Discover the world of Time Series

Gabriele Greco s303435 **Davide Aiello** s303296 **Costantin Clipca** s290214

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Discover the world of Time Series

Abstract—**In the pursuit of detecting anomalies in time series datasets, this project employs four distinct state-of-the-art methods: Hybrid Isolation Forest (HIF), Long Short Term Memory networks for Anomaly Detection (LSTM-AD), Encoder-Decoder for Anomaly Detection (EncDec-AD), and KMeans clustering. The Hybrid Isolation Forest integrates unsupervised and supervised techniques to address limitations in the Isolation Forest (IF) algorithm. LSTM networks, renowned for their capability in learning long-term patterns, are applied to anomaly detection, modeling prediction errors as a multivariate Gaussian distribution. The Encoder-Decoder scheme for Anomaly Detection (EncDec-AD) focuses on unpredictable time-series in mechanical devices, showcasing robust anomaly detection on our dataset. All these methodologies will be evaluated using metrics such as Precision@K, $F1$, and F_β score, AUC-ROC, AUC-PR, PA%K protocol used to compute the area under the curve, and they will further be compared amongst themselves. In the end, we obtained favorable results for EncDec-AD, demonstrating that deep learning algorithms can be a viable approach to detect anomalies, but at the expense of computational resources. It is then followed by isolation tree methods HIF and KMeans with LSTM-AD tied for. Code available at https://github.com/MLinApp-polito/mla-prj-23-fp-01-acg**

I. Introduction

A *data series* is an ordered sequence of data points that describe some object or property on a continuous measure. If the order is based on time, as in our case, the sequence is called a *time series*. The data points of a time series record consist of one variable, in this case, it is called *univariate*, or can be made up of multiple real-valued variables, in which case it is called *multivariate* time series. An *anomaly* in such a time series is a point (*outlier*) or point sequence that deviates from some measure, model, or embedding from the regular pattern of the time series.
We can have different anomaly points, such as:

- An **abnormal time point** which refers to a specific time point in the time series where the monitored variable exhibits a significant difference compared to other time points.
- An **abnormal time interval** which refers to a specific duration within the sequence where the behavior of the variable significantly deviates from the rest of the series.
- An **abnormal time series** which refers to a specific set of data points that form a time series, significantly different from other time series.

In this project, we will focus on dealing with the Anomaly Detection task, where we have to identify data that significantly differs from the ‘normal’ ones (*inlier*). Specifically, we will address multivariate data and abnormal time points.

As explained in [1], there are numerous methods to solve this problem, such as Deep Learning, Stochastic learning, Classic Machine Learning, Outlier Detection, Statistics, Data mining, and Signal Analysis. This class of algorithms can be further subdivided into various methods with different types of algorithm:

- **Forecasting:** these algorithms continuously learn a model to forecast a number of time steps based on a current context window.
- **Reconstruction:** these techniques build a model of normal behavior by encoding subsequences of a normal training time series in a latent space. At test time, the subsequences are reconstructed from the latent space, and the reconstructed subsequences are compared to the original ones.
- **Encoding:** in which algorithms also encode subsequences of a time series in a low-dimensional latent space, but they compute the anomaly score directly from the latent space representations.
- **Distance:** they use specialized distance metrics to compare points of a time series with each other. Anomalous points are expected to have larger distances to other points than points with normal behavior.
- **Distribution:** these algorithms estimate the distribution of the data or fit a distribution model to the data. The distributions are calculated either over data points or subsequences obtained via windowing.
- **Isolation Tree:** as last but not least these algorithms build an ensemble of random trees that partition the samples of the test time series. They recursively select random features and random split values as tree nodes to eventually isolate the samples in the tree leaves.

Due to the vast number of methods and techniques, we have decided to narrow the analysis only to:

- **Classic Machine Learning:** we are going to use K-Means, that belongs to the *distance* family of algorithms.
- **Outlier Detection:** Developing a Hybrid Isolation Forest [2] based on the *isolation tree* methods.
- **Deep Learning:** Expanding the Long Short-Term Memory (LSTM) with two different approaches: standard LSTM [3] for *forecasting* methods and Encoding-Decoding LSTM [4] for *reconstruction* methods.

Anomaly detection algorithms can also be categorized into three main groups based on their training methodology: unsupervised , supervised and semi-supervised.

- In the **unsupervised** context, the goal is to identify anomalous points within a time series without prior knowledge. In this case, no explicit training phase is required, and the algorithm relies on assumptions such as less frequency, distinct shapes or behaviors, or origins from different distributions of anomalous subsequences compared to normal ones.
- In the **supervised** approach, the objective is to model both normal and abnormal behavior in the time series. This requires a training phase where all points in the training time series are labeled as normal or anomalous. The algorithm learns to distinguish between normal and

anomalous behaviors based on the labeled training time series.

- Within the **semi-supervised** framework, the focus is on learning only the normal behavior of a training time series. This involves training on normal time series to build a model of normal behavior. When applied to a test time series, the algorithm labels as anomalous all subsequences that deviate from the learned normal behavior.

This report aims to propose a solution for the task using the mentioned methods and then analyze the results.

Metrics: The evaluation of anomaly detection (AD) tasks is inherently challenging due to the presence of diverse cases and a plethora of metrics. In our specific scenario, we are dealing with non-binary metrics, where the anomaly score predictions consist of real values in the range 0 to 1. Following the methodology proposed in [5], this means assessing the anomaly scores using a single threshold, computing scores at several or all thresholds, and subsequently choosing the optimal score or combining the scores. We are not going to use metrics such as Accuracy since our dataset is imbalanced, Recall or Precision taken alone, due to false positives and false negatives not being penalized respectively. Therefore we will utilize metrics such as Precision at K (*P@K*), *F1 score*, and F_β *score*, where we want to emphasize recall, at a determined threshold. Additionally, we will aim to apply metrics that consider performance across all thresholds, such as Area Under the Receiver Operating Characteristic curve (*AUC_ROC*) and Area Under the Precision-Recall curve (*AUC_PR*) also know as average precision.

Furthermore, a new protocol has been introduced called point adjustment (PA), where the F1 score is computed with adjusted labels. Nevertheless, it is important to notice the underestimation tendency of the F1 score and the overestimation of F1 with PA. Due to these reasons, in alignment with the approach outlined in [6], we will implement point adjustment at K (PA@K). However, it is important to note that this will not be treated as a standalone metric; rather, it will serve as a means to calculate the area under the curve across varying values of the variable K. This comprehensive set of metrics allows for a thorough evaluation of the anomaly detection performance in our specific context

II. Related Work

In this section, we will explore how the original literature work addresses the task of anomaly detection.

K-Means: The anomaly detection task can be approached through classical machine learning techniques like [7], and in this particular case the algorithm for anomaly detection consists of three procedures. The first one is pattern clustering and event extraction for each time series in the data, followed by data mining association rules among the typical patterns or events in more than one time series. In the end, there is real-time monitoring of a spacecraft system and anomaly detection using the acquired association rules. In our work, we don't need the last two procedures but only the first one. The purpose of the pattern clustering procedure is to extract a finite number of representative signal patterns from each continuous time series in the data and label them. Through this clustering labeling, we are going to compute point-to-point anomaly scores.

Hybrid Isolation Forest: Isolation Forest, as presented in [8], is a popular method for anomaly detection tasks due to its capability to process large amounts of data in high-dimensional spaces with computational and spatially controlled efficiency. The anomaly score for the isolation forest is constructed from the analysis of unsuccessful search in a Binary Search Tree (BST). Unfortunately, IF suffers from blind spots, empty portions of the space in which data instances are embedded in an unusual distribution like donuts or a horse shoe shape. HIF [2] fixes this problem by adding two sources of information that will allow designing a more sophisticated anomaly score during evaluation. The first one is an unsupervised extension that exploits distance knowledge to neighboring *'normal'* data, and the second one is a supervised-based extension that exploits distance knowledge to neighboring *'anomaly'* data.

LSTM-AD: LSTM (Long Short-Term Memory) [9] networks are highly effective in understanding sequences characterized by unknown long-term patterns. Fundamental to their success is the implementation of long-term memory mechanisms, which precisely control the flow of information through the use of input, output, and forget gate mechanisms. These mechanisms enable the network to maintain relevant long-term information while forgetting less important details, making LSTMs suitable for a wide range of sequential modeling tasks. Furthermore, through the stacking of multiple LSTM layers [10], the implementation of stacked LSTM networks enables the network to acquire progressively intricate representations of sequences, more naturally capturing the structure of time series and facilitating the processing of time series at different time scales. The stacking of these layers not only enables the network to effectively address long-term patterns but also accelerates the learning process, yielding more efficient representations of temporal sequences. This stacked architecture, with its ability to learn increasingly sophisticated features, proves essential for tackling complex challenges in sequence modeling.

EncDec-AD: Encoder-decoder networks represent a class of deep learning architectures widely employed in various domains, including natural language processing, computer vision, and time series analysis [4]. These networks are particularly adept at tasks involving sequence-to-sequence transformations, where an input sequence is mapped to an output sequence of potentially different lengths. The fundamental idea behind encoder-decoder networks is to leverage two distinct neural network components: an encoder and a decoder. The encoder component processes the input sequence and compresses it into a fixed-length representation or context vector. This context vector encapsulates the salient information extracted from the input sequence and serves as the foundation for subsequent processing by the decoder. In the context of time series data, the encoder analyzes the temporal dependencies within the input sequence and captures its essential features in a condensed form. On the other hand, the decoder component

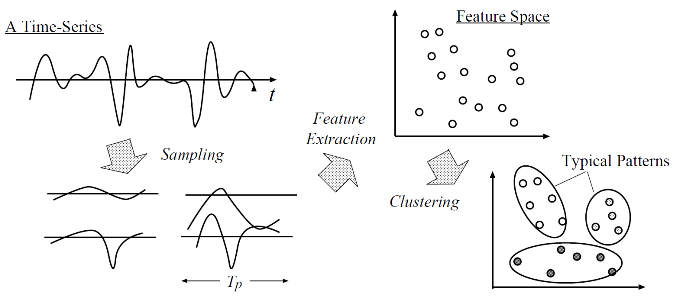

Fig. 1. Pattern clustering procedure for Kmeans.

receives the context vector produced by the encoder and generates an output sequence based on this representation. It unfolds the context vector into a sequence of elements, often with a different length or dimensionality from the input sequence. In tasks such as language translation or sequence prediction, the decoder utilizes the context vector to generate the desired output sequence, leveraging the encoded information to produce meaningful predictions.

III. Methods

In our cases we consider a time series $X = \{x^{(1)}, x^{(2)}, ..., x^{(n)}\}$, where each point $x^{(t)} \epsilon R^m$ in the times series is a m-dimensional vector $\{x_1^{(t)}, x_2^{(t)}, ..., x_m^{(t)}\}$, whose elements correspond to the input variables.

A. *Kmeans*

We are going to extract from X of test a specified number of subsequences with a fixed length (Tp, our window size) from the test data. Each instance originally in the time domain is then transformed into a feature vector space by calculating a set of feature values, as showed in 1. To facilitate this process, we will use the TSFEL library [11], which divides the time series into windows and extracts our features. The different types of features (statistical, temporal, and spectral) are chosen through experiments, as explained in IV-B Afterward, the instances in the feature vector space undergo a normalization process and are clustered using the K-means method. We compute the Euclidean distance between every data point and the cluster to which it belongs. These distances will be used to compute anomaly scores, where greater distances indicate that the point is far away from the cluster and could be considered as an anomaly. In the final step, we'll convert the window-based anomaly score into a point-based anomaly score.

B. *HIF:*

We are adapting the approach described in [2] to our specific case. This forest can also be constructed in a supervised manner by incorporating known anomalies into the external nodes of the forest. Initially, we will construct a forest using non-anomalous data, specifying parameters such as the number of trees t, the size of subsets Ψ used to build each tree, and l_{max}, representing the maximum height of the trees. The forest is obtained by randomly selecting $S_1, S_2, ..., S_t$, t samples from X for all the trees.

For each tree creation, a random feature q from the feature space will be selected and used to compare the nodes of each subset. Following this, a random split value p along dimension q will be chosen between the maximum and minimum values. The value of p will be compared to each element in the subset, and each element will be assigned to the right or left branch based on the result.

After the creation of the forest, anomaly's detection is facilitated through the combination of three scores:

- The first score s for an instance x is defined as
$$s(x,n) = 2^{-\frac{E(h(x))}{c(n))}}, \tag{1}$$
where $E(h(x))$ is the mean of $h(x)$ taken over a collection of isolation trees. Here, $h(x)$ represents the path length, i.e., the number of edges x traverses in a tree from the root node to an external node.
- The second score is an unsupervised extension that introduces a distance-based score equal to
$$s_c(x) = E(\delta(x)) \tag{2}$$
. This is the expectation of the $\delta(X)$ scores evaluated on each tree of the forest.
- The third score is a supervised extension equal to
$$s_a(x) = \frac{E(\delta(x))}{E(\delta_a(x))} = \frac{Mean_{iTree}\delta(x, iTree)}{Mean_{iTree}\delta_a(x, iTree)} \tag{3}$$
, simply being the ratio of the expectation of $\delta(X)$ over the expectation of $\delta_a(X)$. Essentially, it is the ratio of the mean scores of $\delta_a(X)$ over the mean scores of $\delta_a(X)$ evaluated on each tree of the forest.

After normalizing these scores, we can aggregate them to obtain the final HIF score as specified in

$$s_{hif}(x,n) = \alpha_2 \cdot (\alpha_1 \cdot \tilde{s}(x,n) + (1-\alpha_1) \cdot \tilde{s_c}(x)) + (1-\alpha_2) \cdot \tilde{s_a}(x)) \tag{4}$$

Here, α_1 and α_2 are two hyperparameters that control the contribution of the unsupervised and supervised scores, respectively.

C. *LSTM-AD:*

In this methodology, we employ a predictor to model normal behavior and subsequently leverage prediction errors for detecting anomalous patterns within time series data. The dataset is divided into two segments: training and validation. During the training phase, the model is trained on the training dataset, and the validation dataset is used to compute prediction errors, enabling the calculation of mean and variance for a Gaussian distribution.

To ensure that the networks effectively capture the temporal structure of the sequence, we adopt a strategy of predicting several time steps into the future. This implies that each point in the sequence corresponds to multiple predicted values generated at different past instances. Consequently, multiple error values are associated with each point in the sequence.

By analyzing the distribution of errors, we can establish a baseline for normalcy and identify deviations in the test data, facilitating effective anomaly detection.

To quantify these prediction errors, we model error vectors to fit a multivariate Gaussian distribution $N(\mu, \Sigma)$. The probability $p(t)$ of observing a specific error vector $e(t)$ is determined by evaluating the multivariate Gaussian distribution at the point $e(t)$.

The error vectors for the points from the validation dataset are utilized to estimate the parameters μ and Σ through Maximum Likelihood Estimation.

An observation $x(t)$ is categorized as 'anomalous' if $p(t) > \tau$; otherwise, the observation is labeled as 'normal.'

D. *EncDec-AD:*

The methodology for time series anomaly detection using an encoder-decoder network revolves around training the network on normal data to learn the underlying patterns and correlations present in the time series. During testing, the network is tasked with reconstructing incoming data, and any significant deviation between the reconstructed signal and the original input serves as an indicator of anomaly. During the training phase, the encoder-decoder network is exposed exclusively to normal data, allowing it to capture the intricate patterns and correlations inherent in the time series. In this case $X = \{x_1, x_2, ..., x_N\}$ represent the training dataset comprising normal time series samples. The network parameters are iteratively optimized to minimize the reconstruction error, defined as the disparity between the input sequence x_n and its reconstructed counterpart $\hat{x_n}$. Mathematically, the reconstruction error is computed as: $e_i = ||x_i - \hat{x_i}||$. Furthermore, during training, a normal distribution is constructed based on the reconstruction errors of the network. This normal distribution encapsulates the statistical properties of the reconstruction errors and serves as a reference for identifying anomalies during testing. The error vectors for the points in the normal data sequences are used to estimate the parameters and of a normal distribution N, using Maximum Likelihood Estimation. Then, for any point x_i, an anomaly score a_i is calculated as: $a_i = (e_i - \mu)^T \Sigma^{-1} (e_i - \mu)$. In a supervised setting, if $a_1 > \tau$, a point in a sequence can be predicted as anomalous, otherwise normal.

E. *Bayesian:*

During a laboratory session, we developed a Bayesian-based model for anomaly detection tasks. The approach involves an initial step where the confidence of the predictions regarding actions is used to determine the presence of an anomaly within a temporal window. This determination is achieved through the application of a threshold function. In this context, the classification of actions serves as a pretext task, acting as a means to gather information about a different task. This represents a case of self-supervised learning, where the label is obtained through classification, and subsequently, confidence can be derived. Emphasizing the critical nature of this model, it calculates uncertainty over a temporal interval linked to the entire action. Anomalies are identified through a comprehensive evaluation of the complete action within the specified temporal window, marking all points within that interval as anomalous. We choose not to compare this model with other models because, unlike others assigning scores to individual temporal points, this approach involves assigning scores to entire actions, providing a distinctive perspective on anomaly detection.

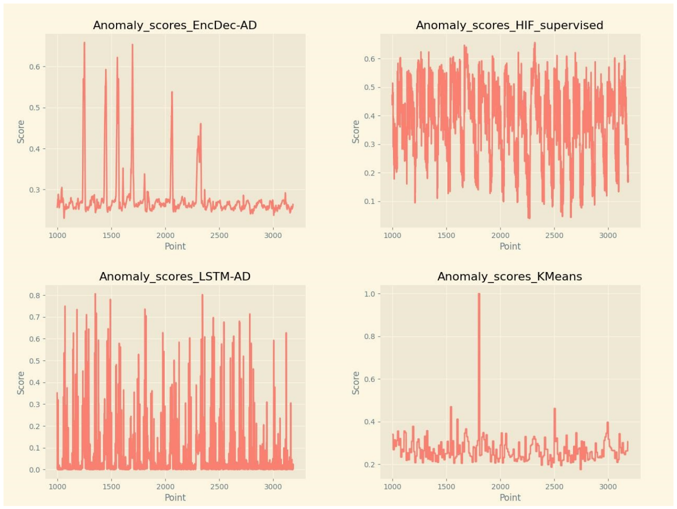

Fig. 2. In the figure are plotted the scores for each models. On the x axis we have the datapoint and on the y axis the score.

IV. Experiments

In this section we present our experiment organization, we will showcase all the setups for our models, which were studied with various parameters. Although we conducted numerous tests to identify the optimal hyper parameters, our primary emphasis will be on comparing the models across different metrics rather than focusing solely on the optimization of each individual model for the task.

A. *Dataset*

Our data consists of a time series dataset collected during several actions performed by a Kuka Robot. We have different recording sessions, with IDs **0**, **2**, **3**, and **4** containing non-anomalous data that can be used for training. Sessions with IDs **1** and **5** contain anomalous data, where the robot encounters anomalies suitable for evaluation and testing. The sensed signals are collected at different sampling frequencies $(1, 10, 100, 200)$ Hz; we are going to test with 1Hz, 10Hz and 100Hz. Increasing the frequency results in larger datasets and more anomalies $(175, 1736, 17297, 34594)$.

B. *KMeans*

Even though KMeans clustering is a straightforward method, it needs correct parameters to work properly. Firstly, despite having the option to use raw data, we will utilize the TSFEL library to perform window sampling and feature extraction. Among spectral, temporal, and statistical features, we decided to use statistical features as they have shown better results. After that, before clustering the data, we perform a little preprocessing on test data by normalizing our data with a $StandardScaler$ and removing zero-variance features. We observe slightly less favorable results when removing highly correlated features. Therefore, we have decided not to do so in order to preserve information and enhance robustness in the clustering process. The only hyper parameters in this case are $n\_clusters$ and $window\_size$. Increasing the number of clusters leads to overfitting, while decreasing it results in fewer clusters and all points being grouped together. We tested several values for $n\_clusters$ and $window\_size$ at 1, 10, 100 Hz frequencies, and we found the best models for $n\_clusters = 7, 15, 20$ and $window\_size = 10, 10, 100$ in that order.

C. HIF

We tested the model in both unsupervised and supervised manners. To standardize the results due to the randomness of the algorithm, we set a seed equal to 42. Both train and test datasets were processed using the same assets: applying of *MinMaxScaler* to normalize the data, use of *VarianceThreshold* to remove low variance features and after that the removal of highly correlated features. The forest was created with a number of trees (n_trees) and the number of samples ($n\_samples$) for each tree. We utilized the same hyperparameters as [2] for the aggregate functions, with $\alpha_1 = 0.3$ and $\alpha_2 = 0.7$, assigning appropriate weights to each score. In the unsupervised model, we set $\alpha_2 = 1$ since anomalies are not present in the nodes. For the supervised approach, we introduced 2% of the test dataset into the forest, corresponding to approximately 4% of the total anomalies (7,71,711) for each frequency (1, 10, 100 Hz).

The results in Tables II, III, and I illustrate the performance of both the unsupervised and supervised forests. While achieving similar results for the F_β score, the supervised approach demonstrates superior performance in other metrics, and given that, we have decided to use this for the comparison with other techniques.

Experimentation with different values of n_trees and $n\_samples$ was conducted to determine the optimal combination for frequencies 1, 10, and 100 Hz, resulting in n_trees (512, 512, 1024) and $n\_samples$ (128, 256, 384).

D. LSTM-AD

In this section, we'll delve into the hyper parameters employed during model training and walk through the preprocessing steps applied.

Initially, we apply *MinMaxScaler* to normalize the data using the min-max scaling technique. Subsequently, we utilize *VarianceThreshold* to selectively remove features with low variance, setting a threshold at 0.001. This process trims down the feature count from 55 to 51.

Moving on to the architecture of our model, it's meticulously configured as a LSTM featuring *2 layers*. The temporal dimension is captured using a *window size* of 10, and the model projects into the future with a *prediction length* of 5. The training unfolds with a *batch size* of 64 for 3 *epochs*, with the learning process guided by a *learning rate* of 0.001.

As already mentioned, the dataset is partitioned into training and validation sets, with 90% of the data allocated for training and an additional 10% utilized as a validation set. Given the lack of noticeable improvements, the linear layer at the beginning of the architecture was not considered. Therefore, following the reference to the original paper, d corresponds to the number of input features, thus 51.

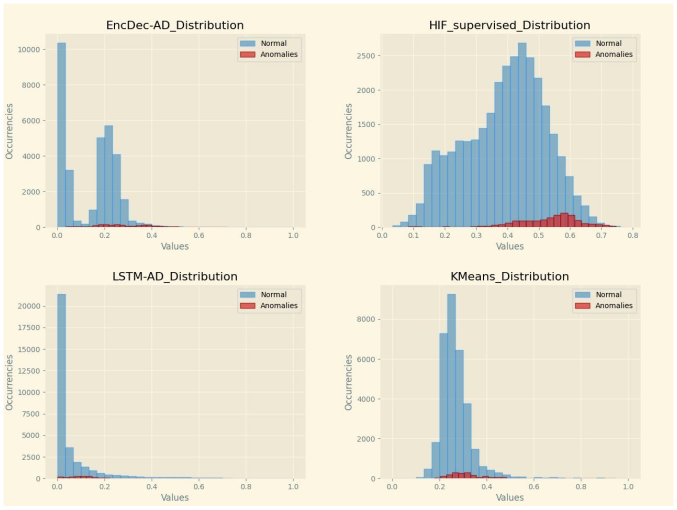

Fig. 3. Diverse distributions for each models. The blue area is the non-anomalous data point and the red area is the anomalous data predicted.

In an effort to closely mimic the behavior outlined in the original paper, our training process aims to predict each of the d dimensions of a sample $x^{(t)}$ l times. This leads to the generation of an error vector with $l \times d$ elements for each sample.

Practically, during the error computation (whether estimating the distribution or evaluating on the test dataset), predictions are made for every possible window associated with each sample. If l is set to 5, for instance, it means there will be 5 windows, each generating a sequence of length 5.

The concept involves iterating through windows, with each window predicting the subsequent sequence.

For each predicted sequence, the relevant sample is extracted. This process is repeated for subsequent sequences, resulting in l predictions for each sample, each at different moments. These predictions are then used to calculate the error with respect to the actual value of the $x^{(t)}$ sample.

This procedure repeats for each index.

E. EncDec-AD

In this section, we present the experiments conducted to evaluate the performance of the LSTM encoder-decoder network for time series anomaly detection. The experimental setup encompasses dataset splitting and preprocessing, as detailed in section LSTM, followed by an exploration of parameter optimization to enhance network performance. To optimize the performance of the LSTM encoder-decoder network, some experiments were conducted to assess the impact of varying key parameters on anomaly detection accuracy. Three primary parameters were targeted for optimization:

- **Latent size:** The latent size refers to the dimensionality of the latent space representation learned by the

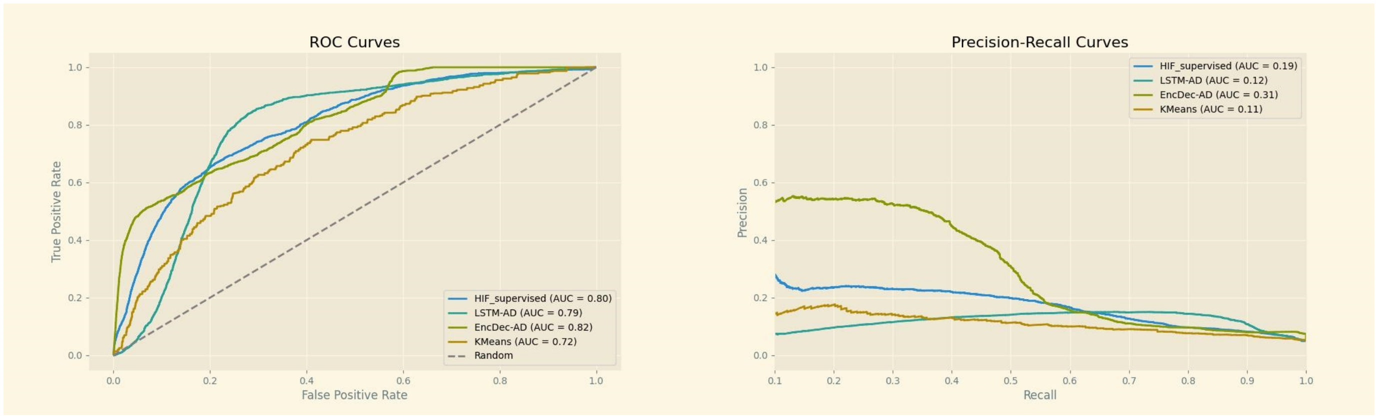

Fig. 4. Plot of AUC-ROC and AUC-PR for each model with f=10Hz

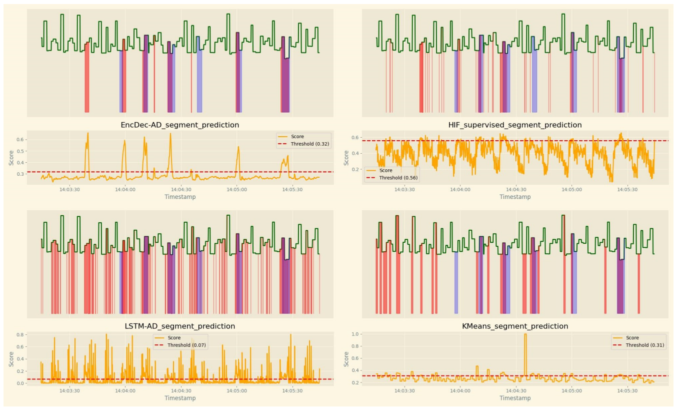

Fig. 5. Predictions for each models made on a sub-segment of test datasets with f=10Hz. The red line are the anomaly points detected by the model, meanwhile the blue area are the true anomalies.

LSTM encoder and decoder. By systematically varying the latent size, ranging from lower to higher dimensions, the network's ability to capture and represent underlying patterns in the data was evaluated.

- **LSTM layers:** The number of LSTM layers in the encoder and decoder modules influences the network's capacity to model temporal dependencies and capture complex patterns within the time series data. Experiments were conducted by varying the number of LSTM layers to ascertain the optimal architecture for anomaly detection performance.
- **Anomaly window size:** The anomaly window size defines the temporal context considered by the network for anomaly detection. By adjusting the size of the sliding window used to extract subsequences from the input time series, the sensitivity of the model to anomalous patterns of different durations was investigated.

For each parameter configuration, the LSTM encoder-decoder network was trained on the training dataset and evaluated on the validation set to monitor performance metrics such as reconstruction error and anomaly detection accuracy. Hyperparameters such as learning rate, batch size, and optimization algorithms are the same as in the LSTM section and were kept constant across experiments to isolate the impact of parameter variations. The overall network performance was evaluated considering the best performing parameters of respectively $(80, 4, 5)$.

F. Scores

The first comparison we want to make is to observe how the scores behave on a plot 2. EncDec-AD presents a few spikes with a steady score distributed evenly, similar to KMeans. Meanwhile, LSTM-AD has numerous score spikes, with a densely concentrated area between 0 and 0.1. HIF is the most peculiar; even though the spikes are not as visible or tangible compared to the others, its distribution is more uniform suggesting that the threshold choice in this model will be crucial.

G. Models Distribution

After examining the scores, let's now explore their distribution in 3, which should be an immediate representation of the already mentioned scores. EncDec-AD exhibits two main columns where the normal scores are distributed, with anomalies under the second column. HIF displays a large Gaussian bell distribution, with anomalies slightly to the right in higher scores, but still within the normal data area. LSTM shows a prominent spike of normal data close to zero, and finally, KMeans has a similar distribution between anomalies and normal data. In all cases we don't observe a clear separation between anomalies and normal data.

H. Prediction segment

After examining the distribution, let's see how the models predict anomalies just for . In 5, all the techniques attempt to predict a predetermined segment of 1500 samples gathered on the first feature. The areas shown in blue represent the true anomalies, while the red stripes indicate the anomalies predicted by the models. These predictions are made using the best threshold for each model. We can immediately observe the performance of EncDec-AD; it has no problem detecting almost all the areas of the anomalies with only one mistake. On the other hand, the other three models can detect most of the anomalies too, but they exhibit too many false positive

	Metrics f=100Hz				
Models	F_1	F_β	AUC_{ROC}	AUC_{PR}	th
$KMeans$	0.223	0.848	0.755	0.131	0.24
$HIF_{supervised}$	**0.299**	0.853	**0.800**	**0.178**	0.52
$HIF_{unsupervised}$	0.186	**0.860**	0.749	0.107	0.57

TABLE I
METRICS FOR EACH MODEL WITH WITH CORRESPONDING THRESHOLD WITH F=100HZ

	Metrics f=1Hz				
Models	F_1	F_β	AUC_{ROC}	AUC_{PR}	th
$LSTM-AD$	0.125	0.845	0.566	0.079	0.26
$KMeans$	0.144	0.857	0.628	0.080	0.47
$EncDec-AD$	0.153	**0.899**	0.641	0.076	0.15
$HIF_{supervised}$	**0.168**	0.846	**0.677**	**0.112**	0.62
$HIF_{unsupervised}$	0.136	0.852	0.648	0.075	0.65

TABLE II
METRICS FOR EACH MODEL WITH WITH CORRESPONDING THRESHOLD WITH F=1HZ

	Metrics f=10Hz				
Models	F_1	F_β	AUC_{ROC}	AUC_{PR}	th
$LSTM-AD$	0.252	0.846	0.791	0.116	0.07
$KMeans$	0.194	0.847	0.717	0.114	0.31
$EncDec-AD$	**0.427**	**0.887**	**0.816**	**0.306**	0.32
$HIF_{supervised}$	0.286	0.857	0.800	0.188	0.56
$HIF_{unsupervised}$	0.188	0.861	0.748	0.106	0.63

TABLE III
METRICS FOR EACH MODEL WITH WITH CORRESPONDING THRESHOLD WITH F=10HZ

stripes within the plot, especially LSTM-AD, indicating a lack of distinguishing power between classes of this feature.

I. AUC-ROC and AUC-PR curves

Let's now analyze the area under the curve graphs for the ROC and PR. In 4, for the ROC curve, we observe that EncDec-AD, HIF, and LSTM-AD are closely positioned, with EncDec-AD at the top, showing a good trade-off between recall and false positives. Meanwhile, KMeans struggles a bit compared to the others. Simultaneously, in the PR curve, we can see that EncDec-AD is the only one achieving a good trade-off between precision and recall compared to the other models. We must acknowledge that, in this particular case, even though AUC_{ROC} is applicable, it may be penalizing due to the presence of an unbalanced dataset where the negative class dominates. Additionally, the integration of AUC_{ROC} over all thresholds can result in a large portion of the score coming from thresholds that may not be relevant for a specific use case. Therefore AUC_{PR} also integrate over all thresholds, but it's more informative than ROC for unbalanced datasets. Therefore, we shall continue exploring other metrics with respect to the anomaly class, such as the F1 and F_β which we discuss in IV-J.

J. $F1$ and F_β scores

Let's now examine the values of F1 and F_β scores reported in table III, II and I. For F=10Hz EncDec-AD has the highest scores, indicating a better balance between precision and recall. Meanwhile, others fall further, especially KMeans with the lower scores. As expected, we obtain good results for all the models with $\beta = 0.1$ emphasizing recall, with a peak on EncDec-AD with $F_\beta = 0.887$ and this is essential in scenarios where missing positives instances is costly. Meanwhile, with f=1Hz EncDec-AD still performs good with F_β, but has similar results in other metrics with all the models. We are not going to analyze values with f=100Hz due to missing results of EncDec-AD and LSTM-AD as explained in IV-N.

K. AUC for PA@K

As we mentioned, we are not going to evaluate point adjustment metrics, but we will examine in Figure 6 the area under the curve for each model at varying values of K with their best thresholds. If K = 0, it is equal to $F1_{PA}$ and if K = 100, it is equal to the normal F1 score evaluated before at IV-J. EncDec-AD performs the best again, followed by a difference of almost a tenth with HIF and a twentieth with LSTM. The results are consistent with other models, with EncDec-AD at the top.

L. Precision@K

Furthermore, when analyzing the P@K metric, we can observe how effectively each model classifies the top scores. This is particularly useful when models exhibit different score distributions (as shown in Figure 3). Ideally, a good classifier should assign higher scores to anomalies and lower scores to normal data, but this may not always be true. In Figure 7, we can see the precision values when choosing the top predictions, in this case selected with a threshold on the percentage of anomalies. In the first 20%, LSTM-AD, KMeans, and HIF unsupervised show similar performance, only being outscored by the usual EncDec-AD and HIF supervised, which significantly improve thier precision after reaching 30% of top anomalies predictions.

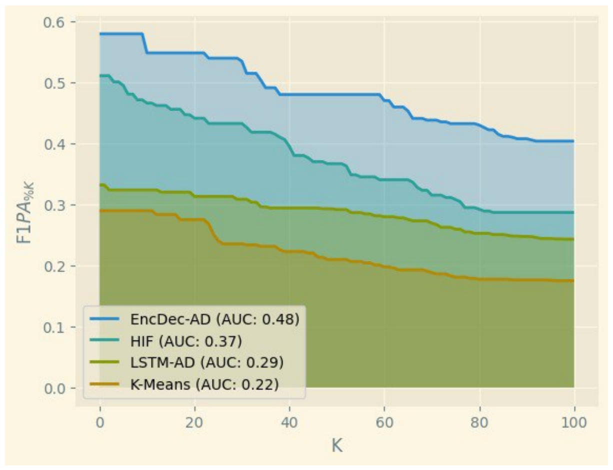

Fig. 6. Area under the curve for PA@K values for all the values of K threshold with f=10Hz.

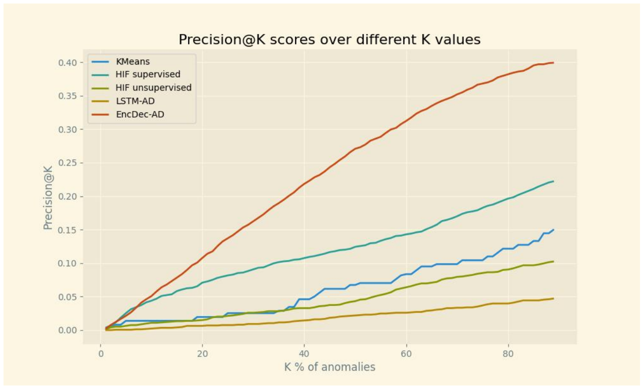

Fig. 7. The figure shows the precision at the top-K values for each model with f=10Hz.

M. *Frequency*

All the tests above were conducted with a dataset at a frequency of 10Hz. Upon analyzing different frequencies, for instance, as shown in II at a frequency of 1Hz, it becomes apparent that EncDec-AD is not the optimal model. This underscores the importance of consistent training data for the EncDec-AD reconstruction method. Unfortunately, we were unable to run Deep Learning methods such as EncDec-AD and LSTM-AD with frequencies higher than 10Hz due to the GPU's RAM limitations.

N. *Computational Resources*

Considering computation time, with f = 10 Hz, each model requires a few minutes of execution: 5 minutes for KMeans, 5 minutes for LSTM-AD, 30 minutes for HIF, and 40 minutes for EncDec-AD. Due to the large amount of data points at a frequency of 100 Hz, the execution time increases to 50 minutes for KMeans and 6 hours for HIF. However, as mentioned in IV-M, EncDec-AD and LSTM-AD demand significantly more computational power from GPUs RAM.

V. Conclusions

Reflecting on our analysis, we have thoroughly examined four state-of-the-art methods for anomaly detection, carefully uncovering their respective weaknesses and strengths. Furthermore, the consensus among various metrics leads to the unanimous recognition of a single model as the best among those studied. Deep learning techniques, such as EncDec-AD, are preferable for this task, but they are complex and require a significant amount of computation resources for training, along with a substantial amount of training data. Simple methods like HIF and Kmeans are the ones to go for, especially with limited time and resources available and they achieve similar results to LSTM-AD. In the end, we indeed acknowledge the difficulty and importance of the anomaly detection task, which still requires more studies. The path forward seems to be hybrid anomaly detection that can combine the strengths of each algorithm.

References

[1] P. T. SCHMIDL Sebastian, WENIG Phillip, "Anomaly detection in time series: a comprehensive evaluation," *Proceedings of the VLDB Endowment*, pp. 15.9: 1779–1797, 2022.

[2] P.-F. M. S. Soheily-Khah and N. Béchet, "Hybrid isolation forest - application to intrusion detection," *arXiv: 1705 . 03800 [cs]. Retrieved 11/05/2020 from http://arxiv.org/abs/1705.03800*, 2017.

[3] P. M. L. V. G. S. P. Agarwa., "Long short term memory networks for anomaly detection in time series," *In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2015.

[4] P. M. A. R. G. A. L. V. P. A. G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *Retrieved 07/23/2020 from http://arxiv.org/abs/1607.00148*, pp. arXiv: 1607.00 148 [cs, stat], 2016.

[5] M. R. Sondre Sørbø, "Navigating the metric maze: A taxonomy of evaluation metrics for anomaly detection in time series," *Retrieved from https://arxiv.org/abs/2303.01272*, p. arXiv:2303.01272 [cs], 2023.

[6] S. K. K. C. H.-S. C. B. L. S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," *In: Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7194–7201, 2022.

[7] K. H. Takehisa Yairi, Yoshikiyo Kato, "Fault detection by mining association rules from house-keeping data," *In Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (-SAIRAS)*, 2001.

[8] Z. H. Z. F. T. Liu, K. M. Ting, "Isolation forest," *In Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*, p. 413–422, 2008.

[9] J. S. S. Hochreiter, "Long short-term memory," *Neural computation*, 1997.

[10] B. S. M. Hermans, "Training and analysing deep recurrent neural networks," *Advances in Neural Information Processing Systems*, 2013.

[11] T. S. H. G. Marília Barandas Duarte Folgado Letícia Fernandes Sara Santos Mariana Abreu Patrícia Bota Hui Liu, "Tsfel: Time series feature extraction library." 2020.