Journal of Computer and Communications, 2026, 14(2), 130-152
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

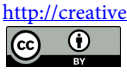
Prototype-Based Dynamic Autoencoders for Real-Time Anomaly Detection and Explainability

Charles Aba Medzo[1*], Handy Kenne Evina[1], Daniel Olle Olle[2], Diane Tchakonte Tchuani[1,2]

[1]Department of Computer Engineering, University of Ebolowa, Ebolowa, Cameroon
[2]Unité de Modélisation Mathématique et Informatique des Systèmes Complexes, Institut de Recherches pour le Développement, Bondy, France
Email: *medzoaba@yahoo.com, junior.evina@facsciences-uy1.cm, georgesdelortolleole@gmail.com, dianetchakonte@gmail.com

How to cite this paper: Medzo, C.A., Evina, H.K., Olle, D.O. and Tchuani, D.T. (2026) Prototype-Based Dynamic Autoencoders for Real-Time Anomaly Detection and Explainability. *Journal of Computer and Communications*, **14**, 130-152.
https://doi.org/10.4236/jcc.2026.142007

Received: January 5, 2026
Accepted: February 23, 2026
Published: February 26, 2026



Abstract

Real-time detection of anomalies in data streams is a foundation of modern applied analysis in complex systems. It enables experts to design rapid, efficient, reliable, and high-performance decision support systems. Rapid identification of abnormal or unexpected behavior is often insufficient in high-risk contexts; therefore, it is essential to provide explanations for these predictions. Although the use of autoencoder-based algorithms has been shown to be effective in the detection of anomalies, it is important to note that their effectiveness decreases with time due to changes in the underlying data distribution. To solve this drawback, we introduce a solution that incorporates the dynamic updating of autoencoders based on the prototypes computed from their latent representations. We propose two algorithms: Update-Based Prototypes-v1, an algorithm specifically designed to detect anomalies through continual, prototype-informed updates, and ARCANA Prototypes-v1, which is developed to provide reliable explanations for anomalies detected over time. We used the Skoltech Anomaly Benchmark database for our experiments. The Update-Based Prototypes-v1 algorithm achieved a detection rate of 80% with a false alarm rate of 20%, compared to 49% for an implementation without updates. The results confirm that the proposed model offers high performance comparable to state-of-the-art architectures in the same field, even when data characteristics change. Additionally, ARCANA Prototypesv-1 provides precise explanations, which enhances the usefulness of the prototypes for explaining anomalies.

Keywords

Anomaly Detection, Autoencoder, Prototype-Based Learning, Explainable AI, Real-Time Systems

1. Introduction

The modern world is becoming increasingly connected and digital. IoT sensors, social networks, or financial transactions continuously produce data, forming what is known as data streams [1]. These streams cannot be stored in totality, which requires real-time analysis. Examples include smartwatches that track heart rate, sleep, and exercise to detect any health anomalies in real-time, satellites that scan the environment, and seismographs that track ground activity to predict natural disasters.

The process of identifying an event that deviates significantly from expected behaviour is known as anomaly detection [2]. In sensitive domains, such as finance, healthcare, or security, it is not enough to identify anomalies; it is also necessary to provide explainable and interpretable explanations that enable professionals to perceive the real underlying causes of the abnormal behaviors and generate trust towards the ML-based predictive models, a requirement that falls within the field of explainable artificial intelligence (XAI) [3].

The effectiveness of an artificial intelligence model depends largely on its learning paradigm, *i.e.*, the nature of the data and the way the algorithm interacts with the available information. Two approaches are particularly prominent in this context. **Supervised learning** relies primarily on the availability and proportion of ground truth labels during training, enabling the model to predict labels for new, unseen data. In contrast, **semi-supervised learning** offers a hybrid solution to the problem of limited labeled data. It combines a small portion of annotated samples with a large amount of unlabeled data, with the aim of exploiting the statistical structure and distribution of the unlabeled data to refine the decision boundary initially inferred from the labeled examples.

Several studies have focused on the detection and explainability of anomalies. Autoencoders [4] [5], particularly the ARCANA method [6], are popular because of their robustness, but the results are not always interpretable. ARCANA takes a bias vector into consideration to create explanations that depict the deviation of the features that cause the identified anomaly. Although the ARCANA model is designed particularly for data streams, there is not an update process in terms of using autoencoders to handle changes in the distribution of data, a process known as concept drift. This can pose challenges to the reliability of the explanation process itself because the autoencoders contribute to the calculation of the reconstruction error. Therefore, although useful, autoencoders face challenges in handling data streams due to the problem of concept drift. To improve the detection of anomalies produced by ARCANA and the explanation of identified anomalies, this study explores the integration of conceptual drift management through dynamic updating of the autoencoder. To resolve this problem, we propose an anomaly detection model named Update Based Prototypes-v1, which uses the prototypes obtained by a VAE. These prototypes capture all the main peculiarities of each class and allow the detection of changes in data distribution, which, in turn, initiates dynamic model changes. We also propose ARCANA Prototypes-v1, which

is an extension of ARCANA that provides reliable and consistent explanations of the observed anomalies to improve the explainability of the model.

The main contributions of this study are as follows:

- Consideration of concept drift in data streams through a dynamic update mechanism for the detection autoencoder.
- Introduction of prototypes to detect distributional changes and guide model updates.
- Extension of the ARCANA method to generate faithful explanations of detected anomalies in an evolving context.

The rest of this paper is structured as follows: Section 2 discusses related work on data stream anomaly detection and explainability. The proposed method is described in Section 3. Section 4 contains the experimental setup and the corresponding results. Section 5 concludes the paper and outlines potential directions for future research.

2. Related Work

Anomaly detection in data streams is a critical challenge in many domains, such as the Internet of Things (IoT), industrial predictive maintenance, and cybersecurity, as a result of the dramatic and diverse increase in data. In this context, traditional methods show their limitations when confronted with noise, variability, and lack of labels, and autoencoder-based models, especially variational autoencoders (VAEs), provide an effective and adaptive model. These architectures allow the learning of a latent representation of the "normal" and the identification of deviations using an error metric or reconstruction probability. Nevertheless, a major challenge remains: the explainability of detected anomalies. Unlike classical models, it is often difficult to justify why a specific instance is considered anomalous, which limits the adoption of these systems in critical or regulated environments. Therefore, this literature review has the purpose of synthesizing the existing work on anomaly detection using autoencoders and the associated explainability, in order to identify the existing gaps.

a) Anomaly detection using autoencoders and variational autoencoders

Autoencoders are increasingly used for anomaly detection due to their robustness against noise and variability in real-world data. By reconstructing normal representations, these models detect anomalies as significant deviations, making them suitable for environments with imperfect, heterogeneous, or noisy data. Numerous studies have explored and refined their applications across various domains, as detailed below.

In 2015, Jinwon An *et al.* [3] introduced a probabilistic approach that leveraged reconstruction probability as an anomaly score. This approach enhanced the process of handling heterogeneous data and unusual cases, and is more sensitive than other autoencoders. However, it was constrained in identifying weakly represented anomalies and weakly interpretable explanations, which limits its use in safety-critical systems. N. Pinon *et al.* (2019) [7] examined the problem of anom-

aly detection in video streams. They fused a Fully Convolutional Network (FCN) to derive spatiotemporal features, PCA-based dimensionality reduction, and a One-Class SVM to achieve classification. Though the model was effective in capturing normal behavior as well as identifying abnormalities in video sequences, its computational cost was still high, especially when dealing with large-scale video sequences or real-time video sequences.

To improve detection in multidimensional time series, Longyuan Li *et al.* (2020) [8] proposed the Smoothness-Inducing Sequential Variational Autoencoder (SISVAE), a sequential extension of the VAE designed to enforce smoothness in the latent space. This approach enhances the detection of point anomalies while preserving temporal structure, yielding superior performance compared with classical autoencoders on sequential data.

Recent studies (2024-2025) have also explored the structural and practical constraints of autoencoder models. In 2024, S. Erniyazov *et al.* [9] compared a classical autoencoder and an LSTM-AE for the detection of anomalies in temperature and sound time series. Both models were difficult to interpret as they are black boxes and were in need of regular retraining to keep up with temporal shifts, although they were effective. Later that year, T. Walczyna *et al.* [10] studied the issue of latent space manipulation and trained lightweight autoencoders, demonstrating that models that are too small simply do not have the capacity to recognize any complex or subtle anomalies. The mode of failure of autoencoders became a research subject in 2025. R. Bouman *et al.* [11] showed that certain autoencoders are able to completely recreate anomalies located far apart in normal patterns, resulting in critical false negatives. In their turn, B. Heleen *et al.* used autoencoders on Intelligent Transportation Systems (ITS) and emphasized that they are highly reliant on the normal data that restricts their capacity to identify new anomalies in mixed sensor streams. The same year, T. Pandiselvi *et al.* [12] investigated the problem of anomaly detection on multimodal sensor networks and highlighted the issues of class imbalance, modality absence, and cross-domain adaptation.

Autoencoders and their variants have a number of common limitations despite their strengths. Their dependence on normal data makes them vulnerable to temporal drift and evolving operational contexts, requiring frequent retraining. They are less interpretable, as the anomalies identified by them are black-box and therefore cannot be used in high-stakes decision-making. Moreover, rare or atypical anomalies are still hard to find because of the reconstruction bias. Lastly, high-dimensional data streams or multimodal data streams are also computationally expensive to process, particularly when dimensionality makes it difficult to extract and select useful features.

b) Explainable Anomaly Detection Methods

A majority of the literature on explainable approaches to anomaly detection uses a two-model architecture, with one model performing detection and the other producing explanations post hoc. Nevertheless, this dichotomy makes the process of detection and interpretability difficult, which tends to lead to discrep-

ancies between the predictions and the explanations thereof. To overcome the limitations of two-model methods, Chong *et al.* [6] introduced a unified framework that simultaneously considers outlier detection and explainability by prototype learning guided by a teacher model. The distance between a sample and its corresponding prototype is used as an indicator of anomaly, which allows detection and interpretability to be combined into a single model. Nevertheless, the combined optimization of these objectives introduces a delicate trade-off between detection and the quality of explanations, and the usage of a teacher model can restrict the generalization ability of the method. Cyriana Roelofs *et al.* [13] introduced ARCANA, a method that identifies the features responsible for an anomaly detected by an autoencoder. For each sample x, x_{bia} a bias is computed to correct the input $x_{corr} = x + x_{bia}$. The L_2 norm ensures the input is corrected, the L_1 norm limits the number of affected features, and optimization is performed via Adam. Biases can be averaged over a temporal window to provide interpretable explanations for anomalies.

Recent studies on explainability in anomaly detection have largely superseded traditional two-model architectures. In 2020, Assaf *et al.* [14] proposed a model based on a convolutional autoencoder for the analysis of multivariate time series, providing a level of explainability suited to engineers' needs. However, this method relies on a limited number of labels, reconstruction error, and historical anomalies, which can limit its robustness and its ability to generalize to rare or never-before-seen behavior. In 2023, Dollon *et al.* [15] suggested a semi-supervised algorithm for anomaly detection in hydroelectric production units, emphasizing statistical explainability to facilitate equipment condition diagnosis and management. However, as with many purely data-driven methods, this approach remains limited in its ability to identify the underlying causes of anomalies, highlighting persistent methodological challenges in linking anomaly detection to actionable insights. He Cheng *et al.* [16] introduced an ESAD model for explainable sequential anomaly detection in a semi-supervised setting, generating multiple prototypes to capture the diversity of normal and abnormal sequences. Nevertheless, in spite of its contributions, this method is highly reliant on the quality of prototypes, has a strong inability to detect rare behavior, and has an excessive computational cost that makes it less practical in large-scale or highly diverse settings.

Noorchenarboo *et al.* [15] suggested an explainability approach for anomaly detection in energy consumption data, based on the SHAP variants, in 2025. Although novel, this approach is characterized by a number of limitations, such as a high level of computation, unstable SHAP-based methods, and lack of a contextual factor, resulting in unstable answers. At the same time, Levshun *et al.* [17] used an autoencoder to perform anomaly detection in industrial IoT systems with the explainability of SHAP. This method is, however, highly reliant on the quality of data and the presence of any applicable variables, and the calculation of SHAP values is computationally intensive, which limits its applicability to large-scale in-

dustrial problems.

In summary, recent approaches combine anomaly detection with explainability, leveraging unified models, prototypes, autoencoders, or SHAP-based techniques. Despite these advances, challenges remain, including dependence on training data, high computational cost, and difficulty in explaining rare or previously unseen anomalies. These limitations motivate the development of a more robust and interpretable methodological framework, which is presented in the following section.

3. Preliminaries

Anomaly detection is based on variational autoencoders, because regularization based on Kullback-Leibler divergence allows the latent distribution to approximate the standard normal distribution. This increases the uniformity and interpretability of the latent space. First, the variational autoencoder encodes the labelled data in this space. Then, a sampling function provides random realizations based on parameters supplied by the encoder. Random sampling in the latent space allows us to obtain more regular and robust representations defined by a vector $\boldsymbol{z}$, which follows a Gaussian probability distribution of the latent space.

In a Variational Autoencoder (VAE), the encoder maps an input $\boldsymbol{x}$ to the parameters of a latent distribution, typically a multivariate Gaussian. The latent vector $\boldsymbol{z}$ is then sampled using the reparameterization trick, as given in Equation (1):

$$\mathbf{z} = \mathbf{z}_{\text{mean}} + \exp\left(0.5 \,|\, \mathbf{z}_{\log\_\text{var}}\right) * \epsilon, \quad \text{with } \epsilon \sim N(0,1), \tag{1}$$

where $\mathbf{z}_{\text{mean}}$ and $\mathbf{z}_{\log\_\text{var}}$ denote the vectors of means and log-variances predicted by the encoder, and ϵ is standard Gaussian noise.

This reparameterization enables the encoder to produce a continuous latent vector $\boldsymbol{z}$ in a differentiable manner, capturing essential input features for decoding or downstream tasks such as anomaly detection.

The training of the variational encoder aims to accurately reconstruct the input data while producing a compact and regular representation in the latent space. This reconstruction is achieved by minimizing the mean squared error (MSE), chosen for its statistical properties, ease of optimization, and intuitive interpretation. Formally, the training consists of minimizing the loss function described in Equation (2):

$$\text{MSE}\left(x_i, x_i'\right) = \frac{1}{m}\sum_{i=1}^{m}\left(x_i - x_i'\right)^2, \tag{2}$$

where x_i and x_i' denote the i-th original and reconstructed features, respectively.

Learning a compact and regular representation of the latent space involves minimizing the Kullback-Leibler (KL) divergence between the learned latent distribution and a reference distribution, typically a standard Gaussian. This regularization term structures the latent space by encouraging the model to produce representations consistent with the target distribution. It is defined in Equation (3):

$$\mathrm{KL}\left(q_\phi\left(\mathbf{z}\mid\mathbf{x}\right)\|\,p_\theta\left(\mathbf{z}\right)\right)=-\frac{1}{2}\sum_{i=1}^{d}\left(1+\log\left(\sigma_i^2\right)-\mu_i^2-\sigma_i^2\right), \tag{3}$$

where μ_i^2 and σ_i^2 are the mean and variance of the latent variable $\boldsymbol{z}_i$ for dimension i, and d is the dimensionality of the latent space. This term measures the divergence between the approximate posterior and the prior, and is commonly included in the VAE loss to regularize the latent representation.

The total loss is defined as the weighted sum of the two components, placing greater emphasis on the reconstruction error, as given in Equation (4):

$$\mathcal{L}_{\mathrm{VAE}}=\frac{1}{n}\sum_{j=1}^{n}\left[\alpha\mathrm{MSE}\left(\mathbf{x}_j,\mathbf{x}_j'\right)+\beta\mathrm{KL}\left(q_\phi\left(\mathbf{z}_j\mid\mathbf{x}_j\right)\|\,p\left(\mathbf{z}\right)\right)\right], \tag{4}$$

where n is the number of samples, $\mathbf{x}_j$ and $\mathbf{x}_j'$ represent the input and reconstructed samples, KL denotes the Kullback-Leibler divergence, and α and β are weighting coefficients that assign greater importance to the reconstruction error.

In the following section, we present our methodological framework, proposing a unified approach for accurate and interpretable anomaly detection on complex datasets.

4. Methodology

This section describes the proposed approach for anomaly detection and explanation of detected anomalies. The approach comprises two main steps:

Step 1: Anomaly detection using the Update-Based Prototypes-v1 algorithm

The data are introduced to the autoencoder with their labels during training. These labels are used to organize the latent space according to classes and determine the corresponding prototypes for each class in this space. During the prediction stage, the error is also assessed not only through the comparison of input data with their reconstructions but also by taking into account the classification in the latent space. This error is compared to a threshold. The observations with an error below this threshold are considered normal, while those above this threshold are regarded as anomalies.

In case the model misclassifies a data point, the difference between the misclassified point and the prototypes of the class in question is measured. In case there is a significant difference, it is a sign of a changing data distribution. Where this occurs, the model is retrained to adapt to the new distribution while updating the class prototypes.

Step 2: The explainability of the detected anomalies using the ARCANA Prototypes-v1 algorithm

After anomaly detection, the ARCANA-Prototypes-v1 model was applied to analyze the data and identify the features that contributed most to each anomaly. Figure 1 describes the general architecture of the proposed approach, including the main stages in the process of detecting anomalies, as well as the explainability mechanisms.

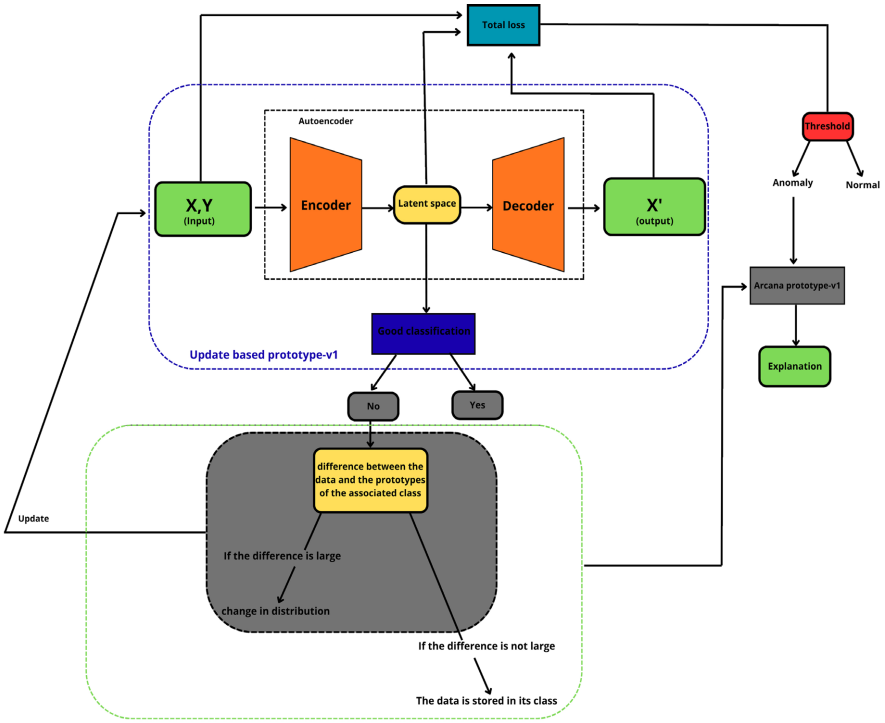


Figure 1. General architecture of the proposed method.

4.1. The Update-Based Prototypes-v1 Algorithm

For anomaly detection, our approach is based on a Variational Autoencoder (VAE). The idea behind this is to train the model to learn to reconstruct the input data as accurately as possible to the original, and the reconstruction error is then used to detect the anomalous instances. The design and training procedure of the VAE process include the following steps:

- Encoder design: In the construction of the encoder, we identified the number of layers, the architecture, and the choice of activation functions. The encoder consists of five layers, successively reducing the number of dimensions of the data, and then a layer that generates the mean and log-variance of the latent distribution.
- Latent space sampling: A sampling function was defined to generate latent vectors from the encoder's distribution.
- Decoder design: In the context of this study, the decoder has been built in a symmetric manner with respect to the encoder and has built a mirror-like structure, reassembling the data using the latent vectors.
- Loss function: It associates the reconstruction error with the VAE regularization term to guide the learning process.
- Model training: The autoencoder was trained with a fixed number of epochs, and in the process, the model had the ability to approximate the underlying data distribution and further improve its reconstruction capabilities.

We chose a symmetric structure for the autoencoder so that the decoder structure mirrors the encoder, ensuring better learning and reconstruction stability.

The labels have to be separated out from the features in the labeled data context. Additionally, the data have been normalized to avoid unwanted effects of particular variables on the model because of their scales. Standardization was chosen due to the fact that all features are treated fairly irrespective of their scale. Table 1 describes the configuration of the update Based Prototypes-v1 Algorithm.

Table 1. Configuration of the VAE model

Configuration	Value
Number of epochs	1000
Normalization	Standard
Batch size	100
Optimizer	AMSGrad
Learning rate	0.0005
Input data dimension	$8 \times N$
Encoder network	5 dense layers with kernel sizes: (128, 64, 32, 16, 8)
Latent space dimension	$4 \times N$
Decoder network	5 dense layers with kernel sizes: (8, 16, 32, 64, 128)
Autoencoder output dimension	$8 \times N$

4.1.1. Prototypes

The autoencoders are supervised trained in our model, and the labels are applied to organize the latent space based on the classes with the help of the Kullback-Leibler divergence. This system enables retrieval of both normal prototype data and anomalies since the latent space has been trained to differentiate anomalies and regular observations. To determine these prototypes, we use a clustering algorithm that captures the class distributions in the latent space. The prototype of a class is the center of each cluster. Thus, the prototype p_c of a class c is computed as the mean of the points in the corresponding cluster. p_c is described by Equation (5). After this algorithm is executed, prototypes are obtained, representing the average representation of the training data.

$$P_c = \frac{1}{n}\sum_{i=1}^{m} x_i, \tag{5}$$

where n denotes the number of observations, c the corresponding class, and x_i the data vector belonging to class c.

4.1.2. Determining the Threshold and Updating the Autoencoder

One of the most critical steps in the anomaly detection process is the selection of an appropriate decision threshold, as an improper choice may lead to misclassification. Several thresholding techniques were evaluated, and the one yielding the best performance was ultimately selected. The evaluation criteria are as follows:

- F1-score: The F1-score, the harmonic mean of precision and recall, is particularly suitable for handling class imbalances, which are common in anomaly detection.
- False Alarm Rate: The proportion of normal instances incorrectly classified as anomalies (false positives) indicates the system's ability to limit unnecessary alerts.
- Missed Alarm Rate: The proportion of undetected anomalies (false negatives) used to assess the model's overall performance.

We propose a supervised learning approach based on prototypes that represent the data of each class in an optimal way. In the latent space, since the distribution of each class is represented as bimodal, two prototypes per class are defined. For each data window, instances are reconstructed with the autoencoder, which ensures an accurate latent representation, and the total reconstruction error is calculated for every data point. Anomalies are then detected by comparing this error to a predefined threshold, with points having errors exceeding the threshold classified as anomalous.

During training, which is conducted in a supervised manner using the labels available within the training windows, class prototypes are learned directly from the latent space. When a data point is misclassified, its distances to the two prototypes associated with its class are considered. A large distance indicates a potential change in the data distribution, which may cause the corresponding prototypes to shift. The VAE model is then retrained on the current data window using the updated labels, resulting in new prototypes and allowing the model to adapt to changes in the data distribution.

The following Algorithm 1 describes the prototype- and autoencoder-based anomaly detection process.

Algorithm 1: Update-Based Prototypes-v1

Require: $X = \{x_i\}_{i=1}^{n}$; $y = \{y_i\}_{i=1}^{n}$ with $y_i \in \{1, \cdots, C\}$; threshold τ

Ensure: the VAE model and prototypes $\{p_{c,k}\}$, with $c = 1, \cdots, C$ and $k = 1, 2$

1: For $i = 1$ **to** n **do**

2: Predict class $\hat{y}_i \leftarrow \text{VAE.predict}(x_i)$

3: If $\hat{y}_i \neq y_i$ **, then**

4: $d_i = \min_{k \in \{1,2\}} \left\| z_i - p_{\hat{y}_i,k} \right\|_2$

5: If $d_i > \tau$, then

6: Fine-tune the VAE using (x_i, y_i)

7: Select

8: $k^* = \text{argmin}_{k \in \{1,2\}} \left\| z_i - p_{y_i,k} \right\|_2$

9: Add z_i to subset S_{y_i,k^*}

10: Update

11: $n_{y_i,k} = \left| S_{y_i,k} \right|$

12: Update prototype

Continued

13: $p_{y_i,k} = \frac{1}{n_{y_i,k}} \sum_{z \in S_{y_i,k^*}} z$

14: **End If**

15: **End If**

16: **End For**

17: Return updated VAE model and prototypes $\{p_{c,k}\}$.

4.2. ARCANA Prototypes-v1

We propose to leverage the presence of two prototypes per class for explainability, providing a diversified representation of the data within each category. We developed an enhanced version of the ARCANA method, in which the corrected sample x is replaced by a class-specific narrative correlate, and the loss function is adapted to our multi-prototype configuration.

For each class c, the prototypes $p_{c,1}$ and $p_{c,2}$ are considered as local means computed from distinct subsets within the class. With a total of two classes $(c=1,2)$, this results in four prototypes in total. The loss function aims to minimize the discrepancy between the VAE-reconstructed output and the closest prototype among those associated with the predicted class $\hat{C}$ obtained via vae.predict(x), while incorporating a correction guided by a bias vector. We propose the loss function defined in Equation (6):

$$Loss = (1-\alpha)\frac{1}{2}\left[\min_k \left\| \hat{P}_{proto,c,k} - g\left(h\left(x_{corr}\right)\right)\right\|_2^2\right] + \alpha \left\| x_{corr} - x \right\|_1, \quad (6)$$

where: $k \in \{1,2\}$ denotes the normalized prototypes extracted from the latent space of the predicted class; $\min_k$ identifies the smallest difference among the two prototypes of the predicted class; x is a data sample, represented as an input feature vector at a given time; $\alpha \in [0,1]$ is a hyperparameter controlling the relative importance of each term; $x_{corr} = x + \beta \cdot \text{bias}$, where β is a scalar controlling the amplitude of the bias, and bias is the bias vector from the ARCANA method.

The bias vector is therefore used to correlate the sample with the normal data prototype. To explain anomalies in the data, we combine the data reconstruction process, bias correction, and prototype comparison to detect and interpret anomalies accurately and in an interpretable manner.

Data processing is performed in batches of N. This method does not load all the data into memory at once, which is convenient when dealing with large data sets. The predicted class of each window is then obtained in order to determine which of the four available prototypes should be used to determine the quality of the reconstruction. The correction vector is applied to each element x of a window, as defined by Equation (7):

$$x_{corr} = x + \beta \cdot \text{bias} \quad (7)$$

This bias compensates for the systematic differences or errors in the data before they are passed to the autoencoder. The reconstruction error is then computed between x_{corr} and x_{pred}, under the assumption that normal data should be re-

constructed more accurately with respect to the prototypes. Furthermore, the discrepancy between the corrected data x_{corr} and the original data x is regularized to limit excessive corrections.

After computing the loss function, the parameters of the autoencoder (weights of the encoder and decoder layers) are updated using a gradient descent algorithm such as AMSGrad. Simultaneously, the bias vector is adjusted on the basis of the errors detected in the current data window, improving the reconstructions in subsequent windows.

The prototypes are updated by calculating the mean of the corresponding subsets in a 2-dimensional space at the end of every window. The algorithm will then give the minimum loss $loss_{min}$ and the bias vector that occurred. The size of this vector is defined as the size of the input features, and each value is a contribution made by that particular feature to the anomaly. Thus, the bias vector can be directly interpreted to provide an accurate and interpretable explanation mechanism. The ARCANA-Prototypev1 is described as follows in Algorithm 2 below.

Algorithm 2 ARCANA-Prototype-v1

Require: $X \in \mathbb{R}^{n\times d}, \alpha, N, \left\{\hat{p}_{proto,c,k} \in \mathbb{R}^d \mid c=1,2; k=1,2\right\}$

Ensure: $\text{bias}_{\min}$, $\text{loss}_{\min}$

1. Initialize bias = 0
2. **For** each window of size N **do**
3. **for** each x in window **do**
4. $\hat{c} \leftarrow \text{VAE.predict}(x)$
5. $x_{corr} \leftarrow x + \beta \cdot \text{bias}$
6. Reconstruction x_{corr} using the VAE:
7. $z \leftarrow f(x_{corr})$
8. $x_{pred} \leftarrow g(z)$
9. $\mathcal{L} \leftarrow \frac{1-\alpha}{2}\min_k \left\|\hat{p}_{proto,\hat{c},k} - g(h(x_{corr}))\right\|_2^2 + \alpha\left\|x_{corr} - x\right\|_1$
10. Update VAE via AMSGrad
11. **end for**
12. Update prototypes: $\hat{p}_{proto,c,k} = \frac{1}{n_{c,k}}\sum_{j\in\mathcal{S}_{c,k}} z_j$
13. **end for**
14. **return** $\text{bias}_{\min}$, $\text{loss}_{\min}$

5. Results and Discussion

5.1. Skoltech Anomaly Benchmark (SKAB) Dataset

The Skoltech Anomaly Benchmark (SKAB) dataset [18] was used in our experiments. It was designed by Ktser and Kozitsin. This industrial test bed is equipped with a number of sensors and features various operating and failure modes, such as isolated, collective, and normal-abnormal state changes. SKAB was specifically designed to test anomaly detection algorithms in a real industrial situation.

The dataset consists of 34 subsets: a normal subset and 33 subsets with anom-

aly-causing conditions. The subset has 10 variables, which are 8 sensor measures and 2 labels: anomaly, with a value of 1 when an anomaly takes place and 0 otherwise; changepoint, which is equal to 1 when a collective anomaly begins and 0 when not. The data are provided as a multidimensional time series. In our experiments, we used the 33 subsets with anomalies, resulting in 37,401 data points, of which 13,067 were cases of anomalies (about 34%). To maintain a balanced proportion of normal and abnormal observations, the dataset was divided into the training, validation, and test sets.

5.2. Evaluation Metrics

To validate our model, we used the following supplementary criteria:

- **Training error**: to verify that the model has correctly learned the training data.
- **Validation error**: to ensure that the model generalizes well to unseen data.
- **Latent space visualization**: to confirm that the learned latent space is semantically coherent and suitable for anomaly detection.

Anomaly detection performance was evaluated using the following metrics:

- **F1-score:** combines precision and recall into a single value ranging from 0 to 1, with 1 representing the best possible performance. It is defined by Equation (8).

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{8}$$

where: $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$ and

$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$

- **False Alarm Rate:** represents the proportion of normal observations incorrectly classified as anomalies. It is defined by Equation (9):

$$\text{False Alarm Rate} = \frac{\text{False Positives}}{\text{True Negatives} + \text{False Positives}} \tag{9}$$

- **Missed Alarm Rate:** describes the proportion of anomalies that were not detected. It is defined by Equation (10).

$$\text{Missed Alarm Rate} = \frac{\text{False Negatives}}{\text{False Negatives} + \text{True Positives}} \tag{10}$$

To evaluate the model's explainability, we used two approaches:

- To tune a characteristic to zero and examine whether the algorithm considers it abnormal.
- Reset the previous characteristics considered abnormal to verify that the model's prediction changes when the anomalies are removed.

This dual approach ensures that the model behaves consistently and is interpretable.

5.3. Detection and Analysis of Anomalies with the Update-Based Prototypes-v1 Model

To deploy the model, we performed training in order to identify normal data.

From this training, we extracted representative prototypes of each class in the latent space. Subsequently, an optimal threshold was found which helped to effectively distinguish between the normal data and the anomalies. Finally, the model is updated every time there are deviations in the data distribution, which allows the model to keep performing over time.

5.3.1. Autoencoder Model Training

We designed a supervised anomaly detector using a variational autoencoder [5]. The encoder consists of five densely connected layers estimating the mean and logarithm of the variance of the latent space, and a symmetrically structured decoder is used for reconstruction. A fixed loss function is used to ensure training stability and reliability in detection. The detailed configuration of the encoder is shown in Table 2.

Table 2. Configuration of the autoencoder.

Configuration	Value
Number of epochs	1000
Normalization	Standard
Batch size	100
Optimizer	AMSGrad
Input dimension	0.0005
Encoder architecture	$8 \times N$
Latent dimension	(128, 64, 32, 16, 8)
Decoder architecture	(8, 16, 32, 64, 128)
Output dimension	$8 \times N$

5.3.2. Latent Space Visualization and Prototype Extraction

The training of the proposed autoencoder enabled the latent space to be organized according to the different classes, as illustrated in Figure 2. The yellow points correspond to anomalies, while the purple points represent normal data. This visualization shows that the model clearly succeeds in separating the two categories, confirming the autoencoder's ability to effectively structure the latent space for anomaly detection.

A closer examination of the data distribution for each class in the latent space revealed a bimodal pattern. Consequently, we applied the K-means algorithm to partition the latent space into clusters and identify the cluster centers that best represent each class. Two prototypes were defined for each class, as using a single prototype could lead to higher classification errors due to the bimodal distribution.

This resulted in a total of four prototypes for the two classes, as shown in Figure 2. Analysis of the data distribution for each class in the latent space revealed a bimodal pattern. Accordingly, we used the K-means algorithm to separate the latent space into clusters and identify the centroids of the clusters that most represent each class. Two prototypes were defined per class, as the dependency

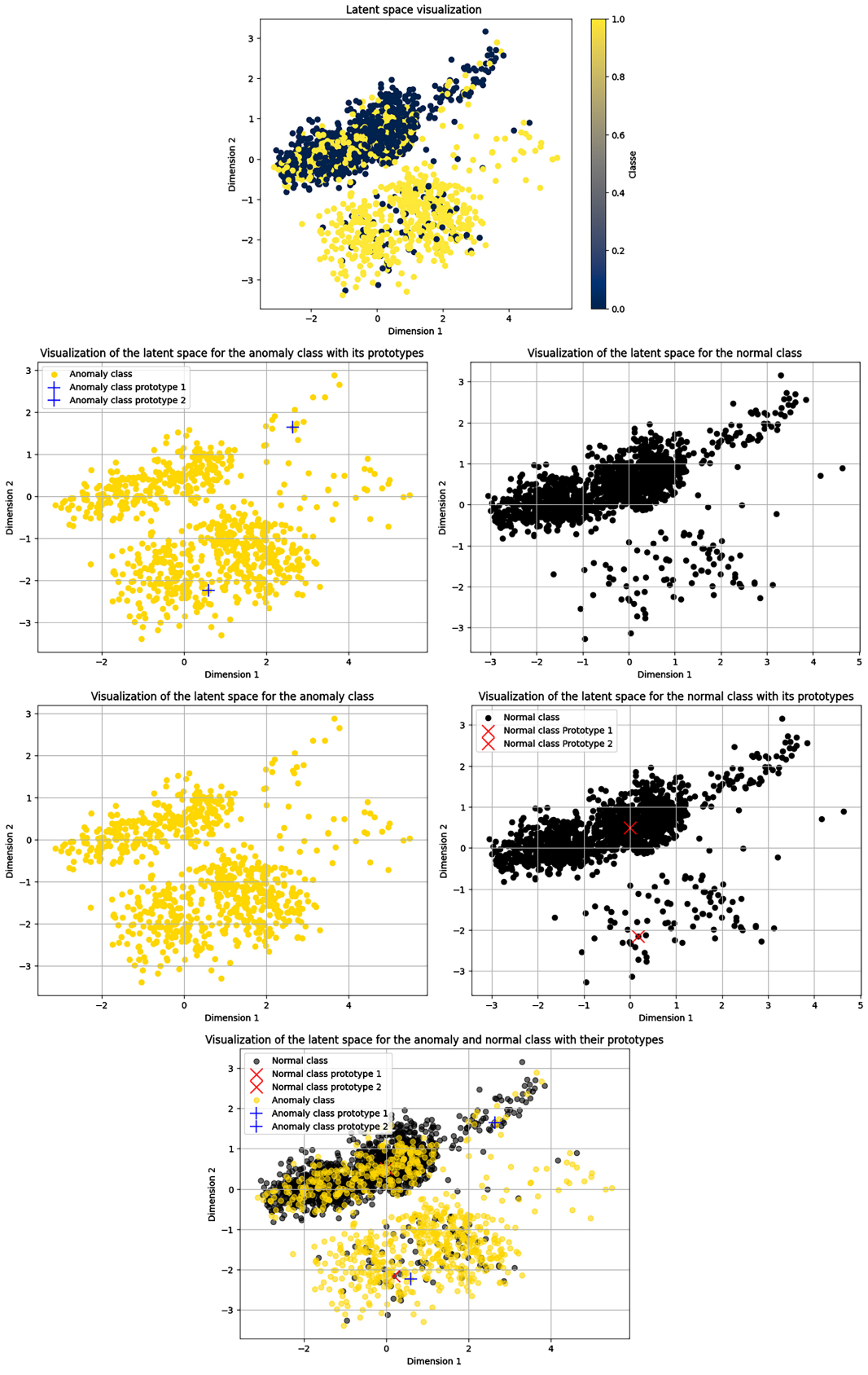


Figure 2. 2D Visualization of the latent space and prototype determination.

on a single prototype may provide high classification error due to the bimodal distribution. This resulted in a total of four prototypes for the two classes, as shown in Figure 2. Principal Component Analysis (PCA) was used as a dimensionality reduction tool to project these clusters into a 2D space.

5.3.3. Optimal Threshold Determination

To identify thresholds, we used the test data set and applied several supervised thresholding methods [19]:

- Area Under the Curve (AUC): The optimal threshold is determined by finding the point on the ROC curve that maximizes the true positive rate (TPR) while minimizing the false positive rate (FPR), typically calculated as the distance from the point (0, 1) on the ROC curve.
- Best F-score: Identifies the threshold that maximizes the F1-score for a given dataset, allowing evaluation of the maximum performance achievable by an anomaly detection algorithm with a fixed threshold.
- Top-k: Selects a threshold that marks exactly k time points as anomalous, where k corresponds to the actual number of anomalies in the test set.
- Quantile-based thresholding: Classifies values (e.g., reconstruction errors) using a threshold defined by a specific quantile of a statistical distribution.

For each algorithm, we evaluated the F1-score, the false alarm rate, and the missed alert rate. The obtained results are presented in Table 3.

Table 3. Determination of the threshold.

The algorithm	F1-score	FAR	MAR	Best threshold
Area Under the Curve (AUC)	0.6337	0.20	0.39	0.1127
Best-F-score	0.6232	0.28	0.42	0.1112
Top-k	0.6362	0.25	0.50	0.1083
Quantile (65%)	0.6585	0.30	0.63	0.1571

The best threshold was determined using the Area Under the Curve (AUC) methodology; this produced a value of 0.1127. At this threshold, the resulting model achieves an F1 score of 0.6337, a false positive rate of 0.20, and a false negative rate of 0.39. This methodological choice is particularly suitable for our context, as not only does it enable the detection of anomalies in an effective way, but it also decreases the occurrence of false alarms, which is a critical criterion for the performance of a detection system. Figure 3 shows the performance metrics obtained using the AUC-based thresholding method.

We applied the LSTM-AE, LSTM-VAE, and Vanilla-AE benchmark algorithms from the SKAB dataset in our experiments. Subsequently, we employed our autoencoder for anomaly detection without any update mechanism and then evaluated our "Update Based Prototypes model." For each model, the F1 score, missed alert rate, false alarm rate, and execution time were calculated. Table 4 presents the results. The results confirm the effectiveness of the proposed solution in reliably detecting anomalies.

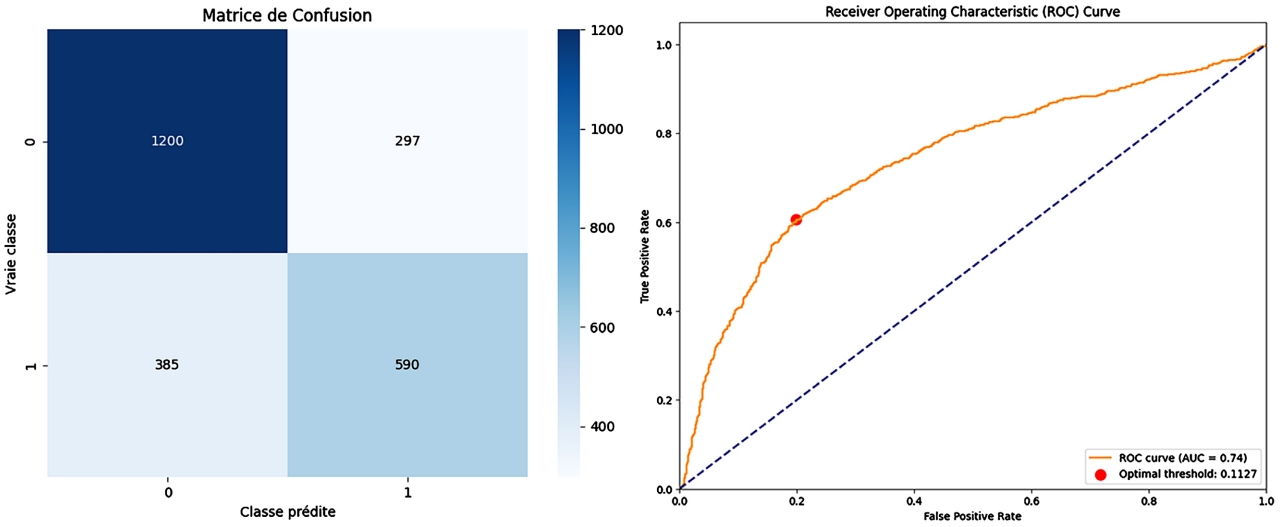


Figure 3. Metrics of the Area Under the Curve (AUC) thresholding method.

Table 4. Results of the experiments.

The algorithm	F1-score	FAR (%)	MAR (%)	Running time (s)
LSTM-AE	0.74	29.96	25.92	520
LSTM-VAE	0.56	9.13	55.03	509
Vanilla AE	0.39	2.59	75.15	415
VAE without update	0.59	48.8	21.85	0.6
Update Based Prototypes-v1	0.63	20.8	20.6	638

5.4. Explainability with ARCANA-Prototypes-v1

To evaluate the explanatory power of our approach, we conducted a series of experiments using the ARCANA and ARCANA-Prototypes-v1 models, then compared their results. The main objective of these experiments was to examine the fidelity and stability of the explanations provided by the models in order to assess their reliability in interpreting predictions. Two types of experiments were performed:

- Characteristic masking test: a specific characteristic was set to zero to check whether the model could identify it as the cause of the detected anomaly.
- Causal characteristic elimination test: characteristics previously identified as responsible for the anomaly were set to zero to confirm that the observation returned to normal in their absence.

The results show that ARCANA-Prototypes-v1 provides stable and reliable explanations, clearly highlighting the most relevant characteristics causing the detected anomalies.

5.4.1. Setting a Characteristic to Zero

For this first experiment, we set the Pressure feature to zero as illustrated in Figure 4, to test whether the explainability models could correctly identify its absence. This modification influences the prediction of the model.

We therefore executed our two algorithms, ARCANA and ARCANA-Proto-

type-v1, twice. During the first run, the explanations suggest that ARCANA detects the 1RMS accelerometer and temperature as the main characteristics associated with the anomaly, while pressure is one of the least influential. In contrast, ARCANA-Prototype-v1 highlights pressure as a key characteristic in detecting the anomaly, demonstrating its ability to recognize its central role. These results are illustrated in Figure 5(a). During the second execution, both algorithms give more importance to the Pressure feature, as shown in Figure 5(b).

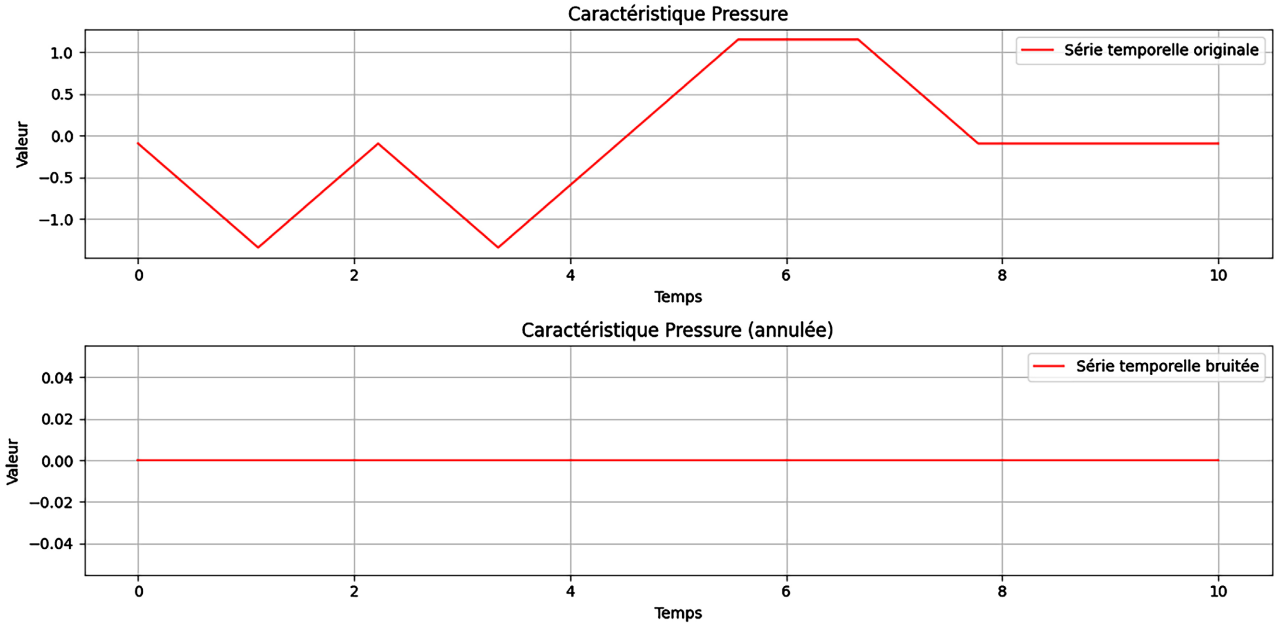
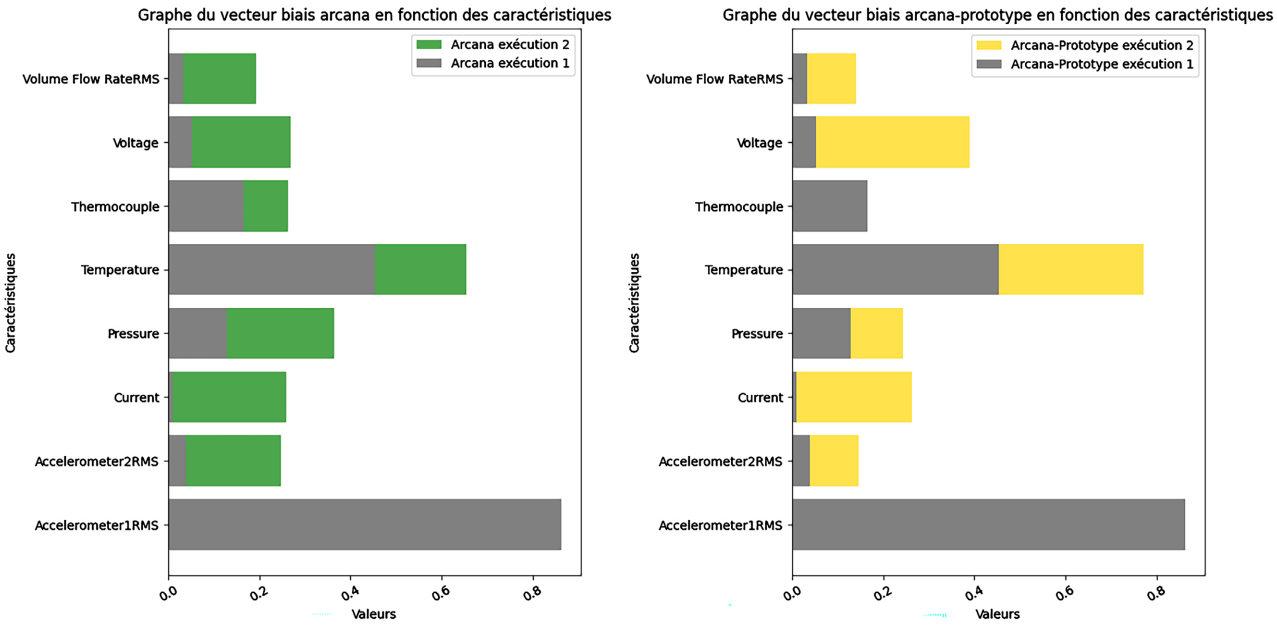


Figure 4. Setting the Pressure feature to zero.



(a) First experiment

(b) Second experiment

Figure 5. Experiment on setting the "Pressure" feature to zero.

5.4.2. Reset Identified Characteristics

For this experiment, anomalous data points correctly classified by the model were selected, and their explanations were generated. The influential features, namely Temperature and Pressure, were then set to zero to simulate drift and assess the impact of this modification on the model's predictions, as illustrated in Figure 6.

We see that, in both iterations of the experiment, ARCANA produces the same predictions, as shown in Figure 7. This is consistent and indicates that ARCANA is not capable of effectively handling concept drift. In contrast, resetting the Temperature and Pressure characteristics resulted in a change in the prediction of ARCANAPrototype-v1. However, the unexpectedly high importance assigned

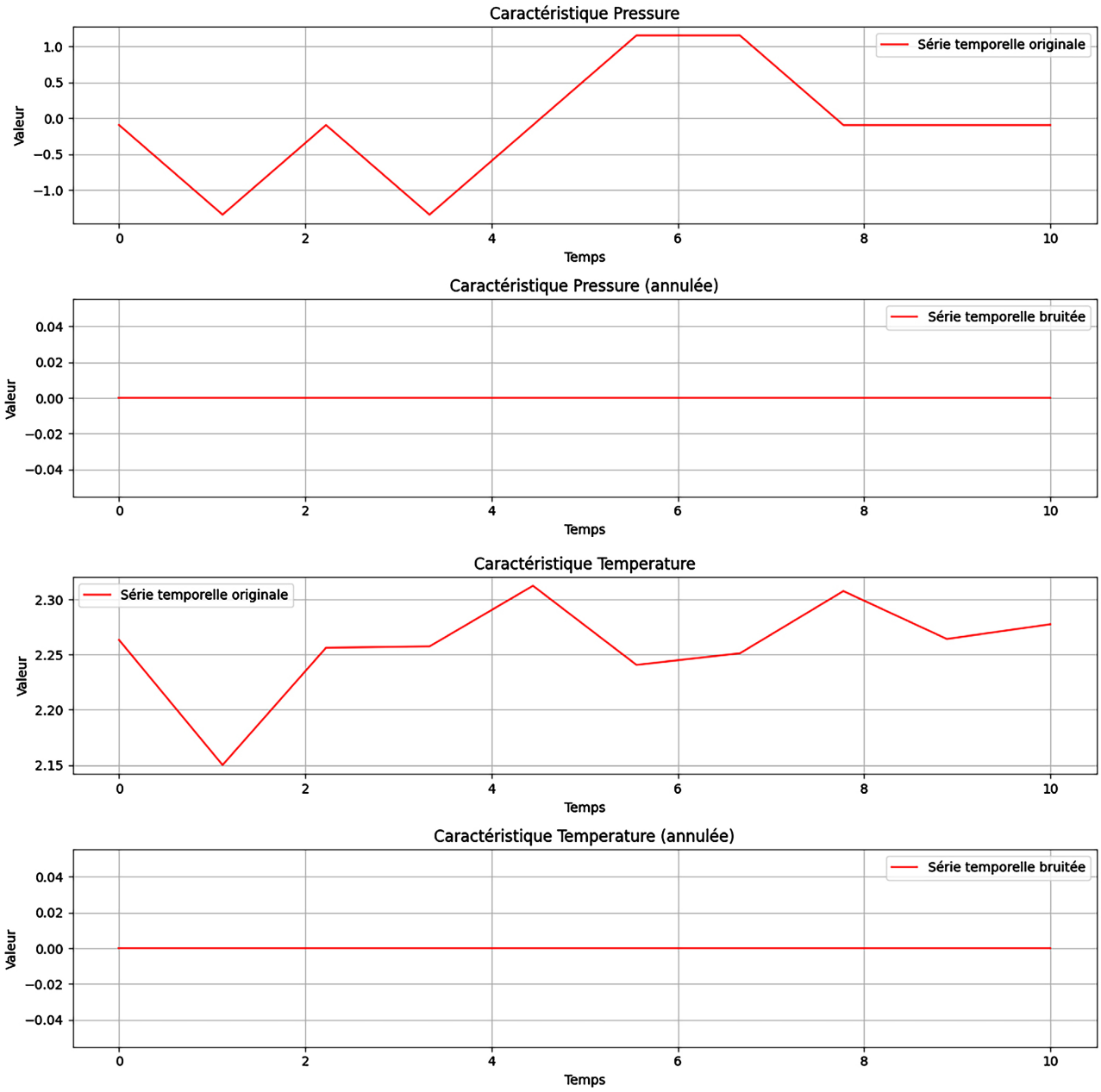
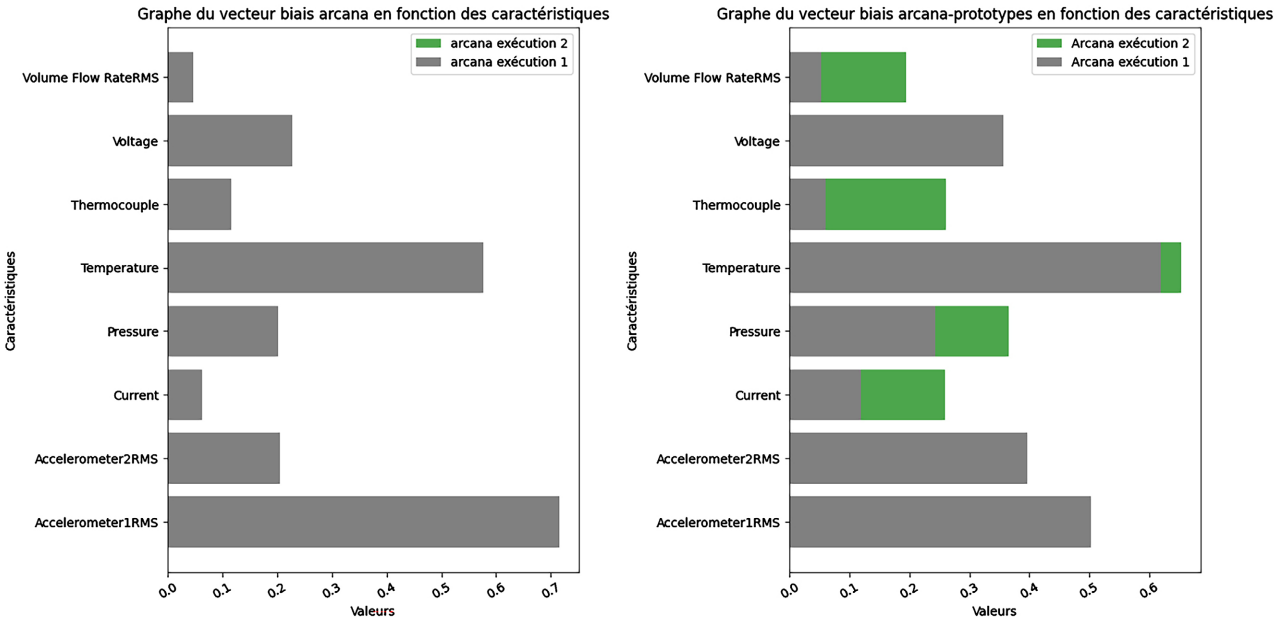


Figure 6. Setting the "Temperature" and "Pressure" features to zero.



(a) First experiment

(b) Second experiment

Figure 7. Zeroing experiment of the "Temperature" and "Pressure" features.

to Thermo Couple, Volume Flow RateRMS, and Current suggests a degree of instability in the model's explainability behavior.

To assess the stability of our model, we use the mean relative variation, which is a simplified application of the local robustness (Lipschitz) score that measures the variation between two nearby points to ensure that the anomaly detection mechanism converges fairly stably towards the responsible variables.

The mean relative variation is computed using Equation (7).

$$\text{VRM} = \frac{\left|\text{Value Ex } 2 - \text{Value Ex } 1\right|}{\text{Value Ex } 1} \tag{11}$$

where Value Ex 1 and Value Ex 2 represent the values of the features that were canceled and those identified by the model as causing the anomaly in the first and second runs, respectively. Table 5 summarizes the mean variations obtained across both experiments.

Table 5. Results of the average relative variation.

Experiments	Pressure	Temperature
Experiment 1	84.6%	
Experiment 2	65.2%	6.4%

We note that in the two experiments for the Pressure feature, we observe VRMs of 84.6% and 65.2%, respectively. These variations reflect an amplification of the model's response to the detected drift. On the other hand, in the second experiment, we observe a VRM of 6.4%, indicating the slight instability of our model. Nevertheless, local robustness is confirmed by the maintenance of the variables

among the top contributors to anomalies during the runs.

In summary, our experiments have enabled us to compare the performance and explainability of the two models. The variations observed over the iterations show how important it is for explainability methods to be robust and stable in order to detect anomalies. These results naturally lead to the next section, which presents the conclusions and practical implications of this study.

6. Conclusions

In this work, we studied anomaly detection and explainability in a data stream. We present a method that is based on two prototypes: Update Based Prototypes-v1 to detect anomalies dynamically and ARCANA Prototypes-v1 to provide coherent explanations for the detected anomalies.

The experiments show that the Update Based Prototypes-v1 method can identify 80% of anomalies in the SKAB dataset with 20% false alarms, compared to 49% for an updated autoencoder. This indicates that dynamic updates to prototypes enable the model to achieve a high level of detection even when new data properties are present or new types of anomalies are encountered.

ARCANA-v1 prototypes provide reliable explanations based on data from the central representation. This allows anomalies to be detected and model predictions to be justified. However, we observe that the stability of explanations can vary from one iteration to another. Hence, there is a need to improve the robustness of the model in dynamic data flows.

In conclusion, the work undertaken improves the performance of decision assistance tools in the presence of anomalies. Indeed, it enables them to be both detected and interpreted. Improving the robustness of the model will be an important contribution to real-time monitoring systems, particularly in situations where the data distribution is constantly changing.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Agrahari, S. and Singh, A.K. (2022) Concept Drift Detection in Data Stream Mining: A Literature Review. *Journal of King Saud University—Computer and Information Sciences*, **34**, 9523-9540. https://doi.org/10.1016/j.jksuci.2021.11.006

[2] Aguiar, G., Krawczyk, B. and Cano, A. (2023) A Survey on Learning from Imbalanced Data Streams: Taxonomy, Challenges, Empirical Study, and Reproducible Experimental Framework. *Machine Learning*, **113**, 4165-4243. https://doi.org/10.1007/s10994-023-06353-6

[3] An, J. and Cho, S. (2015) Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability. https://api.semanticscholar.org/CorpusID:36663713

[4] Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J. (2002) Models and Issues in Data Stream Systems. *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Madison, 3-5 June

2002, 1-16. https://doi.org/10.1145/543613.543615

[5] Belay, M.A., Blakseth, S.S., Rasheed, A. and Salvo Rossi, P. (2023) Unsupervised Anomaly Detection for IoT-Based Multivariate Time Series: Existing Solutions, Performance Analysis and Future Directions. *Sensors*, **23**, Article 2844. https://doi.org/10.3390/s23052844

[6] Chong, P., Cheung, N., Elovici, Y. and Binder, A. (2022) Toward Scalable and Unified Example-Based Explanation and Outlier Detection. *IEEE Transactions on Image Processing*, **31**, 525-540. https://doi.org/10.1109/tip.2021.3127847

[7] Pinon, N., Trombetta, R. and Lartizien, C. (2023) Détection d'anomalies dans l'espace image ou l'espace latent d'auto-encodeurs par patch pour l'analyse d'images industrielles. *GRETSI* 2023: *XXIXème Colloque Francophone de Traitement du Signal et des Images*, Grenoble, 28 Auguest-1 September 2023. https://hal.science/hal-04147526

[8] Li, L., Yan, J., Wang, H. and Jin, Y. (2021) Anomaly Detection of Time Series with Smoothness-Inducing Sequential Variational Auto-encoder. *IEEE Transactions on Neural Networks and Learning Systems*, **32**, 1177-1191. https://doi.org/10.1109/tnnls.2020.2980749

[9] Erniyazov, S., Kim, Y., Jaleel, M.A. and Lim, C.G. (2024) Comprehensive Analysis and Improved Techniques for Anomaly Detection in Time Series Data with Autoencoder Models. *International Journal on Advanced Science, Engineering and Information Technology*, **14**, 1861-1867. https://doi.org/10.18517/ijaseit.14.6.20451

[10] Walczyna, T., Jankowski, D. and Piotrowski, Z. (2024) Enhancing Anomaly Detection through Latent Space Manipulation in Autoencoders: A Comparative Analysis. *Applied Sciences*, **15**, Article 286. https://doi.org/10.3390/app15010286

[11] Bouman, R. and Heskes, T. (2025) Autoencoders for Anomaly Detection are Unreliable. arXiv: 2501.13864.

[12] Pandiselvi, T., Nagavani, C. and Vijaya Barathy, J. (2025) Leveraging Autoencoders for Anomaly Detection in Sensor Data from Critical Infrastructure. In: Muthu, S., Mohan Kumar, G., Kumar, D. and Gurumoorthi, E., Eds., *Deep Neural Networks for Predictive Analytics and Proactive Decision-Making in Securing Critical Infrastructure*, RADemics Research Institute, 191-241. https://doi.org/10.71443/9788197933684-08

[13] Roelofs, C.M.A., Lutz, M., Faulstich, S. and Vogt, S. (2021) Autoencoder-Based Anomaly Root Cause Analysis for Wind Turbines. *Energy and AI*, **4**, Article ID: 100065. https://doi.org/10.1016/j.egyai.2021.100065

[14] Assaf, R., Giurgiu, I., Pfefferle, J., Monney, S., Pozidis, H. and Schumann, A. (2020) An Anomaly Detection and Explainability Framework Using Convolutional Autoencoders for Data Storage Systems. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Yokohama, 11-17 July 2020, 5228-5230. https://doi.org/10.24963/ijcai.2020/752

[15] Dollon, Q., Labbé, P. and Léonard, F. (2023) Promoting Explainability in Data-Driven Models for Anomaly Detection: A Step toward Diagnosis. *Annual Conference of the PHM Society*, **15**, 1-11. https://doi.org/10.36001/phmconf.2023.v15i1.3509

[16] Cheng, H., Xu, D. and Yuan, S. (2023) Explainable Sequential Anomaly Detection via Prototypes. 2023 *International Joint Conference on Neural Networks* (*IJCNN*), Gold Coast, 18-23 June 2023, 1-8. https://doi.org/10.1109/ijcnn54540.2023.10191703

[17] Levshun, D.A., Levshun, D.S. and Kotenko, I.V. (2025) Detecting and Explaining Anomalies in Industrial Internet of Things Systems Using an Autoencoder. *Ontology of Designing*, **15**, 96-113. https://doi.org/10.18287/2223-9537-2025-15-1-96-113

[18] Katser, I.D. and Kozitsin, V.O. (2020) Skoltech Anomaly Benchmark (SKAB). https://doi.org/10.34740/KAGGLE/DSV/1693952

[19] Garg, A., Zhang, W., Samaran, J., Savitha, R. and Foo, C. (2022) An Evaluation of Anomaly Detection and Diagnosis in Multivariate Time Series. *IEEE Transactions on Neural Networks and Learning Systems*, **33**, 2508-2517. https://doi.org/10.1109/tnnls.2021.3105827