LIMESegment: Meaningful, Realistic Time Series Explanations

Torty Sivill
University of Bristol
vs14980@bristol.ac.uk

Peter Flach
University of Bristol
Peter.Flach@bristol.ac.uk

Abstract

LIME (Locally Interpretable Model-Agnostic Explanations) has become a popular way of generating explanations for tabular, image and natural language models, providing insight into why an instance was given a particular classification. In this paper we adapt LIME to time series classification, an underexplored area with existing approaches failing to account for the structure of this kind of data. We frame the non-trivial challenge of adapting LIME to time series classification as the following open questions: "What is a meaningful interpretable representation of a time series?", "How does one realistically perturb a time series?" and "What is a local neighbourhood around a time series?". We propose solutions to all three questions and combine them into a novel time series explanation framework called *LIMESegment*, which outperforms existing adaptations of LIME to time series on a variety of classification tasks.

1 INTRODUCTION

The proliferation of edge devices, digitization of medical records and tracking of online activity has led to extensive Time Series (TS) datasets. TS Classification (TSC) has thus recently grown in popularity. However, when compared to image processing, TSC remains relatively unexplored. The application of traditional statistical models (i.e. logistic regression) to TSC is not trivial, attributed to the ambiguity surrounding TS representation (Esling and Agon, 2012). Deep learning, whose architecture is more suited to high dimensional data, has potential to unlock the information embedded in TS datasets in the same way it has done for image and natural language (Lim and Zohren, 2021). However, the instability of deep approaches to TSC has hindered its adoption in practice (Lim and Zohren, 2021). The adoption of machine learning depends on its trustworthiness (Ignatiev, 2020). Explainable AI (XAI) bridges the gap between complex machine learning behaviour and human understanding and has helped build trust in machine learning. A significant proportion of research in XAI is dedicated to the development of feature importance explanations which identify the most salient parts of the model input given an output. Of these, Locally Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016) has become the most popular. Adapting LIME to TSC improves understanding of TS latent representations in deep architectures, offers insight into the way bias presents in a TS and builds trust for end-users interacting with TS based AI systems. Adapting LIME for TS however, is not trivial. A key component of LIME is that input data is mapped to interpretable concepts for an end user to understand. For example, groups of pixels can easily be grouped together to form objects in an image. In contrast, there is no obvious way of grouping TS observations together to form meaningful concepts. We address these limitations in this paper such that, given a TSC model and an individual TS to be explained, we obtain explanations of the form: "The individual's temperature over the first hour in hospital was most influential to the classifier's prediction of survival from septic shock".

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

2 RELATED WORK

In this section we begin by introducing Locally Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016) which forms the basis of our framework. We outline the challenges of adapting LIME to TS and review current approaches.

In LIME, the original feature space is transformed into an interpretable representation of human understandable concepts dependent on data type. Given as input a dataset $\mathbf{X}$, the black box classifier $f : R^d \to R$, the instance to be explained $\mathbf{x} \in R^d$ its associated classification $\mathbf{Y_x} \in R$, LIME applies a mapping from the original instance $\mathbf{x}$ into its interpretable representation $\sigma(\mathbf{x})$. If $\mathbf{x}$ is an image, $\sigma(.)$ is often a grouping over the original pixel space into human interpretable "super pixels". $\sigma(\mathbf{x}) = \mathbf{1}^{d'}$ can therefore be understood as a vector of ones to indicate each superpixel is "turned on" in the instance to be explained where d' corresponds to the number of "super pixels" ($d' < d$). LIME then generates new samples $\sigma(\mathbf{z})$ by randomly "turning off" dimensions of the interpretable representation through drawing nonzero elements of $\sigma(\mathbf{x})$ uniformly at random. LIME converts the set of generated samples $\sigma(\mathbf{Z})$ into the original feature space to obtain the labels $\mathbf{Y_Z} = f(\mathbf{Z})$. LIME approximates the behaviour of f with an explanation model, $g : R^{d'} \rightarrow R$. Commonly, a linear model such as ridge regression is selected as g. An exponential kernel $\mathcal{K}_{exp}(\sigma(\mathbf{x}), \sigma(\mathbf{Z}))$ is used to weight g such that generated samples which are more similar to the instance to be explained have more of an effect on the resulting explanation. LIME uses the coefficients of g, $\mathbf{w} \in R^{d'}$ to be used as an explanation of $\mathbf{x}$.

Adapting LIME to TS: A univariate time series sample is a temporally ordered set of T observations $\mathbf{x} = [x_1; x_2; ...; x_T]$. Given a dataset of univariate time series samples $\mathbf{X} \in R^{N \times T}$, a TS to be explained $\mathbf{x} \in R^T$ where T denotes the number of observations, a black box classifier $f : R^T \rightarrow R$, and $\mathbf{Y_x} = f(\mathbf{x})$, the predicted class of the example to be explained, we are interested in generating a surrogate model g around $\mathbf{x}$ in order to identify the most salient features of $\mathbf{x}$ with regards to $\mathbf{Y_x}$. There is limited literature on TS explainability compared to image and natural language for which a plethora of established frameworks exist. Feature importance based explanations have limited success when applied to TS as any adaptation requires the consideration of the temporal nature of the input space. Tonekaboni et al. (2020) introduce FIT, an explainability framework which defines the importance of each observation based on its contribution to the black box model's distributional shift. Similarly, Rooke et al. (2021) extend FIT into WinIT which measures the effect on distribution shift of groups of observations. Labaien et al. (2020) instead suggest a framework which finds the minimum perturbation required to change a TS classification. Closest to our framework is that of Guillemé et al. (2019) and Neves et al. (2021) which directly adapt LIME to TS. Any adaptation of LIME to TS data must address the following considerations: 1) How to find an interpretable representation of $\mathbf{x}$ 2) How to generate samples in the neighbourhood of $\mathbf{x}$ 3) How to define locality around $\mathbf{x}$. The following section argues how existing adaptations of LIME to TS make simplifying assumptions which we improve upon in our framework, *LIMESegment*.

Interpretable Representation of a TS: For TS, transformation into an interpretable representation follows the same intuition as images where an explanation involving a single observation would be nonuseful to an end user and fail to capture salient properties of the TS. It follows that the transformation involves segmenting the TS into a lower dimensional representation. However, TS does not lend itself naturally to "conceptualisation", (grouping of observations in a meaningful way), in the same way as super pixels intuitively conceptualise images. In their respective adaptions of LIME to TS, Guillemé et al. (2019) and Neves et al. (2021) use arbitrarily determined, fixed length windows. Segmenting in the time domain assumes that neighboring observations have a similar impact on the predictions of the model. Intuitively we would want the "super segments" of a TS to capture homogeneous regions of behaviour. For example, given a TS recording an individual's activity over the period of a day, "super segments" correspond to the various activities (sleeping, walking). An arbitrary segmentation may result in non-homogeneous segments with conflicting properties or homogeneous regions spanning multiple segments. This segmentation challenge is encapsulated by **Open Question 1:** *"How do we meaningfully segment a TS into an interpretable representation where each "super segment" corresponds to a homogeneous region?"* The challenge of time series segmentation has been studied extensively throughout the change point detection literature which use changes in the statistical properties of a signal to determine the temporal indexes which indicate changes of underlying behaviour. Many change point detection methods segment via optimising over a specified cost function. The L1 and L2 cost functions detect changes in the median and mean of the signal respectively and more recently, cost functions have been proposed which detect changes in the statistical properties of the latent embedded signal Garreau and Arlot (2018). Techniques which efficiently return an ordering over the set of possible segmentations include dynamic programming (Guédon, 2013) and pruning (Killick et al., 2012). From the perspective of explainability, change point detection optimises a different segmentation objective than that outlined above. Consider two TS where the second is exactly the same as the first yet its temporal ordering has been reversed. Change point analysis may not detect a change point in between the two as the statistical properties and associated probability densities of both TS are the same. However, the reversed shape of the second TS corresponds to a pattern with contrasting semantic meaning (intuitively a temperature spike and then fall is different from a temperature fall and then spike). For this reason, we seek a segmentation approach which is sensitive to the changes in shape of the underlying time series. Gharghabi et al. (2017) define this challenge as "semantic sgementation" of a TS and propose a segmentation framework built on the intuition that homogeneous segments are likely to be composed of similarly shaped shorter sub-sequences. In contrast, Zhu et al. (2018) propose an alternative semantic segmentation framework by searching for "TS chains" which can be thought of as consecutive subsequences with similar shape. While these methods are effective on TS with super segments consisting of repeated shape patterns they cannot be applied to TS with segments composed of diverse shapes. This motivates the development of our proposed semantic segmentation framework *NNSegment* which relaxes the assumption that super segments contain solely similar sub-sequences, replacing it with the assumption that a time series will contain a mixture of similarly shaped super segments alongside anomalous super segments, allowing for the semantic segmentation of a wider class of time series.

Generating TS Samples: After conceptualising a TS into its interpretable "super segment" representation it is then a challenge to generate new samples its local neighbourhood. LIME does this by "turning off" concepts. To apply this intuition to "super segments", we must specify what it means to "delete" information from a TS. Perturbation approaches for image data include replacing the super pixel with some constant value, injecting noise, or blurring the image. These techniques can also be applied to TS: Neves et al. (2021) replace segments with mean valued segments and Guillemé et al. (2019) replace segments with randomly selected authentic segments from the original dataset. As is the case with image super pixels, we are generally interested in simulating natural TS samples, leading to more meaningful perturbations. Agarwal and Nguyen (2020) show how applying blur, zero and random perturbations result in visually unrealistic images which retain some proportion of the salient information contained in the super pixel. They propose an alternative generative perturbation, Inpainter, which replaces super pixels with realistic background content. Figure 1 shows the application of blur perturbation to a TS. Unlike perturbed images, there is no obvious visual implication of the perturbation approach on the realism of the resulting TS. This therefore raises **Open Question 2:** *"What constitutes realistic background content for TS?"*

Defining Locality: An important concept in LIME is the weighting over generated samples, used as input to the interpretable model, to encapsulate the intuition that samples closer to the instance to be explained should have more influence on the generated explanations. In LIME, this weighting is determined by the distance between each new sample and the instance to be explained in their interpretable representation. It has been shown by Garreau and Mardaoui (2021) that weights depend only on the number of inactivated super pixels in each generated sample. For a TS, this distance measure fails to take into account the global distance between the generated sample and the original instance. Figure 2 visualises an example TS $\mathbf{x}^0$ with six super segments and change points at indexes $[100, 200, 300, 350, 400]$ where $\sigma(\mathbf{x}^0) = [1,1,1,1,1,1]$. We generate three new samples as $\sigma(\mathbf{x}^1) = [0,1,1,1,1,1]$, $\sigma(\mathbf{x}^2) = [1,1,0,1,1,1]$, $\sigma(\mathbf{x}^3) = [1,1,1,1,0,1]$. Under Euclidean distance each of the generated samples would be equidistant from $\mathbf{x}_0$. However, we can see that perturbation to shorter length super segments ($\mathbf{x}^3$), are more similar to the original instance than longer perturbations. Moreover, even for "super segments" of equal length we can use Figure 2 to argue that $\mathbf{x}^2$ is closer to $\mathbf{x}^0$ than $\mathbf{x}^1$ as $\mathbf{x}^2$ maintains more motif structure than $\mathbf{x}^1$ which raises **Open Question 3:** *"How do we measure distance between two TS that accurately reflects local neighbourhood round* $\mathbf{x}$*?"*

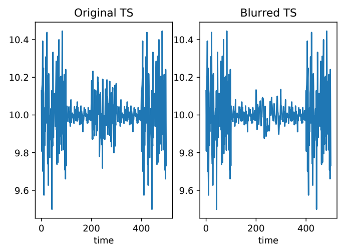

Figure 1: Effect of applying blur (via a Gaussian Filter) to an example "super segment" located at indexes $[200:300]$. Unable to confirm visually whether resulting perturbed TS is realistic.

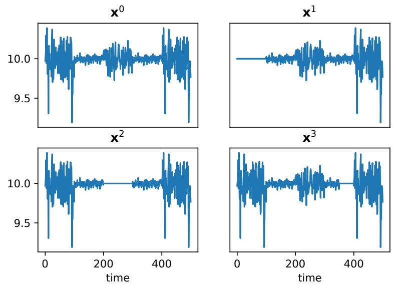

Figure 2: Example TS $\mathbf{x^0}$ and three generated samples with zero perturbations. $\mathbf{x^1}$ at index $[0:100]$; $\mathbf{x^2}$ at indexes $[200:300]$; $\mathbf{x^3}$ at index $[350:400]$.

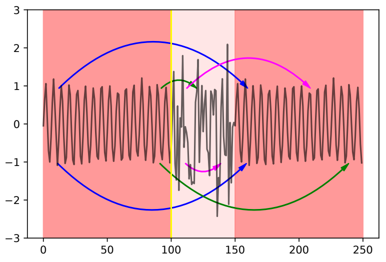

Figure 3: Intuition behind *NNSegment*. TS composed of motifs (shaded deep red) and anomalies (shaded pale red). Arrows connect current window with its nearest neighbour. Blue arrows at indexes 10 and 11 represent windows where adjacency holds. Magenta arrows at indexes 110 and 111 indicate adjacent windows where adjacency is broken. In this case $ws = 10$ and $\rho(\mathbf{w}_{100}, \mathbf{w}_{110}) > \rho(\mathbf{w}_{110}, \mathbf{w}_{120})$, indicating a cp at the beginning of the window, at index $i = 110$ which is added to the set of potential change points **cp**. Green arrows at indexes 90 and 91 also indicate windows where adjacency is broken. In this case, $\rho(\mathbf{w}_{90}, \mathbf{w}_{100}) > \rho(\mathbf{w}_{80}, \mathbf{w}_{90})$ thus $i = 100$ is added to **cp**. In this example, $\rho(\mathbf{w}_{90}, \mathbf{w}_{100}) > \rho(\mathbf{w}_{100}, \mathbf{w}_{110})$ implying that the cp at $i = 100$ is more likely than that at $i = 110$.

3 METHOD

We now present our framework for adapting LIME to TS. We first address each open question raised above and proceed to formalise our resulting explanation algorithm *LIMESegment*.

Meaningful Conceptualisation via ***NNSegment*****:** To address Open Question 1, we propose a segmentation algorithm which uses both the shape and the statistical properties of the TS to identify change points. Given a TS $\mathbf{x}$ of length T, our segmentation algorithm, *NNSegment* returns a set of change points $\mathbf{cp} = [cp_1; cp_2; ...; cp_{T'}]$, where T' is the number of change points in $\mathbf{x}$. Each cp_j $(j \in T')$ indicates the temporal index $i \in T$ of $\mathbf{x}$ where there is a change in behaviour. *NNSegment* is built on the assumption that neighbouring observations are likely to represent the same behaviour. We can thus group these neighbouring observations together into "super segments" which may be either repeating motifs or randomly occurring anomalies. Our segmentation approach is based on the assumption that regularly occurring motifs will be similar in shape and therefore use normalised cross correlation (Definition 1) which has had much success on pattern matching tasks (Zhao et al., 2006).

Definition 1 *Given a TS* $\mathbf{x} = [x_1; ...; x_T]$*, the similarity between two sub-sequences of length* ws *is denoted as* $\psi(\mathbf{x}_{s1}, \mathbf{x}_{s2})$ *where* $\mathbf{x}_{s1} = [x_{s1}; ...; x_{s1+ws}]$ *and* $\mathbf{x}_{s2} = [x_{s2}; ...; x_{s2+ws}]$*, can be defined as the normalised cross correlation function such that* $\psi(\mathbf{x}_{s1}, \mathbf{x}_{s2}) = \frac{E[\mathbf{x}_{s1} - \mu_{\mathbf{x}_{s1}}][\mathbf{x}_{s2} - \mu_{\mathbf{x}_{s1}}]}{\sigma_{\mathbf{x}_{s1}} \sigma_{\mathbf{x}_{s2}}}$ *where* μ *and* σ *represent sub-sequence mean and variance respectively*

NNSegment first decomposes $\mathbf{x}$ into overlapping windows of size ws which are then used to identify "super segments" in $\mathbf{x}$. $\mathbf{x}$ decomposed into its windowed representation is the set $\mathbf{w} = [\mathbf{w}_1; ...; \mathbf{w}_{T-ws}]$. For each window $\mathbf{w}_i = [x_i; ...; x_{i+ws}]$, the index of its nearest neighbour $w_{nn}(i)$ is determined by finding the window with which it shares minimal cross correlation: $w_{nn}(i) = argmin_j(\psi(\mathbf{w}_i \mathbf{w}_j)) : j \in T - ws$. After finding the nearest neighbours of all windows, our intuition, demonstrated in Figure 3, is that adjacent windows which belong to a homogeneous region of behaviour will follow an adjacency pattern (Definition 2). When adjacency is broken, we assume the behaviour of the TS has changed and the corresponding index represents the end of a super segment.

Definition 2 *Given a TS* $\mathbf{x}$ *and two adjacent windows at index* i *and* $i + 1$*. Adjacency holds at window index* i *if* $w_{nn}(i) + 1 = w_{nn}(i + 1)$

For anomalies, adjacent windows are likely to break the adjacency property. However, this does not necessarily signify the end of the "super segment". To account for these potentially erroneous change points we include a normalisation term which uses the statistical properties of the preceding and following window to determined if a true change point has occurred. We define the difference in statistical properties between two windows as $\rho(\mathbf{w}_i, \mathbf{w}_j) = |(\frac{\mu(\mathbf{w}_i)}{\sigma(\mathbf{w}_i)} - \frac{\mu(\mathbf{w}_j)}{\sigma(\mathbf{w}_j)})|$ where μ and σ represent window mean and variance respectively. For each window $\mathbf{w}_i$ which breaks the adjacency property, we calculate the values of $\rho(\mathbf{w}_i, \mathbf{w}_{i-ws})$ $\rho(\mathbf{w}_i, \mathbf{w}_{i+ws})$ to account for change points at the end of a window which have caused the break of adjacency. If $\rho(\mathbf{w}_i, \mathbf{w}_{i-ws}) > \rho(\mathbf{w}_i, \mathbf{w}_{i+ws})$ the change point is more likely to have occured at the beginning of the window whereas if $\rho(\mathbf{w}_i, \mathbf{w}_{i-ws}) < \rho(\mathbf{w}_i, \mathbf{w}_{i+ws})$ the change point is more likely to have occurred at the end of the window. We sort the vector of $\rho(.)$ and return the index i of $\mathbf{w}_i$ corresponding to the greatest T' changes in statistical properties as our vector **cp**. Under the assumption that every observation x_i either forms part of a motif or anomaly, we formally introduce *NNSegment* in Algorithm 1 which requires parametrising with window size, ws and the number of user-specified change points T'.

Perturbing a TS with Realistic Background Content: To address Open Question 2 we employ

Algorithm 1: Nearest Neighbour Segment (*NNSegment*)

```
Input  : TS $\mathbf{x}$, window length $ws$, number of
         Change Points $T'$
Output: cp: vector of change points in $\mathbf{x}$
$\mathbf{w} \leftarrow [\mathbf{w}_i] : i \in T - ws$;
$\mathbf{w_{nn}} \leftarrow argmin_j(\psi(\mathbf{w}_i\mathbf{w}_j)) : i, j \in T - ws$;
cp $\leftarrow []$ ;
for $i \in T - ws$ do
    if $w_{nn}(i+1) \neq w_{nn}(i) + 1$ then
        if $|(\rho(\mathbf{w}_i, \mathbf{w}_{i-ws}))| > |(\rho(\mathbf{w}_i, \mathbf{w}_{i+ws}))|$
         then
            cp $\leftarrow$ cp $+ [i]$
        end
        ; else
            cp $\leftarrow$ cp $+ [i + ws]$
        end
        ;
    end
end
cp $\leftarrow sort($cp$)$ by $\rho(.)$;
cp $\leftarrow [cp_1; cp_2; ...; cp_{T'}]$;
```

findings from harmonic analysis: Any TS $\mathbf{x}$ can be represented as a composition of harmonic oscillations in the frequency domain such that $\mathbf{x} = \mathbf{x}_\omega = \int_{-\infty}^{\infty} x_t e^{-2\pi t\omega} dt$. T_ω can be viewed as a distribution over the frequency content of the signal. Maxima in the frequency domain reflect a high proportion of the signal oscillating at that frequency. It has been shown that realistic background content represents a global property of an image and is not necessarily the local low frequency content but the most commonly occurring global frequency information (Agarwal and Nguyen, 2020). A TS' frequency distribution varies considerably over time. Applying a low pass filter to, or blurring a segment, is therefore not necessarily a true reflection of removing the salient signal content. Instead we propose replacing a "super segment" with a background content segment, artificially generated by identifying the frequency band which has the highest representation, with lowest variance in the original signal.

To understand how the spectral density of a TS changes over time we use the Discrete Short Time Frequency Transform (STFT). Given a TS $\mathbf{x} = [x_1; ...; x_T]$ where $\mathbf{x}$ is considered a discrete time representation of the underlying phenomenon, the STFT converts $\mathbf{x}$ into its time-frequency representation by taking the Fourier transform of $\mathbf{x}$ multiplied by a sliding window which is non-zero only for a fixed small length ws: $STFT(\mathbf{x}, ws, \omega) = \Sigma_{-\infty}^{\infty} \mathbf{x} w(T - ws) DFT_T$ where $DFT_T = e^{-i\omega T}$ represents the Fourier transform, ω is

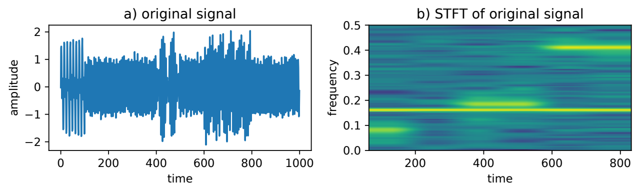

Figure 4: Intuition behind *RBP*. Original signal (a) composed of background signal and varying frequency sine waves at indexes: [0 : 100], [400 : 500] and [600 : 800]. b) shows the spectrogram obtained by applying STFT to the original signal. The spectogram captures the background signal which remains constant through time as well as the shorter length "content" sine waves at their respective frequencies.

the frequency parameter and $w(.)$ is a window function parametrised by window size. The STFT, when applied to a discrete TS results in a matrix which records magnitude and phase for each point in time and frequency. We are interested in filtering the signal by selecting only the background content. Given $STFT(\mathbf{x})$ we therefore find the most persistent frequency by first selecting only the magnitude response as $|f_t|$ and then finding the frequency band which has the highest value over time with minimal variance: $F_{persist} = argmax_f \frac{\mu(|f_t|)}{\sigma(|f_t|)} : f, t \in \{STFT(\mathbf{x})\}$ where $\frac{\mu(|f_t|)}{\sigma(|f_t|)}$ is the mean magnitude response normalised by its standard deviation of a selected frequency band over time. To use this background content to meaningfully perturb our original TS we convert $F_{persist}$ into the original time domain via the inverse STFT, from which, the relevant segments of background content can be chosen to replace parts of the original signal. Our realistic background perturbation algorithm, *RBP* is shown in Algorithm 2 and Figure 4 shows the STFT applied to an example TS, demonstrating our background frequency intuition.

Measuring Distance between two TS: To address Open Question 3, we employ Dynamic Time Warping (DTW), introduced by Bellman and Kalaba (1959) to address the limitations of the Euclidean distance in measuring the similarity between two TS. DTW (formally defined in the Supplementary Material) ignores both global and local shifts in the time dimension to give a more accurate similarity between two TS. For generating explanations, and more specifically, generating the weighting between the instance to be explained and the generated samples, this property of DTW is very useful. For the example samples shown in Figure 2, we obtain $d_{DTW}(\mathbf{x}^0, \mathbf{x}^1) = 16$, $d_{DTW}(\mathbf{x}^0, \mathbf{x}^2) = 10$ and $d_{DTW}(\mathbf{x}^0, \mathbf{x}^3) = 1$ which captures the intuition that perturbations applied to the

Algorithm 2: Realistic Background Perturbation (*RBP*)

```
Input  : TS $\mathbf{x}$ of length $T$, window size $ws$,
         frequency parameter $\omega$, change point
         indexes $\mathbf{cp}$, perturbation segments $\sigma(z)$
Output: Perturbed TS $\mathbf{z}$
$\mathbf{x}_{stft} \leftarrow STFT(\mathbf{x}, \omega, ws)$;
$F_{persist} \leftarrow \mathbf{x}_{stft}[argmax_f \frac{\mu(|f_t|)}{\sigma(|f_t|)}] : f, t \in \mathbf{x}_{stft}$;
$R = STFT^{-1}(F_{persist}, \omega, ws)$;
$\mathbf{z} \leftarrow \mathbf{x}$;
for $i$ in $T'$ do
    if $\sigma(x_i) == 0$ then
        $\mathbf{z}[cp_i : cp_{i+1}] \leftarrow R[cp_i : cp_{i+1}]$
    end
end
```

more significant "super segments" of the original TS should represent a more dissimilar resulting sample.

LIMESegment: Algorithm 3 details our framework, *LIMESegment*, for generating local TS explanations. **Problem Statement:** Given an example TS to be explained $\mathbf{x}$ and an underlying black box classifier $f : R^T \rightarrow R$, alongside the predicted label of $\mathbf{x}$, $Y_{\mathbf{x}} = f(\mathbf{x})$, we build a surrogate model g in the locality of $\mathbf{x}$ to generate explanations in the interpretable domain $\sigma(\mathbf{x})$.

We first transform $\mathbf{x}$ into its interpretable representation $\sigma(\mathbf{x}) = \mathbf{1}^{T'}$ which corresponds to a vector of ones for each "super segment" found by *NNSegment*. We generate n random samples $\mathbf{Z} = [\mathbf{z_1}; ...; \mathbf{z_n}]$ in the locality of $\sigma(\mathbf{x})$ according to a Bernoulli sampler. Each coordinate $\sigma(z_i)$ ($i \in T'$) of $\sigma(\mathbf{z}_j)$ is i.i.d Bernoulli distributed with parameter $\frac{1}{2}$. We use *RBP* to convert each $\sigma(\mathbf{z}_j)$ into TS $\mathbf{z}_j$. Given our transformed TS dataset $\mathbf{Z}$ we can obtain predicted sample labels $Y_{\mathbf{Z}} = f(\mathbf{Z})$. To determine the distances to be used as weightings π to the surrogate model g we first normalise compute $DTW(x, \mathbf{Z})$ between $\mathbf{x}$ and each $\mathbf{z}_j$ ($j \in n$). These distances are then z-normalised $DTW_{z-norm}(\mathbf{Z})$ and passed to an exponential kernel with scale parameter l, $\pi = exp(\frac{DTW_{z-norm}(\mathbf{Z})^2}{l})$. We use Linear Ridge Regression as our surrogate model g and interpret the feature weight vector $\mathbf{w} = [w_1; ...; w_{T'}]$ as our "super segment" importances and resulting explanations. We argue that the combination of a meaningful segmentation algorithm *NNSegment* and realistic perturbation *RBP* alongside the use of DTW results in a more appropriate adaptation of LIME to TS than existing methods (Guillemé et al., 2019; Neves et al., 2021), which we evaluate extensively Section 4. A weakness of our framework is the necessity to parametrise both *RBP* and *NNSegment* with appropriate window sizes, a common problem in TS mining. Additionally, *NNSegment*, does not address the frequency coherence assumption where neighbouring frequency bands have similar impact on the behaviour of the black box. However, we argue that segmentation in the time domain is the most human-interpretable way of conceptualising a TS and trade off this assumption for interpretability. To compute NNSegment we use the STUMPY library (Law, 2019) which runs in O(n^2) if n is the length of the TS. However, the STUMPY library can be approximated with SCRIMP where 1% of the pairwise similarities are computed. The *RBP* algorithm is dependent on the STFT which operates in O($n \log n$) time. While the DTW Algorithm runs in O(n^2) we made use the FastDTW approximation (Salvador and Chan, 2007) algorithm which operates in O(n) time. *LIMESegment* explores what meaningful TS explanations *could* look like and leave amortisation and methods which address longer length, multivariate TS with missing values for future work. All project code and experiments can be found at https://github.com/TortySivill/LIMESegment.

Algorithm 3: *LIMESegment*

```
Input  : TS $\mathbf{x}$, Black Box Classier $f$, no. of
         generated samples to generate $n$
Input  : Bernoulli Sampler $B$, Linear Model
         $RidgeRegression$
Input  : $NNSegment$, window size $ws$, Number
         of Change Points $T'$
Input  : $DTW$, $RBP$, window size $ws$, frequency
         parameter $\omega$
Output: Segment Importances $\mathbf{w}$
$\mathbf{cp} \leftarrow NNSegment(\mathbf{x}, ws, T')$;
$\sigma(\mathbf{x}) \leftarrow [1]^{T'}$;
for $i \in n$ do
    $\sigma(\mathbf{z}_j) \leftarrow [B_0, ..., B_{T'}]$;
    $\mathbf{z}_j \leftarrow RBP(\mathbf{x}, ws, \omega, \mathbf{cp}, \sigma(\mathbf{z}_j))$;
end
$Y_{\mathbf{Z}} \leftarrow f(\mathbf{Z})$;
$DTW_{z-norm}(\mathbf{Z}) \leftarrow \frac{DTW(\mathbf{x},\mathbf{Z}) - \mu_{DTW(\mathbf{x},\mathbf{Z})}}{\sigma_{DTW(\mathbf{x},\mathbf{Z})}}$
$\pi \leftarrow exp(\frac{-DTW_{z-norm}(\mathbf{Z})^2}{l})$;
$\mathbf{w} \leftarrow RidgeRegression(\sigma(\mathbf{Z}), Y_{\mathbf{Z}}, \pi)$
```

4 EXPERIMENTAL EVALUATION

To evaluate the explanations generated by *LIMESegment*, we first evaluate *NNSegment*, *RBP* and the DTW distance measure separately in their capability of addressing Open Questions 1,2,3 raised in Section 2. We then evaluate the explanations generated by *LIMESegment* on a variety of TSC tasks. Full experimental configuration details can be

	NNSegment	FLUSS	DynP	BotUp
F-Score				
Synthetic	**0.60**	0.00	0.23	0.28
Apnea	**0.42**	0.16	0.20	0.25
Hausdorff				
Synthetic	**0.05**	0.76	0.37	0.34
Apnea	**0.30**	0.40	0.74	0.40

Table 1: F-score and Hausdorff distance from applying *NNSegment*, FLUSS, Dynamic Programming with L2 Cost and Bottom Up with L2 cost segmentation algorithms to the Synthetic Dataset and Apnea Dataset. Higher F-score and lower Hausdorff Distance reflects better segmentation.

	Original	*RBP*	Zero	Random	Blur
Acc	1.0	**0.36**	0.49	0.52	0.47

Table 2: Classification accuracy after applying various perturbation methods to Synthetic TS.

	Euclidean	DTW
Simple Synthetic	0.11	**0.49**
Complex Synthetic	0.17	**0.22**
ECG200	0.55	**0.63**

Table 3: Mean RSSI after applying *LIMESegment* with DTW and Euclidean based locality weights

found in the Supplementary Material.

Semantic Segmentation: To evaluate *NNSegment* in addressing Open Question 1, we use Hausdorff Distance and F-Score. The Hausdorff Distance (HD) is the greatest temporal distance between a true change point cp^t and predicted change point cp^p for a given TS of length T and change point vector **cp** of length T': $\text{HD}(cp^t, cp^p) = \frac{max(|cp_i^t - cp_i^p|)}{T}$ for $i \in T'$ (Aminikhanghahi and Cook, 2017). The F-score is the harmonic mean of the Precision and Recall of the predicted change points: $\text{FScore}(cp^p, cp^p) = \frac{cp^t \cap cp^p}{cp^t \cap cp^p + \frac{(cp^t \setminus cp^p + cp^p \setminus cp^t)}{2}}$. We compare *NNSegment* with FLUSS, the semantic segmentation algorithm closest to our work (Gharghabi et al., 2017) as well as two benchmark change point detection algorithms: DynP, Dynamic Programming with L2 Cost Function and BotUp, Bottom Up segmentation with L2 Cost Function. Both change point methods are implemented using the Ruptures Python Framework (Truong et al., 2020). We use synthetic TS and and examples from the Apnea-ECG dataset (Penzel et al., 2000) for evaluation. The Synthetic TS are generated by concatenating six super segments. Each segment and its respective frequency composition is taken to represent a homogeneous region of activity. The Apnea dataset contains ECG recordings of 70 participants with labelled apnea events. Table 1 shows how *NNSegment* outperforms FLUSS and both Change Point detection algorithms across both metrics and both datasets evaluated. We attribute the superiority of *NNSegment* over FLUSS to the difference in intuition driving both algorithms. Unlike *NNSegment*, FLUSS assumes all similar behaviour occurs in the same segment and fails to take into account for repeating patterns. We attribute the superiority of *NNSegment* over *DynP* and *BotUp* to the high proportion of motifs in both datasets. We note that *NNSegment* is not designed to function as well on datasets with a high proportion of anomalies and motivate this for future work.

Realistic Background Perturbation: To evaluate *RBP* on Open Question 2 we ask the following: How capable is *RBP* at generating background content? How capable is *RBP* at generating realistic new samples? To evaluate the former we use the intuition of Agarwal and Nguyen (2020) adapted to TS: after applying a perturbation to a test set, the more successful the perturbation, the worse the classification performance. We generate a synthetic binary TS dataset where each TS has five "super segments". The class-wise difference is contained solely in the final super-segment. We select a 1D Convolutional Neural Network as our black box. We perturb the final segment of each evaluation TS with either *RBP*, blurring, zeroed, or random values and obtain classification accuracy on each perturbed dataset. Results are shown in Table 2 where we can see a significant accuracy decrease following all perturbations. However, the accuracy decrease is most significant for *RBP*. To evaluate whether the TS perturbed *RBP* produces more realistic TS than blurring, noise and zero perturbations we build on the theory of Chen et al. (2020), who show how unrealistic samples allow an adversary differentiate between data points coming from the input distribution and instances generated via perturbation. To test how realistically *RBP* generates new samples we train a new classifier, a 1D CNN, on a binary class synthetic TS dataset. Class A contains TS with five "super segments". Class B contains the TS of Class A after undergoing a perturbation. We train the classifier on the dataset and compare the validation loss curve for varying perturbation strategies including *RBP*, blurring, zeroed, random. The more realistic the perturbation, the more difficult it will be for the classifier to learn and generalise. Results are displayed in Figure 5 which confirms our earlier claim that blur, zero and random perturbation result in unrealistic TS.

Locality: To evaluate DTW on Open Question 3, we assume that a failure to correctly sample in the locality of **x** results in unstable LIME explanations. To measure the stability of explanations we adapt the ex-

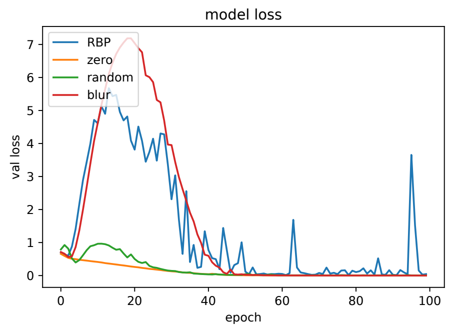

Figure 5: Validation loss of training neural network classifier on synthetic datasets composed of varying perturbation methods. A validation curve which falls quickly to near zero indicates that the model has successfully learned to separate each class and has generalised well to the validation set. *RBP* does not have a smoothly decreasing loss curve and has not reached stable low loss which indicates that the black box is unable to differentiate between perturbed and non-perturbed TS.

planation stability metric of Visani et al. (2020) to introduce Ranked Segment Stability Index (RSSI) as the proportion of concordant (equal) pairs out of all $\frac{n(n-1)}{2}$ pairs of segment importance vectors for a given TS after running the explanation algorithm for n iterations. Full description of RSSI is included in the Supplementary Materials. To show how DTW improves stability of LIME explanations we generate two synthetic binary class TS datasets. Across both datasets each TS has five "super segments". In "simple synthetic", the class difference occurs in the final super segment of each TS. In "complex synthetic" the class difference is spread evenly across the initial and final super segments. We also evaluate the stability of *LIMESegment* on the ECG200 dataset from the UCR TS repository (Chen et al., 2015). We use a 1D CNN as our black box classifier. For each TS we obtain two sets of segment importances by running *LIMESegment* using DTW as well as running *LIMESegment* with Euclidean distance. To establish the stability of each set of segment importances we repeat this process 50 times for each TS. Table 3 shows RSSI of explanations under DTW is greater than those of Euclidean distance implying that comparing the similarity of generated TS to the original TS in their raw form according to DTW results in more stable explanations.

To evaluate *LIMESegment* as a tool for generating useful explanations we evaluate: **1) How faithful is** ***LIMESegment*** **to the original black box classfier?** We define Faithfulness as the decrease in classification confidence of the black-box when removing the most important "super-segment" from the TS as returned by its explanation. If *LIMESegment* has correctly identified the most important "super-segment", its removal will result in large decrease in prediction confidence. We adopt the same approach as Neves et al. (2021) whereby removing a selected segment from the TS corresponds to replacing it with reversed segment values. To measure the faithfulness of *LIMESegment* we measure the mean drop in prediction probability after segment removal for each TS in the test set. **2) How robust is** ***LIMESegment*** **to small variations in the input space?** Ideally, an end-user would want their explanations to be robust to small changes in the input space such that anomalous observations don't influence the resulting explanation. In this work we measure Robustness by observing the difference in explanation generated for a TS before and after it has been perturbed with randomly generated noise. We report the proportion of TS whose explanations are unchanged following perturbation as our measure of Robustness.

To evaluate Robustness and Faithfulness of *LIMESegment* we use three black box classifiers: K Nearest Neighbour, a 1D CNN and the state-of-the-art TS classification LSTM proposed by Karim et al. (2019). We train each classifier on twelve randomly selected binary TS datasets from the UCR repository (Chen et al., 2015). We evaluate the performance of *LIMESegment* against the performance of the LIME TS adaptations of (Guillemé et al., 2019), and Neves et al. (2021). Full details of the experimental setup can be found in the Supplementary Material. For each dataset we measure the mean Faithfulness and Robustness across classifiers alongside the standard deviation. Table 4 reports five individual datasets alongside the mean Faithfulness and Robustness across all twelve datasets evaluated (all). Explanations generated under *LIMESegment* are significantly more Robust than the framework of Neves et al. (2021) and Guillemé et al. (2019) for all datasets evaluated. While *LIMESegment* is also more Faithful than Neves et al. (2021) and Guillemé et al. (2019), individual dataset results are nuanced: *LIMESegment* applied to instances of the Strawberry dataset achieves significantly superior Faithfulness across all three classifiers than the frameworks of Guillemé et al. (2019) and Neves et al. (2021). We observe that generally, the most important segment as returned by *LIMEsegment* occurs in the middle of these TS and is roughly of length 20. de Abreu Fontes et al. (2021) introduce a wavelength importance classifier and show how the Strawberry dataset can be accurately classified by retaining just 4% of the original signal. Our result confirms this finding and shows how *LIMEsegment* successfully identifies the

	strawberry	handout	yoga	ecg200	chinatown	all
			Faithfulness			
L	**0.35**±0.10	0.08±0.05	**0.10**±0.06	**0.20**±0.18	**0.05**±0.09	**0.10**±0.06
G	0.05±0.04	0.05±0.06	0.06±0.03	0.13±0.18	0.03±0.06	0.04±0.05
N	0.07±0.04	**0.16**±0.14	0.05±0.04	0.16±0.14	**0.05**±0.09	0.06±0.05
			Robustness			
L	**1.00**±0.20	**0.40**±0.20	**0.70**±0.36	**0.88**±0.20	**0.98**±0.03	**0.74**±0.14
G	0.20±0.35	0.00±0.00	0.60±0.53	0.33±0.58	0.67±0.58	0.42±0.22
N	0.00±0.00	0.10±0.17	0.52±0.40	0.45±0.05	0.67±0.58	0.34±0.23

Table 4: Mean and Standard Deviation of Faithfulness (F) and Robustness (R) of *LIMESegment* (L), Guillemé et al. (2019) (G) and Neves et al. (2021) (N) explanations after training KNN, CNN and LSTM on 12 datasets from UCR repository (Chen et al., 2015) (all) alongside individual results of five datasets. As L and G require user defined segmentation we report the best results obtained with segment length of 5%, 10%, or 20% of TS length.

most significant segment of the TS whose removal results in a significant decrease in performance, exemplifying the benefits of meaningful over arbitrary segmentation.

LIMESegment in Practice: We now present *LIMESegment* in a healthcare setting to exemplify the insights TSC explanations offer. We apply *LIMESegment* to the prediction of in-hospital survival post-septic shock onset using temperature fluctuations, where it has been shown in the literature how fever in septic shock patients is strongly associated with lower mortality probability (Sundén-Cullberg et al., 2017). For this study, the "Sepsis Cohort" was selected as a subset of the MIMIC III dataset according to the method of Komorowski et al. (2018). The MIMIC III Sepsis Cohort includes all z-normalised vital sign observations 24 hours before until 48 hours after presumed onset of sepsis.

We train a 1D CNN on 1130 TS and randomly select TS from a test set to generate explanations via *LIMESegment*. Four example TS and associated segment importances are shown in Figure 6. For the True Negative (individual dies) and True Positive (individual survives), the segment following sepsis onset is most influential for the resulting classification which aligns with sepsis research reporting elevated peak temperature in the first 24 hours following sepsis onset is associated with decreased in-hospital mortality (Young et al., 2012), This kind of explanation guided observation demonstrates how *LIMESegment* could be used to offer medical insight to an end-user. Figure 6 shows how, for the False Negative example, the most influential segment in its incorrect classification is of similar shape to the True Negative's most influential segment: temperature significantly dips and sharply rises, giving insight into why this sample was incorrectly classified by the black box. To

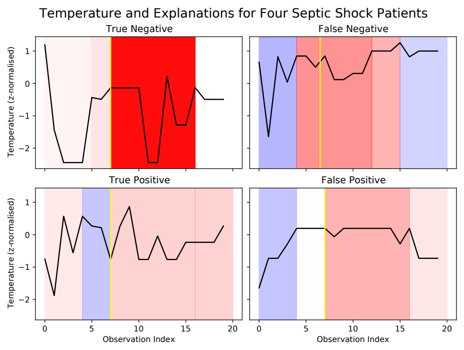

Figure 6: Each TS is labelled as either a True Negative or Positive, where the black box has correctly classified the instance or, as a False Positive or Negative where the black box has misclassified the sample. Each "super segment" as returned by *LIMESegment* is shaded either blue or red. Red shading indicates the segment importance supports the black box prediction and blue indicates the segment importance contradicts the black box prediction. Opacity indicates greater segment importance. The yellow vertical line indicates sepsis onset for each individual. For both correctly classified instances *LIMESegment* has detected the time of sepsis onset in its segmentation.

further evaluate the *LIMESegment* algorithm on the Sepsis Cohort we compare Faithfulness and Robustness of *LIMESegment* with Guillemé et al. (2019) and Neves et al. (2021) on each test sample. *LIMESegment* obtains Faithfulness of 0.29 and Robustness of 1.0, Guillemé et al. (2019) obtains Faithfulness of 0.02 and Robustness of 0.90, and Neves et al. (2021) obtains Faithfulness of 0.12 and Robustness of 0.72, demonstrating how *LIMESegment* outperforms state-of-the-art TS LIME adaptations on real-world data.

5 CONCLUSION

We present our adaptation of LIME to TS by combining solutions to open challenges in TS mining into an explanation framework. Our segmentation algorithm, *NNSegment* outperforms existing work in finding homogeneous regions of TS activity. Our perturbation algorithm, *RBP*, outperforms existing standard perturbation approaches and we have shown how the use of DTW instead of Euclidean distance results in more stable explanations. *LIMESegment* has been shown to produce more Faithful and Robust explanations than the existing state-of-the-art adaptation of LIME to TS (Guillemé et al., 2019; Neves et al., 2021). In future work we wish to generate realistic local TS using generative methods; adapt *LIMESegment* for multivariate TSC and evaluate *LIMESegment* with a user-study to understand how humans interpret TS explanations.

Acknowledgements

We thank Dr James Goulding, the Bristol Intelligent Systems Laboratory and the wider Turing Institute community for their advice during the writing of this paper. We also thank the anonymous reviewers for insightful comments during the review process. This work was supported by an Alan Turing Institute PhD Studentship funded under EPSRC grant EP/N510129/1.

References

C. Agarwal and A. Nguyen. Explaining image classifiers by removing input features using generative models. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

S. Aminikhanghahi and D. J. Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

R. Bellman and R. Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4 (2):1–9, 1959.

J. Chen, Y. Li, X. Wu, Y. Liang, and S. Jha. Robust out-of-distribution detection for neural networks. *arXiv preprint arXiv:2003.09711*, 2020.

Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The ucr time series classification archive, July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

J. de Abreu Fontes, M. J. Anzanello, J. B. G. de Brito, G. B. Bucco, F. S. Fogliatto, and F. Puglia. Combining wavelength importance ranking to the random forest classifier to analyze multiclass spectral data. *Forensic Science International*, page 110998, 2021.

P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):1–34, 2012.

J. Faouzi and H. Janati. pyts: A python package for time series classification. *Journal of Machine Learning Research*, 21(46):1–6, 2020. URL `http://jmlr.org/papers/v21/19-763.html`.

D. Garreau and S. Arlot. Consistent change-point detection with kernels. *Electronic Journal of Statistics*, 12(2):4440–4486, 2018.

D. Garreau and D. Mardaoui. What does lime really see in images? *arXiv preprint arXiv:2102.06307*, 2021.

S. Gharghabi, Y. Ding, C.-C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh. Matrix profile viii: domain agnostic online semantic segmentation at superhuman performance levels. In *2017 IEEE international conference on data mining (ICDM)*, pages 117–126. IEEE, 2017.

Y. Guédon. Exploring the latent segmentation space for the assessment of multiple change-point models. *Computational Statistics*, 28(6):2641–2678, 2013.

M. Guillemé, V. Masson, L. Rozé, and A. Termier. Agnostic local explanation for time series classification. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 432–439. IEEE, 2019.

A. Ignatiev. Towards trustable explainable ai. In *IJCAI*, pages 5154–5158, 2020.

F. Karim, S. Majumdar, and H. Darabi. Insights into lstm fully convolutional networks for time series classification. *IEEE Access*, 7:67718–67725, 2019.

R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

M. Komorowski, L. A. Celi, O. Badawi, A. C. Gordon, and A. A. Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine*, 24(11):1716–1720, 2018.

J. Labaien, E. Zugasti, and X. De Carlos. Contrastive explanations for a deep learning model on time-series data. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 235–244. Springer, 2020.

S. M. Law. STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *The Journal of Open Source Software*, 4(39):1504, 2019.

B. Lim and S. Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

I. Neves, D. Folgado, S. Santos, M. Barandas, A. Campagner, L. Ronzio, F. Cabitza, and H. Gamboa. Interpretable heartbeat classification using local model-agnostic explanations on ecgs. *Computers in Biology and Medicine*, 133:104393, 2021.

R. T. Olszewski. *Generalized feature extraction for structural pattern recognition in time-series data.* Carnegie Mellon University, 2001.

T. Penzel, G. B. Moody, R. G. Mark, A. L. Goldberger, and J. H. Peter. The apnea-ecg database. In *Computers in Cardiology 2000. Vol. 27 (Cat. 00CH37163)*, pages 255–258. IEEE, 2000.

M. T. Ribeiro, S. Singh, and C. Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

C. Rooke, J. Smith, K. K. Leung, M. Volkovs, and S. Zuberi. Temporal dependencies in feature importance for time series predictions. *arXiv preprint arXiv:2107.14317*, 2021.

S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

J. Sundén-Cullberg, R. Rylance, J. Svefors, A. Norrby-Teglund, J. Björk, and M. Inghammar. Fever in the emergency department predicts survival of patients with severe sepsis and septic shock admitted to the icu. *Critical care medicine*, 45(4):591–599, 2017.

A. A. Taha and A. Hanbury. Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool. *BMC medical imaging*, 15(1):1–28, 2015.

S. Tonekaboni, S. Joshi, K. Campbell, D. K. Duvenaud, and A. Goldenberg. What went wrong and when? instance-wise feature importance for time-series black-box models. *Advances in Neural Information Processing Systems*, 33, 2020.

C. Truong, L. Oudre, and N. Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

G. Visani, E. Bagli, F. Chesani, A. Poluzzi, and D. Capuzzo. Statistical stability indices for lime: obtaining reliable explanations for machine learning models. *Journal of the Operational Research Society*, pages 1–11, 2020.

C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1317–1322. Ieee, 2016.

P. J. Young, M. Saxena, R. Beasley, R. Bellomo, M. Bailey, D. Pilcher, S. Finfer, D. Harrison, J. Myburgh, and K. Rowan. Early peak temperature and mortality in critically ill patients with or without infection. *Intensive care medicine*, 38(3):437–444, 2012.

F. Zhao, Q. Huang, and W. Gao. Image matching by normalized cross-correlation. In *2006 IEEE international conference on acoustics speech and signal processing proceedings*, volume 2, pages II–II. IEEE, 2006.

Y. Zhu, M. Imamura, D. Nikovski, and E. J. Keogh. Time series chains: A novel tool for time series data mining. In *IJCAI*, pages 5414–5418, 2018.

Supplementary Material:
LIMESegment: Meaningful, Realistic Time Series Explanations

A DTW ALGORITHM

A central component of *LIMESegment* is the use of the Dynamic Time Warping (DTW) distance measure to weight the surrogate model. DTW is formally specified in Algorithm 4.

Algorithm 4: Dynamic Time Warping (DTW)

```
Input  : TS $\mathbf{x}^1 = \{x_1, ..., x_T\}, \mathbf{x}^2 = \{x_1, ..., x_{T'}\}$
Output: $Dist_{DTW}(\mathbf{x}^1, \mathbf{x}^2)$
$DTW \leftarrow array[0:T, 0:T']$;
$w \leftarrow max(w, |(T - T')|)$;
for $i \in T]$ do
    for $j \in [T']$ do
        $DTW[i,j] \leftarrow \infty$
    end
end
$DTW[0,0] \leftarrow 0$ for $i \in T]$ do
    for $j \in [max(1, i-w) : min(T', i+w)]$ do
        $DTW[i,j] \leftarrow 0$
    end
end
for $i \in T$ do
    for $j \in [max(1, i-w) : min(T', i+w)]$ do
        $c \leftarrow d(\mathbf{x}^1[i], \mathbf{x}^2[j]$;
        $DTW[i,j] \leftarrow c + min(DTW[i-1,j], DTW[i,j-1], DTW[i-1,j-1])$
    end
end
return $DTW[T, T']$
```

B EXPERIMENT DETAILS

B.1 Semantic Segmentation

To evaluate our proposed *NNSegment* segmentation algorithm we compare its performance with the segmentation algorithm FLUSS (Gharghabi et al., 2017) which requires as input the Matrix Profile for a given time series (TS). The Matrix Profile, an algorithm proposed by Yeh et al. (2016) is a vector that stores the (z-normalized) Euclidean distance between any subsequence within a TS and its nearest neighbor. We also evaluated *NNSegment* against the Dynamic Programming Search and the Bottom Up Search change point detection algorithms. To evaluate segmentation algorithms we use F-score and Hausdorff Distance which were selected as they are commonly used in the image processing literature to evaluate semantic segmentation algorithms (Taha and Hanbury, 2015). The datasets selected for evaluation are a Synthetic dataset and the Apnea ECG dataset (Penzel et al., 2000). TS of the Synthetic dataset were generated via Algorithm 5. We generated 50 Synthetic TS in total, each of length 500. When applied to Synthetic TS, *NNSegment* was parametrised with $ws = 10, cp = 5$. FLUSS was parametrised with $ws = 10, R = 5$. DynP and Botup were both paremetrised with the L2 cost function. The Apnea ECG dataset contains 70 records, split into a test and train set each of size 35. Each sample records a continuous ECG signal of a respective human participant sleeping for a duration of seven to ten hours. Each sample is accompanied with a set of human-labelled annotations recording the time of onset and duration of apnea events. We take each apnea event and the intervals of non-apena sleep between as homogeneous regions of activity. For simplicity, each TS was aggregated from recording ten samples per second to one sample per second. For TS of the Apnea dataset, *NNSegment* was parametrised with $ws = 60$ and *cp* set to the number of annotations for each sample. FLUSS was parametrised with $ws = 60$ and R was determined according to the number of annotations for each sample.

Algorithm 5: Synthetic Segmentation TS Generation

```
Output: TS $T$
$\mu, \sigma \leftarrow 10, 0.2$ ;
$s1 \leftarrow Normal(\mu, \sigma, 100)$;
$\mu, \sigma \leftarrow 10, 0.03$ ;
$s2 \leftarrow Normal(\mu, \sigma, 100)$;
$\mu, \sigma \leftarrow 10, 0.1$ ;
$s3 \leftarrow Normal(\mu, \sigma, 100)$;
$T \leftarrow s1 + s2 + s3 + s2 + s1$
```

B.2 Realistic Background Perturbation

To evaluate the proposed *RBP* algorithm we first measure the performance decrease after perturbing TS with *RBP*, zero, blur and random. We begin by generating a Synthetic binary TS dataset. Class A is created via Algorithm 6 generating 500 samples in total. Class B is created via Algorithm 7 generating 500 samples in total. We split the resulting Synthetic Dataset into a train and test set each of size 500. On the train set, we train a 1D convolutional neural network (CNN) of architecture outlined in Figure 7 for 100 epochs, batch size of 64 and validation split of 0.4 using the ADAM optimiser and sparse categorical cross entropy as the loss function. We obtain an accuracy of 1.0 on our test set. To evaluate the performance decrease following each perturbation we perturb each individual TS in the test set from index 400 to 500 with RBP, $\textit{ZeroPerturb}(TS[400 : 500) = [0]^{100}$, $\textit{RandomPerturb}(TS[400 : 500]) = [\textit{RandInt}(-100, 100)]^{100}$ or $\textit{BlurPerturb}(TS[400 : 500]) = \textit{GaussianFilter}(TS[400 : 500], \sigma)$ with $\sigma = 0.1$ We evaluate the model accuracy on each perturbed test set and report this as our measure of performance decrease. To evaluate *RBP* at generating realistic TS we create a new binary dataset for each perturbation type. Class A is generated via Algorithm 6 and Class B is generated via Algorithm 6 and then perturbed as above. Each dataset contains 500 TS samples which are all used to train the CNN as depicted in Figure 7 under the same configuration as above. We record and plot the validation loss after each epoch and use this to compare the realism of each perturbation strategy.

Algorithm 6: Synthetic Class A

```
Output: TS $T$
$\mu, \sigma \leftarrow 0, 0.35$ ;
$noise \leftarrow Normal(\mu, \sigma, 100)$;
$freq \leftarrow \sin([0; ...; 500])$;
$bg \leftarrow freq + noise$ ;
$tc \leftarrow sin([(0; ...; 100] * 20)$;
$fs \leftarrow [0]^{500}$;
$fs[400 : 500] \leftarrow tc$;
$T \leftarrow bg + fs$
```

B.3 Locality

To evaluate the use of Dynamic Time Warping (DTW) distance against Euclidean distance in measuring the similarity between TS instances we construct two synthetic datasets: Simple Synthetic (Class A: Algorithm 6 Class B: Algorithm 7) and Complex Synthetic (Class A: Algorithm 8 Class B: Algorithm 9) . We generate a train and test set of 500 TS. We also evaluate the stability of explanations on the ECG200 dataset (Olszewski, 2001) which tracks the electrical activity of one heartbeat for 100 train and 100 test participants. Each TS is

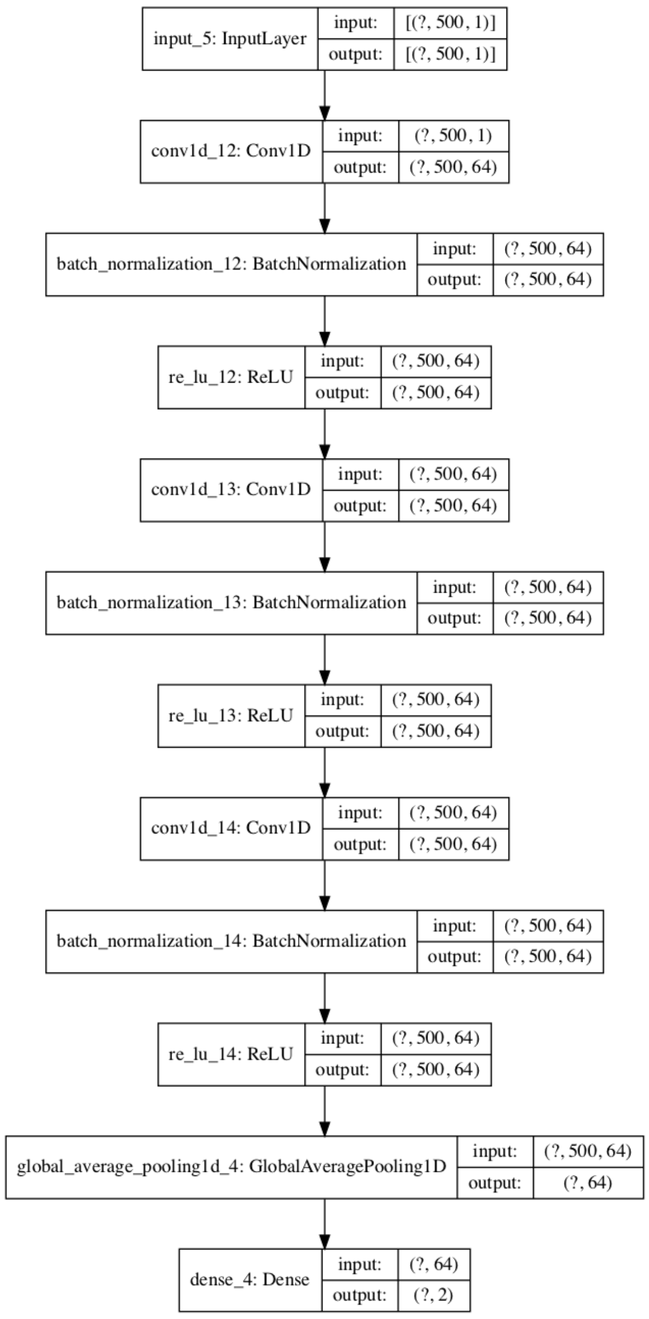
input_5: InputLayer | input: [(?, 500, 1)] | output: [(?, 500, 1)]
conv1d_12: Conv1D | input: (?, 500, 1) | output: (?, 500, 64)
batch_normalization_12: BatchNormalization | input: (?, 500, 64) | output: (?, 500, 64)
re_lu_12: ReLU | input: (?, 500, 64) | output: (?, 500, 64)
conv1d_13: Conv1D | input: (?, 500, 64) | output: (?, 500, 64)
batch_normalization_13: BatchNormalization | input: (?, 500, 64) | output: (?, 500, 64)
re_lu_13: ReLU | input: (?, 500, 64) | output: (?, 500, 64)
conv1d_14: Conv1D | input: (?, 500, 64) | output: (?, 500, 64)
batch_normalization_14: BatchNormalization | input: (?, 500, 64) | output: (?, 500, 64)
re_lu_14: ReLU | input: (?, 500, 64) | output: (?, 500, 64)
global_average_pooling1d_4: GlobalAveragePooling1D | input: (?, 500, 64) | output: (?, 64)
dense_4: Dense | input: (?, 64) | output: (?, 2)

Figure 7: Network architecture of Concurrent Neural Network employed throughout study

Algorithm 7: Synthetic Class B

```
Output: TS $T$
$\mu, \sigma \leftarrow 0, 0.35$ ;
$noise \leftarrow Normal(\mu, \sigma, 100)$;
$freq \leftarrow \sin([0; ...; 500])$;
$bg \leftarrow freq + noise$ ;
$tc \leftarrow sin([(0; ...; 100] * 5)$;
$fs \leftarrow [0]^{500}$;
$fs[400 : 500] \leftarrow tc$;
$T \leftarrow bg + fs$
```

of length 96 and the associated class labels are either "Normal heartbeat" or "Myocardial Infarction". We train the 1D CNN as depicted in Figure 7 on each dataset for 100 epochs, batch size of 64 and validation split of 0.4 using the ADAM optimiser and sparse categorical cross entropy as the loss function and obtain accuracy of 1.0 on both Synthetic datasets and 0.78 on ECG200.

To measure explanation stability we introduce Ranked Segment Stability Index (RSSI) which takes as input a set of t ordered segment importances $\alpha_i = [j_i]^r | \mathbf{w}_j < \mathbf{w}_{j+1}$ where $\mathbf{w}_j^r$ is the segment importance vector returned by the TS explanation framework. After making t calls to each explanation framework, for each pair of ordered super segment importances, $\alpha_i, \alpha_j | i, j \in t$, we count the number of concordant pairs (i, j) for which $\alpha_i = \alpha_j$. RSSI is therefore the proportion of concordant (equal) pairs out of all pairs of segment importance vectors for a given TS. To obtain the results reported in Table 3 of the original paper, we run *LIMESegment* initialised with DTW as its distance measure ($\pi \leftarrow exp(\frac{-DTW_{z_norm}(\mathbf{Z})^2}{1})$) as well as *LIMESegment* initialised with Euclidean distance as its distance measure ($\pi \leftarrow exp(\frac{-Euc(\sigma(\mathbf{x}),\sigma(\mathbf{Z})))^2}{l})$) where $l = (0.75T')^2$ 50 times for each TS. We report the mean (over every TS in the test set) RSSI per dataset and distance measure.

Algorithm 8: Complex Synthetic Class A

```
Output: TS $T$
$\mu, \sigma \leftarrow 0, 0.35$ ;
$noise \leftarrow Normal(\mu, \sigma, 100)$;
$freq \leftarrow \sin([0; ...; 500])$;
$bg \leftarrow freq + noise$ ;
$fs \leftarrow [0]^{500}$;
$tc \leftarrow sin([0; ...; 100] * 15)$;
$fs[100 : 200] \leftarrow tc$;
$tc \leftarrow sin([0; ...; 100] * 20)$;
$fs[400 : 500] \leftarrow tc$;
$T \leftarrow bg + fs$
```

Algorithm 9: Complex Synthetic Class B

```
Output: TS $T$
$\mu, \sigma \leftarrow 0, 0.35$ ;
$noise \leftarrow Normal(\mu, \sigma, 100)$;
$freq \leftarrow \sin([0; ...; 500])$;
$bg \leftarrow freq + noise$ ;
$fs \leftarrow [0]^{500}$;
$tc \leftarrow sin([(0; ...; 100] * 10)$;
$fs[100 : 200] \leftarrow tc$;
$tc \leftarrow sin([0; ...; 100] * 5)$;
$fs[400 : 500] \leftarrow tc$;
$T \leftarrow bg + fs$
```

B.4 Explanation Faithfulness and Robustness

To evaluate the Faithfulness and Robustness of each LIME adaptation we compare the explanations generated for three classifiers: a 1D CNN as detailed by Figure 7, K-Neighbour algorithm with DTW distance measure and neighbours of one as implemented in Faouzi and Janati (2020), a state-of-the-art hybrid LSTM, CNN model as implemented by Karim et al. (2019). We randomly select twelve binary TS datasets from the UCR dataset repository (Chen et al., 2015) whose details are specified in Table 5. The neural network classifiers were trained dependent on dataset as per details specified in Table 5 where we also report resulting test accuracy of the trained models. As the explanation frameworks of Neves et al. (2021) and Guillemé et al. (2019) depend on arbitrary segmentation we document the selected segment size in Table 5. To ensure fairness of results we measured the Fairness and Robustness of the frameworks of Neves et al. (2021) and Guillemé et al. (2019) under multiple segmentation approaches and report the best results in Table 5. We also report the window size parameter and

	Coffee	Strawb	GPAge	Hand	Yoga	ECG	GPGender	Dodger	China	FreezerST	House20	Worms
TS Length	286	235	150	2709	426	96	150	288	24	301	3000	900
Train	28	613	135	1000	300	100	50	20	20	150	34	181
Test	28	370	316	370	3000	100	150	138	345	2850	101	77
Batch Size (CNN&LSTM)	8	32	16	64	32	16	8	8	8	32	8	16
Epochs (CNN&LSTM)	200	200	200	200	200	200	200	200	200	200	200	200
KNN Accuracy	1.00	0.94	1.00	0.84	0.81	0.77	0.99	0.88	0.97	0.71	0.85	0.63
CNN Accuracy	1.00	0.95	1.00	0.91	0.78	0.89	0.99	0.76	0.94	0.69	0.71	0.68
LSTM Accuracy	1.00	0.96	1.00	0.87	0.90	0.87	1.00	0.87	0.96	0.72	0.77	0.78
L: window size/*cp*	10/5	10/5	10/4	100/8	10/5	10/2	10/4	10/5	3/2	10/5	100/8	10/6
N & G: ws: TSlength/5 or TSlength/10	10	5	5	10	5	5	10	10	5	10	10	10
KNN & L: F	0.01	0.04	0.01	0.02	0.10	0.04	0.02	0.00	0.00	0.03	0.00	0.05
KNN & G: F	0.01	0.03	0.00	0.00	0.10	0.04	0.02	0.00	0.00	0.03	0.00	0.01
KNN & N: F	0.01	0.00	0.00	0.00	0.09	0.02	0.02	0.00	0.00	0.03	0.00	0.00
CNN & L: F	0.05	0.49	0.03	0.11	0.15	0.17	0.02	0.12	0.16	0.25	0.25	0.06
CNN & G: F	0.00	0.10	0.00	0.05	0.05	0.01	0.02	0.00	0.10	0.00	0.01	0.00
CNN & N: F	0.00	0.00	0.00	0.27	0.02	0.15	0.02	0.01	0.16	0.16	0.02	0.03
LSTM & L: F	0.04	0.55	0.01	0.10	0.04	0.40	0.02	0.00	0.00	0.01	0.00	0.04
LSTM & G: F	0.00	0.00	0.00	0.11	0.04	0.34	0.02	0.00	0.00	0.01	0.00	0.02
LSTM & N: F	0.00	0.12	0.00	0.20	0.04	0.30	0.02	0.00	0.00	0.00	0.00	0.06
KNN & L: R	1.00	1.00	0.90	0.65	1.00	1.00	0.95	1.00	1.00	1.00	1.00	0.45
KNN & G: R	1.00	0.60	0.00	0.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	0.15
KNN & N: R	0.60	0.00	0.00	0.00	0.70	0.40	0.00	1.00	1.00	1.00	1.00	0.00
CNN & L: R	0.20	1.00	1.00	0.25	0.30	1.00	0.95	0.50	0.95	0.35	0.70	0.15
CNN & G: R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CNN & N: R	0.00	0.00	0.00	0.00	0.05	0.5	0.05	0.00	0.00	0.00	0.00	0.00
LSTM & L: R	0.45	1.00	0.90	0.40	0.80	0.65	1.00	1.00	1.00	1.00	0.90	0.45
LSTM & G: R	0.00	0.00	0.00	0.00	0.80	0.00	0.00	1.00	1.00	1.00	1.00	0.00
LSTM & N: R	0.00	0.00	0.00	0.30	0.80	0.45	0.00	1.00	1.00	0.00	0.05	0.35

Table 5: Overall results table listing all twelve datasets evaluated in this study: Coffee, Strawberry, GunPointOld-vsYoung, HandOutlines, ECG200, GunPointManvsWoman, DodgerLoopGame, Chinatown, FreezerSmallTrain, HouseTwenty, WormsTwoClass. Rows 1 and 2 display the dataset information. Rows 3 and 4 show the configuration details for the neural network classifiers. Rows 5,6,7 show the resulting test accuracy after training each classifier. Row 8 shows the parametrisation of *LIMESegment* (L) algorithm. Row 9 shows the optimal segment length for the explanation frameworks of Guillemé et al. (2019) (G) and Neves et al. (2021) (N). Rows 10 - 18 show the Faithfulness (F) under each classifier and explanation framework. Rows 19 - 27 show the Robustness (R) under each classifier and explanation framework.

threshold *cp* required for *LIMEsegment* in Table 5. For brevity we select just five individual datasets to report in Table 4 of the original paper (Strawberry, HandOutlines, Yoga, ECG200 and Chinatown) whose results are calculated as the mean Robustness and Faithfulness over each classifier alongside the standard deviation for each dataset. The "all" column in the original paper calculates the mean and standard deviation of Faithfulness and Robustness over all twelve datasets evaluated for each explanation framework.

B.5 LIMESegment In Practice

To evaluate the use of *LIMESegment* on the Sepsis Cohort of the MIMIC Dataset we adopt the approach of Komorowski et al. (2018) which generates a dataset of total of 256230 sepsis patients. We further pre-process the dataset by selecting the patients who have 20 observations over the 36 hour period. We create a balanced dataset by selecting 590 temperature TS of patients of each class (0: patient dies, 1: patient survives). We divide this dataset into a train/test split of size 500/90 and train our 1D CNN as specified by Figure 7 for 200 epochs with a batch size of 64 obtaining accuracy of 0.80. We continue by generating explanations for each TS in the test set according to *LIMESegment* (window size = 3, *cp* = 4), and the frameworks of Neves et al. (2021) and Guillemé et al. (2019) (segment size = 4).