DITAN: A deep-learning domain agnostic framework for Detection and Interpretation of Temporally-based multivariate ANomalies.

Michail Giannoulis, Andrew Harris, Vincent Barra

Abstract

We present DITAN, a novel unsupervised domain-agnostic framework for detecting and interpreting temporal-based anomalies. It is based on an encoder-decoder architecture with both implicit/explicit attention and adjustable layers/units for predicting normality as regular patterns in sequential data. A two-stage thresholding methodology with built-in pruning is used to detect anomalies, while root cause and similarities are interpreted in data and units space. Our approach is designed to intersect the 9 fundamental characteristics extracted from the union of related works. We demonstrate the DITAN modules on real-world datasets of 6 multivariate time series contaminated by point and contextual temporal-based anomalies at a varying duration. Experiments show a dominant predictability power of DITAN against the originally proposed models. DITAN is able to determine critical regions and thus identify anomalous events similarly well. Informative similarities between anomalous records are interpreted, since almost all similarities in units space are also verified in data space.

Multivariate Time Series, Anomaly Detection, Neural Networks, Generic Normality Feature Learning, Predictability Modeling

1 Introduction

Recent advances in technology, enable data to travel across a sensoring system from our physical to the digital world in large amount over time. Such data are recorded in an orderly fashion, correlated in time, constituting a *time series*[1]. Each record in a time series is a vector containing the sensor values at a specific time step, introducing both temporal and features dimensionality. A time series with more than one sensor is called *multivariate*, otherwise *univariate*. However, sensors often capture records that all or some of their values significantly differ from their expected or *normal* behavior. These unexpected values are called *temporal-based anomalies* or simply *anomalies*. A temporal-based anomaly has certain temporal (duration) and features (sensors) resolutions. Figure 1 demonstrates the four possible states of a temporal-based anomaly using three sensors, graphically illustrated on the *anomalous exploration space* (AES). In this space, each record is illustrated in three dimensions with depth and width indicating features and temporal resolution respectively, while height indicates the contamination magnitude of each sensor value.

From the features aspect, a record can either be contaminated to a subset or to the full-set of its sensor values. The former case is called *subspace anomaly* with respect to those sensors, while the latter one is called *fullspace anomaly*. In Figure 1, full-space anomalies are depicted on top right and bottom right, while subspace anomalies are proposed on two out of three and one out of three features, on top left and bottom left respectively. Although sub-space anomalies are in the scope of interest of this work, spatial anomalies are not. We indeed assume that each sensor is a one-dimensional signal and we do not deal with pure spatial anomalies. A full-space anomaly can be characterized both low and high dimensional features space. If sub-space anomalies often exhibit evident abnormal characteristics in a low-dimensional space, they become hidden and unnoticeable in high dimensional space. Hence, the detection of sub-space anomalies require a deep understanding of the underlying features correlation.

From the temporal aspect, *point anomaly* is a subspace or fullspace anomaly occurring in an individual record at a specific time instance. In addition, *subsequence anomaly* is a joint subspace or fullspace anomalous behavior of consecutive records in time. In other words, a point anomaly can be seen as a subsequence anomaly of length (width) one. Hence, the detection of subsequence or point anomalies requires both *context* and *horizon* to be determined. The context and horizon are windows of consecutive records, where context controls the locality of temporal-anomalies examined in horizon. Thus, the length of horizon and context are related, in a way that a very far to the future forecast may require a larger context.

[1]Note that the techniques discussed in this paper only consider regularly sampled time series with the same temporal granularity in all dimensions

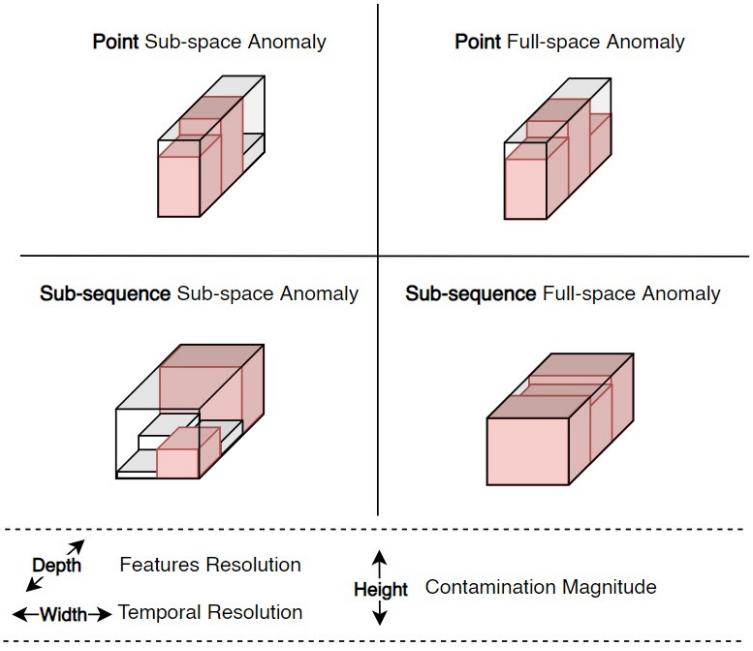


Figure 1: Temporal-based anomalies on the anomalous exploration space (AES). Normal values are color-coded in white, and anomalies in red.

The scale of features and temporal resolution can address local and global anomalies. Intuitively, large scale will allow to observe global anomalies and finer scales will highlight local ones. Local can also be seen as *contextual* anomalies, when abnormality has a context related meaning.

From the magnitude point of view, each sensor value of a record is measured by an anomalous (severity) score obtained using both features and temporal resolution. The higher the score the more anomalous the value is. Especially when labels are not available, a threshold value is crucial to transform the regression problem into binary classification; normal vs. anomalous value. Scores can either be assessed manually from an expert or in an automatic way using an anomaly detection method as a tool, called *detector* for short, which is the objective of this work.

Detection Techniques	**Features Aspect**	**Temporal Aspect**
Model-based (Prediction and Estimation)	Multi/Univariate	Point / subsequence
Density-based	Multi/Univariate	Point
Histogram-based	Multi/Univariate	Point
Dissimilarity-based	Multi/univariate	Point / subsequence
Discord-based	Univariate	Sub sequence
Frequency-based	Univariate	Sub sequence
Information-Theory	Univariate	Sub sequence

Table 1: From temporal-based anomalies to detection techniques.

In a recent survey [4], seven detection techniques were reviewed for detecting point and/or subspace univariate and/or multivariate anomalies. These techniques are summarized in Table 1, concerning the aforementioned characteristics of a temporal-based anomaly. Although histogram-based techniques are applicable on multivariate anomalies, they do not support sub sequence anomalies. Also, density-based, discord-based, frequency-based and information theory methods are only applicable on univariate anomalies and therefore even if applied on a multivariate environment they will not consider any dependency between features. On the other hand, model-based and dissimilarity-based techniques are applicable on point or subsequence multivariate anomalies. Model-based techniques enable to learn the underlying characteristics of normality, and dissimilarity-based techniques, either apply pairwise comparison to detect point anomalies or use a reference of normality to detect subsequence anomalies. Yet, a model requires a

deep understanding of the feature dependencies in a multivariate environment, usually composed by both relevant and irrelevant to the detection process features. Manually covering linear and non-linear dependencies is proved to be a difficult problem for traditional model-based methods, since they additionally require a feature-selection-based or a subspace-based technique. To this direction, deep learning for anomaly detection, *deep anomaly detection* for short, aims at automatically learning feature representations specifically tailored for anomaly detection. A recent comparative study [1] stresses that deep learning methods are better choice when datasets require large training set or contaminated by contextual anomalies, while in majority conventional and ML ones considered a similarly good choice with no significance difference across their evaluated datasets.

Category	**Sub Categories**	**Model/Scoring**
Deep Learning for Feature Extraction	-	Fully Separated
Learning Feature Representations of Normality	Generic/Measure-dependent	Coupled
End-to-end Anomaly Score Learning	-	Fully Joint

Table 2: Model-based deep anomaly detection Frameworks.

Another recent survey [24] shows the efficiency of deep anomaly detection methods, through a comprehensive review of model-based techniques. Authors reviewed twelve diverse modeling perspectives on harnessing deep learning techniques for anomaly detection, categorized under three principle frameworks as shown in Table 2. In the *deep learning for feature extraction* category, a neural network is used as feature extractor and an independent method is used for anomaly scoring. The disjoint behavior between the two modules leads to less interpretable and sub-optimal scores. In *end-to-end anomaly score learning*, a neural network aims to learn scalar anomaly scores simultaneously with feature extraction. However, this framework requires some form of labels or prior knowledge on anomalies and thus may not be able to generalize. A coupled relation of feature learning with anomaly scoring, rather than fully separated or fully joint, is implemented on *learning representations of normality* in two different ways, *measure-dependent* or *generic*. In *measure-dependent*, the feature representations are optimized for a particular (limited) type of anomaly using existing shallow anomaly measurements (e.g. distance-based, cluster-based), while in *generic*, the learning process is forced to capture the key underlying data regularities. Although the *generic* way is not particularly designed for anomaly detection, the learnt representations suppose to capture the varieties of normality (regularities), resulting interpretable anomaly scores whose magnitude is driven by the model itself.

In this paper, we present DITAN, a generic framework for normality feature learning. The main innovative features are the following. (a) It is domain agnostic, applicable on predictable multivariate time series. (b) Normality in DITAN is expressed as regular patterns stored in a compressed latent space. (c) Its inference mechanism is armed of both implicit and explicit attentions. (d) It incorporates a dynamic threshold, optimized on the testing errors per sensor. (e) It provides anomaly interpretation, based on their severity, root cause and similarity aspects. More specifically the contributions are the following:

- We address recent literature and emphasize to the nine main characteristics that methods used to cover temporal-based anomalies (Section 2)
- We stress how different pre-processing operations are applied to time series (Section 3.1)
- We present the DITAN, a predictive modeling approach using an encoder-decoder architecture and managing temporality in both implicit and explicit ways (Section 3.2)
- We propose a regularization scheme as an immune system to our architecture, to allow generalization (Section 3.3)
- We define a methodology to automatically derive the model's hyper-parameters, to enable a domain-agnostic functionality (Section 3.4)
- We provide a two-stage dynamic threshold technique for DITAN to detect anomalies in the form of critical regions, directly on the testing errors sequence (Section 3.5)
- We examine the severity of anomalies, characterizing their root cause as well as similarity on units space (Section 3.6)
- We evaluate DITAN on six multivariate time series contaminated by point, contextual or joint anomalous events and show the benefits of our approach compared to baseline (Section 4)

DITAN is built upon three assumptions. (1) *time series data are predictable*: although multivariate time series data are sequentially placed records, the values of these records may come from a random generator. Such a case

is not in the scope of this framework, because records data are useless to predict future ones and so these data are considered unpredictable ; (2) *normality is identical to regularity*, since we are under the generic normality feature learning, and even further and ; (3) *irregular records are temporally less predictable than regular ones*, since we use a predictability modeling perspective. Benefits and limitations of the main methodological contributions in DITAN are presented in Table 3, where the first column points out sections providing a detailed description and analysis, and briefly highlighted here: in the training phase, we use an attention mechanism to better memorize temporal frames across series, at the cost of a higher number of model parameters. We regularize these parameters to avoid over-fitting, using more training epochs and a slower convergence. The model parameters and number of epochs are then optimized faceted to the given series. Detection of variable-length temporal-based anomalies is supported using critical regions with built-in pruning, at the cost of false negatives in terms of sub-capturing. The relative similarity of the detected-anomalies between sensors is controled by the non-costly root cause formula, and an optimization-based one is introduced to interpret these anomalies from model perspective as well.

.

Section	**Technique**	**Advantage**	**Disadvantage**
3.2	ED with Attention	Temporal Information	More (hyper) parameters Time prone
3.3	Regularization	Avoid over-fitting	More epochs required (slow converge)
3.4	HP Optimization	Dynamic Domain-agnostic Structure	Many Iterations (runs) Time Prone
3.5	Critical Regions	Support variable length anomalies	Prone to FN in the form of miss-alignment
3.5	Built-in Pruning	Reduce False Negatives (FN)	HP Optimization required Time prone
3.6	Root Cause	Relative severity across sensors	-
3.6	Internal Similarity	Model Perspective Interpretation	Optimal Cluster Selection

Table 3: Pros and cons of the core technical choices of DITAN.

The rest of the paper is organized as follows: we first discuss related works and their limitations (Section 2). We then describe the formal foundations for the discovery and interpretation of temporal-based anomalies (Section 3). Next, the effectiveness of our approach is exhaustively evaluated (Section 4). Finally, conclusion and future works are discussed (Section 6).

2 State-of-the-art

The four (deep) model perspectives of the *generic normality feature learning* framework as proposed in [24] are summarized in Table 4. Auto encoder (AE) and generative adversarial network (GAN) use similar paradigms to detect anomalies: they both use a low-dimensional space to reconstruct or generate the input records, so that values of anomalous records cannot be accurately estimated. In addition, although self-supervised classification (SSC) and predictability modeling (PM) are both self-supervised perspectives, their semantics are not similar. In predictability modeling, normality is examined through mapping input records to outputs, which are future records. Values of an upcoming anomaly will deviate from the ones in the predicted normality. In self-supervised classification, normality is expressed by several classifiers learnt to identify inconsistent input records. It is also remarkable that predictability modeling is the only perspective at which temporal aspect is strongly embedded to its mapping (learning) process.

The objective, common for all model perspectives, is to learn the feature representations in the form of general regularities. Since a model is composed of artificial neural networks, regularities are expressed through weights Θ, W between units. During the learning process, these weights are learnt by minimizing a loss function:

$$\Theta^*, W^* = \arg\min_{\Theta,W} \sum_{c \in C} l(\psi_W(\phi_\Theta(c)), h) \tag{1}$$

Here, a context c is a sequence of one or more consecutive multivariate records, representing the input data. A

Model Perspective	Anomaly Definition
Auto-encoders (AE)	hardly to be reconstructed from compressed space
Generative Adversarial Networks (GAN)	hardly to be generated from latent feature space
Predictability Modeling (PM)	temporally less predictable than normal ones
Self-supervised Classification (SSC)	less consistent to classifiers

Table 4: The model perspectives of the generic normality feature learning framework.

function ϕ_Θ, parametrized by weights θ, maps a context $c \in C$ from original space onto a latent representation $z \in Z$. A surrogate learning task ψ_W, parametrized by weights W, operates on latent space and is dedicated to enforce the learning of underlying regularities, mapping the z to a horizon $\hat{h} \in H$ back to original space. A horizon $\hat{h}$ is a sequence of one or more consecutive multivariate records, representing the generated output data. Next, a loss function l relative to the underlying modeling approach compares $\hat{h}$ to the corresponding actual data h. Higher weights are supposed to describe one or more normal patterns. However, a model trained in an *unsupervised* setting enables the presence of anomalies in the training sequence. This may force the model to additionally learn irregularities, since anomalous records can also create some strong weights. To avoid such situations, a good practice is to apply unsupervised settings when anomaly ratio is up to 5%, otherwise apply *semi-supervised* settings using pure normal data.

Since a model is forced to learn regularities over a bulk of normal records, the aforementioned model perspectives can only forecast normality. The forecasting process of this framework utilizes the trained weights W^*, Θ^* along with mapping functions to construct any horizon $\hat{h}$ given a context c, i.e $\hat{h} = \psi_{W^*}(\phi_{\Theta^*}(c))$. The relation of the records between $\hat{h}$ and c is controlled by the forecasting protocol, either estimation-based or prediction-based. In *estimation*-based protocol (*AE*, *GAN* and *SSC*), records in $\hat{h}$ correspond one by one to c, indicating their estimated normal behavior. In *prediction*-based protocol (*PM*), records in $\hat{h}$ are all succeeding to the records in c, indicating the predicted normal behavior of a future sequence. Intuitively, the mapping from c to $\hat{h}$ can be seen as the *inductive* construction of *IF-Then* rules, where c represents the *IF* and $\hat{h}$ represents the *Then* conditions. The rules are also called temporal-based, for prediction-based protocol. Thus, a frequent rule is expected to contain regular patterns addressed by the model to strong weights.

The goodness of a horizon $\hat{h}$ is assessed using a scoring function f, i.e. $score(h) = f(c, \phi_{\Theta^*}, \psi_{W^*})$, where h is equal to c in estimation-based protocol and equal to succeeding record/s of c in prediction-based protocol. The emergence of efficient temporal-based multivariate deep anomaly detection methods has recently led to a large variety of researches, under the aforementioned model perspectives. These model perspectives are also called *meta-networks*, because they are armed by one or more basic, feed-forward, convolutional or recurrent, neural networks.

2.1 Autoencoders

The Auto Encoder (AE) approach aims to learn some low-dimensional feature representation space from which the given records can be well reconstructed, assuming that normal records can be better reconstructed from compressed space than anomalous ones. There is a large family of reconstruction meta-networks with plenty of recent instantiations (table 5).

An *AE* is composed of two identical and symmetrical networks: an encoder E and a decoder D, trained with a collaborative objective. E learns to map the input data c onto a low-dimensional feature representation z, while D attempts to find $\hat{h} \approx h = c$ from z. Hence, the objective of E and D is to identically reconstruct the input data with $D(E(c))$, by minimizing the reconstruction error (*RE*) loss function. Both networks are composed by one or more basic networks. A *standard* AE architecture is instantiated in many works (first line in table 5) to reconstruct the records of a multivariate time series. From these, [2] used only feed forward (MLP, CNN) layers. However, conventional (feed-forward) neural networks make the assumption that data is independent in time, which does not hold for sequential data [25] , such as time series. Therefore, recurrent networks are often used. Specifically, in [13, 18, 19, 27, 36] authors used only recurrent (LSTM, BiLSTM, GRU) layers to capture the temporal aspect in data. [33, 35] used both feed forward (CNN) and recurrent (LSTM) layers to capture both spatial and temporal patterns. That is because in some cases features are all-to-all correlated formulating a spatial terrain (e.g. image pixels). Particularly, in [35] both encoder and decoder consist of CNN layers, connected through an additive attention based *ConvLSTM* layer, to adaptively select relevant hidden states across different time steps. On the other hand, authors in [33] used two networks connected in sequence through a window feature sequence layer . Firstly, CNN layers are employed to extract spatial feature maps from data. Then, these features are reconstructed using LSTM layers

Meta Network	Type	Basic Network	References
AE	Multilayer	MLP	[2]
	Recurrent	LSTM	[18, 19, 27]
		BiLSTM	[36]
	Conv & Recurrent	CNN, ConvLSTM	[35]
		CNN, LSTM	[33]
Ensemble AE	Recurrent	LSTM	[13]
Denoising AE	Recurrent	BiLSTM	[20]
Variational AE	Recurrent	GRU	[28]
		BiLSTM	[26, 25, 34]

Table 5: Recent literature instantiations on AE.

in both encoder and decoder, capturing potential temporal relevance. An ensemble of standard AEs architecture is proposed in [13], where each AE is trained independently and encourages sparsity in the LSTM layers by randomly removing some connections. An interesting modification in the learning process is proposed by [2], where authors replaced the standard learning process with an adversarial one to increase robustness to small anomalies.

Denoising AE (DAE) learn representations that are robust to small variations, by forcing the hidden layers to retrieve more robust features and prevent from simply learning the identity. Authors in [20] used a denoising AE with bidirectional LSTM (BiLSTM) layers to learn robust temporal patterns in both positive and negative directions of the time axis.

Some authors also suggested a *Variational* AE (VAE) based approach, introducing a regularization into the representation space by encoding records using a prior parametrized distribution over the latent space (fourth line in table 5). Basically, these works used only recurrent (GRU and BiLSTM) layers, from which [25] employed a self-attention mechanism to improve the encoding-decoding process and [34] presented some interesting customizations. [34] extended VAE architecture to improve modeling capabilities of normal data, accompanied by a loss function which takes into account these characteristics. BiLSTM layers were used to construct a re-encoder layer after a VAE network, to enable the extraction of more data features including both original and latent space. A constraint network was then stacked to limit model's ability to reconstruct abnormal data. Although feed forward layers are not ideal for capturing temporal information, the network captured multiple local temporal dependencies by applying a series of convolutions (encoder) and deconvolutions (decoder) with different filter sizes over a windowed time series.The difference in these models is that VAE instantiations are stochastic generative models that can give calibrated probabilities, while AE instantiations are deterministic discriminative models that do not have a probabilistic foundation.

The simplicity of reconstruction forces the vast majority of models to assess an anomaly score using its reconstruction error over the original feature space, i.e $score(h) = ||h - \hat{h}||$. In [35], the anomaly score is defined as the number of poorly reconstructed pairwise correlations, using a predefined threshold, whereas in [13] it is defined as the median of its N reconstruction errors over an ensemble architecture. In [19], Maximum Likelihood Estimation is applied to estimate the parameters μ and Σ of a Normal distribution. [36] as for them proposed a division relation, considering both forward $\hat{h}$ and backward $\hat{c}$ reconstruction terms, due to their bi-directional architecture:

$$score(h) = \frac{((||h - \hat{h}||_2 + ||c - \hat{c}||_2)/2 - \mu)^2}{2\sigma^2} \tag{2}$$

In [19], parameters are combined in a multiplicative way;

$$score(h) = (||h - \hat{h}|| - \mu)^\top \Sigma^{-1} (||h - \hat{h}|| - \mu) \tag{3}$$

Other studies [25] make use the advantages of their variational architecture to assess an anomaly score based on the reconstruction probability using Sequential Monte Carlo of L iterations:

$$score(h) = -\frac{1}{L} \sum_{l=1}^{L} log(p(h|\mu_h^{(l)}, \sigma_h^{(l)})) \tag{4}$$

In [26], an anomaly score is assessed using only its (variational) latent representation using a binary (K-means, Spectral and Hierarchical) clustering on μ_z as well as the computation of the median Wasserstein distance in both μ_z

and Σ_z. If they might share the same mean, the variability of anomalous records relatively to normal ones is likely to be higher

In [34], an anomaly score is computed as the normalized addition between the original and latent reconstruction terms, constrained by complementary weights b:

$$score(h) = norm(b||h - \hat{h}||_1 + (1 - b)||z - \hat{z}||_1) \quad (5)$$

Detecting anomalies by thresholding anomaly scores is a very different issue. Several AE models use a trivial approach to determine a threshold classifying records into normal or anomalous. A static threshold is deployed in [34] using a fixed reconstruction error value and in [27] using a fixed quantile over the reconstruction errors. In [27] the quantile is updated over time during the testing phase to the anomaly score maximizing the accuracy. A hyperparameter β can also be introduced into the threshold computation, shifting the median of reconstruction errors [20] or the maximum of the poorly reconstructed errors in validation set [35]. The Extreme Value Theory (EVT) is used in [28] to determine a threshold over reconstruction errors. The advantage of EVT is that it makes no assumption on data distribution when finding extreme values. Similarly, [18] is based on the central limit theorem over reconstruction losses to construct a threshold. Basically, the notion of quantile can be extrapolated using EVT, when there is only normal data.

Thresholds can also be computed using performance measures. [19] selected the threshold maximizing the precision (P) - recall (R) relation over the likelihood values

Some methods do not search for a specific threshold at all. In [26, 25] authors proposed a set of thresholds according to the AUC of the ROC curve. Similarly, [13] used both AUC-PR and AUC-ROC to include all possible thresholds. As an alternate to thresholding, anomalies in [36] (respectively [33]) are detected using a Gaussian Segmentation Model (resp. a softmax classifier).

Apart from scoring and detecting anomalies, only three of the aforementioned instantiations [28, 35] investigate their characteristics, reporting the root cause of each anomalous record as the top-k sub scores of its features [35] or as its features reconstruction probabilities in ascending order [28], assuming that the most anomalous features can provide sufficient clues to understand and troubleshoot the detected anomaly. Note that [35] additionally reported the severity level as the duration (length) of subsequence anomalies.

2.2 GANs

Generative Adversarial Networks (GAN) aim to learn a generative model capturing the normality of the given records, assuming that normal records can be better generated from the latent space than anomalous ones. Table 6) gives an overview of some models using GANs.

Meta Network	Type	Basic Network	References
GAN	Recurrent	LSTM	[15]
AE-GAN	Convolutional	CNN	[37]
	Convolutional and Recurrent	BiLSTM, CNN	[7]
	Multilayer	MLP	[31]
VAE-GAN	Recurrent	LSTM	[23]
AE-E-GAN	Convolutional and Recurrent	CNN, ConvLSTM	[12]
	Convolutional	CNN	[10]

Table 6: Recent literature instantiations on GAN.

A standard GAN is composed of two networks trained simultaneously using adversarial objectives. A generator G network takes as input a noise vector z randomly selected from a latent space Z to generate a fixed size subsequence of synthetic records $\hat{h} \approx h = c$. These records are supposed to be realistic, capturing the actual c data distributions. To this direction, a discriminator D network takes as input either actual c either synthetic h records to estimate a prediction score. The objective of G is to fool D that its synthetic records are real, while the objective of D is to correctly distinguish between synthetic and actual data. *Standard* GAN architectures are proposed in [15] to identically generate the records of a multivariate time series. A recent study [14] suggests that recurrent (LSTM) layers are more suitable for the learning procedure of generative networks over complex time series data, due to their memory blocks. So [15] used a multivariate model with only LSTM layers for both generator and discriminator. If *standard* GANs architectures learn to generate data from a latent space using the generator, they do not learn the inverse mapping G^{-1} back to the latent space.

Many works introduced an encoder E that learns G^{-1}. Hence, the generator takes the role of a decoder, formulating a new network G composed of an encoder G_E and decoder G_D, interacting to minimize a *reconstruction* (a.k.a. apparent) loss, i.e. a distance measure between actual records h and reconstructed (generated) ones $\hat{h}$ on the original space. G_E and G_D are trained simultaneously or in a post-hoc manner to maintain the training complexity. In [7, 31, 37] the generator is a *standard* AE network, to simultaneously map from the latent space to data $G_D(z)$ and vice versa $G_E(c)$. In [23], the generator is a *variational* AE to additionally learn a distribution in the latent space Z. A more advanced architecture is proposed in [10, 12]. Authors introduced an additional encoder E' after a *standard* AE generator, to enforce similar inputs to lie close to each other in both original and latent space. This variation introduces a new loss term for G, the *latent* loss, representing the distance between z and $\hat{z}$, where $\hat{z}$ is the encoded bottleneck representation of $\hat{h}$. In [10], authors introduced a binary cross entropy term over the prediction scores of the discriminator on h in the loss function of G.
The one and only task of the discriminator is to distinguish between actual h and generated $\hat{h}$ records. If [10, 31, 37] used MLP, [7, 12, 23] retained temporal information. In [23] LSTM layers are used to implicitly learn temporality, while in [7] BiLSTM (respectively CNN) layers are defined for G (resp. D), with the aim of capturing local temporal features. In [12], an explicit attention layer is introduced through the use of a smoothed attention mechanism over an additional ConvLSTM layer along CNN layers, to jointly capture the spatial patterns and temporality.

A fundamental problem of GAN architectures, namely *mode collapse*, is that the generator tends to learn only a small fraction of data variability, such that it cannot perfectly converge to the actual distribution. This is mainly because the generator is reluctant to produce records that capture other modes in data beyond the ones which fool the discriminator. To overcome this limitation, [7, 12] applied Wasserstein loss as the adversarial loss. In this way, the generator is forced to not only focus on a subset of distribution and thus to theoretically converge to the actual distribution.
The purpose of GAN is to reconstruct the input time series. However, an adversarial loss alone cannot guarantee mapping an individual record to a desired latent space from which will be then mapped to its reconstruction. To reduce the search space of the mapping function, [7] adapted a cycle consistency loss to time series reconstruction, introducing two critics C_h and C_z to the discriminator, where, C_h is an indicator of how real the actual or generated records are, and C_z measures the goodness of the mapping into a latent space. Interestingly, they concluded that adding backward consistency loss did not notably improve the performance.

The architecture preferences also affect the anomaly scoring methodology. In *standard* GAN architectures, G is fed with a random vector z to generate a horizon $\hat{h}$. It is required a series of λ back propagation steps to update the parameters of G until $\hat{h}$ is close to the input context. Thus an anomaly score can only be determined after λ iterations.

In [15], both generator and discrimination losses are considered. The former term measures the reconstruction error in the original space, while the latter measures the error in a rich feature space of last intermediate layer f in discriminator:

$$score(h) = (1-\alpha)||h - G(z_\lambda)||_1 + \alpha||f(h) - f(G(z_\lambda))||_1 \tag{6}$$

The addition of an encoder G_E in the generator of a GAN architecture is useful also in the anomaly scoring process. Such an architecture does not require λ back propagation steps. Instead, G_E is handed directly by the context c to estimate a latent representation $G_E(c)$, which is then fed to G_D to generate a horizon $\hat{h} = G_D(G_E(c))$. For example, [31] used both terms proposed in Eq. 6 and introduced scalars n_h, n_f, κ, where, n_h is the number of sensor values of the input record h, n_f is the number of neurons of the f layer and κ is a coefficient to adjust the weights of the reconstruction and features losses:

$$score(h) = \frac{\kappa}{n_h}||h - G_D(G_E(c))||_2 + \frac{1}{n_f}||f(h) - f(G_D(G_E(c)))||_2 \tag{7}$$

[37] only retained the first error term of Eq. 7 to define their loss function.

A relatively low prediction score is expected for the input records which do not conform to the distribution of normal data. To this direction, reconstruction error along with the prediction score of discriminator for the actual data were used in [23]:

$$score(h) = (1-\alpha)||h - G_D(G_E(c))|| - \alpha D(h) \tag{8}$$

In [7], authors used reconstruction error along with the critic C_h of their discriminator as an indicator of how real the actual input record is. Due to the contractive definition of the anomaly score, they standardized them into z-scores Z. Two scoring functions were used and for example:

$$score(h) = \alpha Z(||h - G_D(G_E(h))||) \odot Z(C_h(h)) \tag{9}$$

The addition of an encoder E' after the generator of a GAN architecture is useful also in the anomaly scoring process. Such an architecture introduces the error between the latent representation z of h and the reconstructed $\hat{z}$ of $\hat{h}$. For example [10] defined a score taking into account the reconstruction errors in both original (h, $\hat{h}$) and latent (z, $\hat{z}$) spaces:

$$score(h) = \alpha||h - G_D(G_E(h))|| + (1-\alpha)||G_E(h) - E'(G_D(G_E(h)))||_2 \tag{10}$$

In [12], the anomaly score is defined as the number of poorly reconstructed pairwise correlation, using a predefined threshold, considering the reconstruction in both original and latent space, to be less sensitive to severe anomalies.

Detecting anomalous records is usually related to thresholding anomaly scores. A dynamic thresholding methodology can be proposed, where a sensor value is considered anomalous if its anomaly score is higher than the prediction score of the discriminator for its actual sensor value. Static threshold are also proposed, extracted using a fixed quantile over the reconstruction errors in training sequences [15], using a locally adaptive approach [7], or even optimized [12, 23].

Alternatively, some methods used no threshold at all. Anomaly scores in [31, 37] are reported in heat maps, where sensor values highlighted over time, rooting experts attention to the anomalous portions. In [10], authors simply reported the raw anomaly scores, in which the closer to zero the anomaly score is, the more normal the record is considered.

A further exploration is examined over the detected anomalies by [7], where authors noticed that the use of sliding windows may produce many false positives. False positives are mitigated in [7] by applying an anomaly pruning approach, inspired by [9]. In that approach each window is first addressed by its maximum anomaly score and then a decrease percent over the descending scores is computed, re-classifying as normal each window that does not exceed a certain threshold. Beyond filtering, root causes are examined in [12], where the authors automatically extracted the number of root causes using an elbow method on the anomaly scores distribution, as opposed to [35] which fixed the number of root causes in three. With this approach authors aim to find the point where the amount of errors become very small and close to each other.

2.3 Self Supervised Classification

Few more estimation-based models are proposed using Self Supervised Classification (*SSC*). This approach learns representations of normality by building classification models in a self-supervised manner and identifies records that are inconsistent to these models as anomalies. In previous years, shallow methods have been introduced based on cross-feature analysis and feature models [30], in which each model evaluates a sensor value of a record with respect to the rest of its sensor values. Hence, the consistency of a record is measured either as the average prediction results or as the majority voting of binary decisions [30] given the models across all features. Recently, deep methods [8] focused on capturing spatial information on images and used transformation-based feature augmentation to build different models. Formally, a context c gets augmented by T different transformations, parsed through a function ϕ to result a latent representation per transformation $\{z_{(1)}, .., z_{(T)}\}$. The latent representations are then fed to a multi-class classifier ψ to result the corresponding horizons $\{\hat{h}_{(1)}, .., \hat{h}_{(T)}\}$. A standard cross entropy loss is then applied over $(\hat{h}_{(j)}, h_{(j)})$ pairs, where $h_{(j)}$ is a one-hot encoding of the synthetic class for records augmented using the transformation operation $T_{(j)}$. In [8], the classification scores resulting from ψ are aggregated using a simple average associated with different $T_{(j)}$ to compute the anomaly score.

Although the aforementioned models focus on image data, there are some slight variations applicable on broad type of data [3, 11]. Instead of using geometric or audio-inspired transformations, [3] used a different feature-level transformations to map data into a finite number of subspaces, before learning a feature mapping that maximizes the difference between inter-class and intra-class separations. Fully connected layers and resulting softmax classification scores over subspace transformations were used to assess anomaly scores. In [11], authors proposed an autoML pipeline consisting of three key parts: auto representation learning, auto anomaly score calculation and auto negative sample generation. Here, *auto* stands for simultaneously Bayesian optimization of hyper-parameters along with anomaly detection in the latent space using Gaussian Mixture Model to characterize the level of abnormality.

2.4 Predictability Modeling

There are many natural phenomena that require a prediction algorithm for answering important questions, such as estimating future population variations, predicting the orbits of astronomical objects or predicting the occurrence of seismic waves. Predictability Modeling (PM) approach aims to learn feature representations by predicting output data, succeeding to the input, with a possible overlapping, assuming that normal records are temporally more predictable than anomalous ones.

To our knowledge, only few such methods were recently introduced [9, 21, 29].

Formally, a context c is given to an encoder E to result a latent representation z, fed to a decoder D to result a constructed horizon $\hat{h}$, where $\hat{h} \approx h \neq c$. So, *PM* uses an *Encoder-Decoder* (ED) architecture to predict a single or a sequence of records using a sequence to point (S2P) or a sequence to sequence (S2S) learning process respectively. It exhibits a complementary approach to *AE*, since *AE* only captures the case of input data reconstruction at which $h = c$. An *ED* architecture is proposed in [21, 29] using convolutional layers with automatically derived hyper-parameters via grid-search over a series of extensive experiments [21] or over an augmented with synthetic anomalies validation set [29]. In [21] authors learnt to predict a current record using a sequence of previous (prior) records, in S2P manner, by minimizing the regression loss. In [29] a S2S learning process is employed, learn to predict both current and previous records from the prior only. Authors in [29] introduced a reconstruction task to the one decoder. This mapping choice enables to minimize a composite loss function, which takes into account the relative importance of reconstruction and regression errors. Essentially, only the first part of $\hat{h}$ is used in the reconstruction error, while the entire $\hat{h}$ is used in the regression error. To this point, against to the superior performance of LSTM on capturing temporal information, we notice a literature gap in predictability modeling at employing *ED* using recurrent (e.g. LSTM, BiLSTM, GRU) layers. A stand-alone LSTM instantiation with fixed hyper-parameters has been recently proposed in [9], where spacecraft anomalies are predicted in S2P manner by minimizing the regression loss.

Nonetheless, outside the scope of anomaly detection, there is recent literature under *PM* which uses an *ED* architecture with recurrent layers and an explicit attention mechanism. In [5], a S2S learning process is developed for multi-step prediction, using BiLSTM layers to encode sequential input data in both directions, a temporal attention layer and LSTM layers to decode the hidden representation. Several real-world problems have been addressed using these methods. A common conclusion of these studies is that an attention mechanism can effectively improve the model performance, since the prediction ability gradually degrades as the length of input data increases.

Once a prediction is made, it is necessary to assess the anomaly score. The actual horizon h is scored as the L2-norm [21] (or L1-norm [9]) discrepancy from its predicted horizon $\hat{h}$, formulating the regression error. Since h represents a single record, the scores of K consecutive records are aggregated to assess an anomaly score to a fixed-length subsequence. A statistical approach was then proposed [9] to score dynamic-length subsequence records:

$$score(e_{seq}^{(i)}) = \frac{max(e_{seq}^{(i)}) - \epsilon}{\mu(e_s) + \sigma(e_s)} \tag{11}$$

where e_s is a one-dimensional vector appending horizon scores and smoothed using an Exponentially Weight Moving Average (EWMA), $\epsilon = argmax(\mu(e_s)+k\sigma(e_s))$ is a threshold value, where k is an ordered set of positive values representing the number of standard deviations over e_s and e_{seq} is an arbitrary-length subsequence of consecutive smoothed scores from e_s which exceed the ϵ value. Finally, i refers to the sensor (feature) dimension in a multivariate environment.

A composite approach is proposed in [29], where a reconstruction and a regression error are defined as metrics for evaluating the degree of anomaly of an actual horizon h. Both metrics use Frobenius norm to measure the deviation between h and its predicted horizon $\hat{h}$. Here, reconstruction refers to the first p records, while regression addresses all $p+q$ records of h (Eq. 12). Since h represents a subsequence of records, the average of its K_{rec} and K_{reg} scores were respectively used to assess a reconstruction and regression score per record, where K indicates the number of horizons that the record of interest appears into, as a result of overlapping sliding windows.

$$score_{rec}(h) = \sum_{0 \leq i < p} ||h_k^i - \hat{h}_k^i||_F \quad \text{and} \quad score_{reg}(h) = \sum_{0 \leq i < p+q} ||h_k^i - \hat{h}_k^i||_F \tag{12}$$

The detection of an anomalous point or subsequence over the derived anomaly scores is usually a result of a threshold value. A fixed-value is determined in [21, 29], either empirically [21] or as the maximum regression error in the training sequence [29]. A dynamic one is determined in [9], to result a set of anomalous subsequences E_{seq} from the smoothed scores e_s. In addition, authors noticed that the precision of detection heavily depends on the amount of data used to set a threshold. They thus proposed a pruning mechanism to mitigate False Positives (FP), by re-examining the abnormality of subsequences in E_{seq}. To this direction, a vector e_{max} is constructed as a result of appending the sorted E_{seq} along with the maximum score in e_s, where E_{seq} is sorted in descending order based on the maximum error of each $e_{seq} \in E_{seq}$. Then, a vector d is computed by applying a percent decrease on e_{max}. The anomalous subsequences whose score in d exceeds a minimum decrease percentage p remain anomalous, and are reclassified as normal otherwise.

2.5 Analysis and Limitations

In this paper, we examine top-notch instantiations under the *Generic Normality Feature Learning* framework for detecting anomalies over multivariate time series. Due to the multitude of (deep) model perspectives, we considered recent studies [4, 14, 24] to ensure our broadness on this topic. The key concept on the generic nature of this framework

is at its objective function, according to which a model is forced to capture the underlying data regularities. Thus, the only assumption to empower anomaly detection is that *normality is identical to regularity*, without the need for prior knowledge over a particular anomaly measure.

We report the nine characteristics addressed by the aforementioned models to support temporal-based anomalies. Specifically, these characteristics are aggregated along with their corresponding references in two groups, the *Temporal Support* (table 7) and the *Anomaly Support* (table 8). Note that we only consider methods that exhibit at least one temporal characteristic.

2.5.1 Temporal Support

A model requires a forecasting protocol to determine the mapping process from context to horizon, in the form of implicit *IF-Then* rules. The vast majority of methods use the estimation-based protocol to self-map current records, elaborating temporal information limited to the present. The remaining models use the prediction-based protocol to cross-map the current records to future records, elaborating an advanced temporal relation along time.

References	**Prediction-based Protocol**	**Implicit Attention**	**Explicit Attention**	**Dynamic Structure**
[7, 13, 15, 18, 19, 20] [23, 26, 27, 28, 33, 34, 36]	✗	✓	✗	✗
[25]	✗	✓	✓	✗
[12, 35]	✗	✗	✓	✗
[21, 29]	✓	✗	✗	✓
[9]	✓	✓	✗	✗

Table 7: The fundamental characteristics of a temporal-based Model

It also requires an architecture that pays both implicit and explicit attention to the temporal information. An implicit attention is employed using a gating mechanism to control and filter the temporal information stored in the model. LSTM or GRU recurrent layers are used to learn an inference relation type from context c to horizon h (i.e. *If* $\rightarrow$ *Then*), or an equivalence relation type (i.e. *If* $\Leftrightarrow$ *Then*) taking into account their bi-directional inference. The recurrent layers in many cases are used to formulate encoders and decoders, learning a compressed representation of the underlying regularities into a latent space. Many instantiations exhibit an explicit attention to distribute weights over the encoder results, such that the decoder is enabled to subjectively pay attention to more informative parts of the encoded information.

The hyper-parameters of an architecture control the goodness of generalization over the underlying regularities and thus the forecasting precision. To this direction, the final architecture preferences have to be tailored to the problem (domain) features, resulting a model with dynamic structure. Although the majority of models use manually selected hyper-parameters, few of them automatically derived them.

2.5.2 Anomaly Support

A trained model, initialized considering Table 7, is then used to detect a temporal-based anomaly as the records which exhibit an irregular representation with respect to the learnt ones. The context representation is evaluated using reconstruction error ($c = h$), and the horizon representation using regression error ($c \neq h$). Both errors measure the severity of irregularity as the distance between actual and predicted values in a feature space. The vast majority of instantiations apply distance function on the original feature space (OFS), while in few cases a latent feature space (LFS) is considered to be less sensitive to severe anomalies.

The severity of irregularity is the model's confidence of how abnormal a sequence of record(s) is. The higher the error value, the more abnormal it is considered. Since a model learns a general representation of regularities, it is practically impossible to result zero error, even for records that exhibit a regular pattern. Hence, several methods proposed a threshold methodology to determine up to what error value the record(s) are considered normal. Only few of them derives that in a dynamic manner, considering the forecast statistics (in either training or testing phase) to calibrate that threshold value as low as possible to avoid False Negatives, reminding that a low valued threshold may introduce additional False Positives (FP). In this case, an anomaly pruning methodology can be proposed to mitigate FP.

The interpretation of anomalies is not a strictly defined terminology. In recent methods it is implemented in terms of root cause and severity level. The former refers to the features that contributed the most to classify a record as

References	RecE	RegE	DynT	AnoP	AnoI
[13, 18, 19, 23, 25, 33, 36]	OFS	✗	✗	✗	✗
[28]	OFS	✗	✗	✗	✓
[7]	OFS	✗	✗	✓	✗
[20, 27]	OFS	✗	✓	✗	✗
[35]	OFS	✗	✓	✗	✓
[15, 34]	OFS/LFS	✗	✗	✗	✗
[12]	OFS/LFS	✗	✗	✗	✓
[26]	LFS	✗	✗	✗	✗
[21]	✗	OFS	✗	✗	✗
[9]	✗	OFS	✓	✓	✗
[29]	OFS	OFS	✗	✗	✗

Table 8: Characteristics of a temporal-based anomaly detection (RecE: Reconstruction error ; RegE : Regression error, DynT : Dynamic threshold, AnoP: Anomaly pruning ; AnoI : Anomaly interpretation)

anomalous, while the latter refers to its duration.

2.5.3 Limitations

Tables 7 and 8 summarize the limitations of the recent literature. None of the methods is able to support all the nine characteristics, and we consider as a baseline the work in [9]. Our work, aims to upgrade old components, by introducing new ones and modifications to the aforementioned scientific directions. In the major changes is included: (i) an advanced back-bone model, inspired by [29], to replace the 2-LSTM initial layers ; (ii) the use of Bayes' theorem to optimize the depth of LSTM layers along with other hyper-parameters ; (iii) an alternative unsupervised thresholding methodology, which first allocate critical regions over the testing errors and then robustly determine the degrees of freedom of normality in them ; (iv) the interpretation of the anomalies from both data and model features space. These and more are addressed one by one in the following sections, highlighting their importance and setting up their role.

3 The DITAN Framework

The task of detecting anomalous records can be seen as a *many to one* forecasting scenario, in which the sensor values of a record at time t is estimated given a temporal window of records until time t-1. Hence, both temporal and feature information is crucial. It is required an architecture which learns a latent representation of normality, through mapping temporal windows to single records with respect to time. This results a way to build the feature values of a record and compare them with its actual ones. Since construction is based on the normality that has been lernt, the higher the deviation between estimated and actual values of a record, the more the record is considered anomalous.

The high level steps of our framework are graphically illustrated in Figure 2. Section 3.1 stresses the importance of pre-processing time series data in a favorable format for such an architecture. Section 3.2 extensively analyzes the memorization mechanism of DITAN. We strongly recommend in section 3.3 several generalization techniques, that when combined can greatly increase the degrees of freedom of normality. This section also sheds light on the importance of batch size, proposing a methodology to properly select the period of updating and resetting weights during the learning process. Hyper-parameters of this framework as well as the optimizer used to configure them are presented in section 3.4. Section 3.5, introduces a robust methodology to construct a threshold allowing the classification of feature values into normal or anomalous. The severity of anomalies as well as their interpretation are detailed in section 3.6.

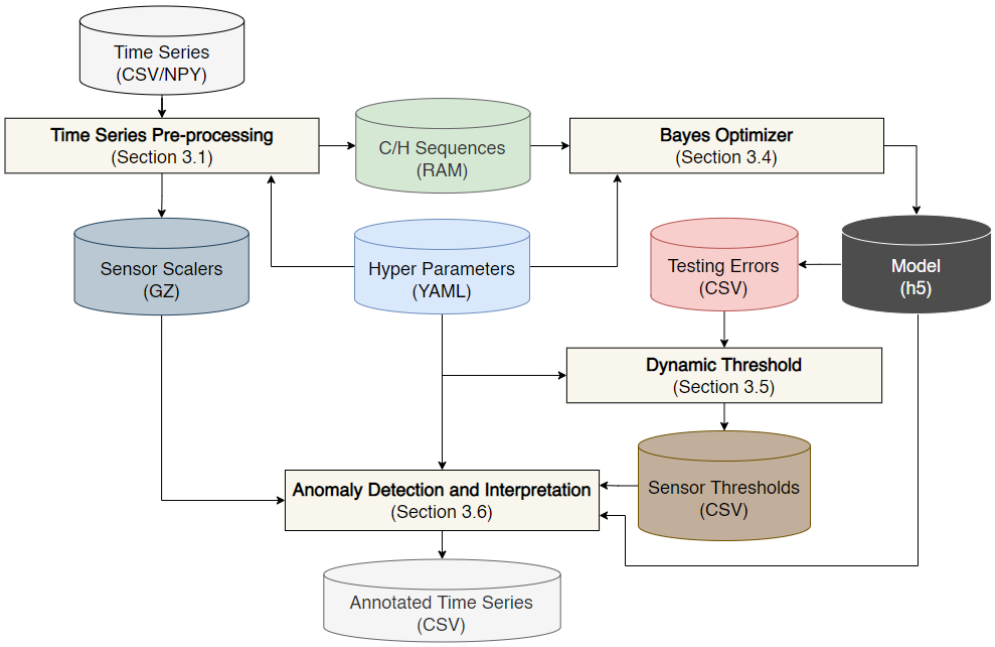


Figure 2: An overview of the DITAN Framework

3.1 Time series Pre-processing

Preprocessing is an integral step in machine learning. The data quality and the useful information that can be derived, directly affect the ability of a model to learn. Therefore, it is important to pre-process data in both temporal and features aspect, before feeding them into a model (Figure 3).

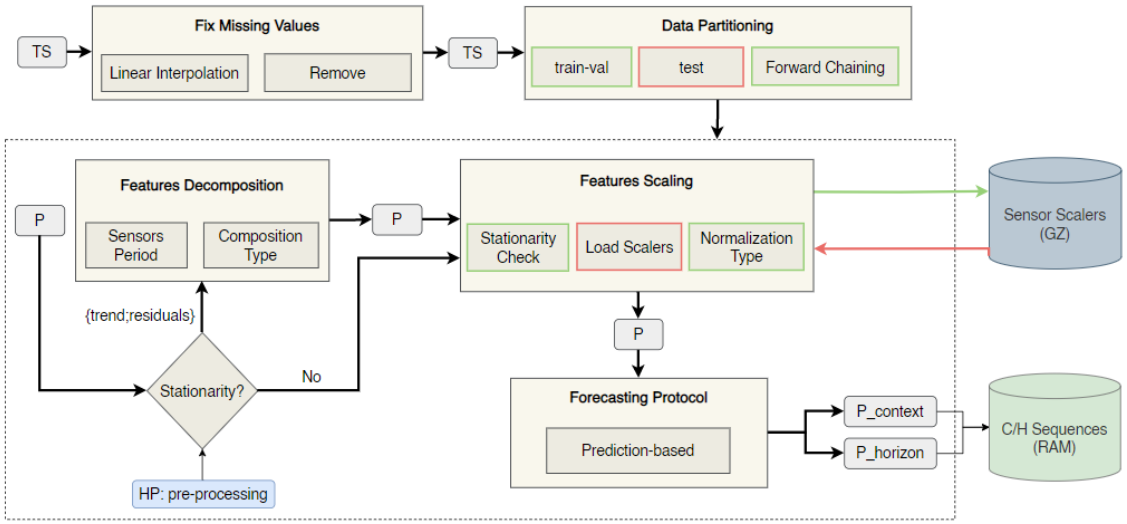


Figure 3: A graphical illustration of the pre-processing steps (red lines: only testing phase ; green lines: only training phase : black lines: both training and testing phase. TS stands for time series, and P for partition.)

Fix Missing Values. When dealing with time series, missing sensor values are frequent. Handling data incompleteness is the very first pre-processing task, since the architectures in our scope of interest can not handle them. To this direction, we impute missing values per sensor and/or remove entire records according to a meta-rule, that takes into account the gap size of missing values. More precisely, the records of missing values gap larger than *context* size gets removed, while the rest of records gets linearly interpolated per sensor (mono-spectral). Although more advanced

(e.g. polynomial) interpolation methods have been introduced, their degree and shape parameters are not easy tunable.

Data Partitioning. In the training phase, the provided sequence namely *train-val* is then partitioned with respect to its records temporal order. Particularly, *Forward Chaining* cross validation is applied on a copy of it, to result *train* and *val* sequences, where the size of *train* is expanding while the size of *val* remains constant. The number of these sequences is conditioned to the *train-val* size, in a way that the largest size of *train* sequences cannot be less than 80% of the *train-val* size and the size of *val* sequences can not be less than 150% of *context* size. This condition ensures that statistical properties between largest train sequence and train-val sequence are similar. Hence, along with the entire series, we also result a large number of partitions.

Features Decomposition. Time is a fundamental factor in time series data. It introduces three important statistical properties; trend, seasonality and residuals. The composition type of these properties, as well as the period of incoming observations determine the sensor itself. Although, these factors are rarely given, we estimate them to decompose each sensor into residuals or trend, if requested. That is because, high frequency anomalies are more visible in residuals, while the low frequency ones can be detected in trend. The period of a sensor is estimated in three steps: (i) remove the mean signal (a.k.a DC component), (ii) calculate its auto-correlation and, (iii) select the largest valid peak from its second-order difference. A peak value is valid when enables at least two cycles. Moreover, the composition type of a sensor is estimated in two steps: (i) compute simple *moving average* (SMA) using 5% size non-overlapping windows and then, (ii) estimate the variance of subtraction as well as division, between its *SMA* and actual values. We assume that a sensor is composed in additive way, when the result of subtracting has less fluctuations than dividing. Otherwise, it is assumed to be composed in multiplicative way. Note that decomposition is optional, however when selected it is applied independently per partition. The hyper-parameter that controls the existence of decomposition is called *stationarity*.

Features Scaling. The main purpose of normalization is to provide a uniform scale for numerical values. As argued in the past, a model performs better or converges faster when features are on a relatively similar scale. Any model that computes distance or assumes normality needs to perform scaling for features before training. However, arbitrary choice of a scaling method is not recommended, due to changes over the statistical properties of a time series such as increasing average, which is usually caused by the presence of trend. In order to take into account the stationarity of each feature, we apply *Min-Max* (-1, 1) normalization when both *Augmented Dickey Fuller* (ADF) and *Kwiatkowski-Phillips-Schmidt-Shin* (KPSS) statistical tests agree that the series is strict stationary. Particularly, in ADF p-value must be ≤ 0.05 (reject) and in KPSS p-value must be > 0.05 (fail to reject), since they use an opposite definition (null hypothesis) of stationarity. Otherwise, we standardize using the *Robust Statistics* from its first (Q1) and third (Q3) quartiles, assuming that changes to the core of its statistics are smoother. Thus, scalers are fitted and transform the training partitions (*train*, *training-val*), while only transform the testing partitions (*val*, *test*). A scaler learns the statistical properties per sensor as the normal behavior during the fitting process, and uses these statistics to normalize partitions during the transforming process. The sensor scalers of *train-val* sequence are then stored for future data. Scalers are saved when the train set is processed.

Forecasting Protocol. Due to the temporal order of records, each partition P has to be analyzed using an iterative process. To this direction, we utilize the *Prediction-based* forecasting protocol to introduce a temporal inference relation between current and next records, formulating a self-supervised environment of *context* and *horizon* windows. Intuitively, context window can be seen as a set of preconditions to satisfy the horizon, in a way that the more the preconditions the more complex rules can be learnt. The selection of context size is not an easy task. If the size is too large, it will increase the computation complexity and time delay by requiring more historical data for analysis. If it is too small, a time series pattern might be lost. On the other hand, the size of horizon is fixed. Note that, sensors in *context* window may differ from the ones in *horizon* if that is requested. The union of these sensors called *active* sensors.

3.2 LSTM Encoder-Decoder with Attention

Modeling such forecasting environment, requires a sequence to point (S2P) architecture to learn relevant feature patterns with respect to time. The forecasting resolution corresponds to the context size, since each record is a step in time also known as *time step*. However, in an S2P environment the attention is given only on the latest records of the context window. To reveal relevant information from the entire context, it is required a composite modeling approach that considers both implicit and explicit attention, memorizing short-term and/or long-term patterns. The information extracted from latest records of a context is known as *short term*, while *long-term* additionally includes information from the former ones. The proposed model architecture is graphically illustrated in Figure 4.

In this work, we employ an Encoder-Decoder architecture, composed of LSTM layers along with a composite

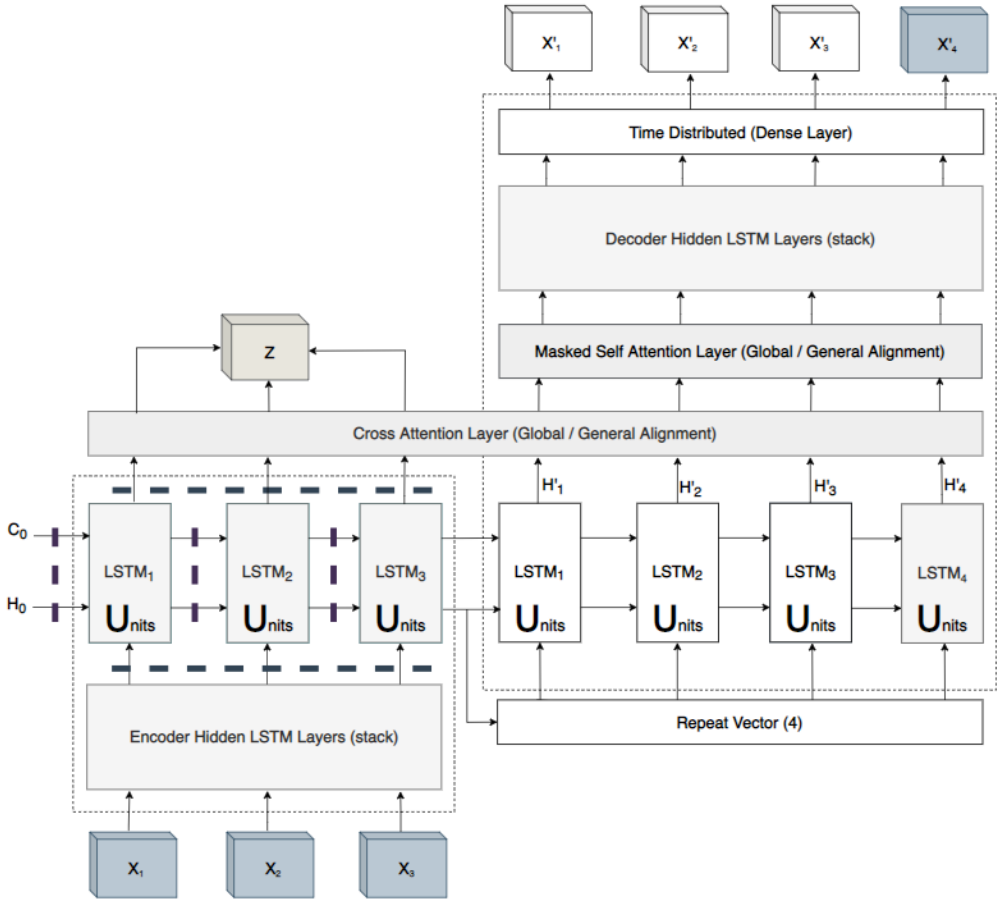


Figure 4: A graphical illustration of the composite Encoder-Decoder with attention. Solid lines represent information flow while dotted lines regularization on the features (horizontal) and temporal (vertical) axes.

decoder and soft attention mechanisms, to capture short-term and long-term normal (regular) patterns. Although the number of layers and units in both encoder (left) and decoder (right) networks are configurable, the number of LSTM cells is fixed. Specifically, the number of cells in the encoder is equal to the context size while decoder has always one LSTM cell for the construction of the next time step and context size more to reconstruct. Such a composite decoder forces the network to stay attentive to all time steps in the encoder, instead of only the last few ones [29]. Each LSTM cell in the encoder is regularized (Figure 4, dotted lines) in both temporal and feature axis to introduce generalization on the memorizing process. On the other side, the first layer of the decoder is initialized by the last memory state of the encoder. Its output is then passed through cross attention to allow the decoder to selectively access encoder information during decoding. That is followed by masked self-attention to introduce relevance importance with respect to time, in terms of probabilities over the hidden states of decoder. Finally, the last decoded hidden states are passed through a dense layer, to be transformed back into the original input feature dimensionality. Despite the composite decoding task, we only evaluate the construction of the next record.

As an example, a context of three records x_1, x_2, x_3 is compressed and decompressed through our architecture, to construct the features of the horizon x_4 along with the reconstruction of the features in the given context. The compression refers to a linear decay of units over the LSTM layers in the encoder, while a corresponding decompression is applied on the LSTM layers of the decoder. This also requires that encoder and decoder LSTM layers must have the same number of units, which stands due to their symmetric relation. The components of our proposed architecture are extensively described in the following paragraphs.

Short-term Memory and Implicit attention. To memorize feature patterns with respect to time, it is recommended to use and maintain memory units. The *Long Short Term Memory* (LSTM) cell was introduced to tackle this issue. It uses a gating mechanism that controls the memorizing process at each time step.

Long-term Memory. LSTM cells can then be connected, formulating an *LSTM layer*. The number of LSTM cells in a layer is fixed and equal to the number of input time steps. These LSTM cells are connected to each other,

from left to right, within their hidden and cell states. The interconnected cell states between LSTM cells in a layer represent the *long-term* memory. Hidden states are connected to iteratively construct predictions. Thus, an LSTM layer is inherently deep in time, since time steps can be seen as the change over time.

Complex Feature Patterns. LSTM layers can also be connected upwards, to construct a *stack* of layers, also known as LSTM *network*. Since LSTM layers operate on sequential data, the additional layers recombine the learned representation from prior layers with respect to time. The goal is to create a more abstract feature representation using the same or different (units) dimensionality. LSTM layers are stacked in a way that each layer takes as input all the hidden states of its previous layer. In other words, at each time step we are going deeper in feature space, looking for more complex feature patterns. However, stacking layers over time is not always an advantage over a single LSTM layer. It depends on the specificity of the problem and the relationships being modeled.

Fading Memory. Although LSTM cells use a gating mechanism to tackle both vanishing and exploding gradients, fading memory is not a closed case. Actually there are two factors that affect the magnitude of gradients: weights and derivatives of activation functions. If either of these is smaller than 1, then gradients may vanish over time steps. Due to the derivatives of their activation functions and the gating mechanism, LSTM may not have an exponential weight decay factor. Since weights decrease linearly with respect to the length of the input time steps, the model often pay less attention to the earlier parts of the input sequence when processing its last parts.

Explicit Attention. Fading memory is usually eliminated by controlling the contribution of time steps though an attention mechanism. Explicit attention can be given to different parts of an encoder-decoder network, and several types are introduced in literature. Among these mechanisms, we employ two attention layers. The first one is a *cross* attention between encoder and decoder, to map the important and relevant hidden states from encoder to the hidden states in the decoder and assign higher weights to them. In this configuration all the hidden states of the encoder's last layer are considered while calculating *attention weights* for each hidden state in decoder. The second one is a *masked-self* attention over the hidden states on the decoder's first layer, used to examine their causal relation with respect to their temporal order. In this configuration, all the prior hidden states of the decoder are considered while calculating *attention weights* for each next time step. Both layers use *global* attention with a general alignment scoring function [17]. Attention weights define a probability distribution over the encoder (cross) or decoder (self-masked) states, to compose a context vector defined with:

$$
\begin{aligned}
c_t &= \sum_s a_{ts} h_s && \text{[Context vector]} \\
\forall s \; a_{ts} &= \frac{exp\left(score(h_t, h_s)\right)}{\sum_{s'} exp\left(score(h_t, h_{s'})\right)} && \text{[Attention weights]} \\
score(h_t, h_s) &= h_t^T W h_s && \text{[Luong's (general) multiplicative style]} \\
h_t^a &= tanh\left(W_c[c_t; h_t]\right) && \text{[attentional hidden state]}
\end{aligned}
$$

where h_s (respectively h_t) are the source (resp. target) hidden states of the encoder's last layer (cross) or decoder's first layer (self-masked) and W_c, W are trainable weights. An attentional hidden state h_t^a is then computed to concatenate the information of h_t and c_t.

Latent-Space of Normality. LSTM networks are used to represent the encoder and composite decoder, with decreasing and increasing number of units respectively. The objective is to train both of them to encapsulate relevant information from the input time steps with respect to the output ones. In a classic ED architecture, relevant information is stored in a single vector corresponding to the encoder's last hidden state (latent space z). In this work, cross attention enables to maintain z as a matrix, corresponding to all the hidden states of the encoder's last layer. Thus, normality is what z encapsulates as regular patterns, given the training sequence.

3.3 Towards Generalization

The ability of a model to generalize is central to its success. In order to avoid overfitting, we propose an immune system to our architecture, by stressing out the importance of two regularization techniques over feature and time dimension, as well as a learning strategy.

Dropout. We build an ensemble of sub-networks by applying *Dropout* on the output units of LSTM cells, vertically (feature axis). During the training process, output units are randomly and uniformly excluded from weights updating, set to zero, with a certain probability. The number of sub-networks corresponds to the number of epochs, since dropout refresh on every epoch. However, dropping units over too many LSTM layers may underfit our model due to over regularized sub-networks. To that reason, dropout is applied only on the LSTM layer(s) in the encoder, regulating the generalization of the latent space z. The concept is to construct an entire context horizon using a subset of the memorized features in z per epoch. The probability to drop a unit is a tunable hyper-parameter. It actually indicates

the volume of regularization. Although, there are horizontal (temporal axis) dropout variations (e.g. recurrent dropout), we do not recommend to drop memory units arbitrary, because there is already a gating mechanism to maintain memory, and memory may be damaged through random temporal disconnections.

Regularization. We can also regularize our architecture in the horizontal (temporal) axis using the *recurrent weight*, imposing norm constraints (L1 and/or L2) on the recurrent weights within the LSTM cells of a layer, instead of dropping them randomly and uniformly. This is known to have the effect of reducing overfitting and improving model performance [16]. We configure recurrent connections using the Elastic net, which make use both L1 and L2 norms in a linear combination. L1 forces the network to drop recurrent weights that do not contribute to the predictive power significantly enough. On the other hand, some weights have strong pairwise correlation coefficients and therefore dropping them out would lead to information loss from memory. L2 forces the model to result relatively small weights. In this way, the cell state (memory) units regularized without dropping them necessarily. Elastic net is applied to the recurrent units of each LSTM layer in the encoder, during the training process using a regularization term in the form $\lambda(\|.\|_1 + \|.\|_2)$. *Strength* λ is a tunable hyper-parameter, common for both L1 and L2.

Learning Scheduler. The *learning rate* (lr) controls how quickly the model is adapted to the problem, and therefore highly affects its stability. A model trained using small lr requires more epochs than in larger lr, since smaller changes of the weights requires more epochs than in rapid changes. However, when learning rate is too large, model may converge too quickly to a sub-optimal solution and is prone to oscillations. The lr value as well as the rate of change, known as *scheduler*, are tunable hyper-parameters. In this work, the *scheduler* is responsible to keep lr value constant or decrease it over step or exponentially along the epochs. Specifically, *exponential* decay decreases lr exponentially by a decay d of 10% at every epoch, while *step* decay drops as for it lr by 25% after every 4 epochs.

Learning Strategy. A major challenge in training neural networks is setting the number of epochs. The objective is to train them enough to learn the mapping, but not so long to overfit and not too little to underfit. In this work, we utilize the *early stopping* strategy to find an optimal epoch to stop the learning process. It uses a criterion, which gives a penalty as long as the *delta* δ of current c from previous p loss is less than 0.0003. A penalty is formally given, when $loss_c > loss_p - \delta$ is satisfied. The number of consecutive penalties that has to be given to end the training process, is a tunable hyper-parameter called *patience*. The performance of the model is monitored on the last 20% overlapping partition of the *train* sequence. This partition is called *internal validation* and the loss used for monitoring is called *internal* loss. The final model weights correspond to the last epoch, because updates are done in a sequential manner.

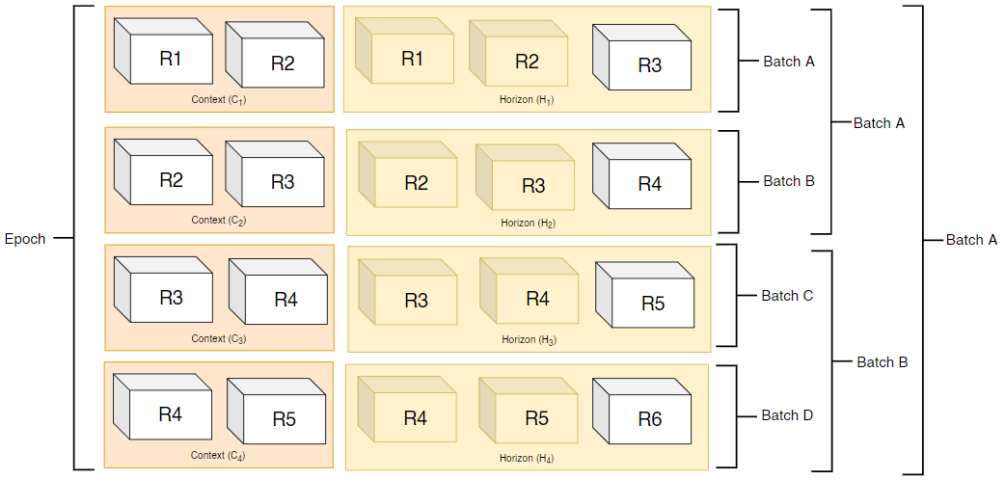


Figure 5: Epoch and batch sizes, over a partition of four context-horizon mappings

Updating Period. Due to recurrent LSTM layers, the model is trained using *Back propagation Through Time* (BPTT). Batch size is the amount of records in context-horizon mappings considered for updating the weights. The impact of different batch size to the partitioning is illustrated in Figure 5. It also affects the resetting period of memory, since in a (stateless) LSTM layer the *cell state* gets cleared at every batch size. In other words, the number of batches is equal to the number of memory resets. Therefore, batch size impacts the stability as well as the duration of the learning process. The majority of works uses 32 as a gold standard size. However, an arbitrary value may end up to an erratic definition of normality. It is well known that decreasing batch size slows down the learning process, while increasing it produces a lack of change in data over the epochs and higher memory cost. In this work, batch size is a tunable hyper-parameter in a range of the aforementioned guidelines. Note that, shuffling across records or batches is not enabled during the fitting process. For example, in Figure 5 batch B always follows after batch A, etc.

3.4 Hyper-parameter Optimization

The hyper-parameters of our framework are presented in table 9. Many of them are fixed or manually driven (see column *Tuned by*). However, in the training phase, some have to be defined automatically to ensure the adaptability to different scenarios. In this section, we introduce these hyper-parameters and propose a method to optimize them.

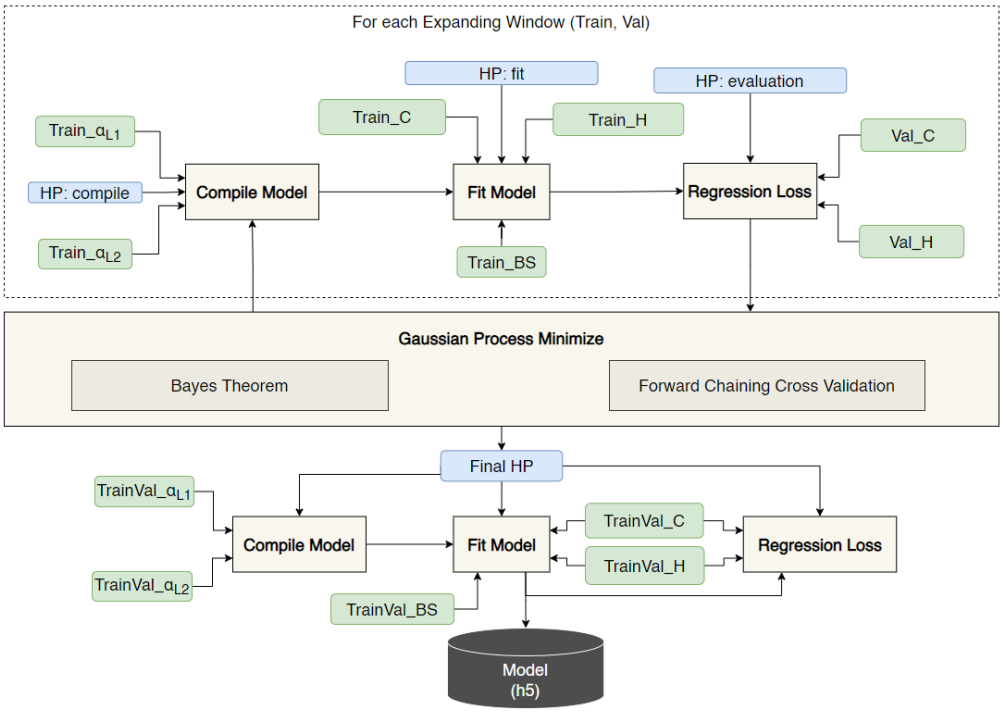


Figure 6: A graphical illustration of the optimization process. In blue we illustrate the corresponding *Type* of hyper-parameter from table 9, while data partitions are highlighted in green.

The training phase is composed of three processes (Compile, Fit and Evaluation in Figure 6), each equipped with hyper-parameters defined in Table 9. During the compile step, the encoder and decoder LSTM networks have three hyper-parameters which cannot be accurately defined using prior data knowledge. The number of LSTM *layers*, the number of *units* in the first layer and the *units decay* which is a ratio of decreasing (encoder) or increasing (decoder) units per layer. Because encoder and decoder are symmetric networks, it is enough to estimate these hyper-parameters only for the encoder. From the features axis regularization, *dropout* is a tunable hyper-parameter, indicating the probability of dropping input units in each LSTM layer. From the temporal axis regularization, *regularization strength* (λ) is a tunable value, common for both L1 and L2 norm in the elastic net. There are four more tunable hyper-parameters, in the *Fit* step: the *learning rate (lr)*, the *learning scheduler*, the *patience* and the *batch size*.

These nine hyper-parameters are optimized using a *Bayesian approach* (Figure 6), as it is a popular approach to deal with the optimisation of expensive black-box functions. Bayesian approaches, in contrast to random or grid search, keep track of past evaluation results to form a surrogate function which maps hyper-parameter configurations to a probability value. The idea is to approximate the objective function using a Gaussian process. It starts by finding a configuration that maximizes the expected improvement of the surrogate function. Then, it hands this configuration to the objective function for evaluation to retrieve the corresponding score. The surrogate function is updated along the feedback of the objective function by applying Bayes' theorem. In this work, each configuration is evaluated on the forward chaining partitions. Specifically, the model is trained on the *train* sequence and evaluated on the *val* sequence of each partition. Due to the expanding window size over partitions, the latter the partition the more available the data for a model to be trained on, and therefore the more reliable evaluation. To this end, the (val) loss of each window is weighted using the ratio of its size to the size of the larger expanding window. This weighting method enables to give more importance on the latter windows. Thus, the objective is to find the configuration C_{obj} minimizing the weighted average loss over expanding windows EW:

$$C_{obj} = Argmin(avg_C(loss_{EW} * weight_{EW})) \tag{13}$$

Note that we evaluate only the regression part of its loss function.

Hyper-parameter	Value (Range)	Type	Tuned By
Layers	[1, 3]	Compile	Bayes Opt.
Units	[32, 128]	Compile	Bayes Opt.
Units Decay	[0.5, 1.0]	Compile	Bayes Opt.
Dropout	[0.0, 0.3]	Compile	Bayes Opt.
Regularization Strength	{0.0; 0.0001; 0.001}	Compile	Bayes Opt.
Learning Rate	{0.001; 0.01}	Fit	Bayes Opt.
Learning Scheduler	{constant; step; exponential}	Fit	Bayes Opt.
Learning Patience	[5, 10]	Fit	Bayes Opt.
Batch Size	{32; 64; 128; 256}	Fit	Bayes Opt.
Loss Function	MSE	Compile/Evaluation	Fixed
Model Optimizer	Nadam	Compile	Fixed
Activation Function	tanh	Compile	Fixed
Stationarity	{residual, trend, no}	Pre-processing	Manually
Context Size	N	General	Manually
Optimization Runs	{0, 20}	Bayes Opt.	Manually
Global Runs	40	Thresholding	Fixed
Local Search	True	Thresholding	Fixed

Table 9: A summary of the hyper-parameters

In addition, there are eight more fixed or manually hyper-parameters determined by a user or an expert based on the application. Two hyper-parameters are used to compile a model during the training phase; *optimizer* and *loss function*. The former one is selected to be *NAdam*.Both training and evaluation process use Mean Squared Error (MSE) as a loss function. *MSE* is used to particularly penalize the presence of abnormal values and shows up broaden levels of abnormality. The *tanh activation function* is applied on each LSTM layer to introduce non-linearities. In addition, *stationarity* determining whether time series need or not to be transformed into stationary (default is *no*). A fundamental application-based hyper-parameter is the *context size*, corresponding to the size of temporal sliding windows. It is recommended to be larger than the maximum expected anomaly length, to avoid being absorbed from an anomalous event. The number of runs using Bayes optimizer is determined by *optimization runs* hyper-parameter, where zero means no optimization (default is *20*). Note that, a single run evaluates a configuration setting across N expanding windows. Last but not least, there are two hyper-parameters corresponding to our dynamic threshold methodology (see Section 3.5). The *global runs* (default is *40*) indicates the number of jumps to a different global point to search for an optimal solution. The *local search* (default is *True*), as for it, indicates the need for going deeper into a local space or not.

3.5 Dynamic Threshold with built-in Pruning

A model uses its latent representation of normality to predict the sensor values of a record. The higher the error over a sensor value the more abnormal it is considered. Yet, due to its generalization in an unsupervised environment, the turning point from normal to abnormal error values is not well defined. As mentioned in Section 2, an upper-bound threshold of normality per sensor is usually estimated using statistics from *training* and/or *testing* errors sequences. Such approach is recommended, only when robust statistics are used. Alternatively, arbitrarily selecting an error (e.g. maximum) value is biased by the presence of rare error values in the form of spikes and therefore there is a high chance of introducing False Negatives (FN) [2]. A single threshold per sensor is then expected to miss spikes of varying magnitude across different regions of an error sequence, introducing FP[3] which are usually next to a large in magnitude error due to the use of sliding windows [7]. In this work, we introduce a two stage thresholding methodology per sensor. First we generate region proposals using a simple moving average (SMA) across the *testing* errors sequence,

[2]FN refers to missed anomalous records, since considered as normal
[3]FP refers to normal records, falsely considered as anomalous

and then we use a robust pruning methodology optimized to shift a threshold of normality upwards from the plateau of smoothed errors per region.

Critical Regions. In the first stage, the testing errors sequence is smoothed using *SMA* using Parzen type windows of $2 * context$ size. Basically, we consider the number of observed records to smooth the (prediction) errors, doubled for more compact error magnitudes. Such windows are appropriate for centralizing errors along time, formulating Gaussian-like bumps. A simple comparison of neighboring errors is then applied across the entire SMA to find peak values on different errors magnitude. However, low height (noisy) peaks located to high frequency bins may be introduced and thus have to be filtered. To this direction, we discretize the SMA values into an optimal number of bins, considering the shape (normal or skewed) of the underling distribution. We assume that the first bin exhibits the mode errors frequency and thus the frequency on the next bins decreases until magnitude fluctuations are introduced. A bin at position k is said to be a part of a decreasing trend if it's frequency is lower than the one at position $k+1$ or $k+2$. The left-bound of the last bin determined the *minimum peak height*. Peaks under that value are considered (irrelevant) noise.
A *critical* region is an expanding window centralized to a peak value. The width of a critical region increases symmetrically by 2, as long as its average error is significantly greater than the SMA average, using Mann Whitney test with a significance level 0.05. Critical regions are further examined for specifying abnormalities, while the errors in non critical regions are considered normal.

Robust Pruning. In the second stage, a batch analysis is applied on the smoothed errors of each critical region cr independently. We apply the Mean-shift algorithm that works by updating candidates for centroids to be the mean of the points within a given *bandwidth*. Since Mean-shift is extremely sensitive to its bandwidth hyper-parameter, we use dual annealing optimization to tune an optimal bandwidth using both local and global search space, where the use of local search and the number of global jumps are configurable (Figure 7). Bandwidth can be seen as the kernel radius and thus candidate values are bounded in the distances range $[min, max](nearest\text{-}dist(losses_{cr}))$.

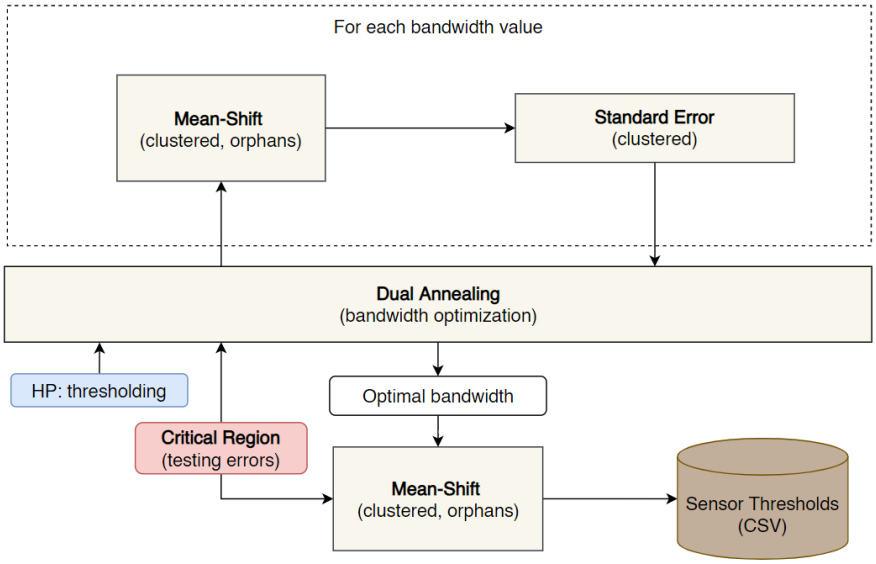


Figure 7: A robustly driven threshold at a critical region of the *testing* errors sequence.

The objective is to select a bandwidth which results in compact (low standard deviation) and heavy (large size) clusters. This is formally achieved by minimizing the maximum standard error (SE) across clusters:

$$Bandwidth_{cr}(C) = argmin\{max(SE(c), \forall\ c \in C_{normal})\}_{cr}. \tag{14}$$

Mean-shift is then handed by the optimal bandwidth to result the final clustered and orphan error values. *Rare* is an error value that is not clustered (orphan) or clustered with one more error, while *Normal* are the error values below the minimum rare error. Hence, the threshold of normality of a critical region is the maximum normal error. By doing this, we reduce both the number of *FP* by centralizing critical regions to (minimum height) non-noisy peaks and the number of *FN* by controlling their (bandwidth) dynamic length. However, this may re-partition a critical region into smaller ones. To that reason, the final critical regions are only the ones above the (aforementioned) minimum peak height.

Note that the semantics here are similar to *decrease percentage differences* applied in [7, 9] to shift threshold upwards. However, in this work Mean-shift is used as a robust metric.

3.6 Towards Anomalies

Thresholding is a crucial step for the anomaly detection process. It determines how much an actual value can deviate from the predicted normal value. A threshold value t is then resulted per record i over a sensor d, equal to 0 (normal) or 1 (anomalous).

Scoring. A severity, or anomaly, score s of a record i over a sensor d is given by:

$$score_d(e^i) = \begin{cases} e^i_d, & \text{if } t^i_d = 1. \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

The score of an anomalous sensor value is equal to its corresponding (original) error value, while it is set to for normal ones. Therefore, an arbitrary length of consecutive records which maintain anomaly to at least one common sensor are considered *subsequence* anomaly, while the individual ones are *point* anomalies.

Interpretation. Detected anomalous records can then be characterized by interpreting their *root cause* [12, 28, 35] aspect. We assume that the most anomalous sensor values can provide sufficient clues to understand and troubleshoot the anomaly. We examine the root cause of each anomalous record s^i using

$$RootCause_d(s^i) = \begin{cases} exp(s^i_d)/sum(exp(s^i)), & \forall s^i_d > 0 \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

Particularly, given an anomalous record i, we apply softmax along to its score values and then report sensor number percentages in descending order, indicating the contribution of each sensor to the abnormality of the record. In normal records the sensors probability is zero.

We then characterize similar anomalies through their internal decoded representation in the units space, using the model's perspective. For each anomalous record i we run its context c through the model to extract a rich feature representation of its actual feature values, as the output $r^i = f_D(E(c^i))$ of the last intermediate (LSTM) layer f in decoder D.

Anomalies $r \in R$ are then grouped into an optimal number of clusters C_{opt}, by applying a Gaussian Mixture Model (GMM) initialized with a general inverse covariance matrix Σ_R^{-1}. The probability of an anomalous record r_i generated by each component C_j, with mixture weights ϕ_j, of center μ_j and a common covariance matrix Σ_R across all components is given by

$$p(r_i) = \sum_{j=1}^{C} \phi_j \mathcal{N}(r_i|\mu_j, \Sigma_R) \tag{17}$$

C_{opt} is constrained in range $[2, min(10, |R|)]$ and determined as the earlier one with the most important *DB-Index* (DBI) score. The importance of a candidate is the number of consecutive candidates for which their DBI scores are higher than it. However, if all candidate values are equally important then maximum percentage difference is used. DBI represents the average similarity of each cluster with a cluster most similar to it, considering its shape using Mahalanobis, instead of Euclidean, distance. The similarities do not require expensive (pairwise) distance operations over records, only over centroids, and thus it is considered an efficient metric for large number of anomalies as well. The covariance matrix Σ_R is nevertheless not always invertible, e.g. when the number of units (features) is higher than the number of anomalous records (samples). Hence, we use a sparse inverse covariance estimation technique based on the Graphical Lassoestimator. From an intuitive standpoint, we might expect that the clustering assignment for some records is more certain than others. Since GMM contains a probabilistic model under hood, instead of only classifying each record to a cluster, we additionally provide the certainty of its prediction in terms of probability. Note that, this grouping methodology is applied only to anomalous records, when more than two are detected.

4 Evaluation and Comparison

We expose here our model to a set of experiments to evaluate its effectiveness on predicting normality and detecting and interpreting anomalies. We use six multivariate time series of varying anomalous types from the experiments in [9]. The scope of this evaluation is to show the adeptness of our model to their environment with similar if not better results. It is not a benchmarking evaluation, since as stated in [32] there are plenty of reasons (triviality, unrealistic density, mislabeled ground truth, etc.) to not to.

Models are trained and executed on a computer with 2 Nvidia TESLA v100 graphics cards of 32 GB each. The entire experiment took approximately 8 hours to be completed. The size of context and the number of layers of the LSTM networks, mainly affected the mount of time required.

4.1 Datasets

From the datasets used in [9][4], we selected three multivariate time series, or *channels*, from Soil Moisture Active Passive Satellite (SMAP) and three more from Curiosity Rover on Mars (MSL), presented in Table 10. We use both of these datasets due to their difference in the amount of (records) telemetry values and features. MSL leads on the number of features while SMAP exhibits consistently more records. These datasets are contaminated by real-world spacecraft anomalies of varying length, annotated by experts as *point* or *contextual* over the test sequence. According to [9], *point* anomalies are values that fall within low-density regions of values, while *contextual* anomalies do not, yet are anomalous with regard to local values. It is also mentioned that MSL performs a much wider variety of behaviors than SMAP, with varying regularity some of it not in the limited training sequences.

Channel ID	Spacecraft	Features	Train-val	Test	Test Anomalies
P-4	SMAP	25	2609	7783	3x point
E-13	SMAP	25	2880	8640	3x contextual
T-1	SMAP	25	2875	8612	1x point, 1x contextual
D-14	MSL	55	3675	2625	2x point
T-13	MSL	55	1145	2430	2x contextual
C-1	MSL	55	2158	2264	1x point, 1x contextual

Table 10: Synopsis of the six multivariate series used in our experiments

Channels are selected based on their anomaly properties over the test sequence. Particularly, *P-4* and *E-13* are selected from SMAP, as well as *D-14* and *T-13* from MSL, because they exhibit the maximum available number of point and contextual anomalies respectively. *T-1* and *C-1* used from SMAP and MSL cover the joint contamination of both anomalous classes to the same test sequence. Specifically, the point anomalies in P-4 (respectively D-14) are subsequences of length 130, 200 and 100 (resp. 20 and 20) records. Moreover, the contextual anomalies in E-13 (respectively T-13) have lengths 101, 40 (resp. 100 and 150). Finally, the joint contamination in T-1 (respectively C-1) refers to subsequences of 1499 and 35 (resp. 200 and 110) records.

These datasets are used without any further pre-processing to maintain common conditions. As a matter of fact, these datasets have no missing values, are not decomposed to stationarity, are already partitioned into train-val, test sequences and are scaled using min-max(-1, 1) transformation. These steps could be part of our automatic pre-processings (Section 3.1). We additionally partition Train-val via forward chaining to construct *train* and *val* sequence for Bayes Optimizer.

4.2 Hyper-parameters

Table 11 reports the hyper-parameter values resulting from 20 *Optimization Runs* per channel of the Bayes Optimizer. Each run evaluates the same configuration up to 4 expanding windows formulated by the forward chaining process. Hence, the estimated number of models that trained/validated is (6 * 20 * 4) *480*. All channels start from the same initial configuration; *layers=1 (per network), units=80, units decay=1.0, dropout=0.3, reg. strength=0.0, le. rate=0.001, le. scheduler=constant, le. patience=10, batch size=64*, which simulates the one reported in [9]. That is to guarantee that the extracted topology per channel is at least as good as the one proposed. The other hyper-parameter values are shared along channels (table 9). Particularly, *context size* is set to 25, instead of 250, training our models to learn to predict a record using 10 times less of previous records. This is an additional challenge to our models, if we consider the previously reported number of records per anomaly which in average are hundreds, not dozens.

Interestingly, we observe that almost every generated model uses a form of regularization (features and/or recurrent) for the encoder, although it is optional. Along with the presence of various batch size and learning strategies, we experimentally confirm our expectations described in Section 3.3. Considering also that all models initialized by the same configuration, the expectations of Section 3.2 are also valid. That is because all models make use of the dynamic modeling architecture (depth and/or width).

[4]https://github.com/khundman/telemanom

	P-4	E-13	T-1	D-14	T-13	C-1
Layers	2	2	2	2	1	2
Units	94	94	128	60	34	32
Units Decay	0.528	0.528	0.5	0.9998	0.615	0.5
Dropout	0.185	0.185	0.0	0.0165	0.052	0.0
Reg. Strength	0.0001	0.0001	0.0	0.001	0.0001	0.0001
Le. Rate	0.01	0.01	0.01	0.01	0.01	0.01
Le. Scheduler	step	step	step	exponential	step	step
Le. Patience	5	5	10	7	8	9
Batch Size	32	32	32	32	64	64
Trainable Param/s	161,393	161,393	284,801	133,991	28,595	24,353
Run Found	3^{rd}	3^{rd}	12^{th}	16^{th}	6^{th}	12^{th}

Table 11: The optimized hyper-parameter values over the six channels

The *trainable parameters* indicate the size of each model, mainly affected by the total number of units and the context window properties. However, in their relative model size comparison context window properties does not count, since all channels share the same context window size. Hence, T-1 is expected to exhibit the highest model size followed by P-4, E-13, D-14, T-13 and C-1 in descending order. Moreover, the *run found* indicates the position out of 20 optimization runs, at which the best configuration is reported per channel. Particularly, we observe that the majority of them are found on the early attempts, with all to perform better than on the default (1^{st}) one. This validates that 20 in total runs where enough to find the best solution.

4.3 Results and Discussion

The trained models are applied on the *Test* sequences of the corresponding channels, to evaluate predictability and detection power. We also used the models of [9]. All models are executed under a common environment, using a random seed *42* for reproducible results.

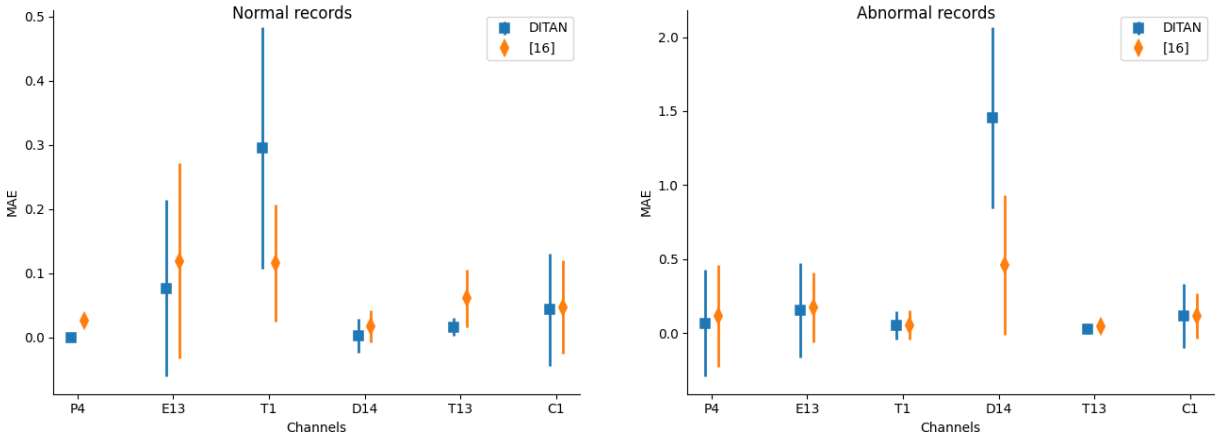
4.3.1 Predicting Normality

Predictability power refers to the ability of a model to predict normality close to the actual records when these are labeled normal and far from them when labeled abnormal. Since ground truth (labels) is provided by experts, we can separately evaluate the mean absolute error (MAE) over normal and anomalous records. Models are trained to predict normality, thus a good quality of predictions corresponds to low MAE over normal records in conjunction with high MAE over anomalous ones. The objective of this experiment is to evaluate how well normality is learned by DITAN models compared to the ones in [9], since their models defined an important and growing challenge within spacecraft operations. In addition, we use the Mann-Whitney non-parametric test to attach a statistical evidence to the results.

The resulted MAE and their standard deviations are presented in figure 8 for both normal and abnormal records. The average STD on both normal and abnormal MAE values are low, in a similar manner between DITAN and [9] methods, indicating a solid mean representation of absolute errors. Individual STD values across normal MAE is considerably lower than in abnormal MAE. The minimum STD is found on normal MAE (channel P-4) at 0.0001 for DITAN and 0.0017 for [9]. The maximum STD is found on abnormal MAE (Channel P-4) at 0.3612 for DITAN and 0.3438 for [9] respectively. Tracing the underlying values, DITAN models result in lower MAE on 5/6 channels across normal records with p-value 0.85 and thus no statistical evidence to reject that observation. Authors in [9] resulted higher MAE on 4/6 channels across abnormal records, with also insufficient evidence for rejection, but lies in a lower 0.62 p-value.

Interestingly, DITAN is favored higher on normal records than [9] on abnormal records. A large improvement is then expected on predicting normality with negligible cost on abnormal records. This is examined in table 12 measuring the *percentage change* of errors between models and their *percentage difference* per model, leading on two research questions.

Question A. *What is the percentage change on normal and abnormal errors by switching from [9] to DITAN modeling?* In overall we observe a decreasing trend on both normal and abnormal error values. Particularly, in P-4,



Normal records Abnormal records

Figure 8: MAE and standard deviations for the 6 channels.

		P-4	E-13	T-1	D-14	T-13	C-1	AVG
% change (RQA)	**Normal**	↓100%	↓36%	↑154%	↓83%	↓73%	↓8%	-
	Abnormal	↓42%	↓11%	↓8%	↑216%	↓32%	↑1%	-
Total Change (RQA)		+16%	+21%	-150%	+478%	+18%	+9%	-
% difference (RQB)	**DITAN**	+200%	+68%	-140%	+199%	+60%	+93%	80%
	[9]	+128%	+37%	-71%	+185%	-32%	+85%	55%

Table 12: Percentage change from [9] to DITAN over normal and abnormal errors, and Percentage difference between normal and abnormal errors per model.

E-13 and T-13 the decrease change is *more than 2 times* higher on normal than on abnormal, indicating a beneficial result against [9]. Generally, a large *improvement* on predicting normal records (i.e. decrease normal errors) is usually accompanied by a small *cost* on predicting abnormal records (i.e. also decrease on abnormal errors). It is the fee we pay against overfitting. Similarly in D-14 and C-1, results continue proving the higher quality on predicting normality, with no cost at all. The only exception is channel T-1. That is because it contains an anomaly of length 1499 records, which is impossible to be captured by observing context windows of 25 records. Finally, *Total Change* is computed per channel, as an indicator of the total benefit (+) or cost (-) in percentage, using $A + B + (A * B)$, where A determines the percentage change over normal and B over abnormal errors. Note that A is positive only when decrease changes while B is positive only when increase changes. Therefore, changing from [9] to DITAN modeling, the predictability power across all channels is improved by 65.3%.

Question B. *How different normal and abnormal errors are, as a factor of overfitting per model?* The quality of learned normality is further examined by measuring the *percentage difference* between normal and abnormal MAE per model. The higher the percentage value the more distinct normal and abnormal MAE values are, while zero indicates no distinction and negative sign (-) miss-conception of normality (i.e. normal > abnormal errors). The high quality of DITAN models is also observed here, maintaining the highest distinction to all channels. Yet, observation is supported statistically at p-value 0.78 with no significant evidence to reject it. These results are an indicator that DITAN is more tolerant to overfitting against [9]. It is also observed that in T-1 and T-13 channels, [9] models exhibit a negative percentage. This is a sign of normality miss-conception to these channels, while DITAN model accomplished T-13. In average, the percentage difference between normal and abnormal errors on DITAN is 80% against the 55% for [9].

4.3.2 Detecting Anomalies

Detection is an extremely important module for the analysis of the resulting scores. The objective of this experiment is to evaluate the detection module of DITAN given the actual state of records labeled by experts. For each channel, except T-1 for which normality is failed to be captured, we compute the confusion matrix together with various metrics (Table 13). Each metric is applied across channels to examine a different aspect. As a matter of fact, we

use *False Positive Rate* (FPR) to examine FP, while TP are examined using both precision (w.r.t. FP) and recall (w.r.t. FN) due to the imbalanced class environment of anomaly detection. An overall performance, mainly affected by precision is also resulted using $\mathbf{F_{0.5}}$ score. Note that all metrics are applied on a record level, while we finally measure Intersection over Union (IoU) on anomalous event level.

	TP	**FP**	**TN**	**FN**	**FPR**	**Precision**	**Recall**	**$\mathbf{F_{0.5}}$ Score**
P-4	242	0	7340	201	0.0%	100%	54.6%	0.86
E-13	113	121	8255	151	1.4%	48.3%	42.8%	0.47
D-14	222	69	2334	0	2.9%	76.3%	100%	0.80
T-13	136	33	2145	116	1.5%	80%	54%	0.73
C-1	201	142	1810	111	7.3%	58.6%	64.4%	0.60

Table 13: Confusion Matrix of the records across channels.

False Positive Rate. The probability of false alarm is examined using $FPR = FP/(FP + TN)$, measuring the portion of normal records (TN) incorrectly identified as anomalous (FP). The low probability of false alarms indicates an accurate estimation of normality across all channels, whatever their type (SMA, MSL). False alarms in D-14, T-13 and C-1 result from different alignments between critical regions and actual anomalies. In E-13, an additional critical region is introduced along with the difference in alignments (Fig. 9, second and third anomalies). In P-4, all critical regions are aligned to the actual anomalies (Fig. 10). Among all the critical regions reported by DITAN, only one is considered FP while the rest are aligned to the 12 actual anomalies.

Precision. The probability of a true alarm when an anomalous record is reported is measured using the *Precision* $P = TP/(TP + FP)$. It is the fraction of records correctly identified as anomalous among all anomalous predicted records. Precision across channels contaminated by only point anomalies (P-4, D-14) is slightly higher than in channels contaminated by at least one contextual anomaly (E-13, T-13, C-1). That is because contextual anomalies require more normal patterns to be memorized and thus introduce more noisy errors. In average, the probability of reporting a true instead of false alarm is more than 70%, while the remaining percentage is mainly due to the critical region displacement on the true alarm. This result exhibits a way for the experts to consider the detection abilities of DITAN to their analysis.

Recall. The probability of a true alarm that is correctly identified is measured using the *Recall* $R = TP/(TP + FN)$. It is the fraction of records correctly identified as anomalous among all discovered and missed anomalies. Here also, channels contaminated by point anomalies are more favored than the ones with contextual anomalies. Particularly, in channels P-4 and D-14, large error bumps cover all actual anomalies, aligned by critical regions. However, due to the pruning methodology, tails are filtered and thus a subset of actual anomalies is considered FN in P-4 (Fig. 10). Same situation occurs in E-13, T-13 and C-1, where an actual anomaly is additionally masked, under the noisy errors of C-1 and E-13 (Fig. 9, first actual anomaly). The average recall mainly indicates the expected intersection of a critical region with an actual anomaly. From the 12 actual anomalies, only two are masked and thus failed to be discovered by DITAN.

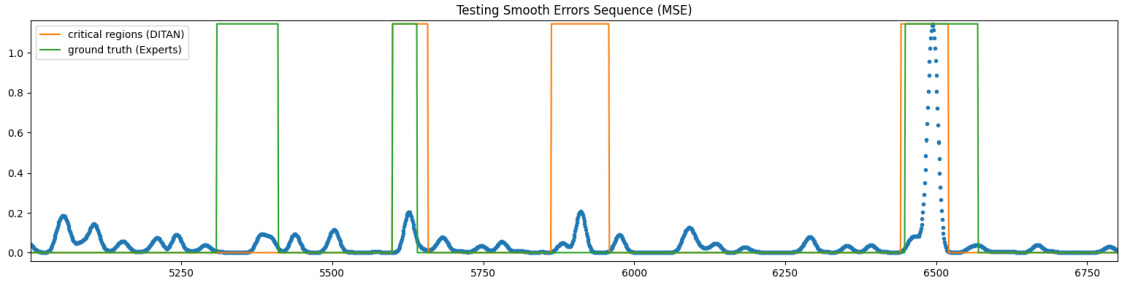


Figure 9: E-13 channel of contextual anomalies. From left to right, FN, miss-alignment, FP, miss-alignment

F-score. The harmonic mean of the precision and recall is examined using the *$F_{0.5}$ score*, $\mathbf{F_{0.5}} = 1.25P.R/(0.25P + R)$. The F-score family is very useful for evaluating results on unbalanced data. Unlike F_1 score, $\mathbf{F_{0.5}}$ score gives more weight to precision (FP) than to recall (FN). This measure is extremely useful in cases such as the sensitive telemetry spacecraft channels of this experiment, because they required to report only high certain anomalies. We observe that precision and recall values are projected across channels, resulting relatively high F-score values. The

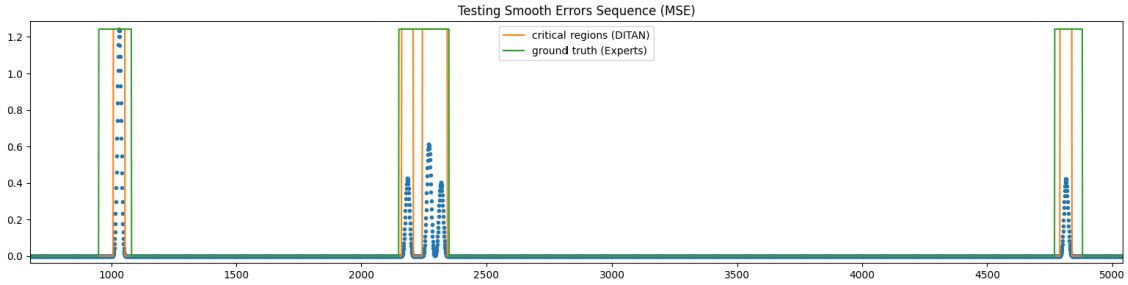


Figure 10: P-4 channel of point anomalies. From left to right, three miss-alignments

highest ones correspond to P-4, D-14 (point anomalies) and T-13 (contextual anomalies), while the lowest ones are computed on C-1 and E-13 (joint and contextual) respectively. In average, SMAP (P-4, E-13) channels result in 0.67%, while MSL (D-14, T-13, C-1) channels in 0.71%. Thus DITAN results similarly well on both datasets, despite their explicit characteristics.

		Actual Range	**Critical Region/s**	**IoU**
(P) Point	**P-4: P1**	950-1080	1008-1054	0.35
	P-4: P2	2150-2350	2161-2207, 2244-2343	0.72
	P-4: P3	4770-4880	4791-4838	0.43
	D-14: P1	1630-1650	1614-1674	0.33
	D-14: P2	1800-2000	1794-2023	0.87
(C) Contextual	**E-13: C1**	5309-5410	masked	-
	E-13: C2	5600-5640	5600-5658	0.69
	E-13: C3	6449-6569	6442-6520	0.56
	T-13: C1	690-790	657-767	0.58
	T-13: C2	1900-2050	1942-1999	0.38
Joint	**C-1: P1**	550-750	415-757	0.58
	C-1: C1	2100-2210	masked	-

Table 14: The intersection over union (IoU) per anomaly, given critical regions of DITAN

Intersection over union. We finally measure how well critical regions are aligned to the 12 actual anomalies, using the intersection over union (IoU) across corresponding records (table 14). Point anomalies are covered similarly well as contextual anomalies, resulting in 0.54 and 0.55 respectively in average, a value close to the alignment over joint anomalies 0.58. This means that DITAN is able to align critical regions over actual anomalies, despite the contamination type. Also the majority of IoU values are above 0.5, which is accepted as a good result, while the ones below are mainly due the presence of FN, such as in P-4 channel (Fig. 10).

4.3.3 Interpreting Anomalies

Last but not least we empirically evaluate how informative similarity is for interpreting the detected anomalies. The goodness on clustering anomalous records to the units space is quantified based on their corresponding DBI scores. DBI is a measurement of similarity between varying shape clusters (see Section 3.6), where the lower the score the better the clustering; further apart and less dispersed. The results of clustering per channel are reported in table 15. Although the number of clusters is constrained up to 10, the early ones are preferred due to early success on their DBI scores. Particularly, the DBI scores across P-4 and D-14 are relatively lower than in E-13, T-13 and C-1 for a similar clustering choice, because the latter channels are contaminated by contextual anomalies and thus their units (decoded) representation is more complex than the former ones. Therefore DBI scores 0.23 for contextual and 0.13 for point contamination conclude to a good choice of clustering on the units space, resulting in a compact clustering.

The role of the decoder is to reconstruct the data representation of a requested horizon, by constructing its internal representation in the units space. Although we showed the goodness on the choice of clustering in the units space, we

	Records	Units Space	Optimal Cluster #	DBI
P-4	242	94	4	0.15
E-13	234	94	3	0.23
D-14	291	60	2	0.11
T-13	169	34	2	0.23
C-1	343	32	4	0.22

Table 15: The scores of clustering in the units space across channels

also have to evaluate how informative it is on the data space. To do so, we examine if the model's understanding of anomalous records is verified w.r.t. their actual representation. This is quantified as the portion of anomalous records considered similar in the units space, that are also similar in the data space. Thus, same clustering semantics are applied on the data space of the univariate horizons, using the actual (instead of sparse) inverse covariance matrix and constrained up to the optimal number of clusters.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
P-4	16/16	133/147	73/73	5/6	227/242 (93.8%)
E-13	32/52	128/144	32/38	-	192/234 (82.1%)
D-14	154/159	100/132	-	-	254/291 (87.3%)
T-13	111/111	58/58	-	-	169/169 (100%)
C-1	182/215	24/51	21/30	23/47	250/343 (72.9%)

Table 16: The units space similarities verified in data space across channels

From the total number of anomalous records, the ones that are considered similar in the units space and verified in data space are reported in table 16. In overall, the average number of records which maintain similarity also in the data space is above 87%, indicating that the choice of clustering in the units space is also intuitive to the data space. Particularly, on channels P-4 and D-14 (point contaminated) the 91% of records similarity is verified in both spaces. On the E-13, T-13 and C-1 contextual contaminated channels!, results are slightly lower, because complexity in data space also appears in units space. Fig. 11 illustrates results of channel T-13 where all similarities are verified. On the top figure, horizons of all records are presented in the data space, where anomalous ones are colored orange (C0) and green (C1) corresponding to the units space clustering on the bottom figure. By tracing the colors, we observe that DITAN properly understands the anomalies. It first recognizes the statistical (e.g. trend) differences between the two events and thus results in no similarity between their anomalous records. It then considers the consecutive anomalies on each event similar, since their previous context results in a similar pattern.

However, depending on the number of clusters, shifts in context may occur a change to clustering (similarities) inside an anomalous event, such as illustrated in Fig. 12. Zooming to the third anomalous event of E-13 channel, we observe that consecutive records of similar context are grouped to the same green colored cluster (C1) which is repeated two times indicating the start and the end of the anomalous event. Also, two intermediate shifts in context are captured, with the corresponding consecutive records to be colored in red (C2) and orange (C0) clusters respectively.

The probabilities of chosen clusters are all equal to one across channels, indicating a high confidence level of DITAN for the chosen similarities.

5 Benchmark across Baselines

In the previous section, DITAN is evaluated across several datasets (channels) using one baseline method. In this section, we evaluate our method using one dataset and many baselines, examining all the top-notch methods from the model perspectives of the generic normality feature learning framework, extensively described in Section 2.

To this direction, we had the opportunity to contribute on the ongoing experiment initiated in [38], by introducing the performance of DITAN on the MIT-BIH Supraventicular Arrhythmia (MBA), a popular large-scale dataset in the management community. MBA is a collection of electrocardiogram recordings from four patients, containing multiple instances of two different kind of anomalies. The dataset is partitioned into *train*, *test* and *labels* sequences of common

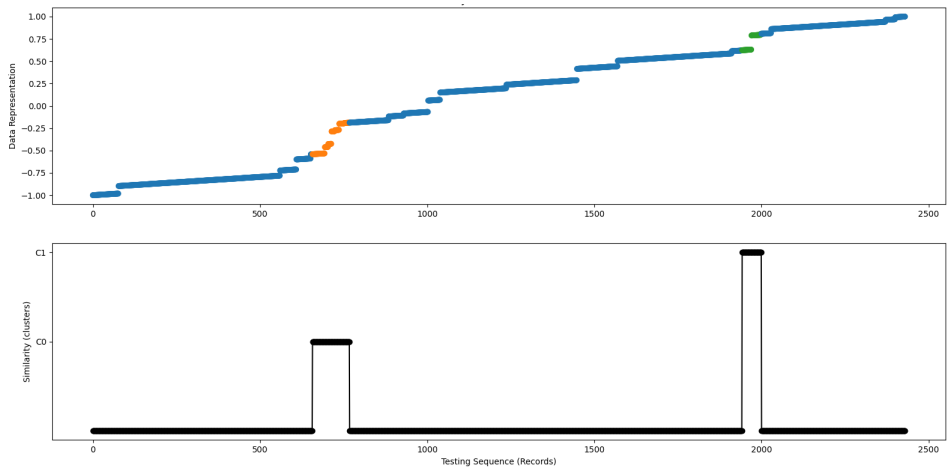


Figure 11: T-13 channel, the two anomalous events and units space clustering

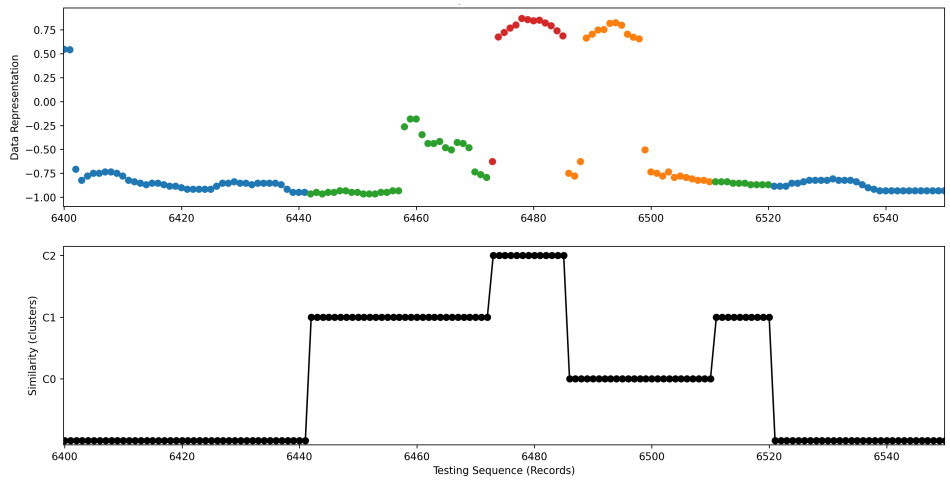


Figure 12: E-13 channel, the third anomalous event and units space clustering

size at 7680 samples. The values in train and test sequences are normalized using min-max normalization, while the duration of each anomaly is fixed to [-20, 20] indicating the left and right distance in samples from each (labeled) anomalous sample respectively.

In this experiment, DITAN uses a context size of *10* and only *5* epochs for the training phase (table 9), to maintain the same values of the common hyper-parameters in [38]. The optimized DITAN model with 22,682 trainable parameters is found at the 13^{th} optimization run (out of 20) over the *train* sequence. It uses 1 layer, 35 units, 0.404 units decay, 0.1049 dropout, 0 reg. strength, 0.001 le. rate, constant le. scheduler, 5 le. patience and 32 batch size.

Performance results are presented in Table 17. We observe lower performance on AUC and F1 as a result of a low recall. That is because true anomalies are defined by a fixed duration while DITAN constructs dynamic length critical regions depending on the time-steps required to the corresponding error values to drop back to its mean level. In other words, all true anomalies are captured with more than the half duration. In addition, DITAN takes the lead on precision since all the detected anomalies are true ones. In fact, precision slightly deviates from the perfect value, due to the only misalignment between an actual anomaly and its corresponding critical region.

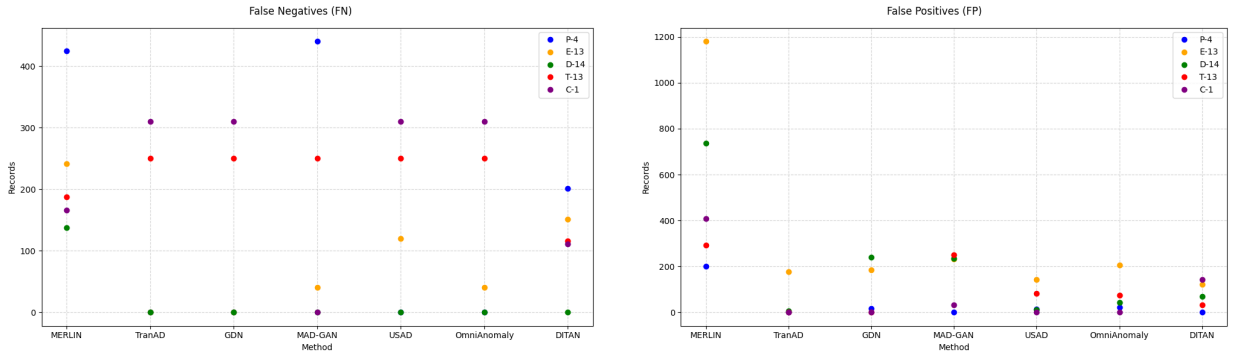
In overall, we observe that DITAN is the most precise method compared to the 11 baselines, able to capture all the true anomalous events in duration more than the half of the defined one.

	P	R	AUC	F1
MERLIN [22]	0.9846	0.4913	0.7828	0.6555
LSTM-NDT [9]	0.9207	0.9718	0.9780	0.9456
DAGMM [39]	0.9475	0.9900	0.9858	0.9683
OmniAnomaly [28]	0.8561	1.0000	0.9570	0.9225
MSCRED [35]	0.9272	1.0000	0.9799	0.9623
MAD-GAN [15]	0.9396	1.0000	0.9836	0.9689
USAD [2]	0.8953	0.9989	0.9701	0.9443
MTAD-GAT [40]	0.9018	1.0000	0.9721	0.9484
CAE-M [41]	0.8442	0.9997	0.9661	0.9154
GDN [6]	0.8832	0.9892	0.9528	0.9332
TranAD [38]	0.9569	1.0000	0.9885	0.9780
DITAN (our)	0.9910	0.7785	0.8878	0.8719

Table 17: performance comparison of DITAN with 11 state-of-the-art methods on the MBA dataset. P: Precision, R:Recall, AUC: Area under the ROC curve, F1: F1 score.

5.1 The effect of Thresholding to Recall

The formation of critical regions follows the model prediction in a series of steps to detect anomalies across multi/univariate series in an unsupervised manner. In Sections 4, 5 we demonstrated that a major contribution of DITAN is its capability of computing extremely high true alarm rates with less than few false alarms, leading to uninterrupted alerts. We clearly stated both quantitatively in Section 4 (Table 14) and qualitatively in Section 5 (Table 17) that on average more than the first-half temporal duration of a true anomaly is captured. So it is fact, more than hypothesis, that recall is *poor* due to false negatives occurred by sub-capturing true anomalies instead of missing them. We strongly believe, and we also asked experts in applicative domains, that in the real world, capturing a sub-part of a true anomaly is enough to trigger the attention of experts to the corresponding time-steps for further examination.
To make things complete, we ran the 6 available implementations of the aforementioned baselines on the channels of Section 4, considering only the results over the first dimension as originally proposed in [9]. The objective is to discuss false negatives (FN) and false positives (FP) from the aspect of threshold positioning across baseline methods (Figure 13). A conservative methodology is expected to set high thresholds and thus introduce more FN. While in a loose one, thresholds are set lower, introducing more anomalies thus more FP. An optimal threshold positioning results in both low FN and FP values.



False Negatives False Positives

Figure 13: FN and FP across channels per method

MERLIN is unable to identify anomalies on P-4 and E-13 channels, since the number of its FN reaches the number of corresponding true anomalies. Similarly, TranAD, GDN, OmniAnomaly and USAD fails on channels T-13 and C-1 as their FNs reach the number of true anomalies of the channels respectively. MAD-GAN fails only on T-13 channel, resulting as many FN as the total number of true anomalies. This is a strong indicator that their threshold value on

these channels is positioned to a higher error magnitude than what is should be considering the true anomalies. On the other hand, DITAN is the only method that do not fail across any channel, critical regions adapting the position of a threshold to the locality of error magnitudes, as opposed of using a global threshold value.
Furthermore, despite the zero FP of TranAD, GDN, USAD and OmniAnomaly on C-1 channel, we showed that their corresponding FN are extremely high. Similarly on channel T-13, TranAD and GDN result in no FP but at the same time no TP. On these two channels, MAD-GAN results the lowest FP on C-1 and DITAN on T-13. In channel P-4, all the TranAD, MAD-GAN and DITAN are leading with zero FP. In channel E-13, DITAN leads with the less FP (121) and in channel D-14 TranAD with 7 FPs. In other words, DITAN leads on the most (3/5) channels maintaining the lowest FP and at the same time able to identify true anomalies.
In overall, DITAN results in conservative thresholding methodology reaching a FN number no more than half of true anomalies, while maintaining similarly low FPs across channels.

6 Conclusion and Future Work

We proposed in this paper DITAN, a domain agnostic framework for the detection and interpretation of temporal-based anomalies. We first lightened the requirements of contamination in multivariate time series, demonstrating an anomalous exploratory space and addressing the limitations of the current literature on their anomalous and temporal support. Particularly, we qualitatively showed that no individual can support all the requirements of its related work. We then showed in both quantitative and qualitative way that DITAN captures all these requirements. In our unsupervised environment, where no labels are required, the method instantiates a neural network model based on Encoder-Decoder architecture with implicit/explicit attention and adjustable layers/units to capture normality as regular patterns over temporal records. Anomalous records are then detected by introducing a two-stage thresholding methodology that reveals critical regions to their errors sequence. Using detected anomalies, root cause is examined on their data space and similarities are seen in their units space using a clustering method.

DITAN is assessed on the well-known MSL and SMAP real-world datasets. From these, 6 multivariate channels are selected to cover all possible contamination types (point, contextual, joint) at a varying duration. To highlight the generic nature of the proposed framework, optimizer is used to automatically configure the hyper-parameters of a model per channel. The proposed models are capable of predicting normality with high tolerance to overfitting, dominating against to the original ones [9] . Critical regions are well aligned to all anomalous events (IoU > 0.5), from which point anomalous records ($F_{0.5} \approx 0.8$) are captured slightly better than contextual anomalies ($F_{0.5} \approx 0.6$). In total, from the 12 anomalous events only 2 are masked, while the rest are detected with more than 70% precision. Finally, a good clustering (DBI < 0.19) is applied to the units space of the detected anomalous records, with more than 87% of them to be verified similar also in their actual data space. The major limitation due to the unsupervised environment is that poor data quality can corrupt the data modeling phase. That requires a good understanding of the training sequence to allow the framework to learn normality. On the other hand, if the contamination level is too high, the system will try to model those instances, hence, considering them as normal at inference time. Also if the duration of an anomalous event is too large, w.r.t. context size, then it may partially covered or considered normal.

We now plan to extend the framework by integrating an expert knowledge level. We more particularly aim at providing an environment enabling experts to annotate the detected events by "physical reasons" that caused them, in a deductive way using IF-THEN temporal rules. DITAN is trained to predict normality and thus detected anomalies that do not conform. Another interesting extension that we are additionally working on is to train a supervised model using the DITAN labels for both normal and anomalous records, to predict future anomalies before they happen.

References

[1] J. Audibert et al. Do deep neural networks contribute to multivariate time series anomaly detection? *Pattern Recognition*, 132, December 2022.

[2] J. Audibert et al. USAD: unsupervised anomaly detection on multivariate time series. In R. Gupta et al. editors, *KDD '20: The 26th ACM Conf. on Knowledge Discovery and Data Mining, August 23-27*, : 3395–3404, 2020.

[3] L. Bergman and Y. Hoshen. Classification-based anomaly detection for general data. In *8th Int. Conf. on Learning Representations, Addis Ababa, Ethiopia, April 26-30*, 2020.

[4] A. Blázquez-Garcí et al. A review on outlier/anomaly detection in time series data. *ACM Computer Surveys*, 54(3)1–56, 2021.

[5] S. Du et al. Multivariate time series forecasting via attention-based encoder-decoder framework. *Neurocomputing*, 388:269–279, 2020.

[6] Deng, A. and Hooi, B. Graph Neural Network-Based Anomaly Detection in Multivariate Time Series. *35^{th} AAAI Conf. On Artificial Intelligence, Feb. 2-9*. pp. 4027-4035, 2021

[7] A. Geiger et al. Tadgan: Time series anomaly detection using generative adversarial networks. In X. Wu, et al. editors, *IEEE Int. Conf. on Big Data, Big Data 2020, Atlanta, USA, December 10-13, 2020*: 33–43, 2020.

[8] I. Golan and R. El-Yaniv. Deep anomaly detection using geometric transformations. In S. Bengio et al. editors, *Annual Conf. on Neural Information Processing Systems 2018 December 3-8, Montréal*, : 9781–9791, 2018.

[9] K. Hundman et al. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In Y. Guo and F. Farooq, editors, *Proc. of the 24^{th} Int. Conf. on Knowledge Discovery & Data Mining, 2018, London, August 19-23*: 387–395. ACM, 2018.

[10] W. Jiang et al. A gan-based anomaly detection approach for imbalanced industrial time series. *IEEE Access*, 7:143608–143619, 2019.

[11] Y. Jiao et al, D. TimeAutoAD: Autonomous Anomaly Detection With Self-Supervised Contrastive Loss for Multivariate Time Series. *IEEE Transactions on Network Science and Engineering*. **9**, 1604-1619, 2022

[12] F. Khoshnevisan and Z. Fan. RSM-GAN: A convolutional recurrent GAN for anomaly detection in contaminated seasonal multivariate time series. *CoRR*, abs/1911.07104, 2019.

[13] T. Kieu et al. Outlier detection for time series with recurrent autoencoder ensembles. In S. Kraus, editor, *Proc of the 28^{th} Int. Joint Conf. on Artificial Intelligence, Macao, August 10-16*, : 2725–2732, 2019.

[14] C.-Ki Lee et al. Studies on the gan-based anomaly detection methods for the time series data. *IEEE Access*, 9:73201–73215, 2021.

[15] D. Li et al. MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks. In I. V. Tetko et al. editors, *Artificial Neural Networks and Machine Learning, Munich, Germany, September 17-19*, LNCS, 11730: 703–716. Springer, 2019.

[16] Z. Liu et al. A regularized LSTM method for predicting remaining useful life of rolling bearings. *International Journal on Automation and Computing*, 18(4):581–593, 2021.

[17] T. Luong et al. Effective approaches to attention-based neural machine translation. In L. Màrquez et al. editors, *Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing, Lisbon, September 17-21*: 1412–1421, 2015.

[18] S. Maleki et al. Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering. *Applied Software Computing*, 108:107443, 2021.

[19] P. Malhotra et al. Long short term memory networks for anomaly detection in time series. In *23rd ESANN Conf 2015, Bruges, April 22-24*, 2015.

[20] E. Marchi et al. Non-linear prediction with LSTM recurrent neural networks for acoustic novelty detection. In *Int. Joint Conf. on Neural Networks, Killarney, July 12-17*: 1–7, 2015.

[21] M. Munir et al. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, 7:1991–2005, 2019.

[22] T. Nakamura et al, MERLIN: Parameter-Free Discovery of Arbitrary Length Anomalies in Massive Time Series Archives. *IEEE Int. Conf. On Data Mining* . 1190-1195, 2020

[23] Z. Niu et al. Lstm-based VAE-GAN for time-series anomaly detection, *Sensors*, 20:3738, 2020.

[24] G. Pang et al. Deep learning for anomaly detection: A review. *ACM Computing Surveys*, 54:1–38, 2021.

[25] J. Pereira and M. Silveira. Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In M. Arif Wani, et al, editors, *17th IEEE Int. Conf. on Machine Learning and Applications, Orlando, USA, December 17-20*: 1275–1282, 2018.

[26] J. Pereira and M. Silveira. Learning representations from healthcare time series data for unsupervised anomaly detection. In *IEEE Int. Conf. on Big Data and Smart Computing, BigComp 2019, Kyoto, Japan, February 27 - March 2*: 1–7, 2019.

[27] O. Provotar et al. Unsupervised anomaly detection in time series using lstm-based autoencoders. In *IEEE Int. Conf. on Adv Trends in Information Theory*: 513–517, 2019.

[28] Y. Su et al. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In A. Teredesai et al. editors, *Proc. 25^{th} ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining, Anchorage, USA, August 4-8*: 2828–2837, 2019.

[29] Y. Tan et al. An encoder-decoder based approach for anomaly detection with application in additive manufacturing. In M. Arif Wani et al. editors, *18th IEEE Int. Conf. On Machine Learning And Applications, Boca Raton, FL, December 16-19*: 1008–1015, 2019.

[30] L. Tenenboim-Chekina et al. Ensemble of feature chains for anomaly detection. In Z. Zhou et al. editors, *Multiple Classifier Systems, 11th Int. Workshop, Nanjing, China, May 15-17, 2013*, Vol. 7872 of *LNCS*: 295–306. Springer, 2013.

[31] P. Wu et al. Unsupervised anomaly detection for underwater gliders using generative adversarial networks. *Engineering Applications of Artificial Intelligence*, 104:104379, 2021.

[32] R. Wu and E. J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. In *Proc. of IEEE ICDE*, 2022.

[33] C. Yin et al. Anomaly detection based on convolutional recurrent autoencoder for iot time series. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*: 1–11, 2020.

[34] C. Zhang et al. Velc: A new variational autoencoder based model for time series anomaly detection. *CoRR*. abs/1907.01702, 2020

[35] C. Zhang et al. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *The 9th AAAI Symposium on Educational Advances in Artificial Intelligence, USA, January 27 - February 1*: 1409–1416. AAAI Press, 2019.

[36] J. Zhao et al. One-step predictive encoder - gaussian segment model for time series anomaly detection. In *Int. Joint Conf. on Neural Networks, Glasgow, United Kingdom, July 19-24, 2020*: 1–7 2020.

[37] B. Zhou et al. Beatgan: Anomalous rhythm detection using adversarially generated time series. In Sarit Kraus, editor, *Proc of the 28th Int. Joint Conf. on Artificial Intelligence, Macao, China, August 10-16*: 4433–4439, 2019.

[38] S. Tuli., G. Casale, & N. Jennings, TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proc. VLDB Endow.*. **15**, 1201-1214, 2022

[39] B. Zong et al., Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *6th Int. Conf. on Learning Representations, Vancouver, April 30-May 3, 2018*

[40] H. Zhao, Y. Wang., J. Duan, C. Huang, D. Cao., Y. Tong , B. Xu, J. Bai, J. Tong & Q. Zhang, Multivariate Time-series Anomaly Detection via Graph Attention Network. *CoRR*. abs/2009.02040, 2020

[41] Y. Zhang, Y. Chen, J. Wang, Z. & Pan, Unsupervised Deep Anomaly Detection for Multi-Sensor Time-Series Signals. *CoRR*. abs/2107.12626, 2021