Unsupervised Change Detection Analysis in Satellite Image Time Series Using Deep Learning Combined With Graph-Based Approaches

Ekaterina Kalinicheva [ID], Dino Ienco [ID], Jérémie Sublime, and Maria Trocan [ID]

Abstract—Nowadays, huge volume of satellite images, via the different Earth Observation missions, are constantly acquired and they constitute a valuable source of information for the analysis of spatiotemporal phenomena. However, it can be challenging to obtain reference data associated to such images to deal with land use or land cover changes as often the nature of the phenomena under study is not known a *priori*. With the aim to deal with satellite image analysis, considering a real-world scenario, where reference data cannot be available, in this article, we present a novel end-to-end unsupervised approach for change detection and clustering for satellite image time series (SITS). In the proposed framework, we first create bitemporal change masks for every couple of consecutive images using neural network autoencoders (AEs). Then, we associate the extracted changes to different spatial objects. The objects sharing the same geographical location are combined in spatiotemporal evolution graphs that are finally clustered accordingly to the type of change process with gated recurrent unit (GRU) AE-based model. The proposed approach was assessed on two real-world SITS data supplying promising results.

Index Terms—Autoencoder (AE), change detection, gated recurrent unit (GRU), object-oriented image analysis, pattern recognition, satellite image time series (SITS), time-series clustering, unsupervised learning.

I. INTRODUCTION

SATELLITE images time-series analysis algorithms have been used widely these days. Among them, there are numerous ecological applications such as the analysis and preservation of the stability of ecosystems [1], the detection and the analysis of such phenomena as deforestation and droughts [2]–[5], real-time monitoring of natural disasters [6], [7], study of the evolution of urbanization [8], crop changes follow up [9], [10], etc. While for some applications, we can create a training database with the objects presented in satellite image time series (SITS) [11], [12], for others it may be a challenging task as the

nature of the researched spatiotemporal phenomena is often not known, unique or, on the contrary, has a lot of variations. For this reason, unsupervised and semisupervised change detection and time-series analysis have been a hot research topic for the past years. However, not so many successful studies are available for the proposed problematic.

Globally, we can divide the existing methods of SITS analysis in first, semisupervised change detection that combines anomaly detection [3], [4], [13] and change detection in land-cover types in dense low/medium resolution time series [14], [15], and second, unsupervised clustering of the whole SITS [16], [17].

Most of semisupervised SITS change detection algorithms aim to detect a particular type of change (mostly in the forest areas) based on previously available images. For example, Kong *et al.* [13] proposed an algorithm for online prediction of forest fires in MODIS time series achieved by comparing real data with nonanomalous behavior modeled by long short-term memory (LSTM) network. In another study, Dutrieux *et al.* [3] use breaks for additive seasonal and trend framework (BFAST) combined with a seasonal-trend model techniques to detect tropical forest cover loss exploiting multiple data sources (MODIS and Landsat-7). The principle of BFAST is based on iterative decomposition of time series into three components—trend, seasonal, and remainder.

Similar approaches are used for change detection in land-cover types. In [14], the previously mentioned BFAST framework is deployed to detect land-cover changes and seasonal changes in MODIS NDVI series. To overcome this approach, in [15], Cai and Liu present a subannual change detection algorithm based on pixel behavior analysis. In this approach, first, temporal evolution of each pixel is presented as a signal with wavelet decomposition and then, each two points corresponding to the same acquisition day of two consecutive years are compared to determine the time of land-cover change.

To correctly model "normal" behavior and detect seasonal trends, the SITS data should be long enough and have regular and frequent temporal resolution for the study area. At the moment, this type of freely available time series can be provided only by long existing missions such as MODIS (250 m–1 km resolution) and Landsat (30 m resolution) that are mostly used for vegetation analysis and, hence, have a low spatial resolution in favor of a high frequency of image acquisition. As the proposed techniques are adapted for low resolution images, they are not suitable to perform SITS analysis at the local level or to distinguish

Ekaterina Kalinicheva and Maria Trocan are with the ISEP—LISITE laboratory, DaSSIP team, 92130 Issy-Les-Moulineaux, France (e-mail: ekaterina.kalinicheva@isep.fr; maria.trocan@isep.fr).

Dino Ienco is with the INRAE—UMR TETIS, University of Montpellier, 34093 Montpellier Cedex 5, France, and also with the LIRMM, University of Montpellier, 34095 Montpellier Cedex 5, France (e-mail: dino.ienco@irstea.fr).

Jérémie Sublime is with ISEP—LISITE laboratory, DaSSIP team, 92130 Issy-Les-Moulineaux, France, and also with the LIPN—CNRS UMR 7030, University Paris 13, Paris, France (e-mail: jeremie.sublime@isep.fr).

different land-cover subclasses. Moreover, the aforementioned methods are adapted for univariate data (usually, NDVI index) that sometimes can not properly characterize all the data present in SITS.

With such high resolution (HR) freely available time series as SPOT-5 and Sentinel-2, it has become possible to obtain more detailed information about the Earth surface. Unfortunately, freely available HR SITS often contain some unexploitable images (flawed images, images with high ratio of cloud coverage, etc.) or do not have sufficient temporal resolution to detect seasonal trends.

For these reasons, unsupervised HR SITS analysis demands different approaches, including ones that do not depend on temporal resolution (for example, general clustering of the whole SITS). *Guttler et al.* [18] introduced a graph-based approach to detect spatiotemporal dynamics in SITS. In this method, each detected spatiotemporal phenomena is presented in a form of an evolution graph composed of spatiotemporal entities belonging to the same geographical location in multiple timestamps. Following this method, [17] proposes an adapted clustering algorithm to regroup the extracted multiannual evolution graphs in different land-cover types. This algorithm exploits spectral and hierarchical clustering algorithms with dynamic time warping (DTW) distance measure [19] applied to the summarized representation of graph structure. It shows promising results both in intracite and intercite studies. Vaduva *et al.* [16] proposed a bag of words (BoW) approach combined with latent Dirichlet allocation (LDA) [20] to detect different abstract clusters of land-cover. Though this algorithm is developed for synthetic aperture radar SITS, it can be adapted for optical images after some data preprocessing. This method first exploits *K*-means clustering to determine the type of temporal evolution of each pixel with excessively large number of clusters (100–150). Second, every pixel is associated to a BoW [21] representing its neighborhood patch. Each pixel of a patch contains an obtained discrete cluster value. Finally, all BoWs are regrouped in abstract classes with LDA method. Despite the fact that the proposed approaches can be used on HR images with irregular temporal resolution, they are not conceived to detect anomalies or land-cover changes. For example, the first method is used for clustering of the prevailing land-cover types without considering eventual land-cover changes or some anomalies. Contrary to this approach, the second method cluster the whole SITS into a large number of abstract classes without associating them to any specific spatiotemporal phenomena, so the land-cover change is not differentiated from seasonal phenological variation.

However, some applications demand another kind of SITS analysis performed at a finer level such as, for example, detection and classification of different types of changes applied to the urban development analysis or some local ecological disturbances.

In this article, we propose an end-to-end unsupervised approach for the detection and classification of nontrivial changes in HR open source SITS. In our work, we imply that nontrivial changes are the changes free of seasonal trend prevailing in SITS (trivial changes). The proposed framework first exploits neural network autoencoder (AE) to create bitemporal change maps that contain only nontrivial changes (permanent changes and

vegetation that do not follow the overall tendency). Afterward, we deploy segmentation techniques on the extracted change areas in order to detect different spatial entities. Then, we use object-based techniques to model temporal behavior of the detected change phenomena in the form of evolution graphs. Finally, we obtain a summarized representation of the extracted evolution graphs and apply gated recurrent units (GRU) [22] AE combined with hierarchical agglomerative clustering [23] to regroup them in different types of spatiotemporal change phenomena. Our algorithm framework does not demand any training data and gives us promising results on two real-life datasets of cities of Montpellier, France and Rostov-on-Don, Russia.

Our proposed method can be used for projects studying the urban evolution of cities: changes in density of residential areas, major urban structure constructions (such as transportation systems or stadiums), trends in the evolution of green areas and parks, etc. Our work is also concerned with the evolution of the landcover in crop cultures. Other applications may include ecological problematics such as the study of SITS to assess phenomena such as deforestation or the impact of droughts.

The proposed change detection approach can be applied to short time series and does not depend on the temporal resolution, as it does not demand the prior knowledge of the seasonal trend that can be computed only if the series is long enough and regularly distributed. For our knowledge, no such approaches have been proposed yet. Moreover, there is no other unsupervised framework that does both change detection and the clustering of the subsequent change graphs.

We summarize the contributions of our work as follows.

1) We propose a novel framework for HR SITS change detection and clustering that do not depend on temporal resolution,
2) We propose a method to model spatiotemporal change phenomena in a form of evolution graphs,
3) We develop a GRU AE-based clustering model that is able to proceed varying length evolution change graphs.

II. METHODOLOGY

In this section, we present the developed framework for the detection and analysis of nontrivial changes in SITS. We start by giving the overall description of the framework and then we detail each step in the corresponding sections.

A. Proposed Framework

The proposed framework has been developed as a part of a SITS analysis algorithm. This algorithm aims to identify spatiotemporal entities presented in SITS and associate them to three different types of temporal behaviors. These temporal behaviors correspond to no change areas, seasonal changes, and nontrivial changes. No change areas are mostly presented by spatiotemporal entities that have the same spectral signature over the whole SITS, such as city center, residential areas, deep water, sands, etc. Trivial (seasonal) changes correspond to cyclic changes in vegetation prevailing in the study area. Finally, nontrivial change areas are mostly represented by permanent
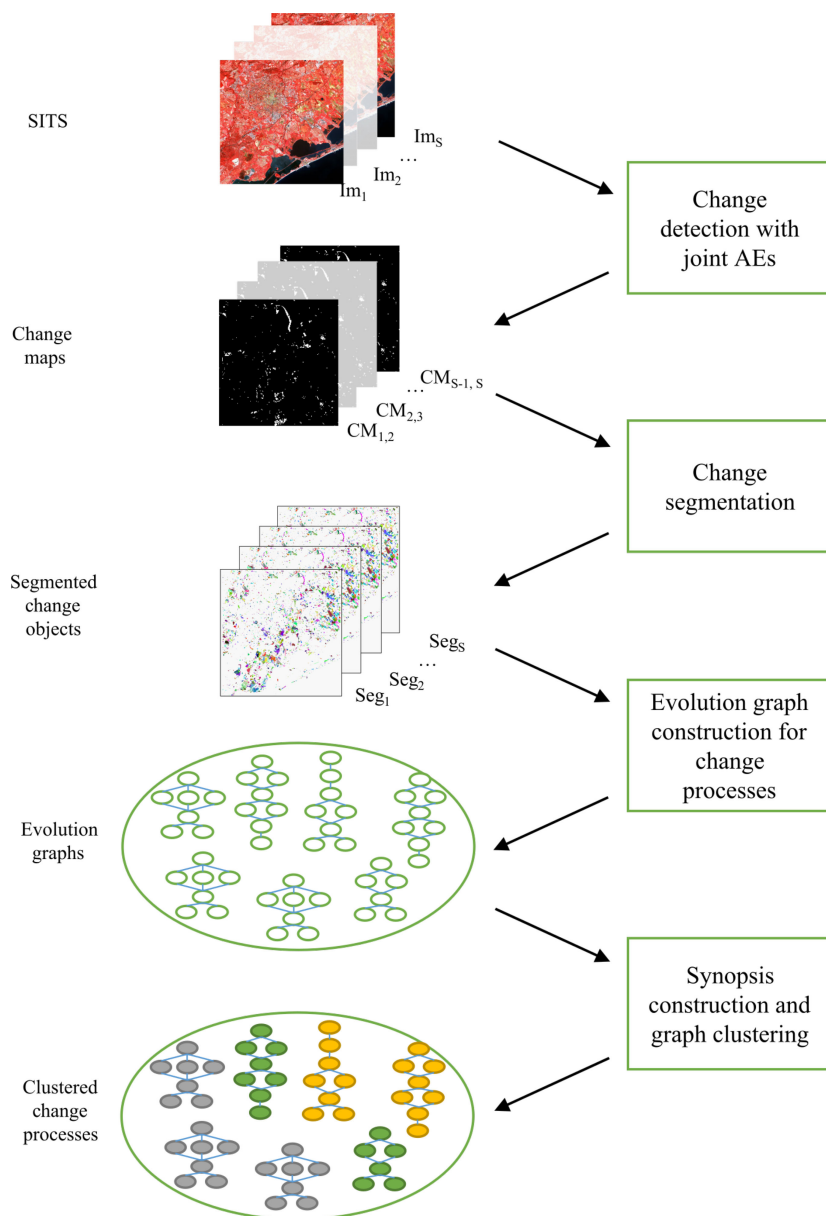
Fig. 1. Proposed framework.

changes such as new constructions, changes caused by some natural disasters, crop rotations, and the vegetation that do not follow the overall seasonal tendency of the study area. As nontrivial changes are less numerous and sometimes unique, they are considered by most clustering algorithms as outliers and, hence, demand a special approach for their identification and analysis that is presented in this article. Moreover, as often we do not dispose of HR SITS with regular temporal resolution, the presented approach does not depend on temporal resolution of SITS.

The proposed approach is composed of several steps. Let R_S be a time series of S coregistered images $Im_1, Im_2, ..., Im_S$ acquired at timestamps $T_1, T_2, ..., T_S$. The algorithm steps are the following (see Fig. 1).

1) We start by applying a bitemporal nontrivial change detection algorithm [24] to every couple of consecutive images $Im_n - Im_{n+1}$ ($n \in [1, S-1]$) in order to get $S-1$ binary change maps $CM_{1,2}, CM_{2,3}, ..., CM_{S-1,S}$ for the whole dataset.

2) We extract spatiotemporal change areas by applying the change masks to corresponding images of SITS. Then, we perform image segmentation within these change areas to obtain changed objects.

3) Afterwards, the change objects located in the same geographic area are grouped in temporal evolution graphs [17], [18].

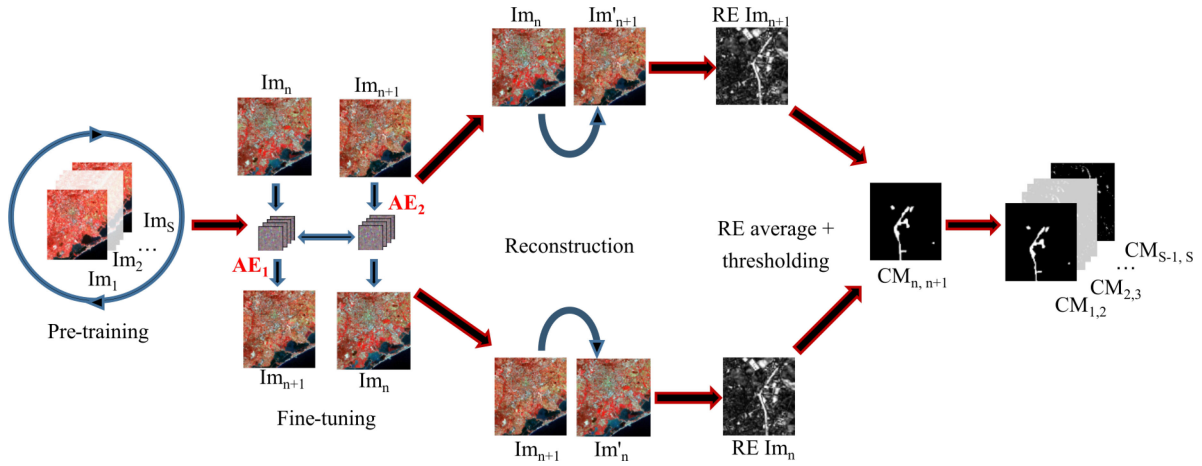4) Finally, we cluster the obtained graphs using the features extracted from the change areas. We use summarized

Fig. 2. Bitemporal change detection algorithm [24].

representation of graph structure—synopsis—as input sequences of clustering approach.

B. Change Detection

1) Bitemporal Change Detection: For the proposed framework, we have developed a bitemporal nontrivial change detection algorithm [24], [25] that is applied to the whole SITS. Contrary to multitemporal approaches that demands regular temporal resolution of SITS to get correct results, the bitemporal approach does not assume the temporal regularity of the SITS. Moreover, when using the bitemporal approach, we can easily detect the begin and the end of a change process in a SITS of fixed size without applying any supplementary methods. Then, the obtained bitemporal change detection results are analyzed in the context of the whole SITS.

The principle of change detection between two bitemporal images is based on a joint AEs model that is able to reconstruct Im_{n+1} from Im_n and vice versa by learning the translation of image features. Areas with no changes or seasonal changes will be easily translated by the model and will have a small reconstruction error (RE). At the same time, as nontrivial changes are less numerous and some of them are unique, they will be considered as outliers and will have high REs that can be used to map areas of nontrivial changes.

We remind the steps of our change detection algorithm [24], which are as follows (see Fig. 2).
1) The first step consists in model pretraining on whole SITS.
2) During the second step, for each couple of images Im_n to Im_{n+1}, we fine-tune the joint AE model by mapping Im_n to Im'_{n+1} and Im_{n+1} to Im'_n.
3) Then, we compute the RE of Im'_{n+1} from Im_n and vice versa for every patch of the images. In other words, the RE of every patch is associated to the position of its central pixel on the image.
4) In the last step, we detect the areas with high RE using automatic Otsu's thresholding method [26] in order to create a binary change map $CM_{n,n+1}$ with the nontrivial changes.
5) Finally, we obtain $S-1$ CMs for whole SITS.

An AE is a type of network where the input is the same as the output that allows us to train the model in a nonsupervised manner. In this work, we use two different types of AEs— convolutional AE and fully convolutional AE—both for change detection and further feature extraction from the change areas. The first type of AE is more complex and allows us to reduce the input 3-D data in a feature vector.

In the standard convolutional AE model, during the encoding pass, some meaningful feature maps (FM) are first extracted with convolution layers and then they are flattened and compressed in a feature vector (bottleneck of the AE) by fully connected (FC) layers. The decoding pass is symmetrical to the encoding one and aims to reconstruct the initial patch from the feature vector. Contrary to the convolutional AE, fully convolutional AE contain only convolution layers for both encoding and decoding steps, so the encoder output is represented only by FM. The models are optimized to minimize the difference between the decoded output and the initial input data. Here, for model optimization and reconstruction loss, we use the mean-squared error (MSE)

$$\text{MSE}(o,t) = \frac{\sum_{n=1}^{N} l_n}{N}, \quad l_n = (o_n - t_n)^2 \qquad (1)$$

where o is the output patch of the model, t is the target patch, and N is the number of patches per batch.

In the previous work [24], we have proposed two different AE models for bitemporal change detection (see Table I, where p is the patch size and B is the number of spectral bands). The choice of model depends mostly on the patch size p. Note that for all convolutional layers the kernel size is 3. We equally apply batch normalization to all convolutional layers, except the one with ℓ_2-norm.

In the proposed method, the change detection is performed in a patch-wise manner for every i,jth pixel of every couple of images ($i \in [1,H]$, $j \in [1,W]$, where H and W are images height and width, respectively). To pretrain the model and avoid overfitting, we randomly select $H \times W/S$ patches from each image to form a pretraining dataset. During the pretraining phase each patch is encoded in some abstract presentation and then

TABLE I
MODELS ARCHITECTURE FOR BITEMPORAL CHANGE DETECTION

	Fully-convolutional AE	Convolutional AE
encoder	Conv(B,32)*+ReLU Conv(32,32)+ReLU Conv(32,64)+ReLU Conv(64,64)+ℓ_2-norm	Conv(B,32)+ReLU Conv(32,32)+ReLU Conv(32,64)+ReLU Conv(64,64)+ReLU Linear($64\times p^2$,$12\times p^2$)**+ReLU Linear($12\times p^2$,$2\times p^2$)+ℓ_2-norm
decoder	Conv(64,64)+ReLU Conv(64,32)+ReLU Conv(32,32)+ReLU Conv(32,B)+Sigmoid	Linear($2\times p^2$,$12\times p^2$)+ReLU Linear($12\times p^2$,$64\times p^2$)+ReLU Conv(64,64)+ReLU Conv(64,32)+ReLU Conv(32,32)+ReLU Conv(32,B)+Sigmoid

*In Conv(a, b) a and b stand for the number of the input and output channels, respectively.

**In Linear(a, b) a and b stand for the sizes of the input and output vectors, respectively.

decoded back with a chosen AE model. We control the model loss and stop the training when the loss stabilizes.

Once the model is stable, we move on to the model fine-tuning for every couple of images Im_n and Im_{n+1}. We use joint AEs model composed of two AEs: AE$_1$ transforms the patches of Im_n into the patches of Im'_{n+1} and AE$_2$ transforms Im_{n+1} into Im'_n (see Fig. 2, Fine-tuning). Both AEs have the same layers as the pretrained model and are initialized with the parameters it learned. We use all image patches to fine-tune the model. The joint AEs model is trained to minimize the difference between (2) the following.

1) The decoded output of AE$_1$ and Im_{n+1} (loss$_1$).
2) The decoded output of AE$_2$ and Im_n (loss$_2$).
3) The encoded outputs of AE$_1$ and AE$_2$ (FM for fully convolutional AEs and bottleneck feature vectors for convolutional AEs) (loss$_3$).

$$\text{modelLoss} = \text{loss}_1 + \text{loss}_2 + \text{loss}_3 \qquad (2)$$

The three losses are not weighted and equally participate in the model optimization.

We transform the patches of Im_n to Im'_{n+1} and vice versa and calculate the REs RE$_{n+1}$ and RE$_n$ in both directions and apply Otsu thresholding to the average of these errors. Otsu thresholding is calculated in a fully automatic manner to detect high RE values to produce a binary change map CM$_{n,n+1}$. Finally, we obtain $S-1$ bitemporal change maps that isolate nontrivial change areas for the whole SITS.

To estimate the quality of the model during pretraining and fine-tuning, we verify the total losses of training epochs and stop the training once the model is stable. We retain the model from the epoch that achieves the highest reconstruction capability, e.g., the smallest value in the loss function.

2) Multitemporal Change Interpretation: Note that bitemporal nontrivial changes can also be interpreted as contextual anomalies [27] as they depend on the overall change tendency presented in the couple of images. Their interpretation might be changed when passing from bitemporal to multitemporal analyses as the context will be changed. To introduce
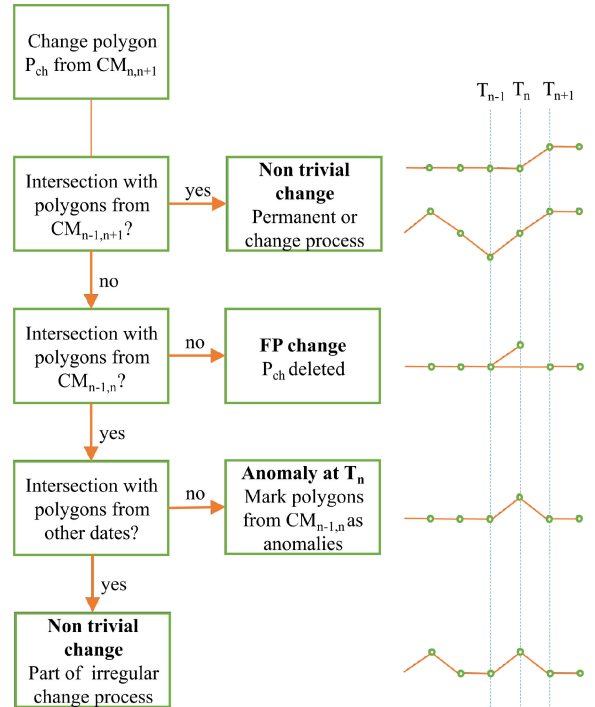

Fig. 3. Correction of detected bitemporal contextual anomalies accordingly to multitemporal context.

multitemporal context when detecting changes that appear between timestamps T_n and T_{n+1}, we propose to check if the detected change polygon has equally appeared at certain other change maps (see Fig. 3).

First, we apply the aforementioned change detection algorithm for every couple of $Im_{n-1}-Im_{n+1}$ images ($n \in [2, S-1]$) to calculate CM$_{n-1,n+1}$. As an isolated change area may contain several change objects, we perform image segmentation to obtain bitemporal change polygons (the segmentation algorithm is explained in the following section and applied directly to the concatenated couples of images Im_n-Im_{n+1} and $Im_{n-1}-Im_{n+1}$). Then, we choose CM$_{n,n+1}$ as reference and check if each change polygon P_{ch} from CM$_{n,n+1}$ is equally present in CM$_{n-1,n+1}$. If P_{ch} has a spatial intersection with any polygon(s) of CM$_{n-1,n+1}$, we mark P_{ch} as change in the SITS context (a permanent bitemporal change or a part of a change process). If P_{ch} does not have spatial intersection with any polygon(s) of CM$_{n-1,n+1}$, it may belong to following different types of temporal behavior.

1) If P_{ch} does not have any spatial intersection with any polygon(s) from CM$_{n-1,n}$, it is marked as false positive (FP) change alarm caused by some image defaults and accuracy issues of unsupervised methods.
2) If P_{ch} has a spatial intersection with any polygon(s) from CM$_{n-1,n}$ and with polygon(s) in at least one other change map (CM$_{1,2}$, CM$_{2,3}$, CM$_{3,4}$, etc.), it is marked as a part of an irregular change process.
3) Finally, if P_{ch} has intersection only with polygon(s) from CM$_{n-1,n}$ and does not have any intersection with polygons from other change maps, it is marked as a one time
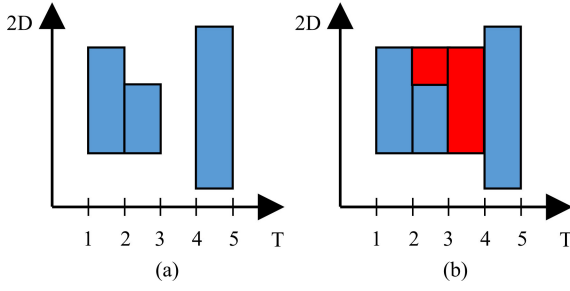
Fig. 4. Transformation of a discontinuous change process into a continuous one. Horizontal and vertical axes represent timestamps and object pixels respectively. (a) Discontinuous change process. (b) Corrected blue polygons correspond to detected change objects, red polygons are added to transform a discontinuous change process into a continuous one.

anomaly that happened at timestamp T_n. In this case, all change polygons from $\mathrm{CM}_{n-1,n}$ that have intersection with P_{ch} are also marked as one time anomalies.

Note that here we use a threshold t_{int} that defines the minimum percentage of spatial intersection of P_{ch} with other change polygon(s), otherwise, it is considered that there is no intersection.

In this work, we suppose that every change process belonging to the same geographical location is continuous. For example, if a pixel i, j has been classified as change in $\mathrm{CM}_{1,2}$, $\mathrm{CM}_{2,3}$, and $\mathrm{CM}_{4,5}$, it should be also marked as change in $\mathrm{CM}_{3,4}$ (see Fig. 4).

We correct the whole SITS in order to transform change processes into continuous ones. Finally, for every image Im_n, we apply the union of change maps $\mathrm{CM}_{n-1,n}$ and $\mathrm{CM}_{n,n+1}$ to extract the change areas. Obviously, we apply only one change map for the first and last images of the SITS.

C. Image Segmentation

In the previous section, we have proposed an approach that introduces multitemporal context to the analysis of bitemporal changes. As it was mentioned, an isolated change area may potentially contain several types of changes. The identification of change segments is needed at two different stages of the proposed framework. First, we detect change objects for the aforementioned algorithm. The segmentation is performed for every couple of concatenated images Im_n–Im_{n+1} within the associated change mask. For the second stage, we perform the segmentation change areas of every image Im_n, so the detected segments will later be used for the construction of evolution graphs.

For both steps of the framework, we leverage a graph-based tree-merging segmentation algorithm [28] due to its ability to produce relatively large segments without merging different classes together. Large segments facilitate further construction and interpretation of evolution graphs as the shapes of some change segments may have important variations from one image to another when they are oversegmented.

The principle of the segmentation algorithm is the following: an image is represented as a graph where the pixels are nodes and resemblance values between two pixels are edges. Usually, each pixel has eight-connected neighborhood. At the beginning

of the algorithm, each pixel belongs to a separate segment, then, in each step of the segmentation algorithm, the adjacent segments are merged if the difference between them is not much less than the internal variability of these segments. To measure the resemblance value between two pixels, we use Mahalanobis distance

$$d\left(\vec{x}, \vec{y}\right) = \sqrt{\left(\vec{x} - \vec{y}\right)^{\top} \mathrm{Cov}^{-1}\left(\vec{x} - \vec{y}\right)} \qquad (3)$$

where $\vec{x}, \vec{y}$ are two pixels values and Cov is the covariance matrix of the image values.

The proposed algorithm requires three parameters which are σ—the standard deviation for Gaussian kernel that is used for image smoothing during the presegmentation phase, k—the constant for the thresholding function used for segment merging, and $\mathrm{min\_size}$ that defines the minimum component size for the postprocessing stage.

D. Evolution Graphs

For our work, we adapt the evolution graph construction algorithm presented in [18] and [17] by passing from the analysis of the whole SITS to the analysis of change areas. This approach combines graph-based techniques and data-mining technology and is used to describe a spatiotemporal evolution of a detected phenomena. The initial approach is the following: given a SITS and its associated segmentation, we choose a set of objects that corresponds to the spatial entities we want to monitor. This set of objects is called Bounding Boxes (BBs). A BB can come from any timestamp and is connected to the objects covered by its footprint at other timestamps. A BB and objects connected to it form an evolution graph. Each evolution graph can have only one BB and has to be continuous. Every object of a graph represents a node and overlapping values between two objects at two consecutive timestamps represent an edge. Objects at timestamp T_n can be connected only to objects from T_{n-1} or T_{n+1}, a timestamp that contains a BB can have only one object that corresponds to this BB.

1) Bounding Box Selection: For BB selection, we use the same strategy as in the initial articles [17], [18], except that we do not work with the whole dataset, but only with change areas. The methodology is the following:

1) We create an empty 2-D grid G that covers all changed areas. If a segment is chosen as a BB, the grid is filled with its footprint.
2) We create an empty list of candidate BB $L_{\mathrm{cand_{BB}}}$.
3) We sort all the segments presented in SITS change areas by their size in the descending order.
4) We iterate through each segment of the sorted list.
5) We calculate the novelty and weight of each element comparatively to already iterated segments. These values are calculated as follows:

$$\mathrm{novelty} = \frac{\mathrm{Pix}(O) - \mathrm{CA}}{\mathrm{Pix}(O)} \qquad (4)$$

where $\mathrm{Pix}(O)$ is the segments size in pixels and CA is the size of a grid area that is covered by this segment and

already filled by previous candidate BBs.

$$\text{weight} = \begin{cases} \text{Pix}(O) & \text{if novelty} = 1 \\ \text{novelty} & \text{if } \alpha \leq \text{novelty} < 1 \\ 0 & \text{if novelty} < \alpha \end{cases} \quad (5)$$

If weight $\geq \alpha$, we select this segments as a candidate BB and add it to $L_{\text{cand}_{\text{BB}}}$.

6) We iteratively update the weights in $L_{\text{cand}_{\text{BB}}}$ and delete previously detected candidate BBs if needed.

7) We iterate through the list until no more segments are left or until the grid is filled.

2) Construction of Evolution Graphs: In our method, we construct graphs in such a manner that every graph contains only coherent information. In other words, every BB is connected only to its best matching segments comparatively to neighbor BBs and each segment can belong only to one or no evolution graph.

In order to construct graphs that contain only objects belonging to the same phenomena, [18] and [17] propose to use two parameters for the construction of evolution graphs that are independent of each other: at least τ_1 of the object should be inside of BB footprint, and the intersection with the object should represent at least τ_2 of BB footprint.

$$\tau_1 = \frac{\text{Pix}(O) \cap \text{Pix}(\text{BB})}{\text{Pix}(\text{BB})} \quad (6)$$

$$\tau_2 = \frac{\text{Pix}(O) \cap \text{Pix}(\text{BB})}{\text{Pix}(O)}. \quad (7)$$

The first parameter τ_1 is the most important and allows to select only the objects that are covered the most by BB footprint. The second parameter τ_2 is used to keep the objects filling only certain percentage of the footprint.

In our experiments, we are interested only in change areas of SITS, hence an isolated evolution processes may not cover the same area at different timestamps (e.g., a growing construction of a residential area).

However, it may cause that some objects will be covered by two or more BBs. To deal with this issue, we propose to associate each object only to its best matching BB, i.e., the one with the highest τ_1 value. In addition, for $\alpha < 0.5$, an object O_i may be a bounding box BB_i of a corresponding graph and at the same time attached to an evolution graph of a neighbor bounding box BB_j. In this case, we compare BB_i's (O_i) weight value with O_i's τ_1 value in the evolution graph of a neighbor bounding box BB_j. If $\tau_1 >$ weight, O_i is attached to BB_j evolution graph and the evolution graph of BB_i is destroyed and its objects are attached to other graphs, otherwise BB_i and its evolution graph are kept and O_i is deleted from BB_j's evolution graph.

Due to pixel shift and to some FP changes, we may observe many parasite objects presented in evolution graphs that usually correspond to crop fields. Parasite objects are the small objects attached to the beginning or to the end of the graph, these objects are the only objects presented at given timestamp and are much smaller than the footprint of the previous/next timestamp. If an evolution graph contains a parasite object, it can influence its further interpretation. Given the size of these objects, we can consider that they correspond to FP changes and need to be deleted. To minimize the number of parasite objects, we introduce τ_3 parameter that represents the minimum ratio between coverage of two consecutive timestamps.

$$\tau_3 = \frac{\sum_1^q \text{Pix}(O_i^{n+1})}{\sum_1^r \text{Pix}(O_j^n)} \quad (8)$$

where q and r are the number of objects at timestamps T_{n+1} and T_n, respectively, $\text{Pix}(O_i^{n+1})$ is ith object at timestamp T_{n+1} and $\text{Pix}(O_j^n)$ is jth object at timestamp T_n.

E. Graph Synopsis and Feature Extraction

To cluster the extracted evolution change graphs, we calculate each graph synopsis as in [17]. A synopsis summarize the information about each graph and allow to compare them with each other. A synopsis Q is defined as a sequence of the same length as the corresponding evolution graph. Each timestamp T_n of the sequence Q contains the aggregated values of graphs objects at this timestamp. The influence of each object at the aggregated value at the timestamp T_n is proportional to its size and calculated as follows:

$$Q_n = \frac{\sum_1^r \text{Pix}(O_j^n) \cdot v_j}{\sum_1^r \text{Pix}(O_j^n)} \quad (9)$$

where Q_n is the synopsis value at timestamp T_n, $\text{Pix}(O_j^n)$ is the size of a jth object at timestamp T_n ($j \in [1, r]$, where r is the total number of the objects presented in the evolution graph E at the timestamp T_n) and v_j is the corresponding mean of the object value.

In our work, each object is characterized by the feature values extracted with deep convolutional denoising AE. We add a Gaussian noise to the initial patches of the dataset during the training to extract more robust features. The noise is added only to the input of the first convolutional layer. To calculate the MSE for model optimization, we use the initial patches without noise as targets. For every pixel i, j, n its features are extracted in a patch-wise manner.

The choice of feature extraction and its efficiency is described in the next section, where we compare the clustering performance for our proposed framework using a feature extraction step and also using raw pixel values for graph descriptors.

All the images are normalized using dataset mean and standard deviation before texture extraction.

In our proposed framework, the feature extraction steps are the following:

1) We extract patches of size p for every pixel of every image of SITS to train convolutional AE model.

2) We divide the extracted patch dataset in training and validation parts (67% and 33%). Validation part is used for early stopping algorithm [29] that prevents our model from over-fitting and works in an unsupervised manner. The more the model is generalized—the more robust features we get.

3) We train the AE model in such manner that every patch from the training dataset is first encoded in a feature vector and then is decoded back to the initial patch. We use mean

TABLE II
FEATURE EXTRACTION MODEL

	Feature extraction
encoder	Conv(B,32)*+ReLU
	Conv(32,32)+ReLU
	Conv(32,64)+ReLU
	Conv(64,64)+ReLU
	MaxPooling(kernel=3, stride=3)
	Conv(64,128)+ReLU
	Conv(128,128)+ReLU
	MaxPooling(kernel=3)
	Linear(128,64)**+ReLU
	Linear(64,32)+ReLU
	Linear(32,f)+ℓ_2-$norm$
decoder	Linear(f,32)+ReLU
	Linear(32,64)+ReLU
	Linear(64,128)+ReLU
	UnPooling(kernel=3)
	Conv(128,128)+ReLU
	Conv(128,64)+ReLU
	UnPooling(kernel=3, stride=3)
	Conv(64,64)+ReLU
	Conv(64,32)+ReLU
	Conv(32,32)+ReLU
	Conv(32,B)+ReLU

*In Conv(a, b) a and b stand for the number of the input and output channels, respectively.
**In Linear(a, b) a and b stand for the sizes of the input and output vectors, respectively.

square error of the patch reconstruction to optimize the model at each iteration.

4) The early stopping algorithm is applied at every epoch and check the loss value when fitting validation dataset to the obtained model. If the validation loss does not improve during chosen number of epoch, the model is considered stable and the training is stopped.

5) We use the encoding part of the AE to encode every patch of SITS change areas in a feature vector.

The feature extraction model is presented in Table II, where f is the size of the encoded feature vector. Note that for all convolutional layers the kernel size is 3. We equally apply batch normalization to all convolutional layers.

F. Clustering

In the presented framework, we propose to use GRU [22] AE combined with hierarchical agglomerative clustering [23] to regroup the obtained evolution graphs. GRU is a recurrent neural networks-based (RNN) type of NN that is able to process time series in order to extract some meaningful information from it. Unlike many other approaches for time-series analysis, RNN is able to deal with varying sequence lengths.

In general, RNN is constructed as follows: let $X = \{x_1, x_2, \ldots, x_n, \ldots, x_S\}$ be a sequence composed of S timestamps. In addition to the sequence values, each timestamp T_n is associated to a hidden state h_n (see Fig. 5), which represent the accumulative value of the previous hidden states of the sequence. The final hidden state h_S characterizes the whole sequence and is used afterward for classification or clustering. The main problem of RNN is that the value of each hidden state h_n depends only on the value of previous hidden state h_{n-1}, hence, RNN networks may suffer from a long-short memory
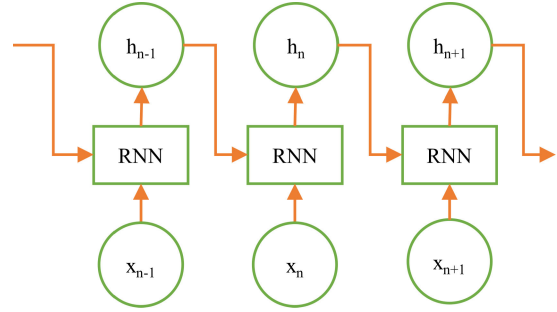

Fig. 5. RNN model.

problem caused by vanishing gradient and does not consider long-term dependencies. To solve this issue, more complex LSTM networks were introduced. Contrary to RNN, LSTM contains input, output, and forget gates as well as memory cell c_n at each timestamp that makes it possible to retain meaningful information from the previous steps and, as a consequence, the value of h_n depends on all the previous hidden states of the sequence and not only on h_{n-1}. Later, to facilitate the computation and implementation of the LSTM model, GRU networks were developed [22] for natural language processing tasks. GRU contains only update and reset gates that allows model to train faster with less memory consuming. GRU were successfully adapted for remote sensing applications and proved to be more efficient than LSTM networks in this research area as they give better or similar results to LSTM using less training parameters and time [30]–[32].

To define the nature of change process, it is necessary to consider the state of sequence at the first timestamp, regardless other timestamps values. Due to the long-short memory problem, RNN does not always keep in memory the beginning of the sequences, especially when it is long. As a result, in our work, we choose more complex GRU model for clustering basing on aforementioned studies in remote sensing.

The principle of GRU AE is the same as of the convolutional one mentioned before. During the encoding pass, GRU AE extracts the accumulated hidden state of the sequence h_S at the last timestamp. The last hidden state is then self-concatenated S times [33] and passed to the decoding part that aims to reconstruct the inversed initial sequence $X_{\text{inv}} = \{x_S, \ldots, x_n, \ldots, x_2, x_1\}$. As it is usually recommended to set hidden state size large (>100), we add FC layers before the GRU AE bottleneck to compress the size of hidden state to ameliorate the further clustering results. The overall GRU AE schema is presented on Fig. 6. Finally, we apply hierarchical clustering to the bottleneck of GRU AE to obtain the associated change clusters.

As the input sequences have varying length, some data preparation is necessary, so the GRU is able to correctly process it. Data preparation is performed for every training batch individually, after the input GRU dataset has been created. For every batch B_i, we perform the following steps (see Fig. 7):

1) We define the maximum sequence length d of B_i.

2) We zero-pad the end of all the sequences of B_i, so they have the same length d.
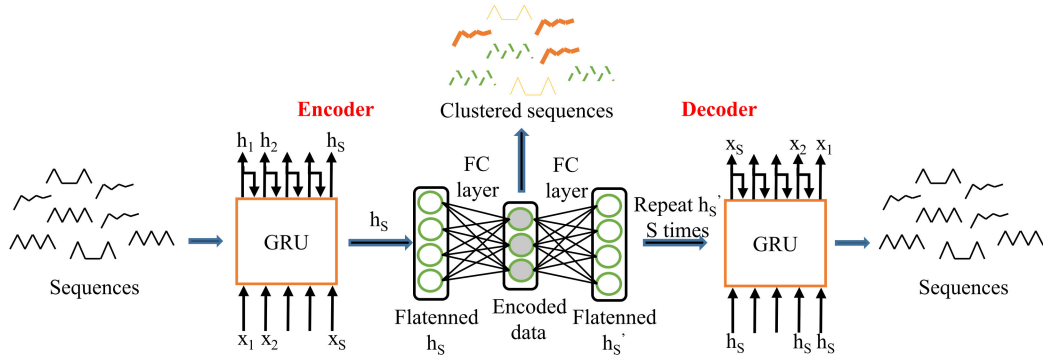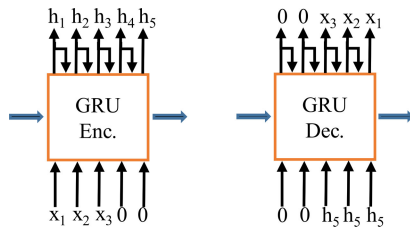
Fig. 6.　GRU AE clustering model.

Fig. 7.　Padding of data sequences. In this example, the initial sequence x_1, x_2, x_3 has the length of three timestamps and the maximum sequence length per batch is $d = 5$. For the simplicity of representation, we do not consider the number of features of each sequence.

3) The padded sequences are passed to the encoder, for each sequence its final hidden state h_S is obtained.
4) As indicated before, we use the cloned h_S as the input of GRU layer in the decoding part, where h_S is repeated S times.
5) We apply the inverted padding mask to the cloned h_S sequence that is fed to GRU layers of the decoder.
6) The output of the decoder should resemble to the inverted padded input sequence.

While the padding of the encoder input sequence allows us to proceed batches with varying length sequences, the padding of the decoder input improves the model quality, especially, it lowers the influence of sequence lengths on the extracted encoded features.

We do not divide the sequence data into training and validation datasets as the nature of some change sequences may be unique. For these reason, we control the training loss changes between two consecutive epochs to prevent the model overfitting. We have used MSE as the loss function.

The model configuration is presented in Table III, where f is number of features of the input sequences, hidden_size is the length of the hidden state vector, d is the maximum sequence length per batch, f_hidden is the size of the encoded hidden state vector.

III. Data and Data Preprocessing

We test the proposed approach on two real-life publicly available time series—SPOT-5 and Sentinel-2. While the first SITS

TABLE III
GRU Model

	Sequence feature extraction
enc.	GRU(f,hidden_size, dropout=0.4)* (2 layers)
	Linear(hidden_size,f_hidden)**+ℓ_2-$norm$
dec.	Linear(f_hidden,hidden_size)+ReLU
	Repeat hidden state d times
	Apply inversed padding mask
	GRU(hidden_size,f, dropout=0.4) (2 layers)

*In GRU(a, b dropout=) a and b stand for the number of features of the input sequence and the size of the hidden state vector.
**In Linear(a, b) a and b stand for the sizes of the input and output vectors, respectively.

TABLE IV
Image Acquisition Dates for the Montpellier Dataset

Acquisition date, yyyy-mm-dd			
1	2002-10-05	7	2006-02-18
2	2003-09-18	8	2006-06-03
3	2004-05-14	9	2007-02-01
4	2004-08-22	10	2007-04-06
5	2005-04-27	11	2008-06-21
6	2005-12-01	12	2008-08-21

contains 12 images that are irregularly sampled over six years, the second one contains 17 images taken over two years with more regular temporal resolution.

A. Montpellier

SPOT-5 dataset was taken over the Montpellier area, France between 2002 and 2008 and belongs to the archive Spot World Heritage.[1] This particular dataset was chosen due to its high ratio of agricultural lands and progressive construction of new areas. We have filtered out the cloudy and flawed images from all the images acquired by SPOT-5 mission over the considered geographic area and obtained 12 exploitable images with irregular temporal resolution (minimum temporal distance between two consecutive images is 2 months, maximum—14 months, average—6 months). Distribution of dataset images is presented in Table IV. All SPOT-5 images provide green, red, near infrared (NIR) and short-wave infrared (SWIR) bands with 10 m resolution. The original images are clipped to rectangular

[1][Online]. Available: https://theia.cnes.fr/

TABLE V
IMAGE ACQUISITION DATES FOR THE ROSTOV DATASET

	Acquisition date, yyyy-mm-dd		
1	2016-07-15	10	2018-04-11
2	2016-08-04	11	2018-05-01
3	2016-09-13	12	2018-05-31
4	2016-11-22	13	2018-07-10
5	2017-05-01	14	2018-08-19
6	2017-06-30	15	2018-09-13
7	2017-08-09	16	2018-10-13
8	2017-09-18	17	2008-11-02
9	2018-01-11		

shapes of 1600×1700 pixels and transformed to UTM zone 31 N: EPSG Projection. The clipped image extent corresponds, respectively, to the following latitude and longitude in WGS-84 system:

1) bottom left corner: $43°30'6.0444''N$, $3°47'30.066''E$.
2) top right corner: $43°39'22.4856''N$, $3°59'31.596''E$.

B. Rostov-on-Don

Sentinel-2 dataset was taken over the city of Rostov-on-Don[2] (the dataset called later Rostov), Russia between 2015 and 2018. This dataset was chosen as the city of Rostov-on-Don underwent the constructions coincided with FIFA world cup 2018 and equally because it has waste agricultural areas. We have filtered out cloudy, snow-covered and flawed images from 2015 (beginning of the Sentinel-2 mission) till the end of 2018 that gave us 26 images. We deleted six more images from the dataset as the temporal distance between two consecutive images did not exceed 15 days. Moreover, we have deleted the first three images of the dataset because of the irregular temporal distance between them (distance between first and second image is around one month, second and third—4 months, third and fourth—5 months). Despite the fact, that our algorithm was developed for the SITS with irregular temporal resolution, it was decided to work with two datasets that differ in the terms of temporal behavior, so the Rostov dataset has "more regular temporal resolution" comparing to the Montpellier dataset. Finally, we have obtained 17 images taken from July 2016 to November 2018 with relatively regular temporal resolution. The average gap between two consecutive images does not exceed 1.5 months, minimum—20 days, maximum—5 months (corresponds to winter with deleted snow-covered images). Distribution of dataset images is presented in Table V.

Sentinel-2 provides multispectral images with different spatial resolution spectral bands. It was decided to keep only 10 m resolution green, red and NIR bands as they contain all the essential information for general change detection and analysis. The images were provided in WGS84/UTM zone 37 N projection and clipped to 2200×2400 pixels areas. The clipped image extent corresponds, respectively, to the following latitude and longitude in WGS-84 system.

1) Bottom left corner: $47°11'3.5664''N$, $39°34'3.2628''E$.
2) Top right corner: $47°24'7.3332''N$, $39°51'41.328''E$.

The preprocessing level of both datasets is 1 C (orthorectified images, reflectance at the top of atmosphere). For this reason, both SITS were radiometrically normalized with the aim to obtain homogeneous and comparable spectral values over each dataset. For the image normalization, we have used an algorithm introduced in [34] that is based on histogram analysis of pixel distributions.

IV. EXPERIMENTS

In this section, we present parameters settings for the proposed framework and the corresponding results. We equally give some quantitative and qualitative analysis of the obtained results to show the advantages and eventual limits of the proposed approach. All the algorithms were tested on 6 cores Intel(R) Core(TM) i7-6850 K CPU 3.60 GHz with 32 Gb of RAM computer with a NVIDIA Titan X GPU with 12 GB of RAM and developed in Python 3 programming language using PyTorch 1.3 library on Ubuntu 16.4. *tslearn* library [35] was used to calculate DTW distance matrices for concurrent approaches.

Source files related to the project are available in the $GitHub$ repository.[3]

A. Experimental Settings

The chosen datasets have some differences in temporal resolution and images quality. For this reason, the proposed framework have some dissimilarities based on the particularities of each dataset.

Due to the high temporal gap between the images of Montpellier dataset, we tend to extract the maximum of the information from it, so all extracted change sequences are kept. On the other hand, for the Rostov dataset, we keep only the change sequences that are at least three timestamps long. As the average temporal resolution between two images is less than two months, we consider that two timestamps change sequences most probably correspond to some false change alarms that could not be corrected when introducing the multitemporal context.

For the Rostov dataset, we manually apply a water mask and a mask for a part of the city with dense constructions. While the first mask is applied to eliminate FP changes caused by boats, the second one is indispensable to lower the number of FP changes caused by the shadows from tall buildings and changing luminosity that can not be corrected.

The water mask was obtained with Sen2Cor[4] classification module. First, the water surfaces were detected for every image of the dataset individually. Second, the pixels that were marked as water for more than 50% of the images of the dataset were added to the water mask. Dense urban area mask was created manually and has almost a rectangular shape that cover a specific city area.

The last difference is related to the fact that the images of the second dataset are not perfectly aligned and some of them have

---

[2][Online]. Available: https://earthexplorer.usgs.gov/

[3][Online]. Available: https://github.com/ekalinicheva/Unsupervised-CD-in-SITS-using-DL-and-Graphs/

[4][Online]. Available: https://step.esa.int/main/third-party-plugins-2/sen2cor/

one pixel size shift from the origin in different directions. This shift could not be eliminated with automatic techniques and, for this reason, we have decided to use bigger patches comparatively to Montpellier dataset for bi-temporal change detection.

1) Bitemporal Change Detection: The parameters for bitemporal change detection were set accordingly to [24]. For the Montpellier dataset, we have chosen the patch size $p = 5$ and the convolution AE model for feature translation as they were proved to give the best results for 10 m resolution images [24]. Due to the shift presented in Rostov dataset, we have increased the patch size to $p = 7$ in order to minimize the shift influence on patch translation. We have equally chosen fully convolutional AE model for feature translation as it gives better results than convolutional AE for bigger patch sizes [24]. The AE models for both datasets are presented in Table I. For both datasets, we use only three bands for the change detection—red, green, and NIR—as they contain the essential information about the images.

Before applying Otsu thresholding, we equally exclude $high_v$% of the highest RE values considering they correspond to some noise and extreme outliers. The variation of the $high_v$ only balances precision/recall values, without influencing kappa value. The higher $high_v$ is —the higher is the recall and the smaller is the precision, and vise versa. To balance these values, we set $high_v$ to 0.5% for both datasets. The analysis of the influence of $high_v$ value on different metrics is detailed in the following section.

The following quality criteria were used to evaluate the performances of the different approaches: precision (10), recall (11), and Cohen's kappa score [36].

$$\text{Precision} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalsePositives}} \tag{10}$$

$$\text{Recall} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalseNegatives}}. \tag{11}$$

To evaluate the performance of the proposed methods on two datasets, we have used ground truth maps extracted for different parts of each dataset. For the Montpellier dataset, the overall ground truth area was 6 00 000 pixels, while for the Rostov dataset this area was 7 80 000 pixels. The ground truth maps were manually created for the chosen areas by comparing the objects presented in the couples of images by one of the authors who is the field expert.

2) Image Segmentation: To segment the extracted change areas, we have used a tree-merging bottom-up segmentation algorithm [28] described earlier. The segmentation parameters were chosen empirically and allow us to produce large segments for both SITS without mixing different classes in one segment. For each dataset, we have manually chosen reference adjacent objects that have to be divided in separated segments. These objects need to have similar spectral properties, but still be easily distinguished by human eye. We set the same parameters for single image segmentation and for the segmentation of concatenated images, so the change objects extracted from concatenated images are relevant to single-image objects.

TABLE VI
PARAMETERS FOR EVOLUTION GRAPHS CONSTRUCTION

Dataset	α	τ_1	τ_3
Montpellier	0.4	0.4	0.2
Rostov	0.5	0.4	0.3

The segmentation parameters for both datasets are the same and equal $\sigma = 0.1$, $k = 7$, min_size = 10.

3) Multitemporal Change Detection: When the preliminary change maps with bitemporal contextual anomalies are obtained, we move on to their analysis in multitemporal context. t_{int} should have the same value as τ_1, because both parameters define the level of spatial relation between segments at different timestamps. In other words, t_{int} defines if changes detected between different timestamps belong to the same change process and τ_1 defines if a candidate object belongs to a reference BB. If we set t_{int} smaller than τ_1 and a candidate change polygon is kept after multitemporal context analysis, it may not be attached to any BB during evolution graph construction. On the contrary, if t_{int} is set much higher than τ_1, it will produce mostly spatially compact change processes, so small τ_1 value is unnecessary.

For both datasets, we have set $t_{int} = 0.4$, as $\tau_1 = 0.4$ with corresponding α gave the best results comparatively to other parameters combinations for both datasets (explained in the next section). Logically, the performance of the proposed approach fully depends on segmentation quality. The approach was qualitatively validated using manually selected reference objects.

4) Evolution Graph Construction: As it was mentioned before, in this work, we tend to create evolution graphs that contain only the objects that belong to the same change process. In the previous works, α value varies between 0.3 and 0.5. Smaller α values provide more overlapping graphs and, on the contrary, if α is high, we may obtain low graph coverage ratio, hence, a lot of change processes will be missed. To obtain the optimal BB coverage and graph overlapping, we choose α values 0.4 and 0.5 for the Montpellier and Rostov datasets, respectively.

As every object can belong to only one evolution graph that ensures the maximum possible spatial overlay with corresponding BB, we propose to omit τ_2 value and to set τ_1 value excessively small. The smaller τ_1 is, the bigger graph footprint we might obtain. However, most evolution graphs are compact independently of τ_1 value. For both datasets, $\tau_1 = 0.4$ provided the best graph coverage. τ_3 value has to be chosen after visual analysis of detected changes and parasite objects that correspond to FP changes. Clearly, its value should not be too high as the footprint of change processes may have important variations at each timestamp. The parameters for evolution graph construction that gave us the best results are presented in Table VI.

5) Feature Extraction AE: To cluster the evolution graphs, it is necessary to extract robust descriptors of change objects. In our work, we deploy a deep convolutional AE model for nonsupervised feature extraction. We use the same model for both datasets (see Table II) with patch size $p = 9$ as it provides sufficient coverage of neighborhood pixels without high influence of objects border pixels. Contrary to the change detection algorithm, we keep all four spectral bands for the Montpelier dataset (Green, Red, NIR, and SWIR) as SWIR band may

ameliorate the clustering of some similar classes. We keep $B = 3$ for the Rostov dataset as in change detection. We encode the patches of the Montpellier and Rostov datasets in feature vectors of size 5 and 4, respectively. Small size of feature vectors was chosen to obtain only the most representative features. Moreover, feature vectors size is equal to the number of spectral bands plus 1 ($f = B + 1$) to avoid potential linear correlation between a spectral band and a corresponding element of the feature vector.

6) Evolution Graph Clustering: After the texture extraction, we create synopses for all evolution graphs that are then passed to the GRU AE model combined with a chosen clustering method. As we do not know beforehand the exact number of change classes presented in a SITS, we choose hierarchical agglomerative clustering algorithm.

Most of the existing clustering algorithms either demand a number of clusters as a parameter, or a set of parameters that will further define the clusters number for a given dataset. In most cases, this set of parameters is difficult to guess and varies from one dataset to another. Often the initialization of the algorithm depends on the chosen number of clusters, hence, clustering model may change drastically. At the same time, data clustering models built by the hierarchical agglomerative algorithm do not depend on a number of clusters as at every step of the algorithm execution, it iteratively merges the two clusters with the highest likelihood value. Given these arguments, hierarchical agglomerative clustering is a good compromise between clustering quality and model complexity and makes the best choice for our generic change clustering approach.

The model parameters were set as follows for both datasets: hidden_size $= 150$, f_hidden $= 20$. Both parameters were achieved empirically, though some general recommendations were followed to set the second parameter. For example, the size of AE bottleneck should correspond to the number of researched clusters [37], although, this number is not known beforehand, we can distinguish around 10–15 the most frequented types of change processes for both datasets. To improve the clustering results, the extracted sequences are normalized by their mean and standard deviation.

After the model is stabilized, we apply the hierarchical agglomerative clustering to the encoded results. As parameters for clustering algorithm, we use Ward's linkage [23] and Euclidean distance between the encoded points. We test the results for different number of clusters in the range from 5 to 50.

Unfortunately, the right choice of the number of clusters is still an open question. No robust methods for the automatic selection of the number of clusters were find in literature and for some datasets it seems impossible to guess it. However, we can visually estimate the number of prevailing change classes and set an optimal range of the number of clusters for the hierarchical clustering. Considering the specificity of the algorithm and that its model does not depend on the selected number of clusters, we do not need to know their exact number. Furthermore, in the case of this article, we consulted experts in cartography specialized in these areas that gave us an idea of what number of clusters and which cluster-class mapping should be expected. Our method

can easily be reused in other geographic areas, with the right number of clusters found by trials and errors, or better with the help of an expert in the specific type of landscape studied.

For both datasets, we have annotated the most frequented of detected change processes to evaluate the clustering results. Note that we do not dispose of the labels for different crops, so the vegetation is annotated accordingly to its variation in the detected change sequence (for example, vegetation-weak vegetation-empty field-vegetation) and that we do not know the exact number of change process classes. For the Montpellier dataset, the annotated change processes that were regrouped in ten classes are:

1) five different vegetation variations (3644, 4429, 4307, 2542, and 2206 pixels per annotated class);
2) changes in deep water (8164 pixels);
3) construction in a dense area (4122 pixels);
4) construction of a new area (13648 pixels);
5) beginning of the construction process (bare soil/construction field) (2196 pixels);
6) changes in buildings luminosity (FP changes) (501 pixels).

For the Rostov dataset, we got 11 classes:

1) five different vegetation variations (2864, 16930, 14805, 7521, and 6183 pixels);
2) construction of a new building (969 pixels);
3) construction of a new area (dense construction) (13669 pixels);
4) construction of a new residential area (sparse small private buildings) (241 pixels);
5) changes in bare soil (2201 pixels);
6) seasonal changes in trees/bushes growing areas (8970 pixels);
7) shadow from tall buildings (FP changes) (3708 pixels).

To estimate the quality of clustering, we use normalized mutual information index [38].

B. Results

In this section, we present quantitative and qualitative results obtained with the proposed framework. We start by presenting the intermediate results of the framework, then we present the final results obtained by the clustering algorithm and discuss the influence of the parameters of the intermediate algorithms on the final results.

We start by bitemporal change detection that defines the final overall accuracy of our framework. The more FP changes there are—the more parasite objects or even parasite graphs are detected. For this reason, during the next steps, we tend to reduce their quantity by using different techniques (for example, by introducing τ_3 value).

The performance of the algorithm for nontrivial bitemporal change detection (permanent changes and vegetation that do not follow the overall tendency) is presented in Table VII. More details about the algorithms performance and its comparison to the state-of-art methods can be consulted in [24]. The advantage of our method is that comparatively to the state-of-the-art algorithms, our method is less sensitive to the changing luminosity (especially in the urban area), hence, produces less FP change

Dataset	Classification performance			
	Precision	Recall	Kappa	Time[a], min
Montpellier	0.72	0.79	0.73	15+8
Rostov	0.60	0.82	0.64	20+10

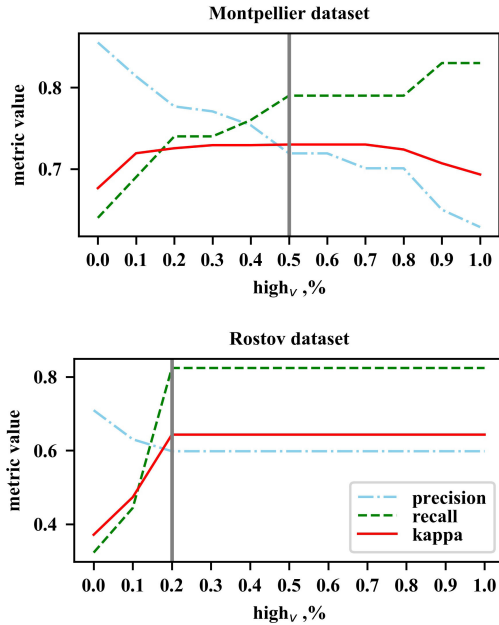[a]Pretraining+fine-tuning per couple without parallelization.

Fig. 8.   Influence of $high_v$ on the recall, precision, and kappa metrics values for the Montpellier and Rostov datasets. The vertical line corresponds to the best $high_v$ for our ground truth data.

alarms. It equally gives us change objects of more "natural" shape, as we perform change detection for every pixel of the image separately by associating it to its neighborhood patch. We observe that both datasets have high recall values that indicate that most of the changes are correctly detected. At the same time, precision value is low for the Rostov dataset and depends mostly on the quality of the images itself (a lot of saturated objects and shadows are detected as changes).

As it was mentioned in the previous section, we set $high_v = 0.5\%$ for our method to balance the precision and recall values. The influence of $high_v$ value on the precision, recall, and kappa values is presented on Fig. 8. It was decided that the optimal threshold $high_v$ should provide the results with recall slightly better than precision, as some FP changes will be deleted during the next steps. We observe that for the Montpellier dataset $high_v = 0.5\%$ gives us the best result. At the other hand, we notice different behavior of metrics values for the Rostov dataset ground truth and we can state less smooth changes in these values. We equally state that the best results are obtained for $high_v \geq 0.2\%$. Therefore, it was decided to set constant $high_v = 0.5\%$ for our method to ensure the most optimal results for both datasets.

Once the obtained change maps are analyzed in multitemporal context, we move on to the construction of the evolution graphs.

Table VIII presents the evolution graph statistics for the chosen parameters that ensure the best coverage. It provides total graph coverage, graph overlapping percentage, minimum graph compactness represented by the ratio of BB footprint to the total graph coverage footprint, dataset average graph compactness, percentage of graphs with compactness inferior to 50% and 75% (to justify the choice of small τ_1 value).

Fig. 9 presents an example of an evolution graph that corresponds to the construction of a rugby stadium. The constructed graph correctly reflects the overall change process, although, we can observe that some extracted changes and corresponding segments are not "pure." For example, in timestamp February 18, 2006, some small vegetation objects are attached to larger construction segments. We can also notice that the presented change process is composed of several subprocesses. One of these subprocesses starts at June 03, 2006 and at this timestamp is presented by segment 7–29 that corresponds to some vegetation fields that are later transformed into constructions.

With the given evolution graph parameters, we obtain 4388 graphs for the Montpellier dataset and 1850 graphs for the Rostov dataset (after excluding two timestamps length sequences) that are used in totality for the AE model training.

To evaluate the proposed clustering framework, we compare it with hierarchical agglomerative clustering with DTW [19] distance measure used in [17] that is equally able to process time-series data with varying length. DTW is a time-series distance measure algorithm that finds optimal match between two time series by scaling one to another. Ward's linkage was used as clustering algorithm parameter. DTW was applied to the extracted graph synopsis calculated for encoded features (called "DTW features") and to the raw image values (called "DTW w/o features"). We highlight the advantage of our proposed framework by comparing it with the same framework without feature extraction (i.e., we compute the synopsis for raw image values, called "Framework w/o features"). The obtained results are presented in Table IX for different number of clusters in order to choose the optimal one (the best results are given in bold).

For both datasets, we equally calculate NMI for primary change classes such as changes in vegetation, construction processes, and changes in water (only for the Montpellier dataset) to verify if the extracted clusters contain only one primary type of changes (see Table X).

The execution time of GRU AE combined with hierarchical agglomerative clustering and hierarchical agglomerative clustering with DTW distance matrix measure are presented in Table XI. Computation of DTW distance matrix is a time-consuming process. hence, with growing number of change graphs, the computation time may drastically increase. At the same time, GRU AE training takes the same time for both datasets, regardless the number of graphs.

We observe that for small number of classes (10–15) our proposed framework gives the best results for the reference ground truth (10 and 11 classes for Montpellier and Rostov datatsets, respectively).

However, for the Montpellier dataset, hierarchical agglomerative clustering with DTW measure outperformed our framework for the number of clusters superior to 15. At the same time,

TABLE VIII
STATISTICS ABOUT EVOLUTION GRAPHS CONSTRUCTION

	Statistics, %						Time, min	
Dataset	Cove-rage	Over-lapping	Min. comp.	Avg. comp.	Comp. <50%	Comp. <75%	BB selection	Graph constr.
Montpellier	96	14	30	94	0.4	7.5	2.5	9.5
Rostov	92	9	34	93	0.7	9.1	7.0	14.5

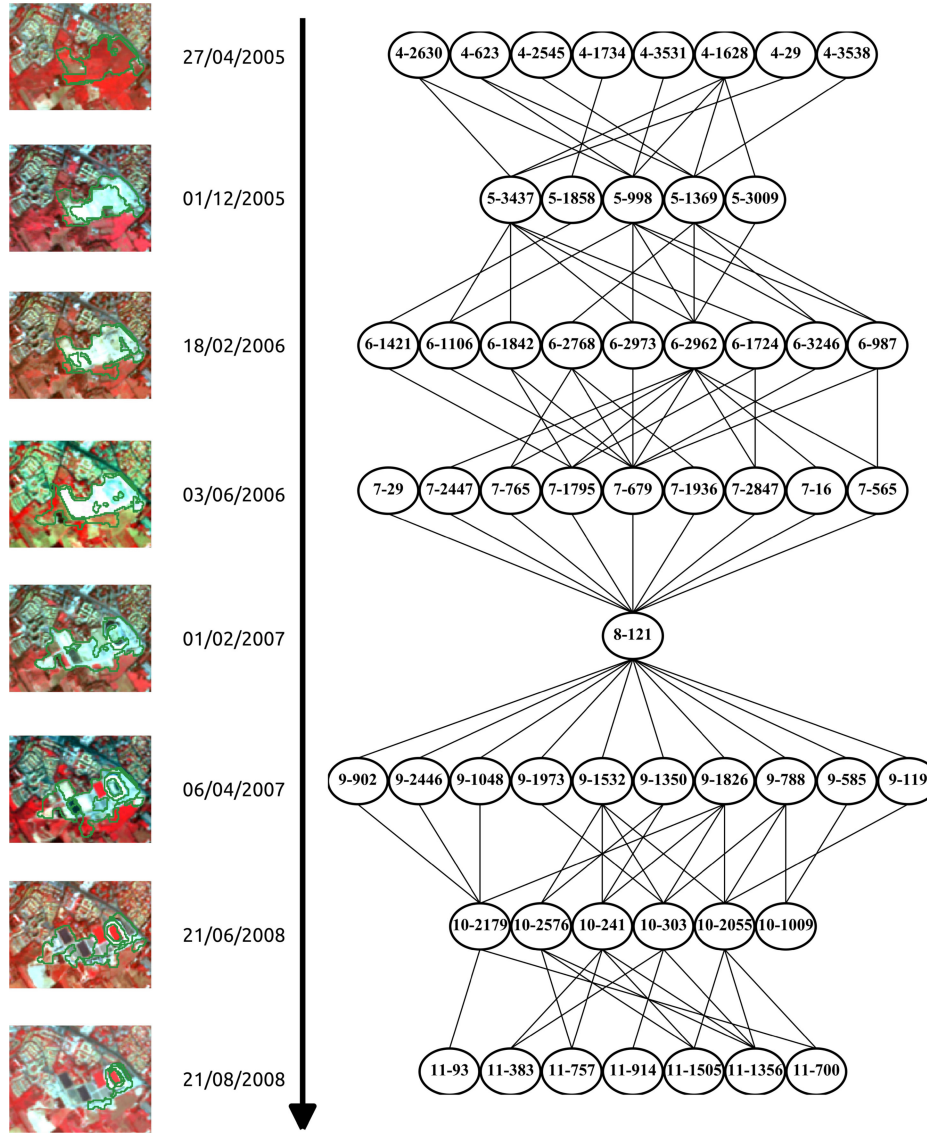Fig. 9. Evolution graph of rugby stadium construction.

TABLE IX
CLUSTERING RESULTS FOR REFERENCE CHANGE CLASSES

		NMI									
		Number of clusters									
	Method	5	10	15	20	25	30	35	40	45	50
Montpellier	Our framework	0.5	**0.57**	**0.55**	0.55	0.55	0.56	0.56	0.57	0.59	0.59
	Framework w/o features	**0.53**	0.56	0.53	0.54	0.52	0.54	0.55	0.55	0.56	0.56
	DTW features	0.43	0.48	0.49	0.51	0.51	0.56	0.6	0.61	**0.61**	**0.65**
	DTW w/o features	**0.53**	0.51	**0.55**	**0.58**	**0.58**	**0.59**	**0.6**	**0.62**	0.6	0.61
Rostov	Our framework	**0.56**	0.55	**0.64**	**0.65**	0.64	0.63	0.7	0.71	0.71	0.71
	Framework w/o features	0.36	0.48	0.52	0.62	0.63	0.65	0.69	0.69	0.71	0.71
	DTW	0.33	0.45	0.58	0.64	**0.67**	**0.71**	**0.71**	**0.73**	**0.73**	**0.73**
	DTW w/o features	0.48	**0.55**	0.56	0.58	0.62	0.62	0.66	0.66	0.67	0.67

TABLE X
CLUSTERING RESULTS FOR PRIMARY CLASSES

	Method	NMI									
		Number of clusters									
		5	10	15	20	25	30	35	40	45	50
Montpellier	Our framework	**0.65**	**0.73**	**0.65**	**0.64**	**0.64**	**0.64**	**0.61**	**0.6**	**0.62**	**0.62**
	Framework w/o features	0.62	0.58	0.53	0.52	0.5	0.49	0.49	0.49	0.47	0.47
	DTW	0.54	0.51	0.5	0.48	0.48	0.48	0.51	0.51	0.51	0.5
	DTW w/o features	0.55	0.52	0.54	0.59	0.59	0.58	0.57	0.56	0.55	0.56
Rostov	Our framework	0.44	0.36	0.34	0.36	0.35	0.35	0.33	0.34	0.34	0.34
	Framework w/o features	0.27	0.27	0.34	0.31	0.32	0.31	0.3	0.3	0.34	0.34
	DTW	0.27	0.29	0.31	0.29	0.38	0.37	0.38	**0.4**	0.4	0.4
	DTW w/o features	**0.5**	**0.43**	**0.41**	**0.43**	**0.42**	**0.42**	**0.4**	**0.4**	**0.41**	**0.41**

TABLE XI
COMPUTATION TIME

Method	Time, min	
	Montpellier	Rostov
GRU AE + hier.	**4**	**3**
Hier. with DTW	16	4

primary classes of the Montpellier dataset are much better separated by our framework, even for small number of clusters. In general, for both datasets, NMI increases with the number of clusters, but at the same time, it may be complicated to interpret the obtained results when its number is elevated.

In our case, as the NMI is calculated at pixel and not object level, we obtain high NMI values for 11 reference classes of the Rostov dataset due to the bigger surface of vegetation changes. As DTW measure does not depend on the classes balance, it has achieved better separation of primary classes for the unbalanced dataset. Nevertheless, for large datasets, DTW computation is time consuming, so GRU AE is the most adapted for balanced datasets of any size, and DTW approach performs better for unbalanced datasets of a small size.

After visual analysis of the obtained results and analysis of Tables IX and X, we can state that for small number of clusters our proposed framework separates the primary classes much better than the framework without feature extraction. It confirms that feature extraction ameliorates the class generalization for GRU, so the primary classes are better distinguished.

We observe that some construction processes that belong to the same classes for human eye are regrouped by the algorithm in different clusters. This fact justifies the necessity to extract large objects during the segmentation as oversegmentation may produce numerous change graphs that correspond to the same change process, but are clustered in different groups. However, visual examination of the results with chosen segmentation parameters shows that some graphs contain several different change subprocesses. For this reason, we may find different classes of objects at the same timestamps in one evolution graph (as in Fig. 9). This class mixture may potentially influence the graph synopsis and lower the performance of our clustering algorithms.

If we lower the threshold k or increase σ values for the image segmentation, we obtain smaller segments that give us more compact evolution graphs. At the same time, the computation time for evolution graph construction will increase with the number of obtained graphs. In this case, we may obtain numerous graphs that correspond to some subprocesses that will decrease the clustering performances due to the lack of generalization of the obtained change graphs. For example, if we change only the segmentation parameters for the Montpellier dataset to $\sigma = 0.3$ and $k = 6$, we obtain 5316 evolution graphs that are clustered with lower precision. For example, for 15 clusters, we obtain a NMI of 0.49 and 0.5 for 10 classes and primary classes, respectively, contrary to 4388 graphs and 0.55 and 0.65 NMI values for the initially chosen segmentation parameters. On the other hand, if we increase the threshold k or lower σ values, we might obtain undersegmented change objects that will lead to the creation of less "pure" evolution graphs.

The influence α, τ_1, and τ_3 values on the evolution graph construction was explained in Experimental settings section. Clearly, we can attend similar effect as in previous paragraph if the number of graphs is elevated and, on the contrary, miss some change processes if the graph coverage is not sufficient.

V. CONCLUSION

In this article, we have presented an end-to-end approach for change detection and analysis in SITS. The approach consists in the extraction of bitemporal change maps for the whole SITS that are then analyzed in multitemporal context in order to construct different change processes that are further regrouped in different clusters. Our framework combines graph-based techniques with unsupervised feature learning with neural networks and does not depend on the temporal resolution of SITS and on its length. The proposed approach is fully unsupervised and gives us promising results on two real-life datasets. However, for unbalanced datasets, we may observe that smaller classes are not well separated from the majority ones.

Later, we are planning to work on more robust graph construction and interpretation. The graphs constructed with our method make the global description of the detected change process without considering that it may potentially contain several subprocesses. We are also looking forward to improve the algorithm for multitemporal interpretation of detected changes by the direct integration of the filtering of parasite objects that at the moment is performed at the graph construction stage.

References

[1] M. Lopes, M. Fauvel, A. Ouin, and S. Girard, "Spectro-temporal heterogeneity measures from dense high spatial resolution satellite image time series: Application to grassland species diversity estimation," *Remote Sens.*, vol. 9, no. 10, 2017, Art. no. 993.

[2] X.-P. Song, C. Huang, J. O. Sexton, S. Channan, and J. R. Townshend, "Annual detection of forest cover loss using time series satellite measurements of percent tree cover," *Remote Sens.*, vol. 6, no. 9, pp. 8878–8903, 2014.

[3] L. P. Dutrieux, J. Verbesselt, L. Kooistra, and M. Herold, "Monitoring forest cover loss using multiple data streams, a case study of a tropical dry forest in bolivia," *ISPRS J. Photogrammetry Remote Sens.*, vol. 107, pp. 112–125, 2015.

[4] Z. Zhu, C. E. Woodcock, and P. Olofsson, "Continuous monitoring of forest disturbance using all available landsat imagery," *Remote Sens. Environ.*, vol. 122, pp. 75–91, 2012.

[5] M. Van Hoek, L. Jia, J. Zhou, C. Zheng, and M. Menenti, "Early drought detection by spectral analysis of satellite time series of precipitation and normalized difference vegetation index (NDVI)," *Remote Sens.*, vol. 8, no. 5, 2016, Art. no. 422.

[6] S. Olen and B. Bookhagen, "Mapping damage-affected areas after natural hazard events using sentinel-1 coherence time series," *Remote Sens.*, vol. 10, no. 8, 2018, Art. no. 1272.

[7] T. Lei, Q. Zhang, D. Xue, T. Chen, H. Meng, and A. K. Nandi, "End-to-end change detection using a symmetric fully convolutional network for landslide mapping," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2019, pp. 3027–3031.

[8] A. Alqurashi, L. Kumar, and P. Sinha, "Urban land cover change modelling using time-series satellite images: A case study of urban growth in five cities of saudi arabia," *Remote Sens.*, vol. 8, 2016, Art. no. 838.

[9] S. Pareeth, P. Karimi, M. Shafiei, and C. De Fraiture, "Mapping agricultural landuse patterns from time series of landsat 8 using random forest based hierarchial approach," *Remote Sens.*, vol. 11, no. 5, 2019, Art. no. 601.

[10] U. Kanjir, N. Durić, and T. Veljanovski, "Sentinel-2 based temporal detection of agricultural land use anomalies in support of common agricultural policy monitoring," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 10, 2018, Art. no. 405.

[11] R. Interdonato, D. Ienco, R. Gaetano, and K. Ose, "DuPLO: A DUal view Point deep Learning architecture for time series classificatiOn," *ISPRS J. Photogrammetry Remote Sens.*, vol. 149, pp. 91–104, Mar. 2019.

[12] C. Pelletier, G. I. Webb, and F. Petitjean, "Temporal convolutional neural network for the classification of satellite image time series," *Remote Sens.*, vol. 11, no. 5, 2019, Art. no. 523.

[13] Y.-L. Kong, Q. Huang, C. Wang, J. Chen, J. Chen, and D. He, "Long short-term memory neural networks for online disturbance detection in satellite image time series," *Remote Sens.*, vol. 10, no. 3, 2018, Art. no. 452.

[14] J. Verbesselt, R. Hyndman, G. Newnham, and D. Culvenor, "Detecting trend and seasonal changes in satellite image time series," *Remote Sens. Environ.*, vol. 114, no. 1, pp. 106–115, 2010.

[15] S. Cai and D. Liu, "Detecting change dates from dense satellite time series using a sub-annual change detection algorithm," *Remote Sens.*, vol. 7, no. 7, pp. 8705–8727, 2015.

[16] C. Vaduva, C. Danisor, and M. Datcu, "Joint SAR image time series and psinsar data analytics: An LDA based approach," *Remote Sens.*, vol. 10, 2018, Art. no. 1436.

[17] L. Khiali, M. Ndiath, S. Alleaume, D. Ienco, K. Ose, and M. Teisseire, "Detection of spatio-temporal evolutions on multi-annual satellite image time series: A clustering based approach," *Int. J. Appl. Earth Observ. Geoinform.*, vol. 74, pp. 103–119, 2019.

[18] F. Guttler, D. Ienco, J. Nin, M. Teisseire, and P. Poncelet, "A graph-based approach to detect spatiotemporal dynamics in satellite image time series," *ISPRS J. Photogrammetry Remote Sens.*, vol. 130, pp. 92–107, 2017.

[19] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, M. B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping," vol. 7, 2013, Art. no. 10.

[20] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[21] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *Int. J. Mach. Learn. Cybern.*, vol. 1, pp. 43–52, 2010.

[22] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Oct. 2014, pp. 1724–1734.

[23] J. H. Ward Jr, ,"Hierarchical grouping to optimize an objective function," *J. Amer. Statist. Assoc.*, vol. 58, no. 301, pp. 236–244, 1963.

[24] E. Kalinicheva, J. Sublime, and M. Trocan, "Change detection in satellite images using reconstruction errors of joint autoencoders," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Image Processing*, I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, Eds. Cham, Switzerland: Springer, 2019, pp. 637–648.

[25] J. Sublime and E. Kalinicheva, "Automatic post-disaster damage mapping using deep-learning techniques for change detection: Case study of the Tohoku tsunami," *Remote Sens.*, vol. 11, 2019, Art. no. 1123.

[26] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

[27] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS One*, vol. 11, 2016, Art. no. e0152173.

[28] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, Sep. 2004.

[29] L. Prechelt, *Early Stopping — But When?* Berlin, Germany: Springer, 2012, pp. 53–67.

[30] E. Ndikumana, D. Ho Tong Minh, N. Baghdadi, D. Courault, and L. Hossard, "Deep recurrent neural network for agricultural classification using multitemporal SAR Sentinel-1 for camargue, france," *Remote Sens.*, vol. 10, no. 8, 2018, Art. no. 1217.

[31] H. Luo, "Shorten spatial-spectral RNN with parallel-GRU for hyperspectral image classification," 2018, *arXiv:1810.12563*.

[32] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, Jul. 2017.

[33] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 4, Sep. 2014.

[34] M. El Hajj, A. Bégué, B. Lafrance, O. Hagolle, G. Dedieu, and M. Rumeau, "Relative radiometric normalization and atmospheric correction of a SPOT 5 time series," *Sensors*, vol. 8, no. 4, pp. 2774–2791, 2008.

[35] R. Tavenard *et al.*, "Tslearn: A machine learning toolkit dedicated to time-series data," 2017. [Online]. Available: https://github.com/rtavenar/tslearn

[36] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[37] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1753–1759.

[38] D. Ienco, R. Pensa, and R. Meo, "From context to distance: Learning dissimilarity for categorical data clustering," *ACM Trans. Knowl. Discov. Data*, vol. 6, pp. 1:1–1:25, 2012.

Ekaterina Kalinicheva received the engineering degree in applied geodesy from the Moscow State University of Geodesy and Cartography, Moscow, Russia, in 2012 and the master's degree in geomatics from the University of Montpellier, Montpellier, France, in 2017. She is currentlyworking toward the Ph.D. degree in tnformatics with Pierre & Marie Curie University (defense programmed for the end of 2020), Paris, France.

She is currently working with LISITE laboratory, ISEP, Paris, France. Her Ph.D. topic is related to unsupervised analysis of satellite image time series, in particular, multitemporal change detection and multi-temporal clustering. Her main research interests include machine learning, image analysis, time series analysis, and unsupervised neural networks applied to different remote sensing applications.

Dino Ienco received the M.Sc. and Ph.D. degrees in computer science both from the University of Torino, Torino, Italy, in 2006 and 2010, respectively.

He joined the TETIS Laboratory in 2011 as a Junior Researcher. His main research interests include machine learning, data science, and spatiotemporal data mining with a particular emphasis on the analysis of Earth Observation data. Dr. Ienco served in the program committee of many international conferences on data mining, machine learning including IEEE ICDM, ECML PKDD, ACML, IJCAI as well as served as a Reviewer for many international journals in the general field of data science and remote sensing including *Machine Learning journal*, *Data Mining and Knowledge Discovery*, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, *ISPRS Journal of Photogrammetry and Remote Sensing* of Environment.

Jérémie Sublime received the Engineer degree in software engineering & computer vision from the EISTI Cergy, Cergy, France, the master's degree in computer and information technologies from INHA University, South Korea, both in 2013 and the Ph.D. degree in applied computer science from Paris-Saclay University (AgroParisTech) in November 2016.

Since September 2016, he has been an Associate Professor with ISEP, where he is a member of the DaSSIP Team (Data Science, Signal, and Image Processing), and the Head of the Data Intelligence Major. He is also a Full Researcher with the LIPN–CNRS UMR 7030, where he is a member of the A3 team specialized in machine learning. His research activities are focused on machine learning and data science problems, and cover a wide spectrum of issues, ranging from unsupervised learning to image processing and change detection. His work can be divided into three main axis: First, Applications of Unsupervised Learning in the field of image processing, with a particular interest for satellite and medical imaging. Second, Unsupervised Learning with a particular focus on new clustering paradigms such as multiview, collaborative, and alternative clustering. Third, Unsupervised neural networks such as Autoencoders coupled with various deep learning architectures (CNN, GRU, etc.), and Kohonen-based neural networks.

Maria Trocan received the M.Eng. degree in electrical engineering and computer science from the Politehnica University of Bucharest, Bucharest, Romania, in 2004, the Ph.D. degree in signal and image processing from Telecom Paris-Tech, Paris, France, in 2007, and the Habilitation to Lead Researches (HDR) from Pierre & Marie Curie University (Paris 6), Paris, France, in 2014.

She joined Joost, Netherlands in 2007, where she worked as Research Engineer involved in the design and development of video transcoding systems. Since May 2009, she has been an Associate Professor, then Professor with the Signal, Image, and Telecommunications Department, Institut Superieur d'Electronique de Paris (ISEP). Between October 2011 and February 2018, she is responsible of the Signal Processing Graduate Program and since March 2018, she has been the charge of the Data Science Program at ISEP. She is currently serving as Reviewer for several major IEEE journals and conferences and was involved in various conference organization and program committees. Her current research interests focus on image and video analysis and sparse signal representations.

Dr. Trocan is Deputy Editor in Chief for IEEE Open Journal on Circuits and Systems, Associate Editor for *Springer Journal on Signal, Image and Video Processing* and Guest Editor for several journals *(Analog Integrated Circuits and Signal Processing*, IEEE Communications Magazine, etc.). Since 2010, she is an Active Member of IEEE France and served as Counselor for the ISEP IEEE Student Branch. In 2014, she has been elected IEEE France Vice-President responsible of Student Activities and IEEE Circuits and Systems Board of Governors member, as Young Professionals representative. She has been re-elected as IEEE France Vice-President responsible of Student Activities in 2017 and Chair of Technical Activities and Membership Development in 2020.