A Deep Learning Approach for Change Points Detection in InSAR Time Series

Francesco Lattari, Alessio Rucci, and Matteo Matteucci

***Abstract*—Interferometric SAR (InSAR) algorithms exploit synthetic aperture radar (SAR) images to estimate ground displacements, which are updated at each new satellite acquisition, over wide areas. The analysis of the resulting time series finds its application, among others, in monitoring tasks regarding seismic faults, subsidence, landslides, and urban structures, for which an accurate and timely response is required. Typical analyses consist of identifying among the numerous time series the ones that exhibit an anomalous displacement, thus deserving to be further investigated. In practice, this is realised by selecting the time series which are characterised by trend changes w.r.t. the historical behaviour. In this work, we propose a Deep Learning approach for change point detection in InSAR time series. The designed architecture combines Long Short-Term Memory (LSTM) cells, to model the temporal correlation among samples in the input time series, and Time-Gated LSTM (TGLSTM) cells, to consider the sampling rate as additional information during learning. We further propose a solution to the lack of ground truth by developing a suitable pipeline for realistic data simulation. The method has been developed and validated through a large suite of experiments. Both quantitative and qualitative analyses have been conducted to demonstrate the detection capabilities of the learned model and how it is a valid alternative to the statistical reference algorithm. We further applied the developed method in a real continuous monitoring project to analyse InSAR time series over the Tuscany region in Italy, proving its effectiveness in the real domain.**

***Index Terms*—InSAR; Deep Learning; LSTM.**

I. Introduction

In the last years, there has been a large increase in applications based on the analysis of remote sensing data. In particular, advances in technology have led to a new era in the field of Earth observation (EO). Satellites equipped with Synthetic Aperture Radar (SAR) systems paved the way for the research of advanced algorithms to exploit the information contained in the data acquired from space. Among them, interferometric SAR (InSAR) techniques [1], [2] exploit the revisiting time of the sensors to perform a spatio-temporal analysis of the Earth surface to measure, with millimeter accuracy, ground deformations over time. The final output of this analysis is a collection of time series representing the ground displacements of measurement points (MPs), or scatterers, on the surface during the observed period. In the last decade, InSAR data increasingly acquired interest for the numerous monitoring applications for which it represents a very valuable source of information, e.g., landslides [3], volcanic activities [4], seismic faults [5], and others.

F. Lattari and M. Matteucci are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milano, Italy e-mail: (francesco.lattari@polimi.it).

A. Rucci is with the TRE ALTAMIRA s.r.l., 20143 Milano, Italy.

In this context, the European Space Agency (ESA) Sentinel-1 mission turned out to be a game changer for the EO community. Deployed in the framework of the EC Copernicus Programme, this constellation of two twin platforms (A and B) is capable of providing wide-scale, systematic, up-to-date, and access-free satellite radar data over Europe, with 6-day revisiting time in both ascending and descending orbits. Permanent scatterers interferometry (PSInSAR) [6], [7] and its enhanced development SqueeSAR™ [8] exploit this huge availability of SAR images to periodically measure the surface deformation through the identification of coherent targets, i.e., *permanent* and *distributed* scatterers exhibiting good phase stability over the whole time period of observation. Distributed scatterers are also considered in other InSAR algorithms like [9] and [10]. A typical analysis of a SAR interferogram consists of identifying among the huge number of MPs the ones exhibiting displacement time series characterised by a trend change, which can be further investigated. This data screening phase is extremely important to support the end-users in the exploitation of frequently updated (every few days) information layers. Nevertheless, the large volume of InSAR data, which increases of hundreds of thousands, or even millions, time series each update, remains an important limitation to its operational use.

An answer to this problem was given in [11], in which a computationally efficient method is introduced to estimate kinematic parameters from interferometric data to monitor the behaviour of points on Earth. This is done by applying multiple hypotheses testing (MHT) to find the optimal model describing the observations, given a library of physically realistic deformation models. In [12] authors focus on the application of polynomials, by introducing a statistical test to rank polynomial approximations for MTInSAR time series, thus finding the ones with the minimum number of parameters. In [13] a model-free method is instead proposed to characterise and select meaningful time series. A different statistical approach is proposed by [14], in which the objective is instead to detect in each time series all those points in correspondence of which a trend change occurs. These points are defined as change points, and the considered task is usually referred to as change points detection or trend change detection [15], [16]. Section II better describes the approach used in [14], which is considered as the baseline in our work, since the proposed method shares the same objective. In the specific, we are interested in identifying three possible classes of change points, depending on the kind of the associated trend change. In correspondence of *step* change points, an abrupt change w.r.t. the historical trend is observable, while keeping a constant average displacement rate. In the case of *velocity*

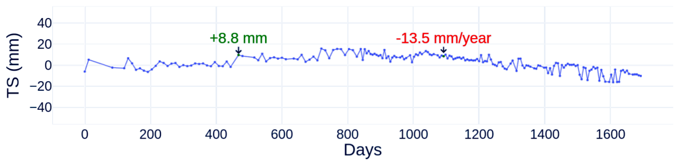


Fig. 1. Example of change points in InSAR time series. In the plot the displacements time series. On the y axis the ground deformation expressed in millimeters (mm), while on the x axis the days related to the corresponding SAR image acquisitions. In the example two trend changes occurs in correspondence of the two coloured points: the green one is related to a step change point of $+8.8$ mm; the red one is related to a velocity change point of -13.5 mm/year.

change points, the sequence undergoes a variation of the average displacement rate, without any abrupt displacement. Finally, the third class includes points where both step and velocity changes occur. Figure 1 shows an example of velocity and step change points.

The identification of change points is not a simple task due to the noisy nature of InSAR time series, resulting from residual errors produced during the estimation process. The main sources of noise are changes in the acquisition geometry, i.e., geometrical decorrelation, changes in the location of the scatterers, i.e., temporal decorrelation, and phase artefacts due to the atmospheric effects. Additional challenges come from the presence of seasonalities in the time series, which have to be estimated and removed from the original data before applying any detection algorithm, and from the variable lack of acquisitions, which produces non-uniformly sampled sequences.

As discussed in Section Section II, statistical approaches have been efficiently applied for detecting change points in InSAR time series. However, they are based on multiple hyper-parameters, which are properly tuned depending on the specific application and the requirements of the final users. Our work shares the same objective of providing a timely and automatic tool of analysis, but we take advantage of deep learning (DL) techniques to *learn* a change point detector, without the need to define any set of possible displacement models. The capability of neural networks of learning patterns from data, coupled with increasingly powerful computing technologies, has made the DL a winning approach in many fields of the computer vision and, in the last years, we are witnessing a growth of DL approaches also in numerous EO applications.

In this work we focused on Recurrent Neural Networks (RNNs), which allow modelling the temporal correlation among samples in sequential data by introducing the concept of *memory* during the learning process. RNNs allow extracting features from data based on the understanding of the information contained in past observations. Indeed, these architectures are well suited to solve problems in which a time dependency among samples exists. This is the case, for instance, of machine translation [17], speech recognition [18], or sounds generation [19]. We demonstrate that RNNs are also suitable for analysing InSAR data and, in particular, that are a valuable approach to the specific task of detecting change points. Other works tried to solve similar tasks in very different domains such as anomaly detection in sensor signals from mechanical devices [20], anomaly detection in multi-modal sensory signals in robot-assisted feeding systems [21], cyber-attacks identification in cyber-physical systems [22], or anomaly detection in electrocardiogram (ECG) signals [23]. In the considered task, the goal is to identify all the trends in InSAR data, which can be characterised by a huge variety of dynamics.

In the proposed work, a recurrent classification network has been developed and trained to discriminate between normal points (class 0), i.e., internal to a trend, and change points (class 1), which are points straddling two consecutive trends. As described in Section III, the designed network combines Long Short-Term Memory (LSTM) cells [24], [25], which allow dealing with the vanishing gradients problem [26], and Time-Gated LSTM (TGLSTM) cells [27], which are a modified version of standard LSTM cells allowing to learn from non-uniformly sampled time series, which is a characteristic of InSAR data. Indeed, the time that elapses between measurements in InSAR time series can be irregular due to the lack of satellite acquisitions because of satellite manoeuvres, conflicts with high priority requests, temporary failure in the radar sensors, or in the downlink system.

One of the main problems to face when dealing with SAR data is the lack of ground truth [28], which makes it difficult both training the model and validating the results. Indeed, this is one of the challenges we tackled in the conducted work, since the huge amount of InSAR time series makes it impossible to perform any manual labelling. Accordingly, we resorted to data simulation to generate both training and validation data. This allows to obtain accurate targets, which is advantageous not only for training, but also for having a reliable evaluation of the results during tests. Furthermore, simulation allows to enrich the training set with a huge variety of scenarios, allowing for higher generalisation capabilities. To fully take advantage of simulation, it is required that the simulated data has to be as similar as possible to real data. For this purpose, we conducted our research closely with InSAR specialists from the industry, which supported us during both the process of data generation and the validation of the obtained results when dealing with real InSAR data, for which specific knowledge and experience is required.

The rest of the article is organised as follows. In Section II we give an overview of the baseline method considered as the golden standard. In section III we present the proposed method, by detailing the designed architecture. In Section IV we describe the suite of experiments we built to train the proposed network, by highlighting the data simulation (IV-A), which is further detailed in Appendix, and the employed training procedure (IV-B). In Section V we prove the validity of the proposed method, both quantitatively and qualitatively, by testing the learned model on a large amount of test data. We demonstrate the capability of the proposed method to identify the change points both in simulated time series, which have been used to perform a quantitative analysis of the performance, but also in real sequences, which have been obtained by estimating displacement values from a stack of SAR acquisitions over the Tuscany region (central Italy). Furthermore, we report the application of the learned model over the Fernandina volcano area, at Galápagos Islands. Finally, we derive our conclusions in Section VI.

II. Reference Solution for InSAR Change Point Detection

In the following a description of the statistical algorithm we considered as the baseline for our experiments is provided. The method has been used in [14], where the authors apply a post-processing on the displacement time series obtained through the SqueeSAR™ algorithm to identify the change points. The detection is performed based on bayesian change point and variable selection algorithm [29]. Given a displacements time series, the algorithm estimates the probability of each possible sub-sequence to contain k change points, which is done through a recursive approach. In particular, the marginal probability density of the data is computed, which is

$$f(Y_{i:j}) = f(Y_{i:j}|X_{i:j}), \tag{1}$$

for $1 \leq i < j \leq N$, where $Y_{i:j}$ is a sub-sequence of the data and $X = [X_1, X_2, ..., X_m]$ is a sub-set of regressors of the regression model. Then, starting from the end of the time series, the probability of any prefix of data $Y_{1:j}$ to contain k change points is computed recursively as:

$$P_1(Y_{1:j}) = \sum_{v<j} f(Y_{1:v}) f(Y_{v+1:j}) \tag{2}$$

$$P_k(Y_{1:j}) = \sum_{v<j} P_{k-1}(Y_{1:v}) f(Y_{v+1:j}) \tag{3}$$

for $j = 1, 2, ..., N$. Finally, Bayes Rule is used to estimate both the number and the location of the change points from the posterior distribution. Given a prior distribution on the number of change points $P(K = k)$ and a prior distribution on the locations of the change points $P(c_1, c_2, ..., c_k|K = k)$, the number of change points is computed as

$$f(K = k|Y_{1:N}) = \frac{P_k(Y_{1:N}) P(c_1, ..., c_k|K = k) P(K = k)}{f(Y_{1:N})}, \tag{4}$$

and the uncertainty of the location of a change is given iteratively, being $c_{k+1} = N$ the last data point, as

$$f(c_k = v|c_{k+1}) = \frac{P_{k-1}(Y_{1:v}) f(Y_{v+1:c_{k+1}})}{\sum_{v\in[k-1,c_{k+1}]} P_{k-1}(Y_{1:v}) f(Y_{v+1:c_{k+1}})}, \tag{5}$$

for $k = K, k-1, ..., 1$.

One of the applications of this approach is in continuous monitoring projects, where displacement time series, for both ascending and descending geometries, are systematically analysed to detect changes in the last d days, e.g., 150 days. When a candidate change point is identified, with the procedure described so far, a breaking point T_b is defined and the average deformation rates are computed for each sub-sequence, i.e., v_1 in the time interval $T_0 - T_b$ and v_2 in the time interval $T_b - T_n$ are computed. Finally, if $|\Delta V| = v_2 - v_1 > \tau$ the point is selected as a change point, where τ is a manually defined threshold which depends on the kind of deformation changes we are interested in.

The algorithm is applied regularly every few days on hundreds of thousands of time series. These numbers refer, in general, to single sites, typically at a regional scale, e.g., thousands of square kilometres, but they become an order of magnitude larger if we imagine to extend the analysis at a national scale, or even more considering a worldwide processing. The increasing volumes of measurement points and the short time between subsequent updates make the statistical analysis inefficient for near-real-time monitoring, for which on-time deliveries are required by the end users. Also, this kind of analysis requires manual operators to adapt the algorithm hyper-parameters depending on the observed site and on the requirements. The proposed approach allows avoiding any manual setting of the model parameters, being the model directly learned from data. In addition, coupled with graphics processing units (GPUs), it allows to perform an high-throughput data processing with a flexible scale-up. Indeed, one of the objectives of the conducted research is to improve the exploitation of InSAR data, paving the way to large-scale applications of ground deformation analysis.

III. Proposed Method

Motivated by the performance reached by RNNs in many tasks involving the analysis of sequential data, we propose a deep recurrent neural network for the automatic detection of change points in InSAR time series. The developed network consists of multiple recurrent layers, which have been stacked to encode the input time series at different levels of abstractions. Each layer is realised through a recurrent unit, or *cell*, which implements a sort of *memory* that is updated at a certain time t based on the current input and the output at time $t-1$. Then, output values are produced based on the memory content, thus conditioning the current understanding of data on past observations.

Let R be the operation computed by a generic recurrent cell, then we have

$$h_t = R(x_t, h_{t-1}) \tag{6}$$

where $h_t \in \mathbb{R}^{N_h}$ denotes the cell output, or features, $x_t \in \mathbb{R}^{N_x}$ is the current input of the cell and $h_{t-1} \in \mathbb{R}^{N_h}$ represents the information coming from previous time step $t-1$ and it is typically called *hidden state*. Considering a multi-layer architecture and defining R^ℓ as the ℓ-th recurrent layer, the following holds

$$h_t^\ell = R_t^\ell(h_t^{\ell-1}, h_{t-1}^\ell), \qquad \ell > 1. \tag{7}$$

Thus, for the ℓ-th layer the input at time t corresponds to the output produced by the lower layer at the same time step. In the considered task, the input x_t of the first layer, i.e., $\ell = 1$, is a single sample from the input InSAR time series and thus we have $x_t \in \mathbb{R}$.

In this work, the time elapsed between displacements, which is measured in number of days, is further included as additional information during learning. In particular, in addition to the displacement value, the first recurrent layer takes as input $\Delta t_t \in \mathbb{N}$, which is the time interval between the sample at time t and the previous one at time $t-1$. Thus, for the first layer, the recurrent operator becomes

$$h_t^1 = R^1(x_t, \Delta t_t, h_{t-1}^1). \tag{8}$$

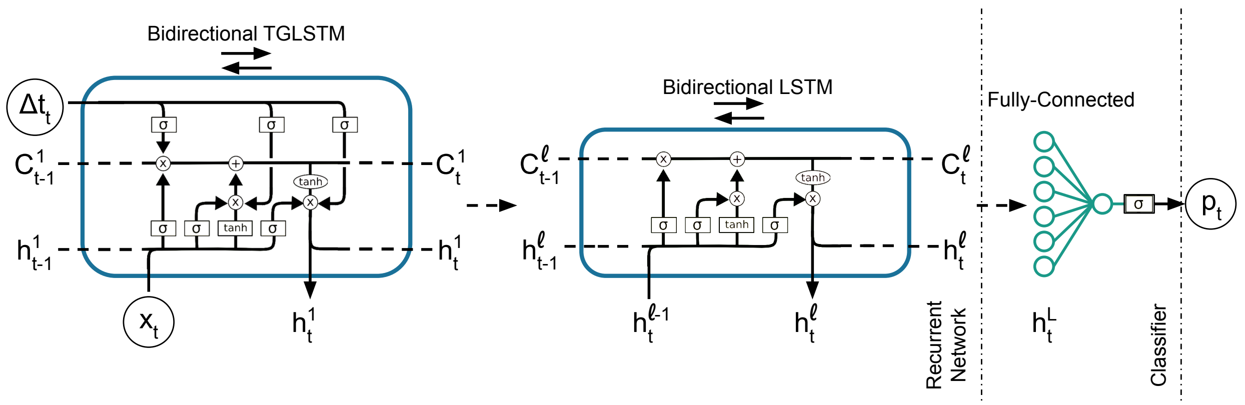


Fig. 2. Designed Neural Network. The recurrent network is composed by stacking a TGLSTM and $L-1$ LSTMs. It takes the displacements x and the sampling intervals between acquisitions Δt as input and produces a set of features as output. Then the classifier, realised through a fully-connected neural network, produces the final probability vector.

Recurrent operations are realised in this work through Long Short-Term Memory (LSTM) [24], [25] cells. An LSTM is composed by an internal *cell state* and a set of *gates*, which allow to read and write to the cell state and whose parameters are learned during training. At each time step t, an LSTM cell decides, based on both the current input and the output of the cell at time $t-1$, which parts of the cell state have to be erased through the *forget* gate, updates the cell state with new candidate values through the *input* gate, and produces output values through the *output* gate. The overall operations are described by the following equations

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \odot \tanh(C_t) \end{aligned} \tag{9}$$

where f_t is the output of the forget gate, i_t is the output of the input gate, $\tilde{C}_t$ are the new candidate values for the cell state, C_t and C_{t-1} are the new and old cell states respectively, o_t is the output gate, and h_t is the final output. W_f, W_i, W_C, W_o, b_f, b_i, b_C, and b_o are the learnable parameters, while σ and $\odot$ are the sigmoid function and Hadamard product respectively [24].

In order to use sampling intervals as additional information, the first layer is realised through Time-Gated LSTMs (TGLSTM) [27]. These units extend the LSTM cells with additional *time* gates to learn a time-dependent scaling function which modifies the response of forget, input and output gates. In particular, the following additional gates are implemented

$$\begin{aligned} \tau^f_t &= \sigma(W_{\tau^f} \cdot \Delta t_t + b_{\tau^f}) \\ \tau^i_t &= \sigma(W_{\tau^i} \cdot \Delta t_t + b_{\tau^i}) \\ \tau^o_t &= \sigma(W_{\tau^o} \cdot \Delta t_t + b_{\tau^o}) \end{aligned} \tag{10}$$

where τ^f_t, τ^i_t, and τ^o_t are the additional time gates which modify the activations of the forget, input and output gates, respectively, while W_{τ^f}, W_{τ^i}, W_{τ^o}, b_{τ^f}, b_{τ^i}, and b_{τ^o} are the corresponding learnable parameters [27]. The remaining operations are the same presented in Equation 9, with the addition of the scaling factors for each gate. Thus, we have

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t \odot \tau^f_t \odot C_{t-1} + i_t \odot \tau^i_t \odot \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \odot \tau^o_t \odot \tanh(C_t) \end{aligned} \tag{11}$$

Both TGLSTM and LSTMs are combined to build the final deep recurrent neural network, which is outlined in Figure 2. The architecture is composed by stacking L recurrent layers and a final classification layer implemented as a fully-connected (FC) network which, given the features $h^L_t \in \mathbb{R}^{N_h}$ extracted by the last recurrent layer at time step t, provides the probability p_t of the t-th point in the InSAR time series to be a change point. The first recurrent layer, i.e., $l = 1$, is realised through a TGLSTM cell, while subsequent layers are implemented through standard LSTM cells. Indeed, the network learns how to produce an output in the first layer based on both displacements and time intervals, and to hierarchically encode this abstract representation further in the following layers.

In the proposed architecture all the recurrent layers operate in a bidirectional way, i.e., there are two cells for each layer which analyse the input sequence forward and backward through time and the outputs of the two cells are concatenated to provide the final output. In this way the network is able to perform a prediction at time t based on the information coming both from past and future observations. Thus, for a bidirectional recurrent cell Equation 6 becomes

$$\begin{aligned} h_t &= R_{Bidirectional}(x_t, h_{t-1}, h_{t+1}) \\ &= R_{FW}(x_t, h_{t-1}) \oplus R_{BW}(x_t, h_{t+1}) \\ &= h^{FW}_t \oplus h^{BW}_t , \end{aligned} \tag{12}$$

where $\oplus$ is the concatenation operator and h^{FW}_t and h^{BW}_t are the features extracted at time step t in the two time directions [30], [31]. Please notice that in order to simplify the notation we denote with h both the hidden state of the forward pass, i.e., h_{t-1}, and the one of the backward pass, i.e., h_{t+1}, but they corresponds to the hidden states of the forward and backward cells, respectively. The same formulation holds when adding the additional input Δt_t. Thus, Equation 8 becomes

$$\begin{aligned} h_t &= R_{Bidirectional}(x_t, \Delta t_t, h_{t-1}, h_{t+1}) \\ &= R_{FW}(x_t, \Delta t_t, h_{t-1}) \oplus R_{BW}(x_t, \Delta t_t^{rev}, h_{t+1}) \\ &= h_t^{FW} \oplus h_t^{BW}, \end{aligned} \tag{13}$$

where Δt_t^{rev} is the time interval between samples at time t and $t+1$. Putting all together we have

$$\begin{aligned} h_t^1 &= TGLSTM_{Bidirectional}(x_t, \Delta t_t, h_{t-1}^1, h_{t+1}^1) \\ h_t^\ell &= LSTM_{Bidirectional}^\ell(h_t^{\ell-1}, h_{t-1}^\ell, h_{t+1}^\ell), \\ p_t &= FC(h_t^L) = \sigma(W_{FC} \cdot h_t^L + b_{FC}) \end{aligned} \tag{14}$$

for $1 < \ell \leq L$, where W_{FC} and b_{FC} are the parameters of the fully-connected layer. The output $p_t \in [0, 1]$ is the probability of input sample x_t to be a change point, produced through a sigmoid activation function σ. In the following sections we describe the employed datasets, with particular emphasis on the data simulation procedure, and the designed suite of experiments, together with the obtained results.

IV. Experiments Setup

In this section we describe the dataset we used during the conducted experiments, by giving an overview on the data simulation pipeline, and the training of the proposed network.

A. Data Simulation

The design of an automatic pipeline for data simulation has been an important stage of our work to overcome the lack of ground truth for InSAR time series. We started by generating a huge amount of time series with no trend changes, which, for the sake of simplicity, we define as *normal* time series. In particular, we collected the temporal baselines of SAR acquisitions from a high number of real InSAR datasets. Given the temporal baselines, we created noisy time series by modelling the noise distribution based on two kinds of possible targets which are point-wise scatterers (PSs) and distributed scatterers (DSs). Under the Gaussian scattering assumption based on the central limit theorem [2], SAR data vector can be described by a zero-mean, multi-dimensional, complex Gaussian p.d.f. Thus, for a complete statistical characterisation of a DS it is sufficient to know the covariance (or correlation) matrix. For a PS, which can be modelled as a strong dominant scatterer affected by a circular Gaussian noise, for SNR, the phase distribution can be approximated by a Gaussian distribution as well [7]. In case of DS, we tried to include the most common sources of decorrelation: temporal, seasonal, geometrical and a random loss of decorrelation which could be related to meteorological events. In Figure 3 we further report the distribution of the temporal coherence used to generate the simulated time series, which is based on the coherence distribution of real datasets. In addition to the training dataset,

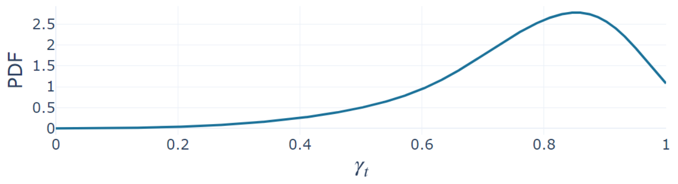


Fig. 3. Probability density function (PDF) of the temporal coherence used to generate the simulated time series.

also test datasets, described when presenting the experiments, are drawn from such a distribution. Finally, we added some outliers to the time series, to make the network robust with respect to them.

During training, normal time series go through a series of random operations to obtain displacement sequences with no or multiple trend changes. Details about the developed simulation process are provided in Appendix. Given a normal displacement sequence $\mathbf{D}$, we first decide with probability p_s if the generated sequence must contain a seasonal component. In such a case, a sinusoidal signal with random amplitude and phase is added. Subsequently, with probability p_{tc} the simulation continues by adding trend changes to the sequence, otherwise it stops and returns the normal, and eventually seasonal, time series. If the trend changes must be generated, the number of change points N_{cp} the time series will contain is uniformly sampled in $[1, N_{max}]$. Then, the position of the change points within the sequence, i.e., $\mathbf{I_{cp}} = [i_1, \ldots, i_{N_{cp}}]$, is randomly sampled with the only constraint that the distance between two change points must be greater than or equal to a minimum number cp_{dis} of acquisitions.

At this point, we start cycling over the selected positions $\mathbf{I_{cp}}$ to generate the different trends. The first operation is to decide the type of the change point, which can be one of the three classes introduced in Section I, i.e., step change, velocity change, or the combination of the two. Once the type is chosen, the time series is divided into two sub-sequences, i.e., $\mathbf{D_l}$, containing all the points between the current change point and the previous one, and $\mathbf{D_r}$, containing all the points between the current change point and the next one. Then, a random step $\Delta S \sim P_{step}$ or velocity $\Delta V \sim P_{vel}$ is added to $\mathbf{D_r}$ depending on the type of change point, where P_{step} and P_{vel} are probability distributions.

The generated trend change might be negligible w.r.t. the local variance of the input signal, i.e., the change is not visible due to the noise, and this leads to wrong targets. To account for this situation a validity test is performed for each candidate point. The validity test consists in estimating the posterior standard deviation given the step/velocity change point, which is then compared with the sampled $\Delta S/\Delta V$. If the value of the change is not negligible w.r.t. the estimated standard deviation the candidate change point is confirmed as target.

All the steps of the presented simulation process have been carefully tuned to obtain realistic realisations. The final output of the whole process is a collection of $< \mathbf{Bt}, \mathbf{D}, \mathbf{Y} >$ triplets, where $\mathbf{Bt}$ is the sequence of temporal baselines, $\mathbf{D}$ is the sequence of corresponding displacements. and $\mathbf{Y}$ is the target vector, having the same length of $\mathbf{Bt}$, which can assume two

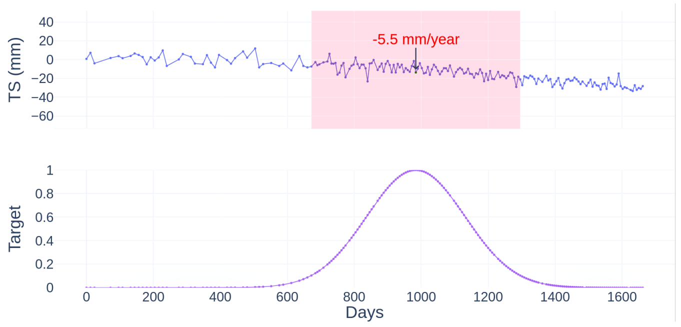


Fig. 4. Example of velocity change point in a noisy time series (first row). The noise makes it difficult to establish the exact location of the change point, which can be identified within the red region. In the second row the corresponding target designed as a gaussian centered in the change point.

values: 1 at change point locations and 0 in the other positions. In the following sub-section, we describe the training of the designed architecture. To simplify the notation, and to be coherent with the notation introduced to describe the proposed method, we will refer to $\mathbf{Bt}$, $\mathbf{D}$, and $\mathbf{Y}$ vectors with t, x and y variables, respectively.

B. Training

We started our investigation by considering a standard setting for solving a supervised classification problem. In particular, given a sample x_t from the input displacement sequence and denoting with ϕ the function realised through the network, we have

$$p_t = \phi(x_t, \theta), \tag{15}$$

where θ are the network parameters and $p_t \in [0, 1]$ is the probability of the input sample to be a change point. Then, parameters θ are learned to minimise the Binary Cross Entropy (BCE) loss, which, for a single time series is given by

$$BCE(p, y) = -\frac{1}{T}\sum_{t=1}^{T} y_t \cdot log(p_t) + (1-y_t) \cdot log(1-p_t), \tag{16}$$

where T is the total number of samples in the sequence and y is the corresponding target.

In this setting, the network is required to predict the exact position of the change points to minimise the objective function. In a real scenario, however, we are uncertain of the exact position where a trend change starts, and this is particularly true in the case of very slow velocity trend changes. In general, the uncertainty depends both on the entity of the change and on the noise. The less the change is evident and the higher the noise, the more is the uncertainty on the change point temporal location. We account for this problem by implicitly forcing the network to learn this kind of uncertainty in the classification. To this purpose, we modified the sparse target vectors y to be a composition of multiple gaussians. In the specific, we define the target associated to each change point as a gaussian centered on the time step corresponding to the simulated change point (t_{CP}) and having variance proportional to the velocity change ΔV measured in mm/year. Thus, for each change point CP the following is defined

$$exp(-\frac{(t-\mu_{CP})^2}{(2 \cdot \sigma_{CP}^2)}), \tag{17}$$

where $\mu_{CP} = t_{CP}$, being t_{CP} the temporal location of the change point, and σ_{CP}^2 the estimated posterior variance measured in $days$. Figure 4 shows an example of noisy time series, with a slow velocity trend change, together with the corresponding target. In this new setting model parameters are optimised to solve a regression problem, by minimizing the Mean Squared Error (MSE), that is

$$MSE(p, y) = \frac{1}{T}\sum_{t=1}^{T}(p_t - y_t)^2. \tag{18}$$

for a single time series.

One of the problems we faced during training was due to class unbalancing. Indeed, the number of change points in the dataset is very small, typically of two orders of magnitude, compared to the number of samples in the time series (e.g., 3-4 change points for a time series of 300 samples). This caused the classification problem to be high unbalanced, which is an unwanted property during training that makes learning challenging. The solution we adopted is to re-weight the losses to give higher weight to the error relative to the change points. Thus, we obtain the weighted BCE (wBCE)

$$wBCE(p, y) = -\frac{1}{T}\sum_{t=1}^{T} w_c \cdot y_t \cdot log(p_t) + (1-y_t) \cdot log(1-p_t), \tag{19}$$

where $w_c = \frac{N_N}{N_{CP}}$, with N_N and N_{CP} the total number of normal points and change points in the sequence, respectively. The same strategy is applied to obtain the weighted MSE (wMSE)

$$wMSE(p, y) = \frac{1}{T}\sum_{t=1}^{T} w_c \cdot (p_t - y_t)^2, \tag{20}$$

where w_c is applied only to the loss values corresponding to the change points in the target. As demonstrated by the conducted experiments, presented later in the document, we obtained our best results by solving for a regression problem minimising the wMSE.

V. Results

In this section we present the obtained results, by providing the performance reached on the suite of tests we designed to validate the learned network. We employed a minibatch learning strategy using the Adam optimisation algorithm [32], with batches of 128 time series and a learning rate of 0.001. Early stopping has been employed in each experiment to avoid overfitting during training. The F1 score is the metric used for model selection and, as for the training, uncertainty about the target change points is considered also during the computation of the metrics by defining as a true positive detection the prediction that falls within the uncertain target region. Predictions are obtained by keeping the peaks in the output probability distribution with values ≥ 0.5.

TABLE I

SCORES OBTAINED ON TEST SET S1 BY DIFFERENT CONFIGURATIONS OF THE BIDIRECTIONAL LSTM NETWORK (BIDILSTM) TRAINED BY MINIMISING THE BCE. GRAY ROWS ARE THE ONES WITH HIGHER F1 SCORES. NETWORKS HAVE BEEN TRAINED ON UNIFORMLY SAMPLED TIME SERIES.

Model	# Recurrent layers	Hidden state size	TP	FP	FN	Precision	Recall	F1
BidiLSTM	3	50	12289	5536	365	0.6894	0.9712	0.8064
BidiLSTM	3	60	12259	5231	395	0.7009	0.9688	0.8134
BidiLSTM	3	70	**12303**	5791	**351**	0.6799	0.**9723**	0.8002
BidiLSTM	4	50	12288	4959	366	0.7125	0.9711	0.8219
BidiLSTM	4	60	12276	4488	378	0.7323	0.9701	0.8346
BidiLSTM	4	70	12275	**4232**	379	**0.7436**	0.97	**0.8419**

TABLE II

SCORES OBTAINED ON TEST SET S1 BY DIFFERENT CONFIGURATIONS OF THE BIDIRECTIONAL LSTM NETWORK (BIDILSTM) TRAINED BY MINIMISING THE MSE. NETWORKS HAVE BEEN TRAINED ON UNIFORMLY SAMPLED TIME SERIES.

Model	# Recurrent layers	Hidden state size	TP	FP	FN	Precision	Recall	F1
BidiLSTM	4	50	11962	**803**	692	**0.9371**	0.9453	**0.9412**
BidiLSTM	4	60	**12007**	930	**647**	0.9281	**0.9489**	0.9384
BidiLSTM	4	70	11948	879	706	0.9315	0.9442	0.9378

A. *Results on Simulated Data*

As already explained in Section IV, to overcome the lack of ground truth for InSAR time series, we performed our investigation by using simulated training data, which has been carefully designed together with InSAR specialists. In the following, the conducted experiments are reported, together with the results obtained on different simulated test sets to assess the detection accuracy of the trained model and its generalisation capabilities.

1) Baseline Network Configuration: In order to find a good baseline architecture for our investigation, we trained several networks which differ for the number of stacked recurrent layers, i.e., the depth of the encoding path, and for the size of the hidden state of each recurrent cell. We started by considering uniformly sampled time series, with a 11-days sampling rate between acquisitions. The first architecture we developed contains simple bidirectional LSTM cells, with no time gates, and the networks have been trained by minimising the weighted BCE loss defined in Equation 19. The training set has been generated through the simulation pipeline described in IV-A. Training time series contain up to 4 change points, corresponding to a combination of step and velocity trend changes. Step change points have a minimum displacement of 3 mm, while velocity change points correspond to slopes of minimum 5 mm/year. At this stage, we gave lower change values an higher probability to be sampled, by forcing P_{step} and P_{vel} to be Rayleigh distributions with $\sigma = 3$ and $\sigma = 5$, respectively. The test set we built for this group of experiments is composed of 10000 anomalous time series, which we refer to as S1 and includes both velocity and step change points. The coherence of the time series in the test set follows the same distribution used for generating the training data (see Figure 3). The sampling rate of this test set is the one of the original SAR acquisitions.

Most relevant results are shown in Table I. Variations of the investigated network hyper-parameters do not affect markedly the performance. On average, networks with 4 recurrent layers have a higher F1 score. Therefore, the same configurations have been considered in later experiments. Please notice that one could have tested a large number of neurons or layers. However, finding the best possible network configuration is out of the scope of our work, which has been focused on demonstrating the proposed method and its applicability.

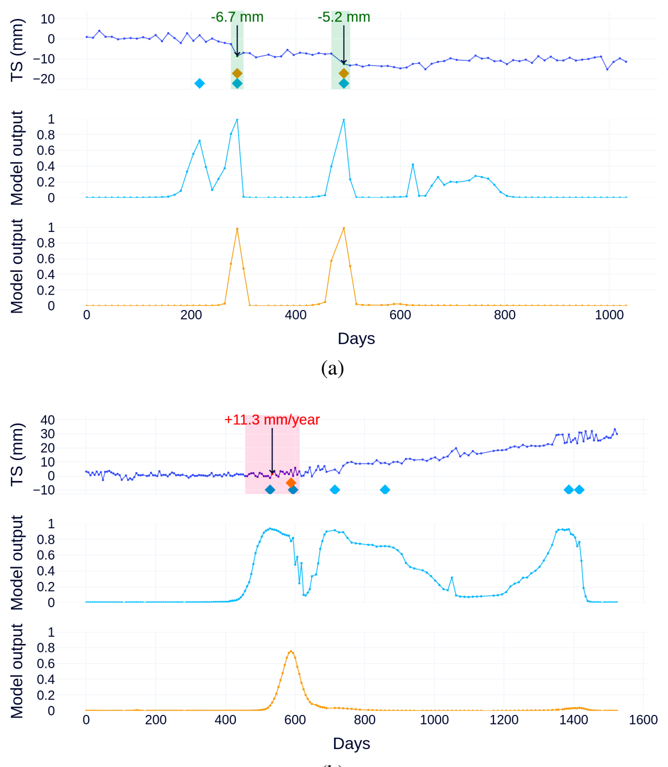


Fig. 5. Examples of inferences on dataset S1, including step change points (a) and velocity change point (b). In each figure: the original time series (1-st row), the output of the network trained with discrete targets (2-nd row), and the output of the network trained with continuous targets (3-th row). Diamond symbols represent the detections performed by the trained networks, coloured as the respective output plots. Transparent coloured regions are the uncertainties related to the targets, which are red for velocity change points and green for step change points. Values of highlighted targets are the entities of the changes, expressed in mm for step change points and mm/year for velocity change points.

2) Continuous Targets: Table II shows the performance on test set S1 with the bidirectional LSTM network trained using the MSE as loss function, defined in Equation 20. True detections slightly decrease but it is possible to observe a drastic drop of FPs, corresponding to an increment of the precision of the learned model. The visual inspection of the results confirmed the quantitative analysis (see Figure 5), outlining noisier predictions when training with discrete targets. Training a regression model allowed to obtain output distributions similar to what is expected by design, which faithfully represent the uncertainty related to the trend changes.

3) TGLSTM: Table III shows the results obtained on dataset S1 by including the sampling rate as additional information. The architecture we trained in these experiments is composed of 4 recurrent layers realised by combining a TGLSTM, employed in the first layer, and LSTM cells, for the following

TABLE III
SCORES OBTAINED ON TEST SET S1 BY DIFFERENT CONFIGURATIONS OF THE BIDIRECTIONAL TGLSTM NETWORK (BIDITGLSTM) TRAINED BY MINIMISING THE MSE. NETWORKS HAVE BEEN TRAINED ON NON-UNIFORMLY SAMPLED TIME SERIES.

Model	# Recurrent layers	Hidden state size	TP	FP	FN	Precision	Recall	F1
GS	-	-	9716	3064	2938	0.7603	0.7678	0.7640
BidiTGLSTM	1 TGLSTM + 3 LSTM	50	12167	446	487	0.9646	0.9615	0.9631
BidiTGLSTM	1 TGLSTM + 3 LSTM	60	12164	**379**	490	**0.9698**	0.9613	0.9655
BidiTGLSTM	1 TGLSTM + 3 LSTM	70	**12219**	423	**435**	0.9665	**0.9656**	**0.9661**
UBidiTGLSTM	1 TGLSTM + 3 LSTM	70	11995	902	659	0.9301	0.9479	0.9389

TABLE IV
SCORES OBTAINED ON TEST SET S1 BY THE BIDIRECTIONAL TGLSTM NETWORK (BIDITGLSTM) TRAINED 4 TIMES WITH DIFFERENT RANDOM SEEDS. MEAN AND STANDARD DEVIATION ARE REPORTED FOR EACH METRIC.

Model	# Recurrent layers	Hidden state size	Precision		Recall		F1	
			mean	std	mean	std	mean	std
BidiTGLSTM	1 TGLSTM + 3 LSTM	50	0.9632	**0.0017**	0.9589	0.0020	0.9610	0.0015
BidiTGLSTM	1 TGLSTM + 3 LSTM	60	0.9642	0.0036	0.9605	0.0026	0.9623	0.0019
BidiTGLSTM	1 TGLSTM + 3 LSTM	70	**0.9648**	0.0022	**0.9651**	**0.0011**	**0.9650**	**0.0011**

ones, as already described in Section III. Performance is noticeably improved with the introduction of the time-gated architecture, which allows an increase of TPs, by keeping low the number of wrong and missed detections. In the first row, the performance of the reference golden standard (GS), introduced in Section II, is also reported. The obtained scores demonstrate that the proposed approach is suitable for analysing non-uniformly sampled time series, allowing to reach higher detection scores w.r.t. the reference statistical method. Additionally, we report the results obtained with the network using a uniform sampling rate (UBidiTGLSTM in the table), which demonstrates to behave similarly to the one without time gates. This result confirms that the network is able to exploit the sampling rate information to improve the detection. Figure 6 shows an example of inference on a time series with non-uniform sampling rate. In the example, the model trained with no time-gates (2-nd row) predicts a wrong change point in the region with a variable sampling rate, while the TGLSTM network (3-rd row) is more robust to such variations.

For completeness, we conducted additional experiments in which we run all the networks configurations with 4 different random seeds to test the robustness of the training w.r.t. the random initialisation of the weights. The results of such investigation are reported in Table IV. Again, the results confirm how the performance does not change drastically among the investigated number of neurons. The BidiTGLSTM network with 4 layers and a hidden state size of 70 units has been considered in later experiments, since it obtained the highest mean and lowest standard deviation in terms of F1 score.

4) Data Augmentation: In the following, we report the tests that have been conducted to assess the generalisation capabilities of the learned network and the role assumed by several data augmentation strategies in the attempt to improve them.

Random slopes and displacements. One of the augmentation techniques that has been introduced to improve

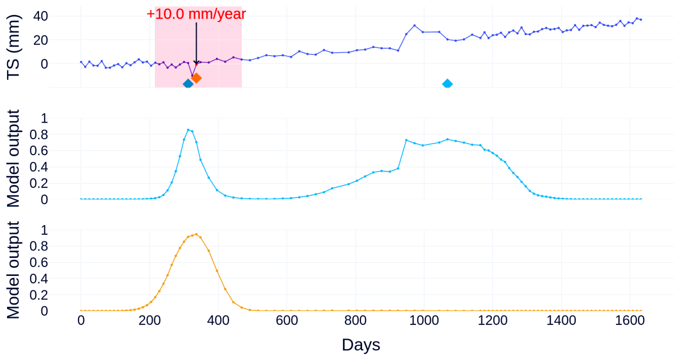


Fig. 6. Example of inference on a time series (1-st row) belonging to dataset S1 with the LSTM and TGLSTM networks, on the 2-nd and 3-rd rows respectively.

TABLE V
SCORES OBTAINED ON TEST SET S2 BY THE BIDITGLSTM NETWORK TRAINED WITH AND WITHOUT DATA AUGMENTATION.

Model	Data augmentation	TP	FP	FN	Precision	Recall	F1
GS	-	9670	2999	2984	0.7633	0.7642	0.7637
BidiTGLSTM	None	7684	6166	4970	0.5548	0.6072	0.5798
BidiTGLSTM	Random slopes and displacements	**11480**	**822**	**1174**	**0.9332**	**0.9072**	**0.92**

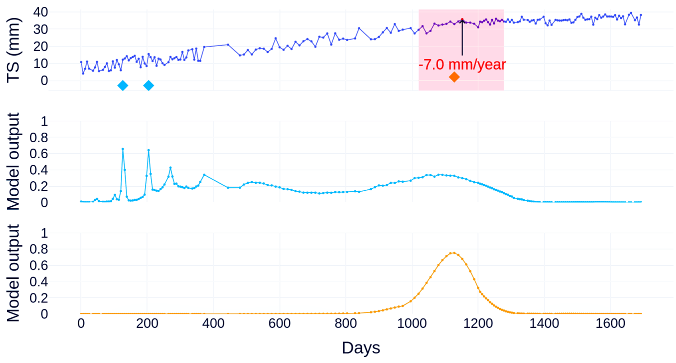


Fig. 7. Example of inferences on dataset S2, including a velocity change point. In the figure: the original time series (1-st row), the output of the network trained without data augmentation (2-nd row), and the output of the network trained with augmented time series (3-th row).

the generalisation of the learned model consists in adding random global displacement and slope to each time series during training. Indeed, the network is encouraged to adapt to different dynamics, thus learning a stronger concept of a trend change. To show the performance reached by introducing this kind of data augmentation we built an additional dataset S2 composed of 10000 time series which have been augmented by adding an initial global displacement uniformly sampled in the range [-20 mm, $+20$ mm] and a global slope uniformly sampled in the range [-20 mm/year, $+20$ mm/year]. The coherence of the time series in S2 follows the distribution reported in Figure 3. In Table V a comparison of two networks trained with and without data augmentation. It is possible to notice a dramatic drop in the performance when applying the model trained without data augmentation to augmented time series. This outlined an overfitting of the learned model to the training set, by demonstrating very poor capabilities to recognise trend changes in time series with different dynamics. Using data augmentation helped to better fit on new unseen

TABLE VI
SCORES OBTAINED ON TEST SET S2 BY THE BIDITGLSTM NETWORK TRAINED WITH AND WITHOUT DATA AUGMENTATION.

W	Data augmentation	TP	FP	FN	Precision	Recall	F1
1	None	10707	**709**	1623	**0.9379**	0.8684	0.9018
	Random deletions	**11012**	759	**1318**	0.9355	**0.8931**	**0.9138**
2	None	10027	**719**	2250	**0.9331**	0.8167	0.871
	Random deletions	**10683**	818	**1594**	0.9289	**0.8702**	**0.8986**
3	None	9201	**749**	3043	0.9247	0.7515	0.8291
	Random deletions	**10344**	788	**1900**	**0.9292**	**0.8448**	**0.885**
4	None	8522	**841**	3604	0.9102	0.7028	0.7931
	Random deletions	**9937**	919	**2189**	**0.9153**	**0.8195**	**0.8648**
5	None	7915	**899**	4121	0.898	0.6576	0.7592
	Random deletions	**9527**	997	**2509**	**0.9053**	**0.7915**	**0.8446**
6	None	7485	**940**	4453	0.8884	0.627	0.7352
	Random deletions	**9365**	1010	**2573**	**0.9027**	**0.7845**	**0.8394**

TABLE VII
SCORES OBTAINED ON TEST SET S3 BY THE BIDITGLSTM NETWORK. p_s IS THE PROBABILITY OF GENERATING SEASONAL TIME SERIES DURING TRAINING.

Model	p_s	TP	FP	FN	Precision	Recall	F1
GS	-	7914	3789	4746	0.6762	0.6251	0.6497
BidiTGLSTM	0.1	6282	2232	6378	0.7378	0.4962	0.5934
BidiTGLSTM	0.2	6413	2235	6247	0.7416	0.5066	0.6019
BidiTGLSTM	0.3	8559	2014	4101	0.8095	0.6761	0.7368
BidiTGLSTM	0.4	8252	**1540**	4408	**0.8427**	0.6518	0.7351
BidiTGLSTM	0.5	**9292**	1980	**3368**	0.8243	**0.734**	**0.7765**

data. Also for this dataset, we report the scores obtained by the GS algorithm, which are lower both in terms of precision and recall. Examples of inferences are shown in Figure 7.

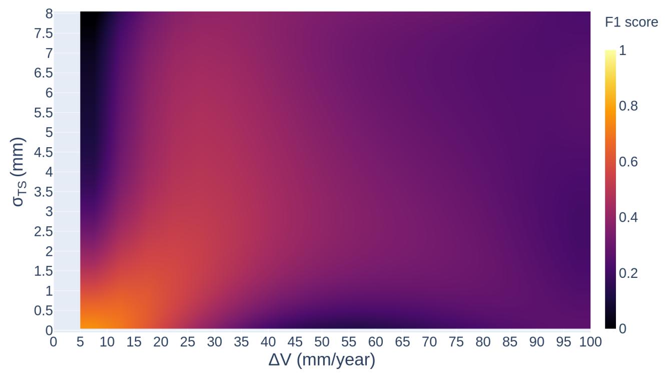
Random deletion of samples. To increase the robustness of the network when dealing with non-uniformly sampled data, we augmented training time series by randomly removing acquisitions, thus increasing the effect of the lack of data and the variability of the sampling rate. To test the performance of this kind of data augmentation we built increasing challenging test sets, in which we enhanced the non-uniformity of the sampling rate by removing windows of length W. Table VI shows the comparison between the network trained with and without the introduced data augmentation. The datasets used for these tests are six versions of the same collection of 10000 time series, in which windows W of increasing length, from 1 to 6 points, have been randomly removed. It is possible to observe that the detection capabilities of the original network get worse as the severity of the lack of data increases, by loosing about 0.16 on the F1 score on the most challenging test set, i.e., $W = 6$. On the contrary, the network trained with data augmentation is more robust by keeping high the number of true detections with a slight increase of false positives. The F1 score obtained on the most challenging test set is only about 0.07 points lower than the score obtained on the less difficult one. Thus, at this point, we obtained a recurrent model able to deal with a irregular sampled time series, which is a good property in real applications.

5) Seasonality: Finally, we tested the robustness of the proposed network when dealing with seasonal time series. This has been done by building a test set composed of only seasonal time series, which we call S3. The number of time series contained in S3 is 10000. We conducted experiments with different values for the probability of generating seasonal time series during training, i.e., $p_s \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ in the simulation process described in Section IV-A. Amplitudes of the seasonal signal have been sampled in the [3 mm, 40 mm] range. The quantitative analysis confirms the need for a correct balancing of seasonal data during training, as it is possible to observe in Table VII. Noticeably, the capability of the network to deal with seasonal time series increases as the probability of introducing a seasonality during training reaches the correct balancing value of 0.5, even if with a slight decrease of about 0.02 points on the precision due to an increase of the false detections, which is negligible compared to the increase of about 1000 correctly predicted change points.

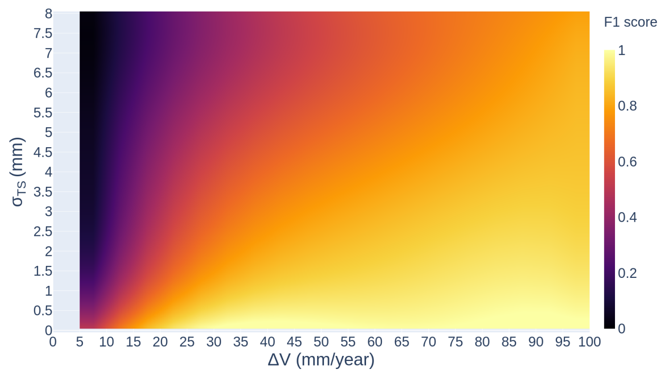
6) Analysis of Velocity Change Points: We integrate the analysis of the network performance with additional tests, which have been conducted to further investigate the behaviour of the learned model in different scenarios and to obtain useful statistical insights on larger test sets. In particular, this section is provided to highlight how the correlation between the noise and the entity of the trend change in a time series impacts on the performance of the trained network in terms of both correct detections, measured with the F1 score, and detection uncertainty, i.e., the difference in days between the detection performed by the network and the exact location of the change point. Here we report the analysis performed on velocity change points, which are the most challenging to detect. To this purpose, we collected a dataset of 50000 time series, which vary for the entity of the trend change ΔV, measured in mm/year, and the standard deviation σ_{TS}. In the conducted test we choose $\Delta V \sim \mathrm{U}(5, 100)$, which is representative of real cases in InSAR data. Furthermore, We defined the generated noise as a random variable, by scaling the original time series from a random quantity such that the standard deviation of the time series is $\sigma_{TS} \sim \mathrm{U}(0.1, 8)$.

Figure 8 describes the results obtained on the 50000 time series that involve velocity trend changes. The heatmaps in the figure, whose points represent the test time series, show how the performance of the learned model changes depending on both the noise and entity of the change. Plots have three dimensions, i.e., the slope difference ΔV on x axes and the σ_{TS} on the y axes. The third dimension, which is the measure of performance, is different in each plot and it is represented through the use of colours. In the plots related to the F1 (Figure 8a-8b) brighter colours refer to higher F1 scores, while darker colours refer to lower scores; in the plots related to the $|\Delta T|$ (Figure 8c-8d) colours from green to red refer to values, from low to high, of the temporal uncertainty related to the detections. The temporal uncertainty is defined as the temporal absolute difference, measured in days, between the day corresponding to the target change point (t_{CP}) and the day corresponding to the detection (t_D), i.e., $|\Delta T| = |t_{CP} - t_D|$.

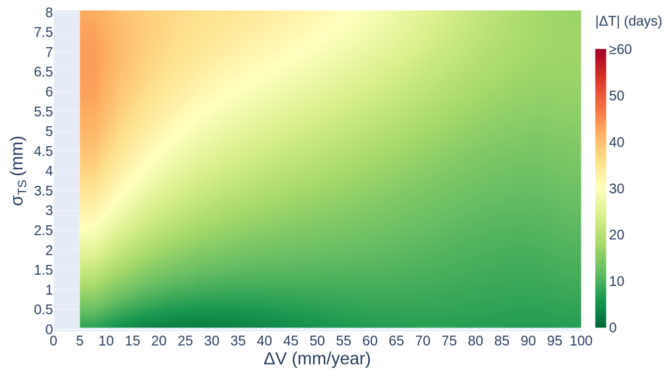
By looking at the plot related to the F1 score on the left (Figure 8a), it is possible to notice how the performance of the network is strictly related to the entity of both the noise and the trend change. In particular, detection scores get worse in the case of very challenging time series, characterised by higher levels of noise and very slow trend change. On the contrary,



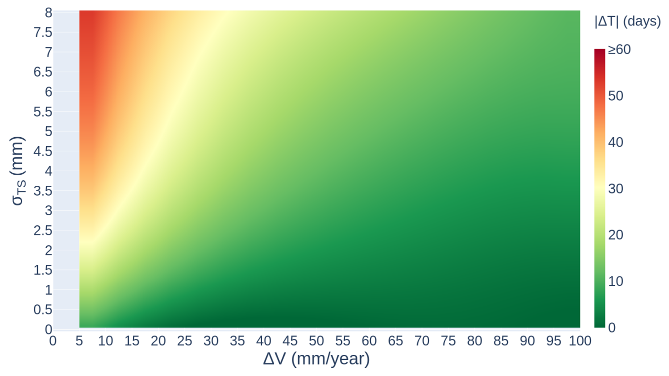
(a)



(b)



(c)



(d)

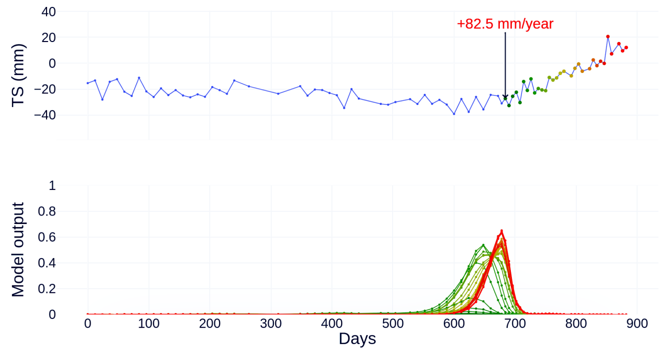
Fig. 8. Results of change point detection performed on 50000 test time series expressed as 3-dimensional plots, one for the F1 score and $|\Delta(T)|$, where each point represent a time series. Plots are heatmaps, which illustrate the performance in relation to the noise σ_{TS} and the slope absolute difference $|\Delta V|$. The third dimension, corresponding to the computed metrics, is realised through the use of colours. In the upper plots (a, b) brighter colours refer to higher F1 scores, while darker colours refer to lower scores; in the lower plots (c, d) colours from green to red refer to values, from low to high, of the temporal uncertainty related to the detections. Plots a and c refer to the result obtained with the original training, while plots b and d refer to the result obtained by balancing the ΔV values in the training dataset.

as soon as the level of the noise decreases and the entity of the change increases, i.e., in the bottom-left region in the plot, performance improves. Nevertheless, contrary to expectations, results get worse as we move towards the right part of the plot. This unexpected performance has been due to the simulation process, which, in the attempt to generate sequences as near as possible to the real InSAR data distribution, assigned slow velocity changes a higher probability of being generated (see Section IV-A). The resulting unbalanced training prevented the network from reaching expected scores in the case of faster velocity changes. In Figure 8b we present the results, in terms of F1 score, which have been improved by correctly balancing the training through the uniform sampling of slope values.

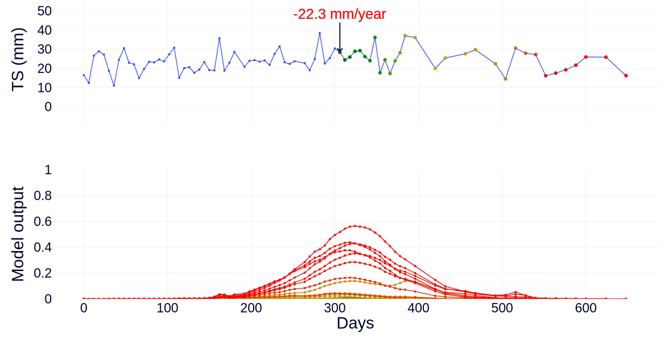
For what concerns the temporal uncertainty related to the performed detection, please consider first to the bottom plot in Figure 8c, whose points refer only to time series for which we have a correct detection, i.e., $TP > 0$. As expected, predicting the exact location of the change point is difficult in the time series characterised by high levels of noise and slow trend changes (upper-left region), as testified by the higher uncertainty. As we move towards easier sequences (bottom-right region), the temporal uncertainty tends to zero. The same plot is shown in Figure 8d after the balancing of the ΔV values during training. The relation between the difficulty of the time series and the temporal uncertainty remains unchanged, but it is much more delineated given the higher number of correct detections.

7) Performance on Continuous Monitoring: Nowadays, Sentinel-1 sensors make it possible to obtain timely displacement time series at regional, or even national, scale. One of the operational use of this data is for monitoring ground deformation over wide areas [33]. In continuous monitoring, as soon as a new acquisitions are available, the displacement time series are updated and the change point detection algorithm is applied to highlight anomalous trends affecting the area of interest. In the following we demonstrate how the proposed method can be successfully employed in continuous monitoring projects, by performing both a quantitative analysis, on simulated time series, and a qualitative one on real InSAR data, as reported in the following Section V-B.

The metrics presented so far, such as the F1 scores and the temporal uncertainty, are no longer enough to evaluate the performance of the proposed approach when dealing with this kind of task. Indeed, in continuous monitoring projects, other than predicting the change points with accuracy, it becomes important to be able to provide timely detections, as soon as new acquisitions are available. Thus, we introduce an additional metric, which quantifies the number of new



(a)



(b)

Fig. 9. Two examples of analysis in a continuous monitoring perspective. For each example, in the first row the time series. Coloured points represent the N updates following the change point, from earlier one (green colour) to later one (red colour). In the second row, N network outputs, obtained by feeding the network with each sub-sequence starting from zero to the n-th update. To facilitate the understanding, line colours in the second row are the same of the corresponding updated point in the time series. The two sub-figures (a-b) show different scenarios. In the upper plot (a) the slope difference related to the trend change is higher (about 83 mm/year), while in the plot on the bottom (b) the change is slower (about 22 mm/year).

updates needed to the network to raise a detection. Figure 9 shows two examples of analysis in a continuous monitoring perspective. For each of the two example in the figure (a-b), the original time series is depicted in the first row, while the network activation is reported in the second row. It is possible to observe how depending on the entity of the trend change, and of the noise, we need different number of acquisitions before having a strong activation of the network. The lower the slope difference (ΔV) and the higher the noise (σ_{TS}), the higher the number of updates needed to raise a detection.

To evaluate the performance in terms of the number of required updates we built a new synthetic test set of 50000 time series of 80 samples each. In each time series a single velocity change point, with different slope values, is placed at the 50-th acquisition. Then, the network is evaluated by computing the output after each of the 30 updates following the change point. As soon as the network identifies a trend change, the current number of updates is registered as the one required to produce a relevant activation. This procedure has been applied for each test sequence and the results have been reported in Figure 10. Again, it is possible to observe how the number of updates required to identify the trend change depends on both the noise level and entity of the trend changes.

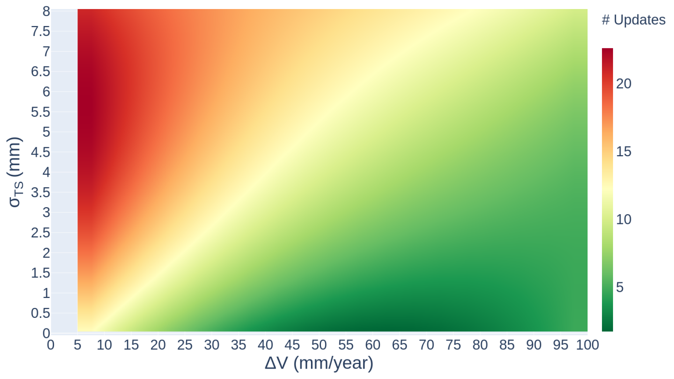


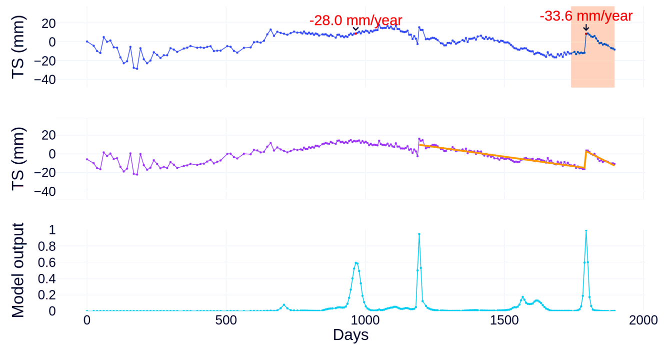
Fig. 10. Results of change point detection performed on 50000 test time series expressed as 3-dimensional plots. Each point is a time series with a change point at the 50-th sample. The number of updates required to raise a detection, i.e., the metric used to evaluate the continuous monitoring performance, is plotted in relation with the noise σ_{TS} and the absolute slope difference $|\Delta V|$.

In particular, for high noise levels and slow changes (upper-left region) a higher number of updates are required to correctly identify the change points, while few updates are needed in the case of easier sequences (bottom-right region).

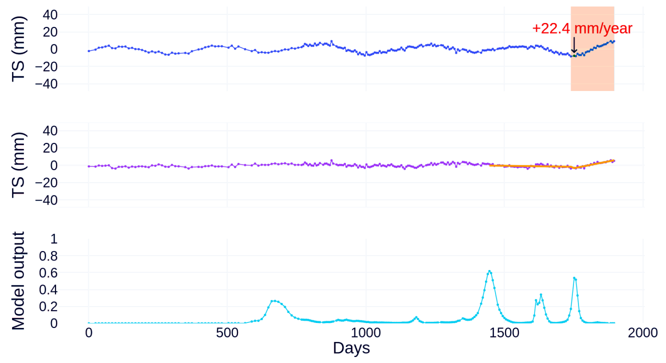
B. Real Data Evaluation

1) Tuscany, central Italy: The developed approach has been tested in a real case study in the context of a continuous monitoring project for monitoring the territory of the Tuscany region (central Italy) using Sentinel-1 data on both ascending and descending geometries. The output of the performed change point detection has been provided to Università degli Studi di Firenze (UniFI), the end-user of the project, which gave us feedbacks during the developing of the approach. InSAR time series have been obtained using the SqueeSAR^{TM} algorithm, which allowed to collect about 1.5 million time series, each one representing displacement values of a measurement point in a period of about 5 years. The continuous monitoring project aims at producing a map of the territory with hotspots, which highlights measurement points associated with trend changes, or, in general, an anomalous behaviour, in the last 150 days.

After the change point detection is performed, a post-processing procedure allows selecting the points considered of interests for further investigation. Besides the temporal restriction to the last 150 days, detections are confirmed as change points if the value of the change ($\Delta S/\Delta V$), which is estimated for each detection, is relevant w.r.t. the noise in the time series. This is done, as during the validity test of the simulation process, by comparing the entity of the change with the posterior standard deviation. In addition, each candidate change point is confirmed only if it has at least two neighbouring change points within a spatial radius of 250 meters. Furthermore, only changes of at least 10 mm and 10 mm/year, for step and velocity change points respectively, are highlighted. In Figure 11 an example of detection performed with the proposed approach on two real InSAR time series from the Tuscany territory (cetral Italy). In both the examples, detected change points are reported directly in the first row



(a)



(b)

Fig. 11. Example of change point detection performed on two real InSAR time series from the Tuscany continuous monitoring project. In both the examples, the first row contains the original time series, where the highlighted period refers to the last 150 days; the second row is the time series deprived of its yearly seasonal component, estimated only for visualisation purposes, where the orange line corresponds to the fitted trends around the last detected change point; the third row is the output of the trained network. Detected change points are reported directly in the first row only if the associated change values are greater than 10 mm and 10 mm/year for step and velocity points respectively.

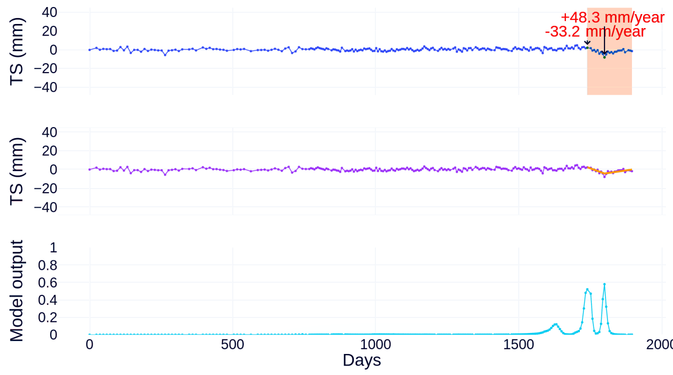


Fig. 12. Example of change point detection performed on a real InSAR time series from the Tuscany continuous monitoring project. In the last part of the sequence the network recognises a coherent group of measurement as a small local trend.

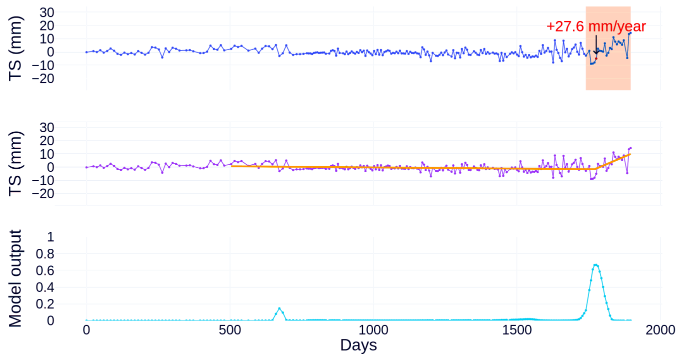


Fig. 13. Example of change point detection performed on a real InSAR time series from the Tuscany continuous monitoring project. In the specific example a change point is identified in the last 150 days (highlighted period) by the proposed method while, for the specific time series, no trend changes have been identified by the statistical approach in the 150 days. The noise makes it difficult to predict the exact location of the change point, causing the detection, of the statistical method in this particular case, to fall out of the period of interest.

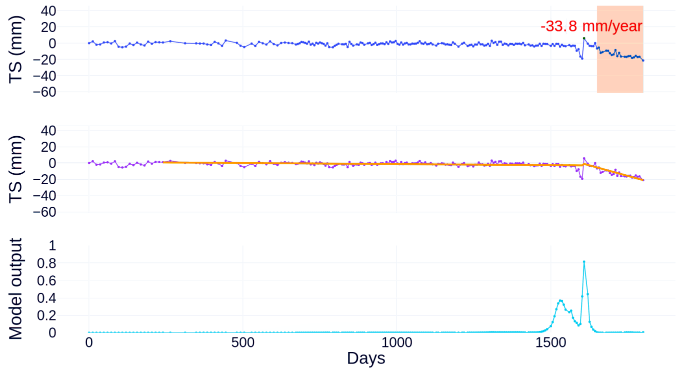
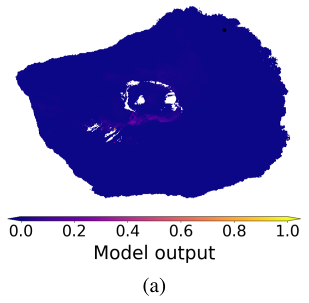
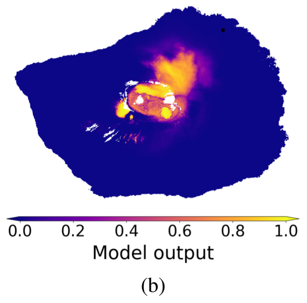
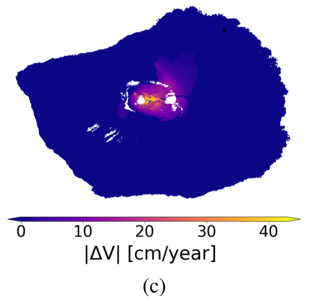


Fig. 14. Example of change point detection performed on a real InSAR time series from the Tuscany continuous monitoring project. In the specific example, the statistical approach raised a detection in the last 150 days while the proposed one identified the change point few acquisitions earlier, resulting in a missed detection in the context of the continuous monitoring task.

only if they meet the aforementioned requirements. It is possible to observe how the learned model can be successfully applied in the analysis of real data, being able to highlight the relevant trend changes in the InSAR time series. Noticeably, the proposed method can deal also with seasonal time series without the need for a pre-processing step to estimate and remove the seasonality, which is instead required by the reference approach. The results have been compared also to the ones obtained through the statistical analysis, reported in Section II, state of the art in the field of the InSAR analysis, to highlight the difference with the proposed data-driven one. What emerged is that the deep learning approach generates a higher number of detections in the last 150 days w.r.t. the statistical one. This is partially due to its sensitiveness to coherent groups of measurements which generates small local trends in the time series. An example of this kind of situation is reported in Figure 12. In other cases, instead, the exact beginning of the trend change is difficult to be identified due to the noise and this produces some differences in the final detection between the two approaches. In Figure 13 a change point is identified in the last 150 days (highlighted period) by the proposed method while, for the specific time series, no trend changes have been identified by the statistical one in the 150 days. As depicted in Figure 14, also the opposite situation has been observed. Indeed, for the specific time series in the example the statistical approach raised a detection in the last







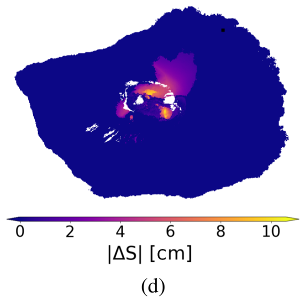
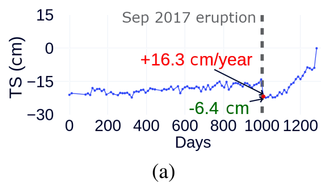


Fig. 15. (a) Network activation before July 2017; (b) Network activation in July-November 2017; (c) Absolute velocity change difference $|\Delta V|$ in correspondence of the detected change points; (d) Absolute step change difference $|\Delta S|$ in correspondence of the detected change points.



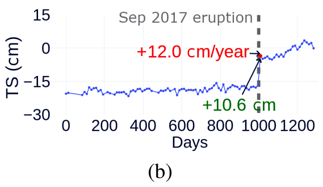
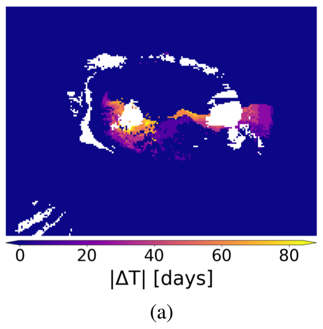
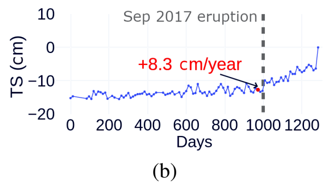


Fig. 16. Examples of InSAR time series from the caldera of Fernandina volcano with negative (a) and (b) positive ΔS. The learned network is able to correctly detect the change points in correspondence of the eruption.





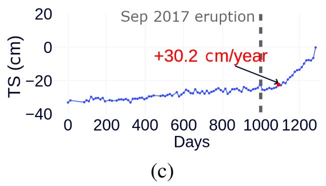


Fig. 17. (a) Detection error measured as the absolute difference $|\Delta T|$ (days) between the detection and the reference day of the eruption computed for the velocity change points; (b) Example of early detection in which the network produced a detection before the ground-truth date; (c) Example of delayed detection in which the network produced a detection after the ground-truth date.

TABLE VIII
SCORES OBTAINED ON 3 SUBSETS OF TIME SERIES EXTRACTED FROM THE SITE AND COMPUTED BY CONSIDERING THE DATE OF THE ERUPTION AS TARGET.

Subset	TP Rate
$\lvert\Delta V\rvert > 5$ cm/year, $\lvert\Delta S\rvert \geq 6$ cm	1
$\lvert\Delta V\rvert > 5$ cm/year, $4 \leq \lvert\Delta S\rvert < 6$ cm	0.89
$\lvert\Delta V\rvert > 5$ cm/year, $0 \leq \lvert\Delta S\rvert < 4$ cm	0.5783

150 days while the proposed one identified the change point few acquisitions earlier, resulting in a missed detection in the context of the continuous monitoring task.

2) Analysis on Fernandina Volcano Area at Galápagos Islands: To further investigate the applicability of the proposed method in a real scenario we report the results of the analysis conducted on an additional area over the Fernandina volcano, at Galápagos Islands. The area experienced several ground deformations due to the known eruption occurred in September 2017 [34]. InSAR time series referring to the site in the period between December 2014 and June 2018 have been obtained through the Python open-source package MintPy, which provides the Sentinel-1 data of the area as an example dataset on the official repository[1].

Figure 15 shows the activation of the network as a 2D map in the period before the eruption (a) and in correspondence of the eruption (b), which has been generated for visualisation purposes by aggregating the output of the network in the period July-November 2017. It is possible to observe how the network has been successfully activated during the event, producing a high probability for the points near to the volcano to be change points. The maps of the estimated absolute velocity and step differences in correspondence of the detected change points are reported Figures 15c and 15d, respectively. In Figure 16 we report two examples of InSAR time series extracted from the dataset, together with the detected change points and the estimated change values. We further report an analysis on the uncertainty related to the detection, by considering the time series exhibiting only a velocity change $|\Delta V| > 5$ cm. The date relative to the eruption has been considered as ground truth for computing the absolute detection error $|\Delta T|$ measured in days (see Figure 17a), which depends on the noise and on the entity of the velocity change, as discussed in Section V-A6.

Finally, we give a quantitative evaluation of the results. Considering again the date of the eruption as target, we selected the set of time series with estimated velocity changes $|\Delta V| > 5$ cm/year. We further divided this set into 3 subsets in which we gradually reduce the contribution of the step change, by obtaining less marked trend changes, and thus more difficult to be identified. As shown in Table VIII, the performance decreases as the visibility of the change decreases, since we go from examples like the ones in Figure 16, in which the change is clearly identifiable, to examples like the ones in

[1]https://github.com/insarlab/MintPy

Figure 17, in which the exact point of the change is difficult to be determined.

C. Discussion

The conducted experiments demonstrate how the proposed recurrent network is suitable for solving the considered task. The results show that the complexity of the network does not affect significantly the performance, while the use of continuous target and of the TGLSTMs over LSTMs have a positive impact on the detection capabilities. We discussed about the importance of data augmentation during training, as well as the correct balancing of the change values during simulation. The noise level in the time series demonstrated to be a determining factor in the detection process, in particular in relation to the entity of the trend change. Obtained scores reflect the distribution of training data, by showing a decrease of performance when considering relatively high level of noise.

The developed network is able to generalise when considering real data. Nevertheless, there is still room for improvement. As previously discussed, the learned model allows detecting also the change points corresponding to local trends. However, these could not be of interest for the final users because their nature could be related to noise artefacts or processing errors. Indeed, one of the limitations of the proposed approach is that it does not identify the phenomena that generate the change points. Thus, the detections provided by the network must go through a post-processing step to select those that are interesting for the specific monitoring project. For instance, the proposed method is not robust to phase unwrapping errors, which would be detected as trend changes. In this case phase unwrapping errors should be corrected before using the learned network. On the other side, specific post-processing chain could be implemented to account for phase unwrapping errors, by removing among the detected change points those exhibiting a change in proximity of 2π.

Another limitation of the proposed approach is related to the problem of the seasonality in the time series, which is not fully resolved. Indeed, the network has been trained on the assumption that the seasonality in the time series can be approximated with sinusoidal signals with constant amplitudes and periods. However, in real cases, the seasonality could not be approximated with such a model and eventual under- or overshoots can be detected as trend changes. A possible solution is to provide the network with the seasonal deformation model w.r.t. which we want to be robust, e.g., the temperature for thermal deformations in urban areas. Furthermore, given that the proposed methodology is based on the approximation of displacement data with linear displacement trends, in case of non-linear displacement, with a constant acceleration for example, the algorithm will approximate the displacement with piecewise linear displacement trends leading to multiple detections every time the local linearisation of the trend is no more valid. Finally, the proposed approach works only with single sequences. Results can be further improved in future works by considering also the spatial correlation among neighbouring points on the ground during the learning process. A spatio-temporal network could be also trained to provide a robust detection algorithm for phase unwrapping errors.

For completeness, we report also the computational time, whose improvement was one of the objectives of our work. The developed method takes about 15 minutes to process a test set of about 630000 time series on a single GeForce GTX 1080 Ti NVIDIA's GPU. The statistical approach instead, takes about 3 hours and 15 minutes on a machine with 2 x Xeon 8-Core E5-2640v3 2.6 GHz 25MB.

VI. Conclusion

We presented a novel approach for change point detection in InSAR time series based on Deep Learning, which is proving to be a valid class of methodologies in the field of Earth Observation. The proposed approach consists of a recurrent neural network, obtained by stacking Long Short-Term Memory (LSTM) layers. We employed a Time-Gated LSTM (TGLSTM) cell to include the sampling rate as additional input information during learning. One of the issues faced in the considered problem, as for many tasks involving SAR data, is the lack of ground truth for training. We overcame this issue by implementing an automatic pipeline to simulate InSAR data, which has been carefully designed with the help of InSAR specialists. Simulated time series have been employed for training the proposed network and to evaluate its detection capabilities. The results of our work have been presented in detail by describing the large suite of experiments conducted during the research. A quantitative analysis of the performance has been performed on a large number of test time series to prove the effectiveness of the learned model. Additionally, we demonstrate how the proposed approach can be employed for continuous monitoring purposes, by performing a quantitative analysis in terms of the number of new SqueeSAR™ updates needed to raise timely detections. Finally, we reported the performance on real use cases, by showing how the learned model demonstrated promising results in the real domain. Furthermore, the limitations of the proposed methodology have been discussed. We additionally reported the performance in terms of required computational time, which is one order of magnitude lower than the one of the reference statistical algorithm. This makes the developed approach a good starting point for future development to provide timely detections at larger scales, e.g., at a national scale, that means analysing millions of InSAR time series every update.

Acknowledgment

This work was performed under a programme of, and funded by, the European Space Agency, contract number N° 4000126733. We would also like to thank Università degli Studi di Firenze (UniFI) for the support provided during the validation process.

Appendix
Procedures for Time Series Simulation

In the following, we provide the pseudocode of the simulation process used in the proposed work. Given a starting collection of normal time series, the GenerateSimulatedTimeSeries (Algorithm 1) procedure is applied to produce simulated trend changes.

Algorithm 1 GenerateSimulatedTimeSeries

```
Require:
    $Dataset$: dataset of $N$ normal time series, each time series is the $(\mathbf{Bt}, \mathbf{D})$ pair of temporal baselines and displacement values
    $p_s$: probability of adding seasonality
    $p_{tc}$: probability of adding change points
    $cp_{dis}$: minimum number of acquisitions between consecutive change points
    $(A_{min}, A_{max})$: range of amplitudes for the seasonal component
    $P_{step}$: probability distribution for the step changes
    $P_{vel}$: probability distribution for the velocity changes
Ensure: $Dataset_{sim}$: dataset of simulated time series

 1: $Dataset_{sim} \leftarrow \{\}$
 2: for $n = 1$ to $N$ do
 3:     $(\mathbf{Bt}, \mathbf{D}) \leftarrow Dataset[n]$
 4:     $len_{ts} \leftarrow$ length of the sequence
 5:     $add\_seasonality \leftarrow$ Bernoulli$(p_s)$
 6:     if $add\_seasonality$ then
 7:         $A \sim U(A_{min}, A_{max})$
 8:         $\phi \sim U(-\pi, +\pi)$
 9:         $\mathbf{D} \leftarrow \mathbf{D} + Asin(2\pi\mathbf{Bt}\frac{1}{365} + \phi)$
10:     end if
11:     $add\_trend\_changes \leftarrow$ Bernoulli$(p_{tc})$
12:     if $add\_trend\_changes$ then
13:         $N_{cp} \sim U(1, N_{max})$
14:         $\mathbf{I_{cp}} \leftarrow \{\}$
15:         for $j = 1$ to $N_{cp}$ do
16:             $i_{sampled} \sim U(1, len_{ts})$
17:             if $|i_{sampled} - i| \geq cp_{dis},\ \forall i \in I_{cp}$ then
18:                 $\mathbf{I_{cp}}$.insert$(i_{sampled})$
19:             else
20:                 $j \leftarrow j - 1$
21:             end if
22:         end for
23:         $\mathbf{D} \leftarrow$ AddTrendChanges$(\mathbf{Bt}, \mathbf{D}, \mathbf{I_{cp}}, N_{cp}, P_{step}, P_{vel})$
24:     end if
25:     $Dataset_{sim}$.insert$(\mathbf{D})$
26: end for
27: return $Dataset_{sim}$
```

The core function for generating trend changes is provided by the AddTrendChanges procedure (Algorithm 2). Given, the sampled locations $\mathbf{I_{cp}}$ of the change points within the sequence, a trend change is generated for each sampled position in accordance with the types of the changes.

CheckStep and CheckVelocity represent the procedures employed to assess that the sampled trend change is valid, i.e., the values ΔS or ΔV are enough to make the change visible given the noise. This is done by estimating the posterior standard deviation given the change point and comparing the magnitude of the change w.r.t. this estimate.

Algorithm 2 AddTrendChanges

```
Require:
    $(\mathbf{Bt}, \mathbf{D})$: pair of temporal baselines and displacements
    $\mathbf{I_{cp}}$: vector of change points indices
    $N_{cp}$: total number of change points to be generated
    $P_{step}$: probability distribution for the step changes
    $P_{vel}$: probability distribution for the velocity changes
Ensure: $\mathbf{D}$: time series with trend changes

 1: for $j = 1$ to $N_{cp}$ do
 2:     $\mathbf{Bt_r} \leftarrow [\mathbf{Bt}[\mathbf{I_{cp}}[j]], ..., \mathbf{Bt}[\mathbf{I_{cp}}[j+1] - 1]]$
 3:     $\mathbf{D_r} \leftarrow [\mathbf{D}[\mathbf{I_{cp}}[j]], ..., \mathbf{D}[\mathbf{I_{cp}}[j+1] - 1]]$
 4:     $\mathbf{D_l} \leftarrow [\mathbf{D}[\mathbf{I_{cp}}[j-1]], ..., \mathbf{D}[\mathbf{I_{cp}}[j] - 1]]$
 5:     $cp_{type} \leftarrow$ random from [“step”, “vel”, “step+vel”]
 6:     $is\_step\_valid \leftarrow True$
 7:     $is\_vel\_valid \leftarrow True$
 8:     if $cp_{type}$ in [“step”, “step+vel”] then
 9:         $\Delta S \sim P_{step}$
10:         $is\_step\_valid \leftarrow$ CheckStep$(\mathbf{Bt}, \mathbf{D}, \mathbf{I_{cp}}, i, \Delta S)$
11:         if $is\_step\_valid$ then
12:             $inverted \leftarrow$ Bernoulli$(0.5)$
13:             if $inverted$ then
14:                 $\Delta S \leftarrow -\Delta S$
15:             end if
16:             $\mathbf{D_r} \leftarrow \mathbf{D_r} + \Delta S$
17:         end if
18:     end if
19:     if $cp_{type}$ in [“vel”, “step+vel”] then
20:         $\Delta V \sim P_{vel}$
21:         $is\_vel\_valid \leftarrow$ CheckVelocity$(\mathbf{Bt}, \mathbf{D}, \mathbf{I_{cp}}, i, \Delta V)$
22:         if $is\_vel\_valid$ then
23:             $inverted \leftarrow$ Bernoulli$(0.5)$
24:             if $inverted$ then
25:                 $\Delta V \leftarrow -\Delta V$
26:             end if
27:             $\mathbf{D_r} \leftarrow \mathbf{D_r} + (\mathbf{Bt_r} - \mathbf{Bt_r}[0]) \cdot \Delta V/365$
28:         end if
29:     end if
30:     if $is\_step\_valid$ or $is\_vel\_valid$ then
31:         $\mathbf{D} \leftarrow$ concat$(\mathbf{D_l}, \mathbf{D_r})$
32:     else
33:         $\mathbf{I_{cp}}$.remove$(j)$
34:     end if
35: end for
36: return $\mathbf{D}$
```

References

[1] A. Ferretti, *Satellite InSAR Data. Reservoir Monitoring from Space*. EAGE Publications, 03 2014.

[2] R. Bamler and P. Hartl, "Synthetic aperture radar interferometry," *Inverse Problems*, vol. 14, no. 4, pp. R1–R54, aug 1998. [Online]. Available: https://doi.org/10.1088

[3] C. Colesanti, A. Ferretti, C. Prati, and F. Rocca, "Monitoring landslides and tectonic motions with the permanent scatterers technique," *Engineering Geology*, vol. 68, no. 1, pp. 3 – 14, 2003, remote sensing and monitoring of landslides. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0013795202001953

[4] A. Peltier, M. Bianchi, E. Kaminski, J.-C. Komorowski, A. Rucci, and T. Staudacher, "Psinsar as a new tool to monitor pre-eruptive volcano ground deformation: Validation using gps measurements on piton de la fournaise," *Geophysical Research Letters*, vol. 37, no. 12, 2010.

[5] C. Colesanti, A. Ferretti, C. Prati, and F. Rocca, "Seismic faults analysis in California by means of the permanent scatterers technique," in *Retrieval of Bio- and Geo-Physical Parameters from SAR Data for Land Applications*, ser. ESA Special Publication, A. Wilson and S. Quegan, Eds., vol. 475, Jan. 2002, pp. 125–131.

[6] A. Ferretti, C. Prati, and F. Rocca, "Nonlinear subsidence rate estimation using permanent scatterers in differential sar interferometry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 5, pp. 2202–2212, 2000.

[7] ——, "Permanent scatterers in sar interferometry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 1, pp. 8–20, 2001.

[8] A. Ferretti, A. Fumagalli, F. Novali, C. Prati, F. Rocca, and A. Rucci, "A new algorithm for processing interferometric data-stacks: Squeesar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 9, pp. 3460–3470, 2011.

[9] P. Berardino, G. Fornaro, R. Lanari, and E. Sansosti, "A new algorithm for surface deformation monitoring based on small baseline differential sar interferograms," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 11, pp. 2375–2383, 2002.

[10] A. Hooper, "A multi-temporal insar method incorporating both persistent scatterer and small baseline approaches," *Geophysical Research Letters*, vol. 35, no. 16, 2008. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2008GL034654

[11] L. Chang and R. F. Hanssen, "A probabilistic approach for insar time-series postprocessing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 1, pp. 421–430, 2016.

[12] F. Bovenga, G. Pasquariello, and A. Refice, "Statistically-based trend analysis of mtinsar displacement time series," *Remote Sensing*, vol. 13, no. 12, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/12/2302

[13] A. Refice, G. Pasquariello, and F. Bovenga, "Model-free characterization of sar mti time series," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[14] F. Raspini, S. Bianchini, A. Ciampalini, M. Del Soldato, L. Solari, F. Novali, S. Del Conte, A. Rucci, A. Ferretti, and N. Casagli, "Continuous, semi-automatic monitoring of ground deformation using sentinel-1 satellites," *Scientific Reports*, vol. 8, 05 2018.

[15] C. Alippi, G. Boracchi, and M. Roveri, "Ensembles of change-point methods to estimate the change point in residual sequences," *Soft Computing*, vol. 17, 11 2013.

[16] C. Alippi, G. Boracchi, and M. Roveri, "A hierarchical, nonparametric, sequential change-detection test," in *The 2011 International Joint Conference on Neural Networks*, 2011, pp. 2889–2896.

[17] S. Yang, Y. Wang, and X. Chu, "A survey of deep learning techniques for neural machine translation," *ArXiv*, vol. abs/2002.07526, 2020.

[18] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[19] J.-P. Briot, G. HADJERES, and F. Pachet, *Deep Learning Techniques for Music Generation - A Survey*. Springer, Cham, 03 2019.

[20] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *CoRR*, vol. abs/1607.00148, 2016. [Online]. Available: http://arxiv.org/abs/1607.00148

[21] D. Park, Y. Hoshi, and C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. PP, 11 2017.

[22] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, Eds. Cham: Springer International Publishing, 2019, pp. 703–716.

[23] S. Chauhan and L. Vig, "Anomaly detection in ecg time signals via deep long short-term memory networks," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–7.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[25] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, pp. 2451–71, 10 2000.

[26] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *30th International Conference on Machine Learning, ICML 2013*, 11 2012.

[27] S. O. Sahin and S. S. Kozat, "Nonuniformly sampled data processing using lstm networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1452–1461, 2019.

[28] F. Lattari, B. Leon, F. Asaro, A. Rucci, C. Prati, and M. Matteucci, "Deep learning for sar image despeckling," *Remote Sensing*, vol. 11, p. 1532, 06 2019.

[29] E. Ruggieri and C. Lawrence, "The bayesian change point and variable selection algorithm: Application to the δo proxy record of the plio-pleistocene," *Journal of Computational and Graphical Statistics - J COMPUT GRAPH STAT*, vol. 23, 01 2012.

[30] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[31] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005, iJCNN 2005.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[33] F. Raspini, S. Bianchini, A. Ciampalini, M. Del Soldato, L. Solari, F. Novali, S. Del Conte, A. Rucci, A. Ferretti, and N. Casagli, "Continuous, semi-automatic monitoring of ground deformation using sentinel-1 satellites," *Scientific Reports*, vol. 8, 05 2018.

[34] Z. Yunjun, H. Fattahi, and F. Amelung, "Small baseline insar time series analysis: Unwrapping error correction and noise reduction," *Computers & Geosciences*, vol. 133, p. 104331, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098300419304194

Francesco Lattari is a PhD student at Dipartimento di Elettronica Informazione e Bioingegneria of Politecnico di Milano, Italy. He received the B.Sc. in Ingegneria Informatica and the M.Sc. in Computer Science and Engineering at Politecnico di Milano in 2014 and 2017, respectively. Starting from his thesis on video object segmentation through deep learning approaches, his research interests are mainly on computer vision and machine learning. Currently, his research focuses on the developing of machine learning approaches in the field of Earth Observation. In particular, he aims to develop innovative solutions to extract information from Synthetic Aperture Radar (SAR) satellite data.

Alessio Rucci was born in Milano, Italy, in 1982. He received the M.Sc. (cum laude) and D.Sc. degrees in electronic engineering from the POLIMI, Milano, in 2007 and 2011, respectively. His thesis is on advanced PInSAR technique and its application in reservoir monitoring and modeling. In 2008, he was with Lawrence Berkeley National Lab as a Visiting Scholar to work on the InSalah Project for Carbon Capture Sequestration (CCS). In 2011, he joined the R&D division of TRE, working on the assessment and estimation of atmospheric effects and surface deformation in non-urban areas.

Matteo Matteucci is Associate Professor at Dipartimento di Elettronica Informazione e Bioingegneria of Politecnico di Milano, Italy. In 1999 he got a Laurea degree in Computer Engineering at Politecnico di Milano, in 2002 he received a Master of Science in Knowledge Discovery and Data Mining at Carnegie Mellon University (Pittsburgh, PA), and in 2003 he got a Ph.D. in Computer Engineering and Automation at Politecnico di Milano (Milan, Italy). His main research topics are pattern recognition, machine learning, machine perception, robotics, computer vision and signal processing. His main research interest is in developing, evaluating and applying, in a practical way, techniques for adaptation and learning to autonomous systems interacting with the physical world. He has co-authored more than 50 (peer-reviewed) papers on international journals, 25 papers in International Books, and more than 150 (peer-reviewed) contributions to international conferences and workshops.