Machine Learning in Applications Time-series Anomaly Detection SOTA Project Report

Youness Bouchari, Federico Bussolino, Marco D'Almo, Haochen He
Politecnico di Torino

Abstract—**Nowadays as systems become increasingly automated, the potential for significant societal impacts due to system failures has grown, making the detection of anomalies within these systems essential for preventing disruptions. The field of study known as Anomaly Detection (AD) achieves results through the use of neural networks in machine learning. In this project we are going to propose different model of neural network such as TadGan, MTGFlow, MSCRED and VAE-LSTM to evaluate the current landscape of SOTA models for this task.**

The goal of the project is evaluating the results and to compare the performance achieved by these model using different statistical criteria; in our tests we concluded that the newest approaches, in particular Gan-based and LSTM based ones seem to perform the best on complex multimodal time series anomaly detection tasks, with some of them being more robust and others being more performant with threshold selection.

I. Introduction

In the context of industrial robots, being able to detect anomalies can be crucial to prevent critical failures, improve the industrial output and avoid accidents in factories. These robots usually have a vast array of sensors, which are able to feed data over time, making them a very specific kind of time-series generating objects. The analysis of these time series seems to be the best ideas to spot anomalies in them, which could represent an array of malfunctions.

In our case we have had the chance to analyze data produced by Kuka, an industrial robotic arm, which has use cases in a vast array of possible industries. The scope of our analysis is testing various deep-learning based approaches to solve the anomaly detection over time-series data problem. With anomaly detection we mean the process of detecting a single reading or a series of reading from the data which shows unusual or unexpected behavior, and which could trigger some automatic solutions, or just show in a record that some sort of unexpected behavior arose. Understanding which models, techniques and architectures are the most effective detecting these anomalies would help obtaining a more reliable system to solve and record anomalies.

In the TadGAN (Time-Series Anomaly Detection using Generative Adversarial Networks) approach, we train the auto-encoder and discriminator using normal dataset where the auto-encoder is used for generating data in the latent space that mimics the normal time-series data, and the discriminator is used for distinguishing between the real time-series data and the synthetic data generated by the generator that enhance the model's ability to detect anomaly behaviors. During the reconstruction phase we pass the segment of the signal (same as the window) to the encoder and transform it into its latent representation, which then gets passed into the generator for reconstruction and mean while we can calculate also critic score using the discriminator,we call the output of this process the reconstructed signal, if the window is abnormal the difference of windows should be high because the trained model is good only to generate the normal window and we will use this difference called reconstruction error combining with critic score to determine the anomalies.

In the MTGFLOW approach [1] we use a Graph Neural Network (GNN) that aims to model the evolving relationship between entities taking into account both time relationship, using an LSTM, and sensors pairwise relationships obtained with self-attention [2]. Then a MAF (Masked Autoregressive Flow) [3] takes as input original window of records and spatio-temporal information (outputs of GNN) to map the probability distribution of entity values to a set of normal distributions (one for each entity), from which we can calculate through a Normalizing Flow the (log-)probability of a certain window to appear in the dataset.

In MSCRED model [4], Raw time-series data are converted into signature matrices that capture correlations at various temporal resolutions. These matrices are processed through a convolutional encoder-decoder network and a ConvLSTM layer, which identifies temporal dependencies. The encoder compresses the data into a latent space, and the decoder reconstructs it. Anomalies are detected by measuring reconstruction errors, which are higher for abnormal patterns the model hasn't seen during training.].

The VAE-LSTM model is built to allow the modeling of complex data distributions and the generation of new, plausible data samples. This model employs a novel LSTM-based Variational Autoencoder, which leverages the temporal modeling strength of Long Short-Term Memory (LSTM) networks to capture the dependencies within sequential data.

In the following sections, we will provide an overview of the architectures of these models, explore their relevance for anomaly detection and we will detail the methodology used in our experiments. The results and analysis will be discussed, along with a comprehensive assessment and comparison of the performance of these approaches. At end, we will conclude with a discussion of the implications and potential future directions for improving anomaly detection in industrial robot systems.

II. Background

Over the past several years, the rich diversity of anomaly types, data types, and application scenarios has driven the development of various anomaly detection methods [5] [6] [7] [8]. In this section, we explore some of the unsupervised approaches. The most basic of these are out-of-limit methods, which identify regions where values surpass a specified threshold [9] [10]. Although these methods are straightforward and easy to understand, they lack flexibility and are unable to detect contextual anomalies. To address this limitation, more sophisticated techniques have been introduced, including proximity-based, prediction-based, reconstruction-based and distribution based anomaly detection.

1) Reconstruction based approaches: reconstruction-based methods use models that attempt to reconstruct the input data. Anomalies are identified when the reconstruction error is high. With this approach several studies are conducted using PCA [11] or an encoder-decoder network to reduce then reconstruct the data like MSCRED [4]. Also some researches are done on Generative Adversarial Networks (GANs) like Variational Auto-Encoder(VAE) [12] and TadGan [13], GANs can be used to generate realistic time series data. The discrepancy between the generated data and the actual data can be used to detect anomalies.

2) Distribution based approaches: Relies on generative model that can model explicitly the probability density of normal samples. MTGFlow [14] is an example of distribution based approach.

III. Materials and methods

The data used in this set of experiments has been divided to make the unsupervised approach easier, with a clear distinction between data without anomalies (or ”normal” behaviour) and data with anomalies (or ”collisions”) and their timestamps. The code was developed as a mono-repo, in order to leverage common code fragments and streamline different experiments with the various architecture. We chose the Tensorflow Python library, to leverage a well known and approachable deep learning library, and we kept track of our experiments with Weights and Biases, an experiment tracking framework which helped us compare results and differences between the tasks. The computing resources to perform the experiments where drawn from a mixture of sources, as Google Colab and our own machines. The explorative nature of the research demanded the use of various experiments setups, to increase the number of tests and records.

A. TADGAN

1) Architecture: Generative Adversarial Networks (GANs) are a class of machine learning frameworks designed to generate new data samples that resemble a given training dataset. GANs consist of two main components: a generator and a discriminator. In the TadGan(Time-Series Anomaly Detection using Generative Adversarial Networks) the model specific designed by MIT team [13] for time series anomaly detection consists in two mapping function :

$$\xi : X- > Z \tag{1}$$

$$\Gamma : Z- > X \tag{2}$$

The X represents the input data domain, and Z represents the latent domain where random vectors z are sampled to represent white noise.

With the mapping functions, we can reconstruct the input time series:

$$x_i \rightarrow \xi(x_i) \rightarrow \Gamma(\xi(x_i)) \approx \hat{x_i}$$

Furthermore, in this version of GAN, there are also two discriminators called adversarial critics C_x and C_z. The goal of C_x is to distinguish between the real time series sequences from X and the generated time series sequences from $G(z)$, while C_z measures the performance of the mapping into the latent space. (Fig.1)

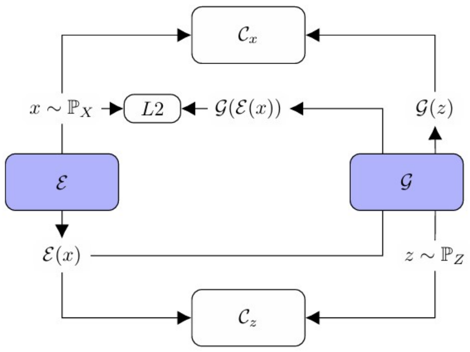

Fig. 1. TadGan model architecure

2) Wasserstein loss: The original formulation of GAN, using standard adversarial losses (Eq 1), is known to suffer from the mode collapse problem

$$L = E_{\mathbf{x}\sim p_x}\left[\log C_f(\mathbf{x})\right] + E_{\mathbf{z}\sim p_z}\left[\log(1 - C_f(G_f(\mathbf{z})))\right] \tag{1}$$

In this context, C_f produces a probability score between 0 and 1, indicating the authenticity of the input time series sequence. Specifically, the *Generator* tends to capture only a small fraction of the data's variability, preventing it from perfectly converging to the target distribution. This behavior arises because the *Generator* prioritizes generating samples that have previously successfully fooled the *Critic*, rather than exploring new samples that could potentially represent other 'modes' within the data.

To overcome this limitation, in the Tadgan model is applied the Wasserstein loss [15] as the adversarial loss to train the GAN. We make use of the Wasserstein-1 distance when training the *Critic* network. For Critic C_f, we have the following objective:

$$\min_{\mathcal{G}} \max_{C_f \in \mathcal{C}_f} V_1(C_f, \mathcal{G}) \tag{3}$$

with

$$V_1(C_f, \mathcal{G}) = E_{\mathbf{x}\sim p_x}[C_f(\mathbf{x})] - E_{\mathbf{z}\sim p_z}[C_f(\mathcal{G}(\mathbf{z}))] \tag{4}$$

For Critic C_z, The objective is expressed as:

$$\min_{\mathcal{E}} \max_{C_z \in \mathcal{C}_z} V_2(C_z, \mathcal{E}) \tag{5}$$

The purpose of the second Critic C_z is to distinguish between random latent samples $\mathbf{z} \sim P_z$ and encoded samples $\mathcal{E}(\mathbf{x})$ with $x \sim P_X$.

3) Cycle Consistency Loss: The aim of TadGAN is to reconstruct the input time series : $\hat{\mathbf{x}} = \mathcal{G}(\mathcal{E}(\mathbf{x})) \approx \mathbf{x}$. But, training the TadGAN with adversarial losses for Critc X and Critc Z alone is not enough to guarantee mapping individual input $\mathbf{x}$ to a desired output $\hat{\mathbf{x}}$ which will be further mapped back to $\mathbf{x}$. To reduce the possible mapping function search space, in TadGan is adopted also the cycle consistency loss to time series reconstruction, which was first introduced by Zhu et al. [16]for image translation tasks. In this case the generative network $\mathcal{E}$ and $\mathcal{G}$ is trained with the adapted cycle consistency loss by minimizing the L2 norm of the difference between the original and the reconstructed samples:

$$V_{\text{L2}}(\mathcal{E}, \mathcal{G}) = E_{\mathbf{x} \sim P_X}[||\mathbf{x} - \mathcal{G}(\mathcal{E}(\mathbf{x}||_2)] \tag{6}$$

4) Full Objective: Combining all of the objectives given in (4), (6) and (7) leads to the final MinMax problem:

$$\min_{\mathcal{E},\mathcal{G}} \max_{C_f \in \mathcal{C}_f, C_z \in \mathcal{C}_z} V_1(C_f, \mathcal{G}) + V_2(C_z, \mathcal{E}) + V_{\text{L2}}(\mathcal{E}, \mathcal{G}) \tag{7}$$

B. VAELSTM

1) Architecture: The Long Short-Term Memory-based Variational Autoencoder (LSTM-VAE) leverages the temporal dependencies inherent in time-series data by integrating LSTM networks into the VAE framework. Instead of using a feed-forward network as in traditional VAEs, LSTMs are employed. The figure illustrates the unrolled architecture, featuring LSTM-based encoder and decoder modules.

2) Reconstruction Loss: To improve the robustness of the LSTM-VAE model, we apply a denoising autoencoding criterion. This is achieved by introducing Gaussian noise to the input, resulting in a corrupted input $\tilde{x} = x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_{\text{noise}})$. The goal is for the model to learn to reconstruct the original, uncorrupted input from this noisy version.

$$\begin{aligned}\mathcal{L}_{\text{dvae}} = -D_{\text{KL}}\left(\tilde{q}_\phi(z_t \mid \tilde{x}_t) \parallel p_\theta(z_t)\right) + \\ E_{\tilde{q}_\phi(z_t|\tilde{x}_t)}\left[\log p_\theta(\tilde{x}_t \mid z_t)\right]\end{aligned} \tag{8}$$

Here, $D_{\text{KL}}(\cdot)$ represents the Kullback-Leibler divergence, which measures the difference between the approximate posterior $\tilde{q}_\phi(z_t \mid \tilde{x}_t)$ and the prior $p_\theta(z_t)$. The expectation term $E_{\tilde{q}_\phi(z_t|\tilde{x}_t)}\left[\log p_\theta(\tilde{x}_t \mid z_t)\right]$ represents the reconstruction loss, where the decoder attempts to reconstruct the original input x_t from the latent representation z_t, even though it was derived from the corrupted input $\tilde{x}_t$.

3) Anomaly Detection: The original paper introduces an online anomaly detection framework for multimodal sensory signals using state-based thresholding. An anomalous event is detected when the current anomaly score of an observation x_t exceeds a predefined score threshold η. where $f_s(x_t, \phi, \theta)$ is the anomaly score estimator. The anomaly score is defined as the negative log-likelihood of an observation with respect to the reconstructed distribution of that observation through an encoding-decoding model:

$$f_s(x_t, \phi, \theta) = -\log p(x_t; \mu_{x_t}, \Sigma_{x_t}),$$

where μ_{x_t} and Σ_{x_t} are the mean and covariance of the reconstructed distribution, $\mathcal{N}(\mu_{x_t}, \Sigma_{x_t})$, from an LSTM-VAE with parameters ϕ and θ. A high anomaly score indicates that the input has not been well reconstructed by the LSTM-VAE, suggesting that the input has significantly deviated from the non-anomalous training data. We didn't implement this part because it is an additional component not completely linked to the architecture, but it is an interesting possible addition.

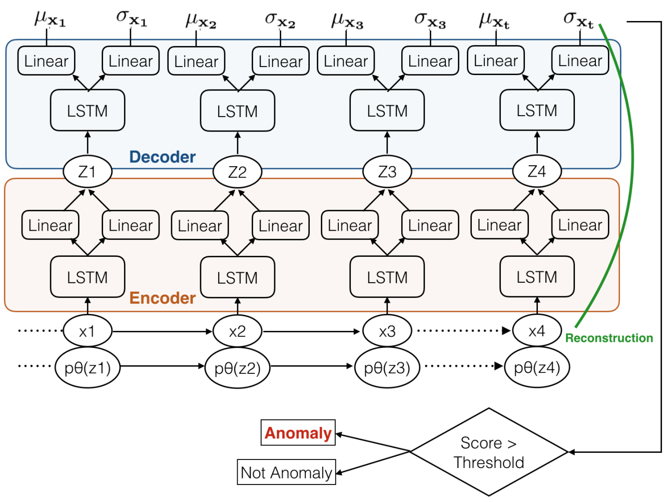

Fig. 2. VAE-LSTM architecture and data flow

C. MTGFLOW

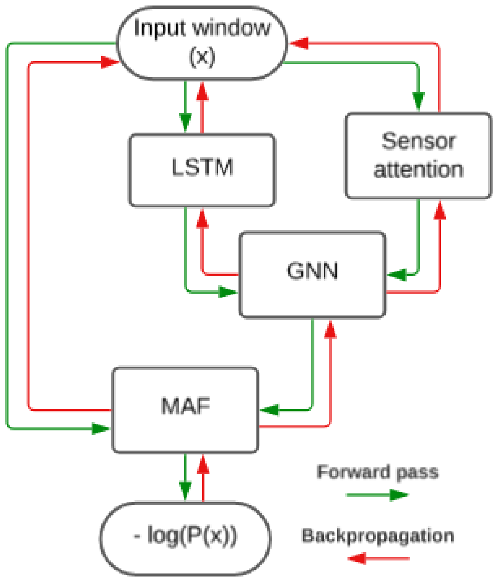

Fig. 3. MTGFlow architecture and data flow

1) LSTM: It is used to extract temporal information $\mathbf{H}$ about the sequence of input data (of length T). It has a number of hidden state equal to the number of entities (K), which, in our case, is an hyperparameter.

2) Sensor-wise Self-attention: Self-attention [2] module extract in a time-agnostic way the pairwise relationship graph between sensors producing attention weights $\mathbf{A}$ as follows:

$$a_{ij} = \frac{e_{ij}}{\sum_{j=1}^{K} e_{ij}} \tag{9}$$

$$e_{ij} = \frac{(x_i \mathbf{W^Q})(x_j \mathbf{W^K})^T}{\sqrt{T}} \tag{10}$$

where $\mathbf{W^Q} \in R^{T\times T}$ and $\mathbf{W^K} \in R^{T\times T}$.

3) Graph Neural Network: One way to join different sensors inter-relationship graph along with temporal information is the graph convolution $\mathbf{AH_t}$ weighted by $\mathbf{W_1}$ summed to history information $\mathbf{H_{t-1}}$ weighted by $\mathbf{W_2}$, finally a ReLU and $\mathbf{W_3}$ are used to enable the model to learn more complex relationships:

$$\mathbf{C_t} = ReLU(\mathbf{AH_tW_1} + \mathbf{H_{t-1}W_2})\mathbf{W_3} \tag{11}$$

4) MAF: It is used to model complex probability distribution as an invertible transformation of a base density (e.g. a standard Gaussian). This is achived by the property of normalizing flows (NF) that allow the area under probability density being modeled to remain constant after a transformation by multiplying the base density by the Jacobian of the transformation w.r.t. input. In our case NF is applied to estimate:

$$P_{X_k}(x_k) = P_{Z_k}(f_k^\theta(x_k|\mathbf{C})) \left|\det\left(\frac{\partial f_k^\theta}{\partial x_k}\right)\right| \tag{12}$$

where $P_{Z_k} \sim N(\mu_k, I)$ with different μ_k for each entity repeated T times. Parameters $\boldsymbol{\theta}$ are in common between entities, differently to original MAF, to reduce model size and are estimated by one or more MADE block.

5) MADE: It is a model based on auto-regressive autoencoder that can estimate conditional probability by masking connection of a regular autoencoder [3]. Auto-regressive property is useful in this context since it guarantees the Jacobian to be triangular and hence allow faster computation of its determinant.

In our context, supposing that parameters are not shared among entities, auto-regressive property can be expressed as the model capability to learn parameters $\theta = \{\mu_1, \alpha_1, ...\mu_K, \alpha_K\}$ as follows:

$$\mu_k = f_{\mu_k}(x_{1:k-1}) \text{ and } \alpha_k = f_{\alpha_k}(x_{1:k-1}) \tag{13}$$

Then we can express z_k as function of x_k as:

$$z_k = (x_k - \mu_k)\exp(-\alpha_k) \tag{14}$$

And we have that:

$$\left|\det\left(\frac{\partial f^\theta}{\partial \mathbf{x}}\right)\right| = \exp\left(-\sum_k \alpha_k\right) \tag{15}$$

So given this result we can estimate Eq.12 and obtain sum of log-likehoods over all entities. Negative log-likehood is then used as loss during training and as anomaly score at inference time.

D. MSCRED [4]

1) Multi-Scale Signature Matrix Construction: The first step in the MSCRED model is to construct multi-scale signature matrices that capture both the temporal and spatial correlations in the time series data. For a given time window w, the model generates a signature matrix $M^t \in R^{n\times n}$ at time step t, where n is the number of variables in the multivariate time series. Each element m_{ij}^t in the signature matrix represents the inner product between the i-th and j-th time series within the window:

$$m_{ij}^t = \frac{1}{\kappa}\sum_{\delta=0}^{w} x_i^{t-\delta} x_j^{t-\delta}$$

where κ is a normalization factor, typically set to the length of the time window w. The use of inner products allows the model to capture both the magnitude and the direction of the relationships between different variables.

2) Convolutional Encoder: The concatenated multi-scale signature matrices are then passed through a convolutional encoder, which is designed to extract high-level spatial features from the input tensor. The convolutional encoder consists of multiple convolutional layers, each of which applies a set of filters to the input tensor to produce a feature map.

For the l-th convolutional layer, the output feature map $X^{t,l} \in R^{h_l \times w_l \times c_l}$ is computed as follows:

$$X^{t,l} = f(W^l * X^{t,l-1} + b^l)$$

where W^l denotes the convolutional filters, b^l is the bias term, and $f(\cdot)$ is the activation function. The MSCRED model uses the Scaled Exponential Linear Unit (SELU) as the activation function, which helps maintain a self-normalizing property in the network.

3) ConvLSTM Network with Attention Mechanism: To capture the temporal dependencies in the feature maps produced by the convolutional encoder, the MSCRED model employs a Convolutional Long Short-Term Memory (ConvLSTM) network. The ConvLSTM network is a variant of the traditional LSTM that incorporates convolutional operations within the LSTM architecture, making it well-suited for handling spatial-temporal data.

To further enhance the model's ability to focus on relevant temporal features, the MSCRED model incorporates an attention mechanism. The attention mechanism assigns a weight to each hidden state based on its relevance to the current time step. The attention-weighted hidden states are then aggregated to form a context vector, which is used in the decoding process.

4) Convolutional Decoder: The final stage of the MSCRED model is the convolutional decoder, which aims to reconstruct the original multi-scale signature matrices from the context vector generated by the ConvLSTM network. The decoder consists of several deconvolutional layers (also known as transposed convolutions) that progressively upsample the context vector to match the dimensions of the original input tensor.

For each layer in the decoder, the feature map $Y^{t,l}$ is computed as:

$$Y^{t,l} = f(W^l * Y^{t,l-1} + b^l)$$

where W^l are the deconvolutional filters, and b^l is the bias term. The decoder reconstructs the multi-scale signature matrices by reversing the encoding process, allowing the model to generate an approximation of the input tensor.

5) Anomaly Detection: Anomaly detection is performed by comparing the reconstructed signature matrices with the original ones, obtaining the difference between the two, known as the residual matrix The magnitude of the residuals serves as the anomaly score; larger residuals indicate a greater likelihood of an anomaly. The intuition is that the model will struggle to accurately reconstruct signature matrices corresponding to anomalous events, resulting in higher residuals.

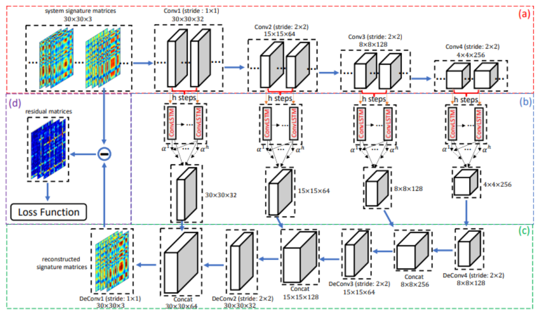

Fig. 4. MSCRED architecture

IV. Results and discussion

A. Dataset & Data prepossessing

The dataset consists of time-series data collected from the sensors of an industrial robot named Kuka at 10hz frequency. Each sample in the dataset consists of 76 features,which encompass various information such as different angles movement, current, and power usage values. The dataset is comprised of 95,815 instances of normal data and 34,275 instances of abnormal data.

For preprocessing, we excluded certain features based on expert input, as they provided additional domain-specific information like the feature where contain the ’temperature’ or ’abb’ are irrelevant for the anomaly detection task,at end we have obtained 51 feature with mean and standard deviation shown as in the figures 2 and 3, then we applied scaler accordingly to the one used in respective paper, we have chosen W=100 for window size of time series representation where the model used such representation; in other cases a point based approach was used.

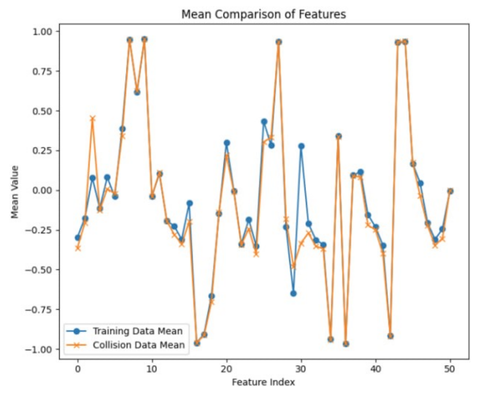

Fig. 5. Dataset features mean

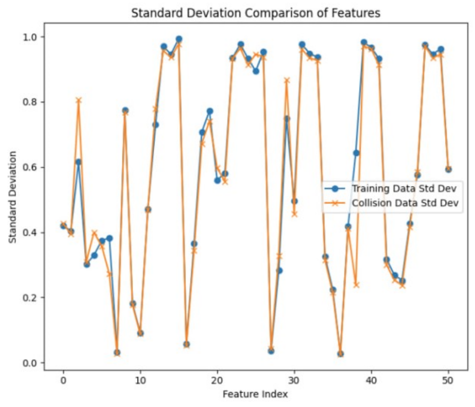

Fig. 6. Dataset features standard deviation

B. Evaluation metrics

The considered metrics are threshold-less since in the context of anomaly detection defining the best threshold is quite heuristic and should be based on a validation set which includes labeled anomaly data. The metrics selected are hence AUPRC, AUROC and maxF1.

These metrics are calculated in two different context avoiding point-adjustment [1] and compared with a random score baseline to make sure methods are effective:

1) Point-wise evaluation: Since our examined methods output is in all case a window score, the score for an instance is obtained as average of the window-scores of windows that include that instance. This way of calculating score results in better prediction than associating the score directly with the starting timestamp of the window and latency of results in a real-time context can be the almost the same.

2) Window-wise evaluation: In order to evaluate a window, its label is considered positive if there is an overlapping with an anomaly interval so this bring to an overestimate of anomalous ground-truth and hence higher AUPRC and maxF1 scores.

C. Individual experiment processs

In this paragraph we will present the experiment process for each model we have presented before

1) TadGan: In the TadGan, the model is initialized with window size, and feature number as the value cited in the previous paragraphs, while the latent space which is a proper hyperparameter of the model is initialized with Latent dimension = 30. The main component of the model are:

Encoder that consist in LSTM Layers: Bidirectional LSTM Layer with 100 units that capt temporal dependencies in both forward and backward directions,Flatten Layer they flatten the data for the further dense processing , Dense Layer which is fully connected with latent dim * feature number units that transform the flattened data into dense representation , the final reshape layer that reshapes the output into 3d tensor of shape (latent dim, feature number) in the latent space representation.

Generator which consist in Flatten Layer, Dense layer with (window size // 2) * feature number units which expand the flattened latent representation into a larger tensor,Reshape Layer, two Bidirectional LSTM layers with 64 units each to capture temporal patterns in the sequence, activation layer that uses tanh activation function to ensure that the generated values are within [-1,1].

Discriminator,which are Cx and Cz in the Tadgan model as citied in the previous paragraphs where in the Critic x we have convolutional layers, Flatten layer and Dense Layer, in the Critic Z we have a Dense Layers and a Flatten Layer.

The model is trained for 2500 epochs while for the optimization, the Adam optimizer is employed.

To detect anomalies, the trained TadGAN model processes test data to generate anomaly scores using various statistical measures such as critic score and reconstruction errors, as in the reference paper, we used different method for calculating the reconstruction error scores, here we will have a brief explanation:

- **Mean Squared Error**:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{X}_{\text{test},i} - \hat{\mathbf{X}}_i\right)^2 \tag{16}$$

  This method calculates the mean squared error (MSE) between the original test data $\mathbf{X}_{\text{test}}$ and the reconstructed data $\hat{\mathbf{X}}$ and it provides a straightforward measure of how well the model has reconstructed the time series on a score-by-score basis.

- **Area-based Reconstruction Errors**:

$$\text{Area\_Diff} = \int |\mathbf{X}_{\text{test}} - \hat{\mathbf{X}}|\, dx \tag{17}$$

  This method uses the area under the curve (AUC) to quantify the difference between the original and reconstructed time series,it provides an aggregate measure of the difference over the entire time series, accounting for both differences of the magnitude and the shape.

- **Dynamic Time Warping (DTW) Reconstruction Errors**:

$$\text{DTW}(\mathbf{X}_{\text{test}}, \hat{\mathbf{X}}) = \sqrt{\sum_{i=1}^{n}(x_i - y_{\pi(i)})^2} \tag{18}$$

  where π is the optimal path that minimizes the distance between $\mathbf{X}_{\text{test}}$ and $\hat{\mathbf{X}}$.

  DTW measures the similarity between two time series by finding the optimal alignment between them [17],it allows for non-linear alignments between the original and reconstructed series.

To evaluate TadGan performance, as in the paper of reference we deployed the combination in sum and in product of critic score and these 3 score cited above using the evaluation metrics cited on the previous section. In the sum approach we have a variable α set to 0.3 that takes count of contribution of critic score and calculated score

	TadGan Point Wise Evaluation		
Method	Max F1	AUROC	AUPRC
CRITIC ONLY	0.19	0.15	0.07
CRITIC + MSE	**0.39**	**0.82**	**0.27**
CRITIC + AREA	0.19	0.18	0.06
CRITIC + DTW	**0.39**	**0.82**	**0.27**
CRITIC x MSE	0.19	0.27	0.07
ONLY MSE	0.32	0.75	0.18
ONLY DTW	0.32	0.75	0.18
ONLY AREA	0.19	0.15	0.06

	TadGan Window Wise Evaluation		
Method	Max F1	AUROC	AUPRC
CRITIC ONLY	0.54	0.25	0.25
CRITIC + MSE	0.63	0.59	0.37
CRITIC + AREA	0.54	0.29	0.26
CRITIC + DTW	0.63	0.59	0.37
CRITIC x MSE	0.54	0.40	0.30
ONLY MSE	**0.64**	**0.61**	**0.38**
ONLY DTW	**0.64**	**0.61**	**0.38**
ONLY AREA	0.54	0.25	0.25

Based on the results highlighted in the two tables, we can draw the following conclusions:

Using the point-wise evaluation, we can see that relying solely on the critic score produced by the discriminator does not provide stability in evaluation. However, by combining the critic score with the reconstruction error calculated using methods such as DTW and MSE, we achieved reasonably high performance in terms of AUROC and AUPRC.

On the other hand, when using the window-wise approach, we observe that, on average, the MAX F1 and AUPRC scores are higher compared to the point-wise approach. Furthermore, in the window-wise approach, there is no added benefit in using the critic score. In fact, we achieved better performance by using only the reconstruction error calculated by MSE and DTW methods, without adding the critic score.

2) MTGFLOW: Since original method is window based the point-wise score is given by the mean over all windows that include the point.
Thresholds used for calclation of ROC, PRC and F1s are all the scores to ensure a complete evaluation.

The model show big variability of metrics between different runs. This can be related to:

- Poor initialization

- Absence of batch normalization
- Overflow problems during training

Overflow problems are originally addressed by clipping gradients. In addition to this in my experiment I added a step scheduler since it seems effective to avoid overflow and help convergence.

Since training different models can result in different metrics I trained and evaluated 10 different models for each set of hyperparameters. Reported metrics are expressed as $\mu \pm \sigma$.

MTGFlow Point-Wise evaluations, window size = 100				
Stride	Action	Max F1	AUROC	AUPRC
100	yes	0.35 ± 0.04	0.72 ± 0.04	0.24 ± 0.04
100	no	0.34 ± 0.03	0.70 ± 0.04	0.22 ± 0.04
30	yes	0.34 ± 0.06	0.70 ± 0.05	0.22 ± 0.05
30	no	0.33 ± 0.04	0.70 ± 0.04	0.22 ± 0.04

MTGFlow Point-Wise evaluations, window size = 30				
Stride	Action	Max F1	AUROC	AUPRC
30	yes	0.42 ± 0.06	0.77 ± 0.04	0.30 ± 0.06
30	no	0.40 ± 0.05	0.77 ± 0.03	0.28 ± 0.04

3) MSCRED: In this study, we implemented a modified version of the MSCRED, specifically, our implementation does not incorporate the multi-scale component of the original model. This modification was necessitated by constraints related to computational resources and storage capacity and it also addresses one of the model's limitations by being highly dependant on hyperparameteres, as well as the window lenght of an anomaly might be highly variable in real applications even by considering multiple scales. Moreover, excluding the multi-scale functionality simplifies the process of comparing our model with other anomaly detection approaches.Nevertheless we compare the results of many window scales.

The results produced by the model consist of reconstructed signature matrices with three channels. We focus on the the maximum reconstruction error computed but from the first channel of the signature matrices. By isolating the first channel, we can effectively assess the model's ability to accurately reconstruct the critical aspects of the input data, ensuring that our evaluation remains both robust and computationally efficient. Considering the maximum of the matrices reconstruction error is the most conservative approach, an anomal behaviour of one sensor is enough to consider the whole window as an anomaly, other approches can still lead to significant results based on the application

MSCRED Point-Wise Evaluations			
Window	Max F1	AUROC	AUPRC
40	0.27	0.59	0.13
60	0.31	0.67	0.16
80	**0.41**	**0.79**	**0.27**
80+PCA	0.46	0.79	0.29
100	0.38	0.78	0.25
120	0.34	0.72	0.20

The model results are very promising; the employment of a singular scale structure not only significantly reduces the computational complexity but also leads to competitive performance. This is due to the robustness of the model in capturing essential features within the time-series data. Hence, the application of PCA to this specific dataset results in improved outcomes.

4) VAE-LSTM: The variational autoencoder hybrid method was the only method not using windowing for the data, and using single points both for training and evaluation. The model presented various issues in the training phase, which resulted in prediction results not at the level of the other methods and architectures discussed in this paper. During the training phase the model loss has a tendency to reach a NaN (Not-A-Number) value in a fairly unpredictable manner. This phenomenon happened in various phases of the training process, either at the beginning, or after several epochs of regular behavior, with a descending loss value. Before discussing the possible causes, it's worth mentioning that this behavior resulted in an harder training process, and possibly to a much less efficient model; training for longer periods inevitably resulted in incurring in this behavior which made to evaluate the effects of higher number of epochs. For these reasons, the results which will be presented here are to be considered non-representative of the possible performance of this model and architecture in other cases, and with additional techniques to mitigate the numerical issues during training. The possible causes for this behavior are multiple, but we'll list here some of the most probable:

- *Numerical Instability in KL Divergence*: he KL divergence term in VAE can cause instability if the variance becomes too small or large;
- *Exploding/Vanishing Gradients in LSTM*: LSTMs can suffer from exploding/vanishing gradients, leading to NaNs;
- *Division by Zero/Log of Zero*: Operations like division or log can cause NaNs if zero values appear;
- *Improper Initialization*: Poor weight initialization can lead to unstable activations;
- *Unstable Latent Space*: An unbalanced loss function may cause the latent space to become unstable;
- *Data Normalization*: an incorrect data preprocessing could lead to this kind of instability;
- *Incorrect Implementation*: error of implementation of loss functions or of the model architecture may be at the base of these issues;
- *High Learning Rate*: this is probably not the cause but more a triggering factor.

To address the NaN issues during training, several mitigation strategies were employed. The learning rate was reduced significantly, including testing very low values, to improve training stability. Scaling parameters were adjusted to fine-tune the latent space, while the batch size was varied to observe its effect on gradient behavior. Gradient clipping was applied both within the optimizer and during the training step to prevent gradients from becoming excessively large. Additionally, different activation functions were tested to ensure they did not contribute to instability. To further align the model with the dataset characteristics used in the reference paper, aggressive PCA was applied to reduce the number of features, bringing them closer in scale and distribution to the reference dataset. Despite these efforts, the persistence of NaN values suggests that more advanced techniques or further model adjustments may be necessary. The results of the successful training runs resulted in an *maxF1* score slightly above the one of a random model, with 0.24 as best result obtained.

D. Final Result & Model comparison

Considering the results obtained in the range of our experiments, we can highlight the following conclusions:

- It appears that the more recent methods based on the MSCRED architecture are the most performing when threshold refinement is possible;
- At the same time, GaN-based architectures are more robust on all possible thresholds spectrum;
- It is almost certain that window-based training processes are most effective to capture the underlying structure of the data;
- While being slightly less performant on other metrics, the MTGFLOW model obtained the best result according to its AUPRC;
- Latency comparison highlights MSCRED as the heaviest model. Despite achieving good results, it is outperformed by the other models in terms of computational demands;
- The results on the MSCRED experiments show a positive effect of dimensionality reduction techniques like PCA on the model performance, but this could also be due to the nature of the data.

Comparison table

Model	Max F1	AUROC	AUPRC	Latency(ms)
TadGAN	0.39	**0.82**	0.27	**10**
MTGFlow	0.42	0.77	**0.30**	41
MSCRED	**0.46**	0.79	0.29	288
VAE-LSTM	0.24	0.55	0.10	56
baseline	0.22	0.50	0.12	–

TABLE I
COMPARISON BETWEEN METRICS AND LATENCY OF THE BEST PERFORMANCE SETUP FOR EACH MODEL.

V. CONCLUSIONS AND FUTURE WORKS

While the data to be analyzed becomes progressively more complex and autonomous systems more pervasive and crucial in our society, models able to accurately and timely detect unusual behavior become highly necessary. In this paper we analyzed some of the approaches used to solve this problem, and we compared their results on a fairly complex time series AD dataset and problem.

Newer approaches like the TadGan model and the MSCRED one sparked our interest in terms of results and prove to be the most reliable to detect anomalies on a complex multimodal time serie like the one used in our experiments, probably being worth the higher computational resources needed in terms of training.

Future endeavours on these topic might explore different and more sophisticated data preprocessing, which was out of scope for this research, and more iterations on architecture and hyperparameters of the best performing models, building the foundations to have reliable and lightweight model usable for any kind of analysis, possibly even on-edge detection.

REFERENCES

[1] Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. Towards a rigorous evaluation of time-series anomaly detection, 2022.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[3] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation, 2018.

[4] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V. Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1409–1416, Jul. 2019.

[5] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[6] Dequan Zheng, Fenghuan Li, and Tiejun Zhao. Self-adaptive statistical process control for anomaly detection in time series. *Expert Systems with Applications*, 57:324–336, 2016.

[7] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[8] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.

[9] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. In *Proc. of the 24th ACM SIGKDD*, 2018.

[10] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*, pages 703–716. Springer, 2019.

[11] Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. Sensitivity of pca for traffic anomaly detection. In *Proc. of the 2007 ACM SIGMETRICS*, pages 109–120, 2007.

[12] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1), 2015.

[13] Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Tadgan: Time series anomaly detection using generative adversarial networks, 2020.

[14] Qihang Zhou, Jiming Chen, Haoyu Liu, Shibo He, and Wenchao Meng. Detecting multivariate time series anomalies with zero known label, 2023.

[15] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, 54:30 – 44, 2019.

[16] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2242–2251, oct 2017.

[17] Donald J Bemdt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *AAAI Workshop on Knowledge Discovery in Databases*, Seattle, Washington, 1994.