This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2025.3644604

A Representation Learning Approach to Feature Drift Detection in Wireless Networks

Athanasios Tziouvaras *, Blaž Bertalanič †, George Floros ‖**, Kostas Kolomvatsos §, Panagiotis Sarigiannidis‡ and Carolina Fortuna†
* Business and IoT Integrated Solutions LTD, Nicosia, Cyprus
† Jožef Stefan Institute, Slovenia
‖ Trinity College Dublin, Dublin, Ireland
** University of Thessaly, Volos, Greece
§ University of Thessaly, Lamia, Greece
‡ University of Western Macedonia, Kozani, Greece
attziouv@bi2s.eu, blaz.bertalanic@ijs.si, florosg@tcd.ie, kostasks@uth.gr, psarigiannidis@uowm.gr, carolina.fortuna@ijs.si

***Abstract*—Artificial Intelligence (AI) is foreseen to be a centerpiece in next generation wireless networks enabling ubiquitous communication as well as new services. However, in real deployment, feature distribution changes may degrade the performance of AI models and lead to undesired behaviors. To counter for undetected model degradation, we propose ALERT; a method that can detect feature distribution changes and trigger model re-training that works well on two wireless network use cases: wireless fingerprinting and link anomaly detection. ALERT includes three components: representation learning, statistical testing and utility assessment. We rely on Multi-layer Perceptron (MLP) for designing the representation learning component, on Kolmogorov–Smirnov (KS) and Population Stability Index (PSI) tests for designing the statistical testing and a new function for utility assessment. We show the superiority of the proposed method against ten standard drift detection methods available in the literature on two wireless network use cases.**

***Index Terms*—feature drift detection, machine learning, artificial intelligence, wireless networks, fingerprinting, link anomaly detection**

I. Introduction

Artificial Intelligence (AI) is foreseen to be a centerpiece in next generation wireless networks, including 6th Generation Wireless Cellular Networks (6G) and beyond cellular networks [1] by enabling ubiquitous communication, new services including high-accuracy localization [2], anomaly fault and detection [3] as well as replacing traditionally networking functionality by AI based realization towards so-called AI native functionality [4], [5]. AI models are typically developed offline by using a pre-defined amount of data and a set of Machine Learning (ML) techniques that are tuned (semi-)manually to find the best performing combination for the respective training data [6]. However, when deployed in a real, production environment, the model development workflow is managed by the so-called AI/ML workflow [7] realized through Machine Learning Operations (MLOps) tools [8]. The combination of those tools and their deployment enable MLOps pipelines that automatically manage the data preparation, model training, evaluation, selection and serving in production systems.

Once deployed in a production setting, the data collection, MLOps pipeline and integrated AI models are typically managed by different teams, sometimes without significant coordination between each other [9]. In such setting, it may happen that input data distribution changes occur naturally due to changes in the observed systems, but it may also happen that unstable data dependencies such as re-calibrations done by the team responsible for data collection is not propagated to the teams managing the MLOps or AI systems. Therefore, the performance in production degrades and adjustments are reactive rather than proactive [9]. A recent study across several hundreds of eNodeBs and three categories of wireless KPIs such as resource utilization has also confirmed the existence of drifts in cellular networks [10] while [11] identified the challenges surrounding the implementation of drift detection and mitigation schemes in resource-constrained networks.

To detect and signal distribution changes that may degrade the performance of AI in production, libraries able to detect feature drifts while integrating with existing MLOps tools have been developed [12]. These libraries incorporate several *drift detectors*, defined as methods that observe a stream of data over time and determine for every new data point if the current distribution of the data has changed compared to a reference data set [13]. Several drift detection techniques as part of three such libraries have been recently benchmarked on two use-cases: occupancy detection and prediction of energy consumption [14]. To date, the only investigations of the drift phenomenon on wireless data are available in [10] including a Kolmogorov-Smirnov based detection technique and [11] that considered Isolation Forests and threshold to detect drifts in an illustrative example.

Aiming to provide a better insight into the suitability of existing drift detection techniques on wireless data as well as improve the existing state of the art for detection in wireless networks, we propose a new feature drift detection method (named ALERT), and benchmark it against ten standard methods on two use cases: wireless fingerprinting and link fault or anomaly detection. The contributions of this paper are:

- ALERT, a new feature drift detection method consisting of three components: representation learning, statistical

testing and utility assessment.
- Validation on two wireless Use Cases that utilize real-world data. We show that the ALERT method outperforms all the baseline models, achieving an overall F1-score of 0.9 in the *fingerprinting* use case and 0.88 in the *links* use case.
- Analysis (i) identifying feature drift; (ii) assessing their impact on the model; (iii) attempting to answer "when" to retrain the model with the new data.

This paper is organized as follows. Section II summarizes related work, Section III provides background related to drift detection, Section IV provides the problem statement while Section V introduces the proposed ALERT method. Section VI details the evaluation methodology Section VII while Section VIII concludes the paper.

II. Related work

Data and concept drift are sometimes interchangeably used in the literature while in some cases one is considered as a superset of the other. In this paper we follow the definition from [15] where *feature drifts* are categorized into four primary types: *covariate drift* also referred to a feature drift in this paper, *prior probability drift*, *concept drift*, and *dataset shift*. We group related works in three categories: works that develop new or analyze existing drift techniques, works that develop drift detection tools and systems and works that focus on studying specific use-cases.

A. Drift Detection Techniques

Most of the drift, or change, detection techniques can classified as follows based on the type of performed analysis: 1) sequential analysis, 2) control charts, 3) difference between distributions and 4) contextual as discussed in [16] with [17] providing a different grouping. One of the foundational sequential analysis test is the Sequential Probability Ratio Test (SPRT) that detects at a point p a change from a distribution to another. Other tests such as Cumulative Sum (CUMSUM) use principles from SPRT. The Page-Hinkley (PH) test is a sequential adaptation of the detection of an abrupt change in the average of a Gaussian signal [16]. The methods from the second category are based on based on statistical process control represented by standard statistical techniques to monitor and control the quality of a product during a continuous manufacturing. One such example is the exponentially weighted moving average (EWMA) [16].

The methods from the third group, that monitor distributions on two different time-windows, compare the two distributions computed over the two windows using statistical tests and signal a drift when the distributions are not equal. The Kullback-Leibler (KL) divergence test, the Population Stability Index (PSI), a variation of the KL [18], as well as ADaptive WINdowing (ADWIN) all fall under this category [16]. Some other statistical tests as follow probably also fall in this group. The Energy Distance (ED) [19] that computes a statistical distance between two probability distributions, the Earth Mover's Distance (EMD) [20] that computes the minimal cost that must be paid to transform one distribution into the other, the Kolmogorov-Smirnov test (KS) the quantifies the distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples, the Kuiper test that is closely related to KS, etc [21].

Finally, the contextual detectors rely on learning, with examples such as the Splice that is a meta- learning technique that implements a context sensitive batch learning approach and the Incremental Fuzzy Classification System algorithm.

B. Drift Detection Tools and Systems

The authors in [14] proposed D3Bench, a benchmarking tool that enables the functional and non-functional evaluation of drift detection tools. In their analysis they benchmark three open source tools for drift detection and found that Evidently AI stands out for its general feature drift detection, whereas NannyML excels at pinpointing the precise timing of shifts and evaluating their consequent effects on predictive accuracy. The authors of [22] start from the observation that observe that not all feature drifts lead to degradation in prediction accuracy and propose a new strategy which, using decision trees, is able to precisely pinpoint low-accuracy zones within ML models. The work triggers model improvement through active learning only in cases of harmful drifts that detrimentally affect model performance. Rather than triggering model retraining when drift is noticed, [23] proposed Matchmaker, a tool that dynamically identifies the batch of training data that is most similar to each test sample, and uses the ML model trained on that data for inference.

C. Drift Detection Use Cases

The authors in [14] benchmarked three drift detection tools on univariate data falling under two use-cases: occupancy detection where CO2 and room temperature were used as features while occupancy was the target variable and energy consumption prediction where energy consumption was the feature. The authors of [24] study the impact of industrial delays when mitigating distribution drifts on a financial use-case. Focusing on cellular wireless data, [10] introduce a methodology for concept drift mitigation that explains the features and time intervals that contribute the most to drift; and mitigates it using forgetting and over-sampling. An illustrative demand prediction use case for multimedia service in a 5G network was briefly considered in [11]. Isolation Forests and threshold were used to conceptually illustrate drifts detection.

III. Drift Definition

As briefly mentioned in Section II, the terminology related to data, concept and model drift varies across works. The formal mathematical definitions are generally similar in [17], [25] and [24], however, in the remainder of the paper we will align with the terminology from [15]. Assume a model M_i is trained to fit a dataset $D_i = \{d_0, d_2, d_3, ..., d_j\}$, where $d_j = \{X_k, y_k\}$. In this sense, $\{d_0, d_2, d_3, ..., d_j\}$ represent the data points of the dataset D_i, X_k represents the feature vector

and y_k represents the label for the corresponding data point d_j. Evidently, since D_i can be described under a distribution $F_{0,j}(X, y)$, M_i learns to identify this distribution through the model training process. Drift can occur when new data points are inserted in D_i, namely $d_{j+1}, d_{j+2}, d_{j+3}, ..., d_{j+n}$, if $F_{0,j}(X, y) \neq F_{j+1,\infty}(X, y)$. For this inequality to hold true, there should be a j that satisfies the following inequality: $P_j(X, y) \neq P_{j+1}(X, y)$. Since $P_j(X, y) = P_j(X) \times P_j(y|X)$, we can rewrite the drift equation as follows:

$$\exists j : P_j(X) \times P_j(y|X) \neq P_{j+1}(X) \times P_{j+1}(y|X) \quad (1)$$

Following Eq. 1, and in line with [15], the four types of drifts are as following:

1) The covariate shift [15] or drift [24], also referred to as source 1 concept drift in [17], feature drift in [26], is observed when $P_j(X) \neq P_{j+1}(X)$, while $P_j(y|X) = P_{j+1}(y|X)$. In such cases, the feature distribution changes, when new data $\{d_{j+1}\}$ are entered into the D_i dataset, thus the reason we refer to *covariate drift* also as *feature drift* in this paper.
2) The prior probability shift [15] or drift [24] phenomenon can be identified when $P_j(x|Y) = P_{j+1}(X|y)$, while $P_j(X) \neq P_{j+1}(X)$. In this case, the label distribution changes, while in the case of the covariate drift, the distribution of the features changed.
3) The concept shift [15], drift [24] or source 2 concept drift [17] phenomenon can be identified when $P_j(y|X) \neq P_{j+1}(y|X)$, while $P_j(X) = P_{j+1}(X)$. In this case, the relationship between the labels and features changes. In [15], it is additionally also defined as when $P_j(X|y) \neq P_{j+1}(X|y)$, while $P_j(y) = P_{j+1}(y)$.
4) Dataset shift [15] or source 3 concept drift [17] is combination of covariate drift and concept drift and occurs when $P_j(y|X) \neq P_{j+1}(y|X)$ and $P_j(X) \neq P_{j+1}(X)$. This phenomenon requires both the data distribution and the feature-data mapping to change.

IV. Problem Statement

In this paper, we focus on feature drift as defined in Section III and assume a model M_0 is trained on a dataset D_0, which we call the *original training dataset*. Given a *new dataset* D_1, our goal is to: **(i)** asses the existence of the feature drift phenomenon; **(ii)** estimate its effects on the M_0 model performance and **(iii)** decide whether the M_0 model should be re-trained with the D_1 dataset in order to increase its quality.

For M_0, we consider two different datasets that correspond to the two distinct validation scenarios we employ in this work. Both validation scenarios leverage supervised classification tasks, one performed on a multivariate dataset (named *fingerprinting*) collected from the LOG-a-TEC testbed [27] and one implemented over a univariate wireless dataset (named *links*) collected from the Rutgers WinLab testbed with synthetically injected anomalies/faults [3].

The labels of the *fingerprinting* dataset consist of discrete measurement positions in a grid and represent the location of the BLE transmitter. The dataset was collected using the LOG-a-TEC testbed in two different seasons: winter and spring. It

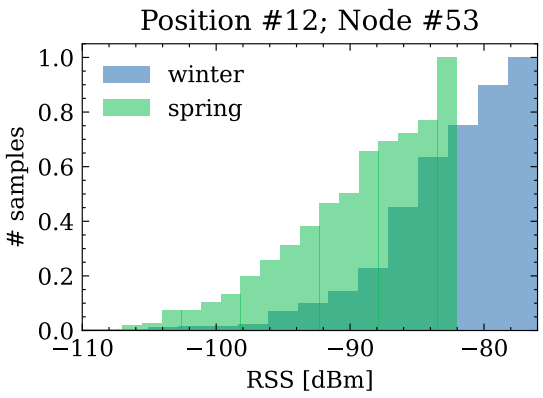

Fig. 1. An example of feature drift between winter and spring data in the *fingerprinting* dataset.

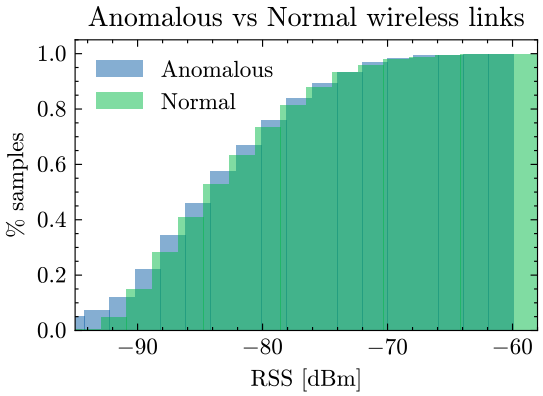

Fig. 2. An example of feature drift between Normal and Anomalous wireless links in the *links* dataset.

comprises of Received Signal Strenght (RSS) data from 25 BLE nodes deployed outdoors in a campus park, with nodes mounted on light poles and building walls at varied heights. In the experiment, a BLE transmitter broadcasted signals every 100 ms across a localization grid with each grid point sampled for about one minute. The data was gathered in a realistic environment with natural ambient interference. Figure 1 represents the distribution of collected RSS measurements at node #53 in localization position #12 for both winter (blue bars) and spring (green bars) data. Although the two histograms partially overlap, it can be seen that the winter data range is between -102 and -78 dBm, while the range for the spring data is between -108 and -82 dBm. This shift, or feature drift, illustrates how environmental factors can alter signal propagation between seasons even when measurements are taken at the same location and from the same transceiver pair. In this example, the testbed area is abundant with trees, bushes, and other vegetation that is fully leafed in the spring and mostly bare in the winter. The difference in foliage between seasons leads to variations in signal propagation, as the dense vegetation in spring can cause additional attenuation of the signals compared to the winter, while the absence of leaves results in less signal interference.

The *links* dataset is a univariate wireless dataset with synthetically injected anomalies. As presented by authors in [3],

there are 4 common types of anomalies that can be observed in wireless link layer monitoring. As mentioned by the authors, these anomalies are rare events that can indicate different causes, such as a broken wireless nodes, software issues, or a slowly dying nodes. Figure 2 depicts the distribution of RSS values for Anomalous (blue bars) and Normal (green bars) wireless links. As it can be seen from the figure, there is a significant overlap between both type of links, with really subtle difference between the two. The Anomalous values range between -95 to -60 dBm, while Normal values range between -92 and -58 dBm.

We consider that the M_0 model is trained on the D_0 dataset (which can be either *fingerprinting* or *links*), and then it is deployed in a production environment where new data points (D_1) correspond to the location or type of anomaly respectively. As feature drift appears, through D_1, the M_0 responses decrease in quality. As we run a controlled experiment in which we also have labels for D_1, we can measure the actual decrease in performance. However, in a real production set-up, D_1 is a dataset which is yet to be labeled, therefore we have to develop a way to detect the feature drift in a reliable way without relying on labels.

As no labels for D_1 are available in production setting, detecting changes between D_0 and D_1 using techniques such as discussed in Section II-A to measure distribution changes (i.e., perform statistical tests) seems the most suitable approach. Then, we can verify which technique is the most suitable for the considered use cases.

V. Feature drift detection using the ALERT method

In this work we propose ALERT, a new feature drift detection method that rather than monitoring the distribution shift of the raw data or traditionally engineered features, it monitors the shift of a learnt representation (or embedding). The intuition behind ALERT is that the learnt representations tend to be lower dimensional and contain less noise making the subsequent distribution change computation faster and more accurate. ALERT includes three components: representation learning, statistical testing and utility assessment. Figure 3 depicts the proposed method detailing the representation learning component.

A. Design of the Representation Learning Component

We employ a supervised approach to learn the representation of D_0 through a lightweight Multi-Layer Perceptron (MLP) rather than relying on more complex and computationally expensive and data-hungry Convolutional Neural Network (CNN) or transformer-based architectures.

Supervised representation learning: The representation is learnt by training an MLP on D_0, the original dataset including the labels as depicted on the top left of Figure 3. Essentially the MLP is a network of feed-forward layers where each layer consists of a number of neurons. The output y_i of the i-th neuron can described by the following equation:

$$y_i = \phi_i\left(\sum\left(W_i X_i\right) + B_i\right) \tag{2}$$

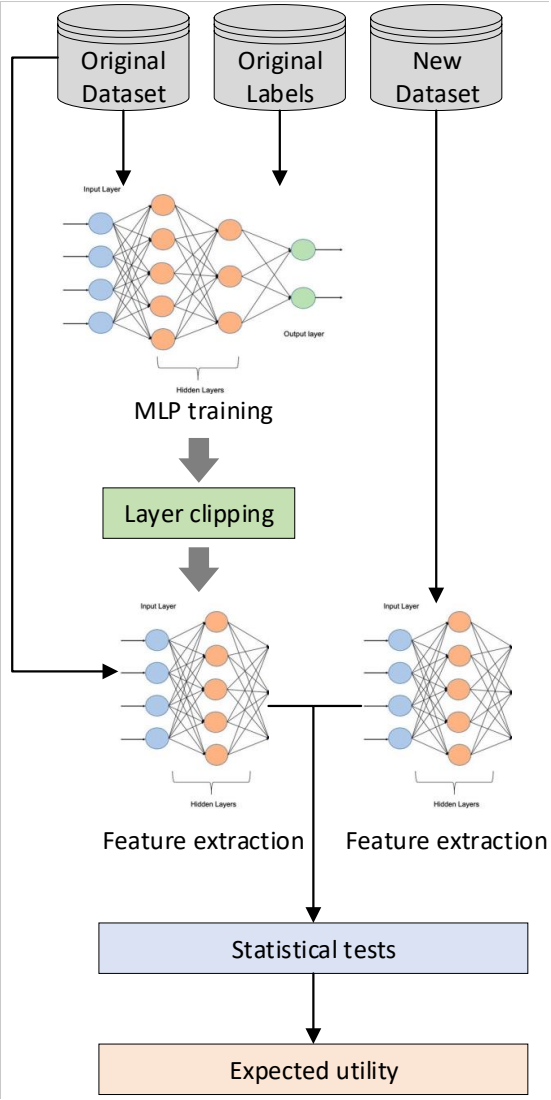

Fig. 3. The proposed ALERT methodology for assessing the feature drift given 2 datasets D_0 and D_1.

which is a linear combination of its inputs X_i, weights W_i and biases B_i where ϕ_i is the activation function of the neuron. Then each layer concatenates the outputs of all its neurons y_i into a vector $Y = y_1, y_2, ..., y_n$, which is forwarded to the next layer.

Extracting the feature representation: For this process, we utilize the trained MLP and we clip its lower layers as depicted in the mid area of Figure 3. We opt to discard the lower layers of the MLP, since the upper layers tend to capture higher-level features [28] and thus, they can provide useful information related with the input data distribution. In the sequel, we perform two forward passes using the clipped MLP model: one using the D_0 and one using the D_1 dataset. These forward passes do not update the weights of the model and they extract two sets of features: the R_0 feature set which is extracted from the D_0, and the R_1 feature set which is extracted from the D_1 dataset. The extracted feature representations contain useful information that can be leveraged by the utility function designed in the next module.

B. Design of the Statistical Testing Component

For the statistical testing depicted in the blue box on the lower side of Figure 3, we opt to use 2 well-established methods (KS [29], [30] and PSI) [18], instead of relying on a single one. This decision aims to: **(i)** reduce the input parameters of ALERT, as much as possible; **(ii)** cancel out the weakness of KS and PSI by aggregating their outcomes; **(iii)** reinforce ALERT with the capacity of detecting both small and medium distribution drifts; **(iv)** introduce sampling symmetry to ALERT; and **(v)** stabilize and simplify ALERT's outputs so that to be easily usable and interpretable by decision-making mechanisms.

Rather than relying on pre-set input parameters or prior knowledge regarding the (Probability Density Function (PDF)s), KS is able to internally estimate the distribution characteristics of the input data. As a result, KS swiftly detects small distribution changes (even in large datasets) and produces a p-value that can be easily interpreted by external systems [31]. However, KS is prone to false positives and is asymmetric thus, if the user swaps the baseline and sample distributions it will produce different outcomes. Also, it requires a large number of samples to accurately function and may struggle with non-normal distributions [32]. PSI, on the other hand, can perform well, even if a lower amount of data is used [33] and is considered a stable and robust measure to assess drifts even when non-normal distributions are considered. Also, PSI produces interpretable outputs, is very good at identifying moderate or larger drifts, and is fully symmetric. On the flip side, it requires the data binning parameter to be set externally, which separates the input data into predefined intervals. The combination of KS and PSI provides a unified framework that cancels out the weaknesses of each method and increases the robustness of ALERT. The importance of both KS and PSI is further discussed in Section VII-E, in which we perform an ablation study and we showcase the individual contributions of KS and PSI to the overall ALERT's utility score. There, it is evident that if the KS or the PSI component is removed, the performance of ALERT drops by a large margin. To accomplish the unification of KS and PSI, ALERT computes the following statistical tests:

- The $KS_{(D_0,D_1)}$ that represents the KS between the dataset D_0 and the dataset $D1$.
- The $KS_{(R_0,R_1)}$ that represents the KS between the extracted features R_0 and the extracted features R_1.
- The $PSI_{(D_0,D_1)}$ that calculates the PSI between the dataset D_0 and the dataset D_1.
- The $PSI_{(R_0,R_1)}$ that calculates the PSI between the extracted features R_0 and the extracted features YR_1.

KS Statistical Test: The KS is invoked to check if two sets of samples belong to the same distribution. To assess this, the test utilizes a p-value which designates that the samples belong to different distributions if $p < 0.05$. We calculate this p-value similarly with [29], as follows:

$$p_{x,y} = 2\sum_{i=1}^{z}\left((-1)^{i-1}\cdot e^{-2c^2(a)\cdot i^2}\right) \tag{3}$$

where z is the total number of samples, x and y are the corresponding sets that are being checked, and the $c(a)$ can be calculated through the following formula:

$$c(a) = D_{x,y}\sqrt{\frac{n_x \cdot m_y}{n_x + m_y}} \tag{4}$$

where n_x and m_y is the number of samples of the x and y datasets correspondingly. $D_{x,y}$ is calculated via the following equation:

$$D_{x,y} = \sup_t |F_x(t) - F_y(t)| \tag{5}$$

Where $F_x(t)$ and $F_y(t)$ are the empirical distributions of the data belonging in the x and y sets.

PSI Statistical Test: The PSI is used to measure the relative entropy between two distributions. This can be interpreted as the measurement of divergence between two different sets of samples. PSI values that are lower than 0.1 indicate that there is no significant difference between two data distributions. We calculate PSI as suggested by previous work in [18]:

$$PSI_{x,y} = \sum_{i=1}^{z}\left(P(x_i) - P(y_i)\cdot \ln\left(\frac{P(x_i)}{P(y_i)}\right)\right) \tag{6}$$

where z is the number of samples of the x and y data sets, while $P(x_i)$ and $P(y_i)$ represent the frequencies of samples i in the x and y datasets.

C. Design of the Utility Assessment Component

Since our ultimate goal is to provide a decision-making mechanism for when to retrain the M_0 model, we formulate a utility function, depicted in the orange box at the bottom of Figure 3, that combines a KS utility with a PSI utility as follows:

$$U = \frac{U_{KS} + U_{PSI}}{2} \tag{7}$$

$U \in (0, 1)$ encapsulates the final utility we obtain if the model retraining action is selected, given the datasets D_0 and D_1. The expected utility U considers the outputs of KS and KL tests to evaluate the statistical difference of the data D_0 and D_1 and the extracted features R_0 and R_1.

Eq. 7 combines Eqs. 8 and 9 defined as follows into a unified utility function. The KS-based utility for retraining the M_0 model is:

$$U_{KS} = \frac{1 - KS_{(D_0,D_1)} + 1 - KS_{(R_0,R_1)}}{2} \tag{8}$$

Since $KS \in (0, 1)$, U_{KS} is also a bounded function ($U_{KS} \in (0, 1)$). This function uses the information derived from the datasets D_0 and D_1 and averages it with the information extracted from the features R_0 and R_1 to assess the KS drift. Evidently, when $U_{KS} \to 1$ the drift phenomenon is more prominent, while when $U_{KS} \to 0$ no drift is detected.

We also devise a function to calculate the PSI-based utility for retraining the M_0 model:

TABLE I
THE DETAILS OF THE DRIFTED AND NON-DRIFTED DATASETS WHICH ARE CREATED BASED ON THE *fingerprinting* AND *links*.

	fingerprinting	*links*
Data types	BLE measurements	Time series
Total samples	505.000	8.493×302
# Classes	25	5
M_0 training	D_0	D_0
Drifted data	$D_3, D_6, D_9, D_{12}, D_{15}, D_{18}, D_{21}, D_{24}, D_{27}, D_{30}$	D_4, D_5, D_6, D_7, D_8
Non-drifted data	$D_1, D_2, D_4, D_5, D_7, D_8, D_{10}, D_{11}, D_{13}, D_{14}, D_{16}, D_{17}, D_{19}, D_{20}, D_{22}, D_{23}, D_{25}, D_{26}, D_{28}, D_{29}$	D_1, D_2, D_3
Cause of drift	Spring data is mixed with winter data	New anomalies are added to the data

$$U_{PSI} = \sigma\Big(\frac{PSI_{(D_0,D_1)} + PSI_{(R_0,R_1)}}{2}\Big) \tag{9}$$

The U_{PSI} averages the $PSI_{(D_0,D_1)}$ and the $PSI_{(R_0,R_1)}$ and uses the sigmoid function σ to bound the result so that $U_{PSI} \in (0, 1)$. Similarly to Eq. 8, the expected utility of the model retrain operation is higher when $U_{PSI} \to 1$ and lower when $U_{PSI} \to 0$.

VI. EVALUATION METHODOLOGY

A. Dataset Description

For our experiments we focus on two use cases with two datasets: *fingerprinting* dataset summarized in Figure 1, and the *links* dataset summarized in Figure 2.

The *fingerprinting* dataset [27] contains received signal strength (RSS) measurements made with Bluetooth Low Energy (BLE) technology, measured in dBm. The dataset consists of 505.000 data points, organised over 25 classes that represent 2D coordinates, which are collected during the spring and during the winter, as described in Section IV. It can be used for outdoor fingerprint-based localization applications, similarly to [27]. We organise the data into 31 smaller datasets each containing 16.290 samples, as follows: The first dataset D_0, which contains samples collected during the winter season only, is used to train a random forest classifier, as our M_0 model. For this reason, we refer to the D_0 as the *original dataset*, as depicted in Figure 3. Then we edit the rest of the datasets D_1 - $D30$ so that each third dataset contains samples stemming from spring measurements, which are drifted (i.e., D_3, D_4, D_6, D_7, D_9 etc.). The rest of the datasets (i.e., D_2, D_5, D_8, D_{11}, D_{14} etc.) contain samples that are collected during winter and belong to the same distribution with D_0. Through this process, we can simulate various scenarios of feature drifts which are encountered in different time frames i.e., in different versions of datasets collected in the field. Table I summarizes the drift creation details for the *fingerprinting* dataset.

In the sequel, we test the M_0 model's Macro Precision, Macro Recall and Macro F1-score with each created dataset ($D_1 - D_{30}$). As illustrated in Figure 4, the performance of the M_0 drops significantly when it is tested with a dataset that contains drifted samples. This experimental set-up shows the existence of the drift and enables its detection through ALERT and selected baselines.

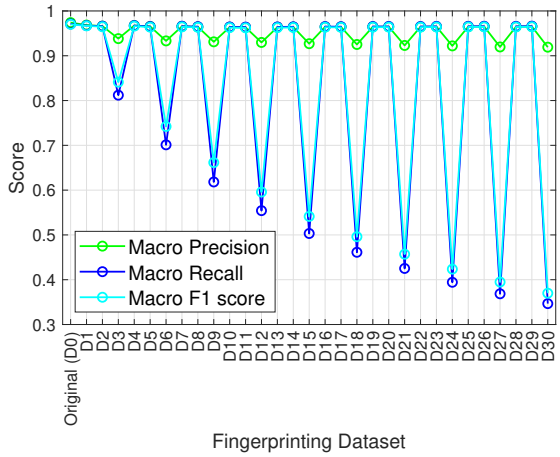

Fig. 4. The performance of the M_0 model with different *fingerprinting* datasets. The performance drops when the feature drift phenomenon is present, since the M_0 is trained with the D_0 dataset.

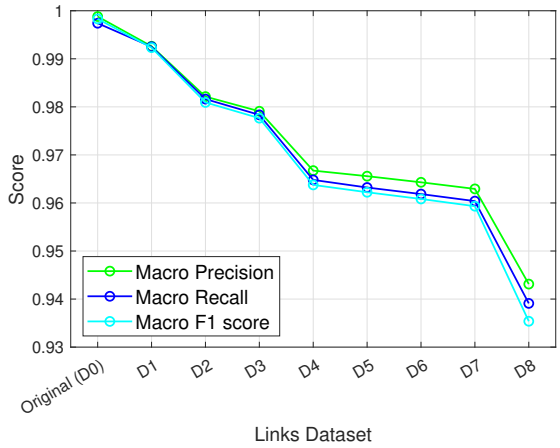

Fig. 5. The performance of the M_0 model with different *links* datasets. The model performance deteriorates when the M_0 is tested with datasets that contain feature drifts.

The *links* dataset contains data from 8492 timeseries, each one of which having 302 data samples. The dataset is labeled and is organized into 5 anomaly classes. We split the data into 9 smaller datasets each one containing 943 timeseries, as follows: the first dataset D_0 is used to train a random forest classifier, as our M_0 model. Similarly to our approach when using the *fingerprinting* case, the D_0 is again the *original dataset*, which is illustrated in Figure 3. Then we split the rest of the datasets D_1 - D_8 so that the D_1, D_2 and D_3 to contain samples from the same distribution as D_0, while the D_4, D_5, D_6, D_7 and D_8 to contain drifted samples, from new anomaly types. Table I summarizes the details of this process. We then test the M_0 model's Macro Precision, Macro Recall and Macro F1-score with each created dataset and we present our findings in Figure 5. We observe that the performance of the M_0 drops when used with drifted samples, similarly to the *fingerprinting* model.

B. Parameter Search for the Representation Learning Components

In order to design an efficient model for representation learning, we should consider tuning the MLP's layers, number of neurons contained in each layer and the number of training epochs. It is our objective to avoid complicated models for the representation learning process, due to the computational and training requirements they would impose during the MLP training operation. For this reason, we try to minimize as much as possible the number of layers, the amount of neurons and the training epochs of the model. On the contrary, we are aware that we risk underfitting if we design the model to be very simple since, in that case it would be unable to capture the representations of the input data. To solve this issue, we perform a 3-dimensional parameter search regarding the number of layers, neurons and training epochs.

C. Baseline Selection

We compare the ALERT technique with state-of-the art methods that exist in the literature. We choose three broader types of methods for comparison, namely statistics-based methods, distance-based methods and ML-based methods. Statistical methods leverage statistical indexes (such as mean values, sampling variations and variance) to predict feature drifts. In this work, we formulate a baseline using the following statistical methods: (i) Kuiper test [34]; (ii) Cramer-Von Mises (CVM) [35]; (iii) Welch Test (WelchT) [36]; (iv) Chi Square test [37]; (v) Mann-Whitney U test (Mann Whitney) [38]; (vi) Andreson Darling Test [39]; and (vii) Kolmogorov-Smirnov (KS) test [40]. On the contrary, distance-based methods focus on estimating the distance between two data distributions by measuring the dissimilarity between them through distance functions. In this work we use as a baseline the following distance-based methods: (i) Population Stability Index (PSI) [41]; (ii) Energy Distance [42]; and (iii) Earth Mover's Distance (EMD) [43]. ML-based methods utilize Deep Learning or Machine Learning models to either form an intermediate representation of the data under investigation, or to estimate the features of data distributions. In this paper we use the following ML-based methods as baselines: (i) Adversarial Detection (ML-AD) [44]; (ii) Embedding-based domain classification (ML-EDC) [45]; and (iii) Data-driven deep Density Estimation (ML-DDE) [46].

D. Training and Evaluation

We use the D_1 - D_{30} stemming from the *fingerprinting* and the D_1 - D_8 stemming from the *links* datasets to evaluate ALERT and to assess whether it is able to identify the feature drift phenomenon accurately. For the implementation of ALERT, we use the python programming language; whereas the code base for the Baseline methods is provided by [44], [45], [46] and [47]. For each validation Use Case, we train the ALERT method's MLP for 3 epochs using the D_0 dataset and then, we utilize the trained model to assess the existence of the feature drift phenomenon.

Since the distance-based and statistics-based methods are provided in a ready-to-use form in GitHub repositories [47], their evaluation is trivial. On the other hand, ML-based methods require some adaptations in order to properly work with our datasets. For this reason, we have performed the following adaptations:

For the ML-AD method: we use the open source code provided by the author [44] and we opt to train a ResNet-28-10 model from scratch to fit the D_0 dataset. Then, we fine-tune the model using the provided Bayesian ensemble method, which utilizes one optimizer (SGD) and one regularizer (Prior Regularizor) in parallel. In the sequel, we save the trained model and test its performance with each new dataset (D_1 - D_{30} for the *fingerprinting* data, or D_1 - D_8 for the *links* data). This approach essentially performs adversarial detection for each new dataset. The outcomes of each test determine if the drift phenomenon is detected. Thus, when the model detects adversarial samples, we assess that the drift phenomenon is present.

For the ML-EDC method: we use an unsupervised representation learning approach for drift detection. More specifically, we train the DriftLens model [45] with our D_0 dataset in order for the model to learn the distribution properties of the input data. Then, we utilise the D_1 dataset to estimate the threshold distance values that discriminate between drift and no-drift conditions, as suggested by the authors. Afterwards, we utilise the rest of the datasets to perform inference of the trained model.

For the ML-DDE method: we first train a new model to recognize the Probability Density Function (PDF) characteristics of our D_0 dataset. For the *fingerprinting* dataset we use a 2-Dimensional data structure (X and Y coordinates), while for the *links* dataset we use a 5-Dimensional data structure (1 dimension for each class). In the sequel we conduct inference using the trained model and the rest of the datasets. In order to assess the presence of drift phenomena, we use the KL-divergence metric, as suggested by the authors [46], to measure the distance between the estimated PDF of the D_0 and the estimated PDF of the dataset under investigation.

E. Performance Metrics

Since ALERT uses a utility function that designates whether the M_0 model should be retrained or not, it is difficult to compare it with existing solutions. This happens because each state-of-the-art method uses a different prediction confidence threshold that is uniquely tailored according to its requirements. To resolve this, we define the following scoring function that can be commonly used among several methods to compare their efficiency:

$$Score = \begin{cases} F1_{gain}, & \text{if decision is true positive.} \\ T_s, & \text{if decision is true negative.} \\ F1_{gain} - T_s, & \text{if decision is false positive.} \\ -F1_{gain}, & \text{if decision is false negative.} \end{cases} \tag{10}$$

The function is designed to increase the score for correct feature drift predictions and to decrease for incorrect ones. The larger the feature drift, the bigger score is allocated

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2025.3644604

for correct assessments, and the bigger the penalty is given for incorrect predictions. We assume that each method under examination outputs a prediction on whether the M_0 should be retrained. We distinguish 4 different possible scenarios for this assessment:

- A method's assessment is true positive and correctly identifies the existence of feature drift. There, the score equals to the macro $F1$-score that the model will gain if retrained ($F1_{gain}$). The larger the feature drift, the higher the ($F1_{gain}$), and thus, larger scores are allocated for correct assessments.
- A method's assessment is true negative and correctly identifies the absence of feature drift. In such scenarios the score equals to the T_s constant, that is set by the user. In our experiments, we set the T_s to 0.1.
- A method's assessment is false positive and incorrectly identifies the existence of feature drift. In this case the model M_0 is retrained, and the method is penalized by an amount of $F1_{gain} - T_s$ where $F1_{gain}$ is $F1$-score the model gains after retraining. We expect a small $F1_{gain}$, due to the absence of feature drift and as a result, the score will be negative.
- A method's assessment is false negative and incorrectly identifies the absence of feature drift. The penalty of this error is $-F1_{gain}$, which is the $F1$-score that the model would gain if it was retrained with the new data.

In our experiments, we apply this formula for each method under examination and for each tested dataset. At the end of each experiment, we sum each method's scores which are collected after assessing the aforementioned datasets and we calculate the final value.

F. Utility Component Contribution

We perform an analysis of the contribution of each of the following components to the total utility score of the ALERT method: (i) $KS_{(D_0,D_1)}$; (ii) $KS_{(R_0,R_1)}$; (iii) $PSI_{(D_0,D_1)}$; and (iv) $PSI_{(R_0,R_1)}$. The aim of this analysis is to validate our initial hypothesis that both the KS and the PSI tests are essential for the calculation of the utility score. Through this, we also aim to analyze the impact of the representation learning technique to the overall utility score of the ALERT. To achieve this, we perform an ablation study by measuring the contributions of each component separately, in terms of percentages ($\%$) for each Use Case, and we present the results we obtain in section .

VII. Results

In this section we analyze the performance of the ALERT method proposed in Section V and evaluated according to the methodology elaborated in Section VI to solve the problem identified in Section IV.

A. Parameter Search for the Representation Learning Components

Figure 6 illustrates the results of the 3-dimensional parameter exploration, which is conducted for the MLP. We experiment with two datasets in which the feature drift phenomenon

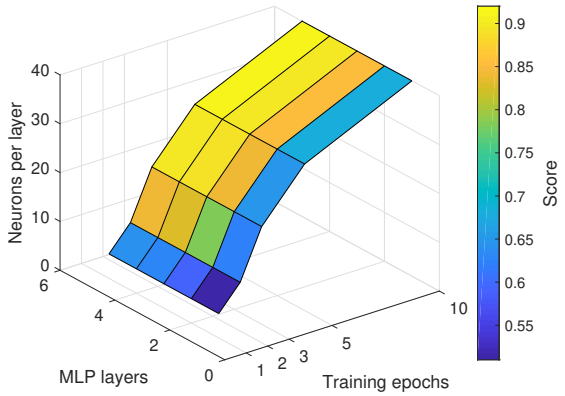

Fig. 6. The contributions of different MLP parameters and training epochs to the obtained utility score.

TABLE II
THE MLP PARAMETERS USED FOR THE IMPLEMENTATION OF ALERT.

MLP parameters	Values
Layers	3 (Dense, Dense, Dense)
Neurons	20, 20, Number of Classes
Activations	Sigmoid, Sigmoid, Softmax
Training epochs	3
Batch size	20
Optimizer	Adam
Loss	Categorical Cross-Entropy

is prominent, in order to calibrate the MLP parameters. We should note that the utility function should be represented by a high value, to clearly predict the existence of feature drifts. Results indicate that the increase of model complexity is directly correlated with higher utility scores. This is expected, since more complex models manage to properly capture input data representations and thus, to produce better assessments. Nonetheless, there is an optimal parameter space, over which larger and more complex MLPs do not provide significant contributions to the utility score. Empirically, an MLP with 3 layers, 20 neurons per layer, and a training period of 3 epochs achieves a utility score of 0.85, which we deem adequate to assess the presence of feature drifts. On the contrary, an MLP with 5 layers, 40 neurons per layer, and a training period of 10 epochs achieves a utility score of 0.92 which, despite being higher than 0.85, does not provide us with better information than the previous configuration. This is true especially if we consider that the training requirements of more complex MLP are significantly larger compared to models that leverage simpler designs.

Table II lists the MLP parameters used for the ALERT method. As discussed in the paragraph above, we opted for a small model with 3 layers, each one of which containing 20 neurons. Since ALERT is a supervised method, the last layer contains a number of neurons equal to the amount of data classes. We utilise the Sigmoid activation function for the first two layers and the Softmax function for the last layer correspondingly. We train the model for 3 epochs, using a batch size of 20, selecting Adam as the optimiser and the Categorical cross-entropy as loss. Due to the small size of the MLP we do not employ any regularization methods, and since

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2025.3644604

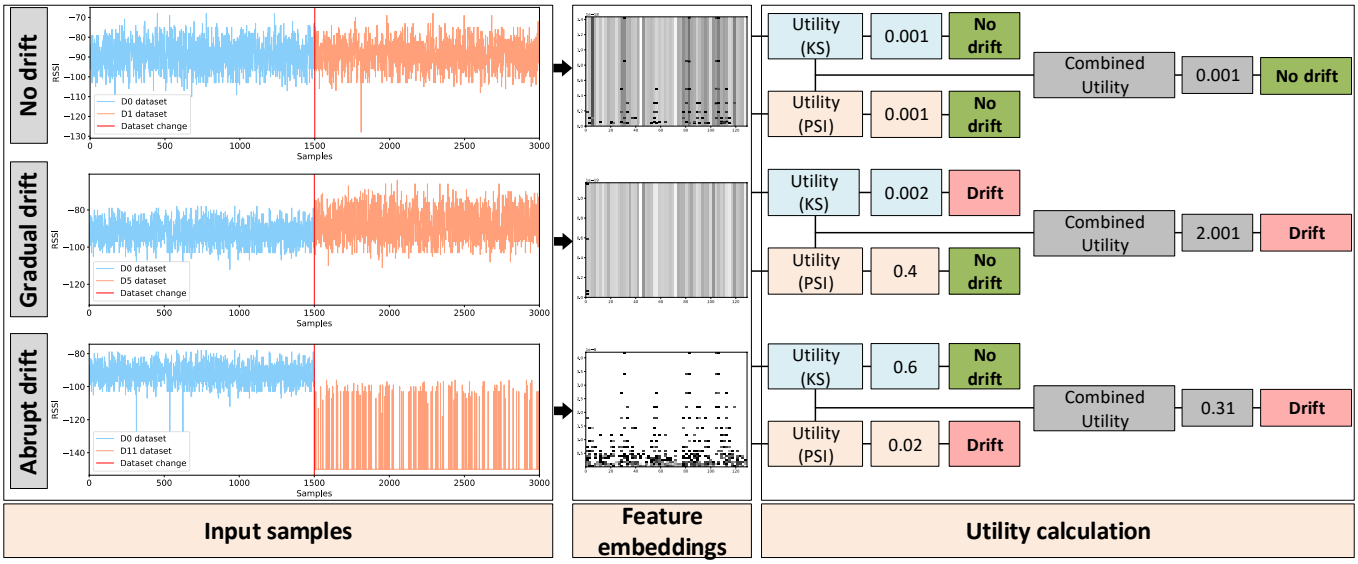

Fig. 7. The behavior of PSI and KS in different drift scenarios (no drift, gradual drift and abrupt drift). PSI excels in identifying abrupt drifts, while KS is good at detecting gradual drifts. The combination of both detectors, along with the proposed representation learning technique give ALERT better coverage for multiple drift scenarios.

the training is conducted for 3 epochs only, we do not use any early-stopping techniques. In order to avoid temporal leakage, we use the D_0 dataset for training, and the rest of the datasets for inference. This way, future data points are not accounted for, during the model training.

B. Performance under different feature drift scenarios

Figure 7 illustrates the performance of ALERT under different different drift scenarios. To properly assess the effects of KS and PSI in the design of the utility function in Eq. 7 we opt to visualise their outputs for 3 scenarios, using the *fingerprinting* dataset, which is illustrated in Figure 1: (i) when no drift exists in the data; (ii) when a gradual drift occurs; and (iii) when an abrupt drift appears. Evidently, for each scenario, ALERT produces different feature embeddings which are then interpreted by the utility calculation module. As discussed in Section V-C, this module utilises both KS and PSI to assess if a drift phenomenon exists. The results confirm the complementarity of KS and PSI incorporated in Eq. 7. More specifically, the feature analysis conducted by KS achieves good results when gradual drifts are under examination, while in such cases PSI underperforms and thus, it cannot properly identify their presence. The opposite observation for abrupt drifts holds true. There, PSI excels in drift detection, whereas KS often fails to identify abrupt drifts. To solve this, ALERT combines the outputs of both estimators and manages to capture both abrupt and gradual drifts. ALERT, KS and PSI behave similarly when dealing with the *Links* dataset as well, which is illustrated in Figure 2.

C. Performance with the "Fingerprinting" Dataset

Figure 8 depicts the performance, in terms of the score function established by the Eq. 10, for the proposed ALERT method and the baseline techniques described in Section VI-C.

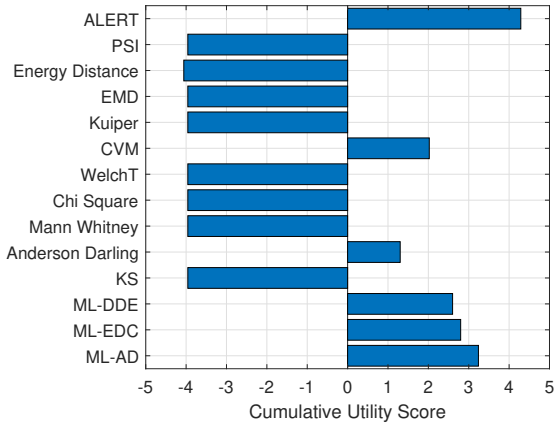

Fig. 8. Performance comparison between the ALERT and the baseline tests, using the *fingerprinting* dataset.

The maximum achievable score for this Use Case is 4.5, which is achieved if all the predictions are correct, and the lowest is -4.5 which is obtained if all the predictions are incorrect. We observe that the proposed ALERT method achieves the best score (4.28), followed by the ML-AD (3.24), ML-EDC (2.8), ML-DDE (2.6), CVM (2.0) and Anderson Darling (1.3) tests. The rest of the methods underperform by a large margin and they obtain negative scores. The superiority of ALERT is due to the learnt representation that keeps only the relevant signals related to drift, and to the design of the utility function in Eq. 7 that provides balanced sensitivity, as discussed in Section V-C. This also contributes to ALERT's robustness, reducing false positives and false negatives in drift detection.

Table III depicts the detailed prediction statistics for the four best performing methods on the "*fingerprinting*" dataset. The first column of the table represents the methods under examination, while the rest of the columns depict each method's performance in terms of true positive, true negative,

TABLE III
PERFORMANCE IN TERMS OF PRECISION, FRD, RECALL AND F1 OF DIFFERENT DRIFT DETECTION METHODS OVER THE *fingerprinting* DATASET. THE PROPOSED ALERT METHOD OUTPERFORMS ALL THE BASELINE MODELS, ACHIEVING AN OVERALL F1-SCORE OF 0.9. IN TERMS OF F1-SCORES, ML-AD FOLLOWS WITH 0.66, ML-DDE WITH 0.64 AND ML-EDC WITH 0.0.64

Methods	True positives	True negatives	False positives	False negatives	Precision	FDR	Recall	F1
ALERT	9/10	19/20	1/20	1/10	0.9	0.1	0.9	0.9
CVM	8/10	10/20	10/20	2/10	0.44	0.56	0.8	0.56
Anderson Darling	5/10	11/20	9/20	5/10	0.35	0.65	0.5	0.41
PSI	2/10	2/20	18/20	8/10	0.1	0.9	0.2	0.13
ML-AD	9/10	12/20	8/20	1/10	0.53	0.47	0.9	0.66
ML-EDC	8/10	15/20	5/20	2/10	0.61	0.39	0.8	0.62
ML-DDE	8/10	13/20	7/20	2/10	0.53	0.47	0.8	0.64

TABLE IV
PERFORMANCE IN TERMS OF PRECISION, FRD, RECALL AND F1, OF DIFFERENT DRIFT DETECTION METHODS OVER THE *Links* DATASET. ALERT ACHIEVES THE BEST F1-SCORE, NAMELY 0.88. ML-DDE ACHIEVES 0.75, ML-AD 0.72, CVM AND PSI 0.6

Methods	True positives	True negatives	False positives	False negatives	Precision	FDR	Recall	F1
ALERT	4/5	3/3	0/3	1/5	1.0	0	0.8	0.88
CVM	3/5	1/3	2/3	2/5	0.6	0.4	0.6	0.6
Anderson Darling	2/5	2/3	1/3	3/5	0.66	0.34	0.4	0.49
PSI	3/5	1/3	2/3	2/5	0.6	0.4	0.6	0.6
ML-AD	4/5	1/3	2/3	1/5	0.66	0.34	0.8	0.72
ML-EDC	2/5	3/3	0/3	3/5	1.0	0	0.4	0.57
ML-DDE	3/5	2/3	0/3	2/5	1.0	0	0.6	0.75

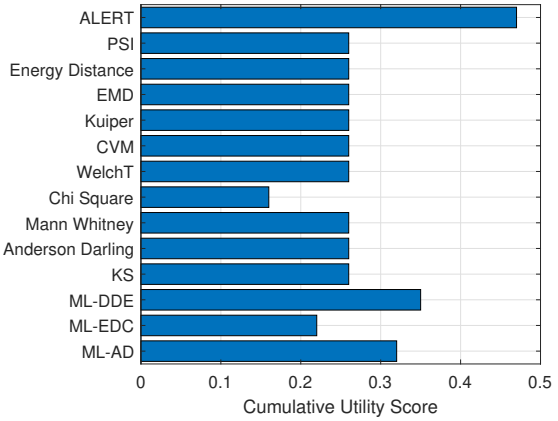

Fig. 9. Performance comparison of the ALERT method, with baseline techniques, using the *links* dataset.

false positive and false negative predictions, as described in Section VI-E. We have also included 4 additional columns that present the precision, false discovery rates (FDR), recall and F1-score of each method. The ALERT method outperforms the baseline techniques since it achieves the best F1-score, namely 0.9. ML-based methods follow with ML-AD (0.66), ML-DDE (0.64) and ML-EDC (0.62). The rest of the methods perform poorly, as the CVM achieves 0.56 F1, the Anderson Darling 0.41 and the PSI 0.13. Evidently, ALERT manages not only to accurately detect the existence of drift phenomena (9/10 true positive score), but also to correctly assess the absence of feature drift (19/20 true negative score), thus saving the M_0 model of unnecessary re-training operations. Also, ALERT minimizes the rate under which it falsely identifies drifts, achieving a low FDR score (0.1). ML-AD, which is the best performing baseline method, achieves the same true positive prediction score (9/10), but a lower true negative score (12/20). Further, ML-AD has a higher FDR score (0.47) which would result in unnecessarily re-training operations of the M_0 model. The same observation also holds true for both the rest of the baseline methods.

D. Performance with the "Links" Dataset

Figure 9 illustrates the comparison of the ALERT technique with existing works over the *links* dataset. The maximum score for each method is 0.48. The ALERT achieves a score of 0.47 and outperforms the rest of the baseline techniques. Distance-based and statistics-based methods acquire an equal score of 0.26, with the exception of Chi-Square that severely under-performs (0.16 score). On the other hand, ML-based techniques perform well, with ML-DDE leading with 0.35 followed by ML-AD (0.32) and ML-EDC (0.22). ALERT manages to successfully capture the data distribution properties of the *links* dataset, containing both gradual and abrupt drifts, and to assess with high accuracy the existence of feature drift.

Table IV showcases the performance of top-scoring methods, considering their predictions. Similarly to Table III, the first column depicts the methods under examination which are the ALERT, CVM, Anderson Darling, PSI, ML-AD, ML-EDC and ML-DDE. The rest of the table columns contain information regarding the true positive, true negative, false positive, false negative, precision, FDR, recall and F1 scores of each method. The ALERT and the ML-AD technique achieve the best true positive score (4/5), followed by ML-DDE, CVM and PSI (3/5). In terms of true negatives, ALERT and ML-EDC come first with 3/3 correct assessments, Anderson Darling and ML-DDE second with 2/3 and the rest follow with 1/3. This has a direct impact to each method's F1, with ALERT scoring 0.88, closely followed by ML-DDE (0.75) and ML-AD (0.72), while CVM and PSI score 0.6. We also

This article has been accepted for publication in IEEE Transactions on Emerging Topics in Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TETC.2025.3644604

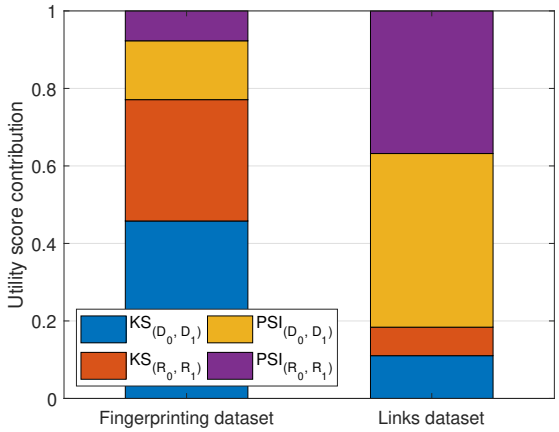

Fig. 10. The contribution of $KS_{(D_0,D_1)}$, $KS_{(R_0,R_1)}$, $PSI_{(D_0,D_1)}$ and $PSI_{(R_0,R_1)}$ to the expected utility score.

observe that all ML-EDC, ML-DDE and ALERT achieve 0 FDR scores and thus, they minimize the unnecessary model re-training operations. Apart from the FDR, the rest of results are in line with the results we observed under the *fingerprinting* dataset. The ALERT method not only identifies the existence of drift phenomena, but also it properly assesses their absence as well.

E. Ablation Study for the Utility Assessment Component

Figure 10 depicts the outcomes of the ablation study, which we conducted in order to evaluate the contributions of the proposed representation learning technique along with the KS and PSI tests described in Section V-B to the utility score of ALERT. As discussed in Section VI-F, we perform the study using the *fingerprinting* and the *links* datasets to confirm our hypothesis that all of the aforementioned methods play a major role to the overall utility score. Results indicate that the contributions of each statistical test ($KS_{(D_0,D_1)}$, $KS_{(R_0,R_1)}$, $PSI_{(D_0,D_1)}$ and $PSI_{(R_0,R_1)}$) are indeed significant. Evidently, all four methods have quantifiable impact to the expected utility, ranging from 7% to 45% depending on the dataset. Therefore, the exclusion of any of these tests would reduce the effectiveness of the utility function for the ALERT method.

F. Execution time requirements

In Table V we present the execution time requirements of each method under examination, in seconds. The first column of the table refers to the method name, while the next two columns present the time requirements of each technique to perform a feature drift assessment, using the *fingerprinting* and the *links* datasets correspondingly. Generally, the distance-based and statistics-based methods that exist in the literature are very fast, since they complete their assessments within 0.02 to 0.4 seconds. On the other hand, ALERT has larger execution times, ranging from 17.2 - 3.5 seconds, depending on the use case. This is expected, since ALERT partially trains an MLP model, which severely affects its execution time. Similarly, ML-based methods are slower compared to the others, since

TABLE V
THE EXECUTION TIME REQUIREMENTS OF THE METHODS UNDER EXAMINATION FOR BOTH THE *fingerprinting* AND *links* DATASETS. THE EXECUTION TIME IS MEASURED IN SECONDS AND IS CALCULATED FOR EACH METHOD INDEPENDENTLY.

Methods	*fingerprinting* dataset	*links* dataset
ALERT	$17.2s$	$3.5s$
PSI	$0.04s$	$0.4s$
Energy Distance	$0.04s$	$0.3s$
EMD	$0.04s$	$0.3s$
Kuiper	$0.07s$	$0.3s$
CVM	$0.03s$	$0.19s$
WelchT	$0.02s$	$0.02s$
Chi Square	$0.06s$	$0.4s$
Mann Whitney	$0.03s$	$0.1s$
Anderson Darling	$0.03s$	$0.3s$
KS	$0.06s$	$0.2s$
ML-AD	22s	$4s$
ML-EDC	25s	$5.2s$
ML-DDE	28s	$7s$

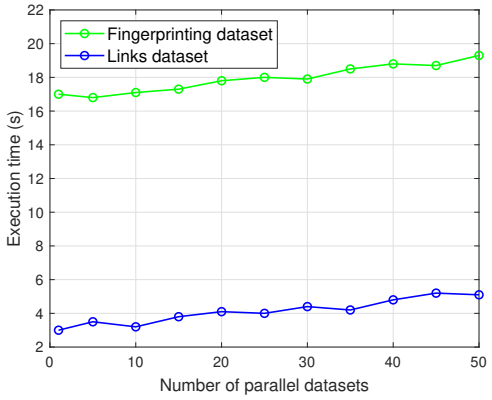

Fig. 11. The execution time requirements of the ALERT methods when conducting several drift assessment tasks in parallel.

during the inference process they utilize a fully trained ML model. We should note that the execution times depicted in Table V correspond to the inference process of the ML-based methods, and they do not account for the model training time requirements. We conclude that, in terms of absolute numbers, the execution time requirements of ALERT are affordable for real-world applications, even when large data volumes are involved.

Further, gradual drifts usually occur in longer periods of time, from several hours, as noted by [48], or several months, as seen in our *fingerprinting* dataset. Considering this, our method's higher execution time can still be regarded as relatively fast, especially in the context of adapting to such evolving conditions. ALERT's capacity to scale is illustrated in Figure 11, where we evaluate its performance when executing multiple drift assessments in parallel, simulating large-scale monitoring of diverse data sources. Each dataset contains 16,290 samples and is processed on a single machine with an NVIDIA GeForce GTX 1660 Ti (6 GB GPU) and an Intel i7-10750H (2.6 GHz CPU). The x-axis shows the number of datasets assessed in parallel, while the y-axis indicates the average execution time (in seconds) per assessment. The results show that ALERT scales efficiently, as the difference between

TABLE VI
THE DETAILS, ADVANTAGES AND DISADVANTAGES OF THE DRIFT DETECTORS UNDER EXAMINATION.

	Statistical-based methods	Distance-based methods	ML-based methods	ALERT
Method type	Unsupervised	Supervised	Unsupervised	Supervised
Parameters	p-value threshold	Probability bins, cut-off thresholds	ML hyperparameters	MLP hyperparameters, T_s
Drift assessment	Over data	Over data	Over features	Over features
Low parameter sensitivity	✗	✗	✓	✓
Fast training	✓	✓	✗	✓
Fast inference	✓	✓	✗	✗
Resistant to noise	✗	✗	✓	✓
Do not require regular re-training	✗	✗	✗	✓
High accuracy with small datasets	✗	✗	✗	✓

1 and 50 datasets is ≈2 seconds of execution overhead.

G. Qualitative evaluation of drift detectors

In this section, we discuss the drift detectors which are evaluated throughout this paper. We focus on breaking down their main advantages and disadvantages, and we present their input parameters. Our goal is to analyze the requirements of each technique and to assess their functionality in different scenarios. Under this premise, we also perform an analysis on how ALERT departs from the standard drift detection paradigm and what innovations it brings to the current discourse. Table VI depicts several characteristics of the method groups under examination i.e., statistical-based, distance-based and ML-based.

Both statistical-based and distance-based methods conduct their drift assessments over the input data, are fast to train and support lightweight inference. On the other hand, they require frequent re-training in order to maintain high accuracy, they are not resistant to noise and they underperform when the training dataset is small. Further, they are very sensitive to input parameters which include the p-value threshold (for statistical techniques) and the number of probability bins, as well as the cut-off threshold, over which a drift is detected (for distance-based methods). A key difference between statistical-based and distance-based detectors is that the former are mostly unsupervised methods, while the latter usually fall into the category of supervised methods.

The ML-based techniques, including ALERT, utilise features (instead of raw data) to assess drifts, they have high resistance to noise and they are not very sensitive to their input parameters. On the flip side, their inference process is slower compared to statistical and distance based methods. The ML-based methods that exist in the literature require a significant amount of training time (ranging from minutes to hours), they can work with both labeled and unlabeled data, they need regular re-training to maintain high quality models, and their performance drops when the training dataset size is small. On the contrary, ALERT is a supervised method that does not require frequent re-training operations and works well with both large and smaller datasets.

VIII. CONCLUSION

In this paper we have introduced ALERT—a novel feature drift detection method that comprises representation learning, statistical testing, and utility assessment components. We demonstrated ALERT's superior performance on two real-world wireless use cases, fingerprinting and link anomaly detection, where it outperformed ten established methods by ensuring that the AI models keep high F1-scores, namely of 0.90 and 0.88 even in the presence of feature drift when it is suitably detected and re-training triggered. Beyond raw detection accuracy, our work provides a comprehensive analysis workflow that not only pinpoints when and where feature drift occurs but also quantifies its impact on model performance and informs optimal retraining decisions. By rigorously benchmarking ALERT against standard and state-of-the-art approaches, we advance the state of the art in feature drift detection for wireless networks and offer practitioners a robust, end-to-end solution for maintaining reliable AI models in dynamic radio environments.

Although ALERT was evaluated on wireless network data, its architecture is domain-agnostic. Because it operates on learned feature representations rather than on specific radio-signal characteristics, the method can generalize to a wide range of time-series or tabular data domains. This includes applications such as financial transaction monitoring, healthcare sensor analysis, and industrial IoT systems, where data distributions evolve over time. Future work will explore adapting ALERT to these contexts, investigating how representation learning can capture domain-specific dynamics while maintaining robust drift detection across heterogeneous data sources. Additionally, a study of using such detection systems in large scale systems where feature dimensionality or incoming data volumes vary would also be relevant.

ACKNOWLEDGMENTS

This work was funded by the Slovenian Research and Innovation Agency (grant P2-0016) and by the European Union's Horizon Europe Framework Program SNS-JU (grant agreement No. 101096456, NANCY). Instead of traditional spell-checking and language correction tools, some manually written paragraphs in the introduction and related work were refined using AI to improve flow and language quality.

REFERENCES

[1] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6g: Ai empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[2] S. E. Trevlakis, A.-A. A. Boulogeorgos, D. Pliatsios, J. Querol, K. Ntontin, P. Sarigiannidis, S. Chatzinotas, and M. Di Renzo, "Localization as a key enabler of 6g wireless systems: A comprehensive survey and an outlook," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 2733–2801, 2023.

[3] B. Bertalanič, M. Meža, and C. Fortuna, "Resource-aware time series imaging classification for wireless link layer anomalies," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 8031–8043, 2023.

[4] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6g ai-native air interface," *IEEE Communications Magazine*, vol. 59, no. 5, pp. 76–81, 2021.

[5] W. Wu, C. Zhou, M. Li, H. Wu, H. Zhou, N. Zhang, X. S. Shen, and W. Zhuang, "Ai-native network slicing for 6g networks," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 96–103, 2022.

[6] Á. A. Cabrera, M. Tulio Ribeiro, B. Lee, R. Deline, A. Perer, and S. M. Drucker, "What did my ai learn? how data scientists make sense of model behavior," *ACM Transactions on Computer-Human Interaction*, vol. 30, no. 1, pp. 1–27, 2023.

[7] O.-R. Alliance, "O-ran working group 2 ai/ml workflow description and requirements," *ORAN-WG2. AIML. v01.03*, July 2021.

[8] A. Čop, B. Bertalanič, and C. Fortuna, "An overview and solution for democratizing ai workflows at the network edge," *Journal of Network and Computer Applications*, p. 104180, 2025.

[9] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," *Advances in neural information processing systems*, vol. 28, 2015.

[10] S. Liu, F. Bronzino, P. Schmitt, A. N. Bhagoji, N. Feamster, H. G. Crespo, T. Coyle, and B. Ward, "Leaf: Navigating concept drift in cellular networks," *Proceedings of the ACM on Networking*, vol. 1, no. CoNEXT2, pp. 1–24, 2023.

[11] D. M. Manias, A. Chouman, and A. Shami, "Model drift in dynamic networks," *IEEE Communications Magazine*, vol. 61, no. 10, pp. 78–84, 2023.

[12] Y. Swathi and M. Challa, "From deployment to drift: A comprehensive approach to ml model monitoring with evidently ai," in *International Conference on VLSI, Signal Processing, Power Electronics, IoT, Communication and Embedded Systems*. Springer, 2023, pp. 307–320.

[13] T. Simonetto, M. Cordy, S. Ghamizi, Y. L. Traon, C. Lefebvre, A. Boystov, and A. Goujon, "On the impact of industrial delays when mitigating distribution drifts: An empirical study on real-world financial systems," in *International Workshop on Discovering Drift Phenomena in Evolving Landscapes*. Springer, 2024, pp. 57–73.

[14] R. Müller, M. Abdelaal, and D. Stjelja, "Open-source drift detection tools in action: Insights from two use cases," in *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 2024, pp. 346–352.

[15] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[16] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[17] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[18] J. F. Kurian and M. Allali, "Detecting drifts in data streams using kullback-leibler (kl) divergence measure for data engineering applications," *Journal of Data, Information and Management*, vol. 6, no. 3, pp. 207–216, September 2024. [Online]. Available: https://doi.org/10.1007/s42488-024-00119-y

[19] C. Harald, "On the composition of elementary errors," *Scandinavian Actuarial Journal*, vol. 1, pp. 141–80, 1928.

[20] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, pp. 99–121, 2000.

[21] M. Z. A. Mayaki and M. Riveill, "Autoregressive based drift detection method," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[22] S. Dong, Q. Wang, S. Sahri, T. Palpanas, and D. Srivastava, "Efficiently mitigating the impact of data drift on machine learning pipelines," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, pp. 3072–3081, 2024.

[23] A. Mallick, K. Hsieh, B. Arzani, and G. Joshi, "Matchmaker: Data drift mitigation in machine learning for large-scale systems," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 77–94, 2022.

[24] T. Simonetto, M. Cordy, S. Ghamizi, Y. Le Traon, C. Lefebvre, A. Boystov, and A. Goujon, "On the impact of industrial delays when mitigating distribution drifts: an empirical study on real-world financial systems," *KDD Workshop*, 2024.

[25] S. Amos, "23when training and test sets are different: Characterizing learning transfer," in *Dataset Shift in Machine Learning*. The MIT Press, 12 2008. [Online]. Available: https://doi.org/10.7551/mitpress/9780262170055.003.0001

[26] S. Ackerman, E. Farchi, O. Raz, M. Zalmanovici, and P. Dube, "Detection of data drift and outliers affecting machine learning model performance over time," 2022. [Online]. Available: https://arxiv.org/abs/2012.09258

[27] B. Bertalanič, G. Morano, and G. Cerar, "Log-a-tec testbed outdoor localization using ble beacons," in *2022 International Balkan Conference on Communications and Networking (BalkanCom)*, 2022, pp. 115–119.

[28] M. Bello, G. Nápoles, R. Sánchez, R. Bello, and K. Vanhoof, "Deep neural network to extract high-level features and labels in multi-label classification problems," *Neurocomputing*, vol. 413, pp. 259–270, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220311115

[29] P. Porwik and B. M. Dadzie, "Detection of data drift in a two-dimensional stream using the kolmogorov-smirnov test," *Procedia Computer Science*, vol. 207, pp. 168–175, 2022, knowledge-Based and Intelligent Information and Engineering Systems: Proceedings of the 26th International Conference KES2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705092200922X

[30] Z. Wang and W. Wang, "Concept drift detection based on kolmogorov–smirnov test," in *Artificial Intelligence in China*, Q. Liang, W. Wang, J. Mu, X. Liu, Z. Na, and B. Chen, Eds. Singapore: Springer Singapore, 2020, pp. 273–280.

[31] C. Whitnall, E. Oswald, and L. Mather, "An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis," in *Proceedings of the 10th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications*, ser. CARDIS'11. Berlin, Heidelberg: Springer-Verlag, 2011, p. 234–251. [Online]. Available: https://doi.org/10.1007/978-3-642-27257-8_15

[32] D. Steinskog and N. Kvamstø, "A cautionary note on the use of the kolmogorov–smirnov test for normality," *Monthly Weather Review - MON WEATHER REV*, vol. 135, pp. 1151–1157, 03 2007.

[33] S. A. Alex, U. Ghosh, and N. Mohammad, "Weather prediction from imbalanced data stream using 1d-convolutional neural network," in *2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22)*, 2022, pp. 1–6.

[34] N. H. Kuiper, "Tests concerning random points on a circle," *Indagationes Mathematicae (Proceedings)*, vol. 63, pp. 38–47, 1960. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1385725860500060

[35] H. C. and, "On the composition of elementary errors," *Scandinavian Actuarial Journal*, vol. 1928, no. 1, pp. 13–74, 1928. [Online]. Available: https://doi.org/10.1080/03461238.1928.10416862

[36] B. L. Welch, "The generalization of 'student's' problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947. [Online]. Available: http://www.jstor.org/stable/2332510

[37] K. P. and, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900. [Online]. Available: https://doi.org/10.1080/14786440009463897

[38] H. B. Mann and D. R. Whitney, "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50 – 60, 1947. [Online]. Available: https://doi.org/10.1214/aoms/1177730491

[39] F. W. Scholz and M. A. Stephens, "K-sample anderson-darling tests," *Journal of the American Statistical Association*, vol. 82, no. 399, pp. 918–924, 1987. [Online]. Available: http://www.jstor.org/stable/2288805

[40] F. J. M. J. and, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1951.10500769

[41] D. Wu and D. L. O. and, "Enterprise risk management: coping with model risk in a large bank," *Journal of the Operational Research Society*, vol. 61, no. 2, pp. 179–190, 2010. [Online]. Available: https://doi.org/10.1057/jors.2008.144

[42] G. J. Székely and M. L. Rizzo, "Energy statistics: A class of statistics based on distances," *Journal of Statistical Planning and Inference*, vol. 143, no. 8, pp. 1249–1272, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378375813000633

[43] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, Nov. 2000. [Online]. Available: https://doi.org/10.1023/A:1026543900054

[44] Z. Deng, X. Yang, S. Xu, H. Su, and J. Zhu, "Libre: A practical bayesian approach to adversarial detection," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 972–982.

[45] S. Greco, B. Vacchetti, D. Apiletti, and T. Cerquitelli, "Unsupervised concept drift detection from deep learning representations in real-time," *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, no. 10, pp. 6232–6245, 2025.

[46] P. Puchert, P. Hermosilla, T. Ritschel, and T. Ropinski, "Data-driven deep density estimation," *Neural Comput. Appl.*, vol. 33, no. 23, p. 16773–16807, Dec. 2021. [Online]. Available: https://doi.org/10.1007/s00521-021-06281-3

[47] J. Céspedes Sisniega and Álvaro López García, "Frouros: An open-source python library for drift detection in machine learning systems," *SoftwareX*, vol. 26, p. 101733, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711024001043

[48] A. Costa, R. Giusti, and E. M. dos Santos, "Analysis of descriptors of concept drift and their impacts," in *Informatics*, vol. 12, no. 1. MDPI, 2025, p. 13.

BIOGRAPHIES

Athanasios Tziouvaras received his B.Sc. and M.Sc. degrees in electrical engineering, and his Ph.D. in computer architecture and data-intensive applications from University of Thessaly in Greece. He joined Business and IoT integrated solutions Ltd. (BI2S) SME in 2021 and he is actively involved in research and innovation activities. His research interests include hardware acceleration, edge computing, machine learning, resource-aware computational methodologies and distributed computing. He has participated in more than 10 European and National research projects and has co-authored several publications in the domain of computer science.

Blaž Bertalanič received his Ph.D. degree with highest distinction at the Faculty of Electrical engineering, University of Ljubljana. He is currently working as a researcher at Sensorlab, Jožef Stefan Institute. His main research interests are connected to advancement of machine learning and AI algorithms, especially in the context of time series analysis and smart infrastructures. Blaz is an IEEE member and with several leadership positions in the Slovenian chapter with over 15 IEEE publications.

George Floros is an Assistant Professor in the Department of Electronic & Electrical Engineering at Trinity College Dublin. Previously, he was a Postdoctoral Researcher and Lecturer at the Department of Electrical and Computer Engineering at the University of Thessaly, Greece. He received his Diploma in Engineering, MSc, and PhD degrees from the same department in 2013, 2015, and 2019, respectively. His research interests primarily focus on the fields of electronic design automation (EDA), semiconductor device modeling, circuit simulation, model order reduction, thermal analysis of ICs, as well as VLSI design techniques and embedded systems. Throughout his academic career, he has authored over 10 journal articles, 30 conference papers, 3 posters, and abstracts.

Kostas Kolomvatsos received his B.Sc. in Informatics from the Department of Informatics at the Athens University of Economics and Business, his M.Sc. and his Ph.D. in Computer Science from the Department of Informatics and Telecommunications at the National and Kapodistrian University of Athens. Currently, he serves as an Assistant Professor in the Department of Informatics and Telecommunications, University of Thessaly. He was a Marie Skłodowska Curie Fellow (Individual Fellowship) at the School of Computing Science, University of Glasgow. His research interests are in the definition of Intelligent Systems adopting Machine Learning, Computational Intelligence and Soft Computing for Pervasive Computing, Distributed Systems, Internet of Things, Edge Computing and the management of Large-Scale Data. He is the author of over 130 publications in the aforementioned areas.

Panagiotis Sarigiannidis received the B.Sc. and Ph.D. degrees in computer science from the Aristotle University of Thessaloniki, Greece, in 2001 and 2007, respectively. He is currently the Director of the ITHACA Laboratory, the Co-Founder of the 1st spin-off of the University of Western Macedonia, MetaMind Innovations, and a Full Professor with the Department of Electrical and Computer Engineering, University of Western Macedonia, in Greece. He has published more than 360 papers in international journals, conferences, and book chapters, including IEEE COMST, IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE INTERNET OF THINGS, IEEE TRANSACTIONS ON BROADCASTING, IEEE SYSTEMS JOURNAL, IEEE ACCESS, and Computer Networks. He has been involved in several national, European, and international projects, including H2020 and Horizon Europe. His research interests include telecommunication networks, IoT, and network security. He received six best paper awards and the IEEE SMC TCHS Research and Innovation Award 2023.

Carolina Fortuna is a research associate professor with the Jožef Stefan Institute where she leads Sensorlab. She was a Post-Doctoral Researcher with Ghent University, Ghent, Belgium. She was a Visiting researcher in Infolab with Stanford University, Stanford, CA, USA. She has led and contributed EU funded projects such as H2020 NRG5, eWINE, WISHFUL, FP7 CREW, Planetdata, ACTIVE, under various positions. She has advised/coadvised more than six M.Sc. and Ph.D. students. She has consulted public and private institutions. She has coauthored over 100 papers including in IEEE COMST, IEEE WICOM Magazine, IEEE OJCOMS and Access. Her research interest includes developing the next generation smart infrastructures that surround us and improve the quality of our life. Dr. Fortuna contributed to community work as a TPC member, the track chair, and a TPC member at several IEEE conferences including Globecom and ICC.