and third folds are used as training and first fold as test. Similarly in the third run, first and third as training and second as test set so that each fold has taken as test set. The average results of these three runs obtained for the proposed and the reference models are included in table III. The proposed UCDNet gives cross validation results that are almost the same as the model's result given in table II. The AGCDetNet, ADSNet and FCNPP show considerable variations in the cross-validation results from the results shown in table II. With respect to these models, FDCNN is showing large variations in the results, i.e. 19.08% in $F1$, 19.57% in *Ka* and 13.17% in *JI*.

Fig. 8 shows the learning curves of the proposed model for the 3-fold cross validation. In all the three times, the model shows almost the same learning behaviour; this suggests that the proposed model is generalized independent of the dataset.

C. Statistical analysis

A boxplot is a standardized method of depicting data distributions by using five numbers - minimum, first quartile, median, second quartile and maximum. The boxplots of $F1$ and *JI* of the proposed and existing models are given in Fig. 9. From Fig.9, it is clear that the proposed model performs well. Both metrics' median and class-wise (changed and unchanged area) values are higher for the proposed UCDNet model than for the reference models. The difference of class-wise values is also minimal for the proposed model. The least median value for both metrics are scored by FDCNN, and also class-wise values are lower for this model. AGCDetNet gives better results with respect to reference models. The highest and lowest inter-quartile ranges for both metrics are shown by FDCNN and the proposed UCDNet respectively.

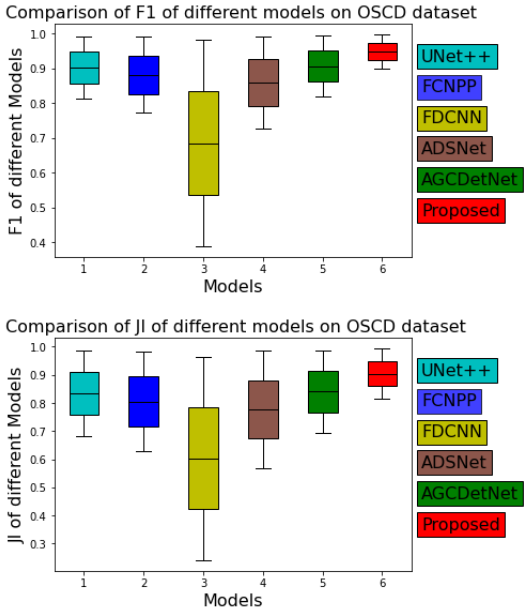


Fig. 9. Comparison of F1 score and JI of different models

TABLE V
TRAINING AND PREDICTING TIME REQUIREMENTS OF ALL MODELS

Model	**Training time (Hours)**	**Prediction time per image (sec)**
U-Net++	0.32	4.14
FCNPP	**0.09**	**3.15**
FDCNN	0.75	3.27
ADSNet	0.29	6.64
AGCDetNet	0.65	6.44
Proposed UCDNet	0.15	4.02

D. Computational complexity analysis

The complexity of a model can be evaluated using the number of trainable parameters and the number of floating point operations(FLOPs) required to run the model. These values for the proposed UCDNet and the benchmark models are reported in Table II. The proposed UCDNet model utilizes the least number of parameters with respect to all the existing models compared here except FDCNN which used pre-trained VGG16 weights optimized on aerial image data (AID). The AGCDetNet model requires the highest number of parameters and FLOPs. The least FLOPs are utilized by the FCNPP model i.e. 30308 millions while proposed UCDNet utilizes 50671 millions.

The time required for training and prediction is also a measure of the complexity of a model. These values for the proposed UCDNet and the other benchmark models for 30 epochs are given in Table IV. The time required for training the OSCD training set and prediction per image is at its minimum for FCNPP, with 0.09 hrs and 3.15 sec respectively. The proposed UCDNet requires 0.15 hrs for training and 4.02 sec for predicting one image; these times are longer than FCNPP but still comparable with other state of art methods. Moreover, as discussed previously, they come with significantly better values on result quality metrics.

VI. CONCLUSION

This paper highlighted proposed UCDNet architecture and loss function for change detection from bi-temporal multi-spectral Sentinel-2 satellite images. The UCDNet gives better results compared to a selection of significant reference models. UCDNet is inspired from the FC-Siam-conc model. The proposed UCDNet is showing improvement of Pr - 6.7%, Re - 4.33%, $F1$ - 5.55%, Ka - 5.74%, and JI - 8.61% from the FC-Siam-conc model. The most recent model AGCDetNet gives better results with respect to the benchmark models. The UCDNet achieves an improvement of 11.74%, 10.32%, 11.05%, 11.42%, and 16.29% in Pr, Re, $F1$, Ka and JI respectively as compared with AGCDetNet. The proposed UCDNet could preserve the shape of the changed area and predict the boundary pixels more accurately than the reference models considered in this paper. The limitation of this work is, in the encoder part, calculating the difference between feature maps reduces the values assigned to pixels, and some of them may be ignored, lowering the recall metric. With the availability of high resolution remote sensing images, semantic change detection with deep learning methods has vast possibilities. In the future, this work can be extended for semantic change detection, and a mechanism must be investigated to solve the lowering of the recall metric and compensate for the decrease in recall.

REFERENCES

[1] A. Singh, "Review article digital change detection techniques using remotely-sensed data," Int. J. Remote Sens., vol. 10, no. 6, pp. 989–1003, Jun. 1989.
[2] H. Luo, C. Liu, C. Wu, and X. Guo, "Urban change detection based on Dempster–Shafer theory for multitemporal very high-resolution imagery," Remote Sens., vol. 10, no. 7, pp. 20–22, 2018.
[3] G. Xian, C. Homer, and J. Fry, "Updating the 2001 national land cover database land cover classification to 2006 by using Landsat imagery change detection methods," Remote Sens. Environ., vol. 113, no. 6, pp. 1133–1147, 2009.
[4] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: A systematic survey," IEEE Trans. Image Process., vol. 14, no. 3, pp. 294–307, Mar. 2005.
[5] Y. Fan, Q. Wen, W. Wang, P. Wang, L. Li, and P. Zhang, "Quantifying disaster physical damage using remote sensing data—A technical work flow and case study of the 2014 Ludian Earthquake in China," Int. J. Disaster Risk Sci., vol. 8, no. 4, pp. 471–488, 2017.
[6] M. Lu, E. J. Pebesma, A. Sanchez, and J. Verbesselt, "Spatio-temporal change detection from multidimensional arrays: Detecting deforestation from MODIS time series," Isprs J. Photogramm. Remote Sens., vol. 117, pp. 227–236, 2016.
[7] Department of Economic and Social Affairs, United Nations. New York, NY, USA. 2018 Revision of World Urbanization Prospects. Accessed: 2018. [Online]. Available: https://population.un.org/wup/
[8] L. Bruzzone and F. Bovolo, "A Novel Framework for the Design of Change-Detection Systems for Very-High-Resolution Remote Sensing Images," Proc. IEEE, vol. 101, pp. 609–630, 2013.
[9] M. Vakalopoulou, K. Karatzalos, N. Komodakis, and N. Paragios, "Simultaneous registration and change detection in multitemporal, very high resolution remote sensing data," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2015, pp. 61–69.
[10] M. Vakalopoulou, C. Platias, M. Papadomanolaki, N. Paragios, and K. Karantzalos, "Simultaneous registration, segmentation and change detection from multisensor, multitemporal satellite image pairs," in Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS), Jul. 2016, pp. 1827–1830.
[11] J. S. Deng, K. Wang, Y. H. Deng, and G. J. Qi, "PCA-based landuse change detection and analysis using multitemporal and multisensor satellite data," Int. J. Remote Sens., vol. 29, no. 16, pp. 4823–4838, 2008.

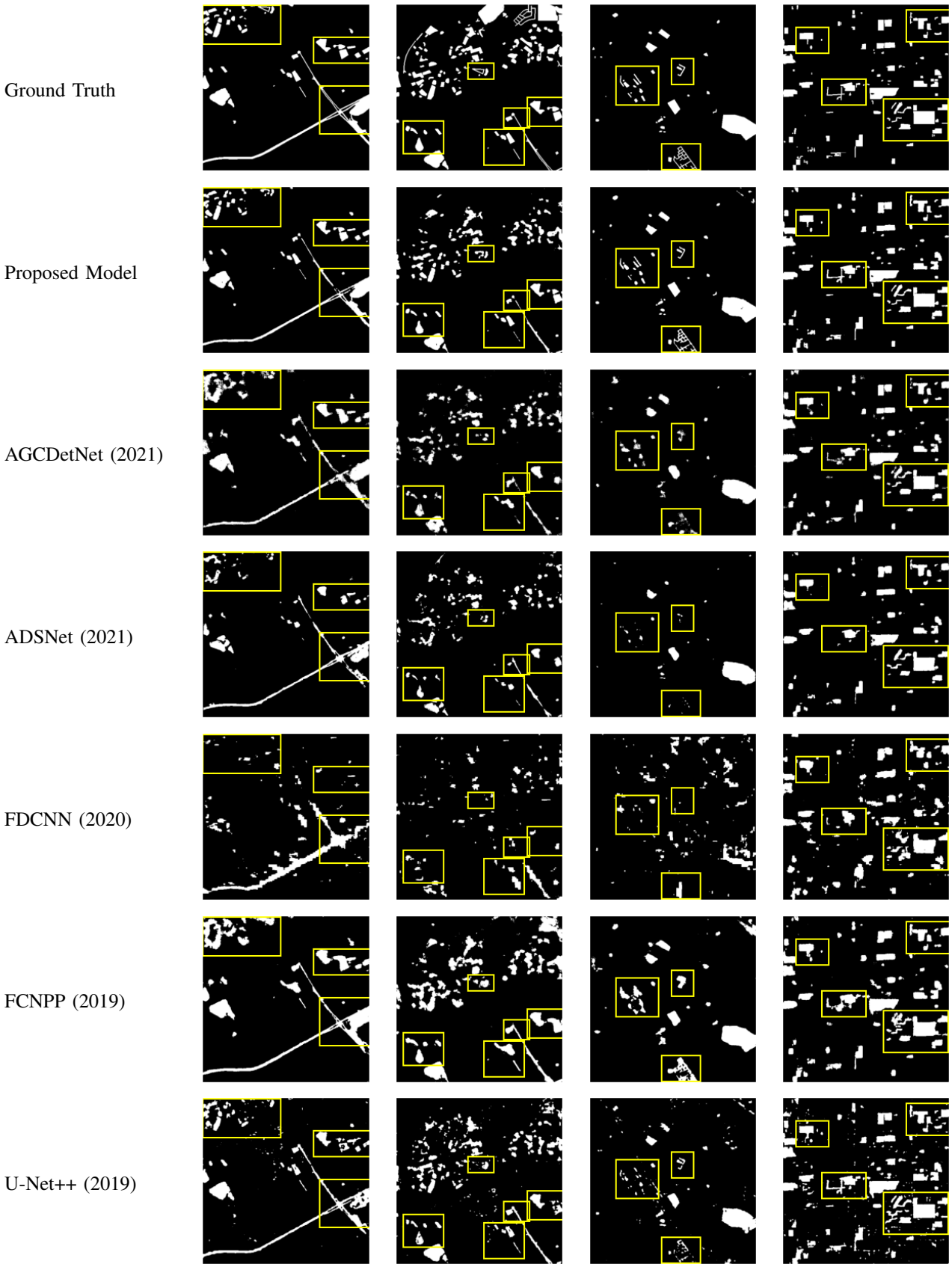


Fig. 10. Prediction results of deep models on OSCD dataset

[12] P. Singh, Z. Kato, and J. Zerubia, "A multilayer Markovian model for change detection in aerial image pairs with large time differences," in Proc. 22nd Int. Conf. Pattern Recognit. Stockholm, Sweden: IEEE, Aug. 2014, pp. 924–929.

[13] M. Volpi, D. Tuia, G. Camps-Valls, and M. Kanevski, "Unsupervised change detection with kernels," IEEE Geosci. Remote Sens. Lett., vol. 9, no. 6, pp. 1026–1030, Nov. 2012.

[14] J. Liu, M. Gong, K. Qin, and P. Zhang, "A deep convolutional coupling network for change detection based on heterogeneous optical and radar images," IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 3, pp. 545–559, Dec. 2016.

[15] M. Gong, J. Zhao, J. Liu, Q. Miao, and L. Jiao, "Change detection in synthetic aperture radar images based on deep neural networks," IEEE Trans. Neural Netw. Learn. Syst., vol. 27, no. 1, pp. 125–138, Jan. 2016.

[16] A. M. El Amin, Q. Liu, and Y. Wang, "Zoom out CNNs features for optical remote sensing change detection," in Proc. 2nd Int. Conf. Image, Vis. Comput. (ICIVC), Jun. 2017, pp. 812–817.

[17] Y. Zhan, K. Fu, M. Yan, X. Sun, H. Wang, and X. Qiu, "Change detection based on deep Siamese convolutional network for optical aerial images," IEEE Geosci. Remote Sens. Lett., vol. 14, no. 10, pp. 1845–1849, Aug. 2017.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017.

[19] M. Vakalopoulou et al., "Atlasnet: Multi-atlas non-linear deep networks for medical image segmentation," in Proc. MICCAI, 2018, pp. 658–666.

[20] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics, (EACLPapers), 2017, pp. 1–10.

[21] J. Y.-H. Ng, J. Choi, J. Neumann, and L. S. Davis, "ActionFlowNet: Learning motion representation for action recognition," in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Mar. 2018, pp. 1616–1624.

[22] S. Saha, Y. T. Solano-Correa, F. Bovolo, and L. Bruzzone, "Unsupervised deep transfer learning-based change detection for HR multispectral images," IEEE Geosci. Remote Sens. Lett., May 2020.

[23] M. Vakalopoulou, K. Karantzalos, N. Komodakis, and N. Paragios, "Building detection in very high resolution multispectral data with deep learning features," in Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS), Jul. 2015, pp. 1873–1876.

[24] M. Papadomanolaki, M. Vakalopoulou, and K. Karantzalos, "A novel object-based deep learning framework for semantic segmentation of very high-resolution remote sensing data: Comparison with convolutional and fully convolutional networks," Remote Sens., vol. 11, no. 6, p. 684, Mar. 2019.

[25] N. Audebert, B. L. Saux, and S. Lefèvre, "Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks," ISPRS J. Photogramm. Remote Sens., vol. 140, pp. 20–32, Jun. 2018.

[26] M. Liu, Q. Shi, A. Marinoni, D. He, X. Liu and L. Zhang, "Super-Resolution-Based Change Detection Network With Stacked Attention Module for Images With Different Resolutions," in IEEE Trans. on Geosci. and Remote Sens., doi: 10.1109/TGRS.2021.3091758.

[27] H. Zhang, M. Lin, G. Yang and L. Zhang, "ESCNet: An End-to-End Superpixel-Enhanced Change Detection Network for Very-High-Resolution Remote Sensing Images," in IEEE Trans. on Neural Net. Learn. Sys., doi: 10.1109/TNNLS.2021.3089332.

[28] C. Wu, H. Chen, B. Du and L. Zhang, "Unsupervised Change Detection in Multitemporal VHR Images Based on Deep Kernel PCA Convolutional Mapping Network," in IEEE Trans. on Cybernetics, doi: 10.1109/TCYB.2021.3086884.

[29] H. Chen, Z. Qi and Z. Shi, "Remote Sensing Image Change Detection With Transformers," in IEEE Trans. on Geosci. and Remote Sens., doi: 10.1109/TGRS.2021.3095166.

[30] H. Chen, W. Li and Z. Shi, "Adversarial Instance Augmentation for Building Change Detection in Remote Sensing Images," in IEEE Trans. on Geosci. and Remote Sens., doi: 10.1109/TGRS.2021.3066802.

[31] S. Fang, K. Li, J. Shao and Z. Li, "SNUNet-CD: A Densely Connected Siamese Network for Change Detection of VHR Images," in IEEE Geosci. and Remote Sens. Lett., doi: 10.1109/LGRS.2021.3056416.

[32] J. Li, X. Huang, and J. Gong, "Deep neural network for remote-sensing image interpretation: Status and perspectives," Nat. Sci. Rev., vol. 6, no. 6, pp. 1082–1086, Nov. 2019

[33] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep learning Earth observation classification using ImageNet pretrained networks," IEEE Geosci. Remote Sens. Lett., vol. 13, no. 1, pp. 105–109, Jan. 2016.

[34] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015 IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR), 2015, pp. 3431-3440

[35] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," J. Mach. Learn. Res., vol. 11, pp. 3371–3408, 2010.

[36] H. Noh, S. Hong and B. Han, "Learning Deconvolution Network for Semantic Segmentation," 2015 IEEE Int. Conf. on Comput. Vis. (ICCV), 2015, pp. 1520-1528, doi: 10.1109/ICCV.2015.178.

[37] L. Mou and X. X. Zhu, "Rifcn: Recurrent network in fully convolutional network for semantic segmentation of high resolution remote sensing images," CoRR, vol. abs/1805.02091, 2018.

[38] R. C. Daudt, B. L. Saux, and A. Boulch, "Fully convolutional siamese networks for change detection," in Proc. 25th IEEE Int. Conf. Image Process. (ICIP), Oct. 2018, pp. 4063–4067.

[39] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in Med. Image Comput. Comput.-Assist. Interv. – MICCAI 2015, Cham, 2015, pp. 234–241.

[40] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. "Rethinking atrous convolution for semantic image segmentation". arXiv:1706.05587, 2017.

[41] Y. Wang, B. Liang, M. Ding, and J. Li, "Dense Semantic Labeling with Atrous Spatial Pyramid Pooling and Decoder for High-Resolution Remote Sensing Imagery," Remote Sens., vol. 11, no. 1, 2019, doi: 10.3390/rs11010020.

[42] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," in IEEE Trans. on Pattern Anal. Mach. Intell., vol. 39, no. 12, pp. 2481-2495, 1 Dec. 2017

[43] D. Marmanis, K. Schindler, J. D. Wegner, S. Galliani, M. Datcu, and U. Stilla, "Classification with an edge: Improving semantic image segmentation with boundary detection," ISPRS J. of Photogrammetry and Remote Sens., vol. 135, pp. 158–172, 2018.

[44] D. Peng, Y. Zhang, and H. Guan, "End-to-end change detection for high resolution satellite images using improved UNet++," Remote Sens., vol. 11, no. 11, p. 1382, Jun. 2019.

[45] M. Papadomanolaki, S. Verma, M. Vakalopoulou, S. Gupta, and K. Karantzalos, "Detecting urban changes with recurrent neural networks from multitemporal Sentinel-2 data," in Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS), Jul. 2019, pp. 214–217.

[46] H. Chen and Z. Shi, "A spatial-temporal attention-based method and a new dataset for remote sensing image change detection," Remote Sens., vol. 12, no. 10, p. 1662, May 2020.

[47] M. Zhang and W. Shi, "A Feature Difference Convolutional Neural Network-Based Change Detection Method," in IEEE Trans. on Geosci. Remote Sens., vol. 58, no. 10, pp. 7232-7246, Oct. 2020

[48] D. Wang, X. Chen, M. Jiang, S. Du, B. Xu, and J. Wang, "ADS-Net:An Attention-Based deeply supervised network for remote sensing image change detection," Int. J. Appl. Earth Obs. Geoinfo., vol. 101, p. 102348, 2021

[49] S. R. Abdani, M. A. Zulkifley and M. Mamat, "U-Net with Spatial Pyramid Pooling Module for Segmenting Oil Palm Plantations," 2020 IEEE 2nd Int. Conf. on Artif. Intell. in Eng. and Tech. (IICAIET), 2020, pp. 1-5

[50] Q. Wu, F. Luo, P. Wu, B. Wang, H. Yang and Y. Wu, "Automatic Road Extraction from High-Resolution Remote Sensing Images Using a Method Based on Densely Connected Spatial Feature-Enhanced Pyramid," in IEEE J. of Selected Topics in Appl. Earth Obs. Remote Sens., vol. 14, pp. 3-17, 2021

[51] R. C. Daudt, B. Le Saux, A. Boulch and Y. Gousseau, "Urban Change Detection for Multispectral Earth Observation Using Convolutional Neural Networks," IGARSS 2018 - 2018 IEEE Int. Geosci. and Remote Sens. Symposium, 2018, pp. 2115-2118

[52] K. Song and J. Jiang, "AGCDetNet:An Attention-Guided Network for Building Change Detection in High-Resolution Remote Sensing Images," in IEEE J. of Selected Topics in Appl. Earth Obs. Remote Sens., vol. 14, pp. 4816-4831, 2021

[53] R. singh and rani Rajneesh, "Semantic Segmentation using Deep Convolutional Neural Network: A Review," Proc. Int. Conf. Innov. Comput. Commun. ICICC, 2020, [Online]. Available: http://dx.doi.org/10.2139/ssrn.3565919.

[54] T. Lei, Y. Zhang, Z. Lv, S. Li, S. Liu and A. K. Nandi, "Landslide Inventory Mapping From Bitemporal Images Using Deep Convolutional Neural Networks," in IEEE Geosci. and Remote Sens. Lett., vol. 16, no. 6, pp. 982-986, June 2019

[55] S. Jadon, "A survey of loss functions for semantic segmentation," 2020 IEEE Conf. on Computat. Intell. in Bioinfo. and Computat. Bio. (CIBCB), 2020, pp. 1-7