IEEE ComSoc IEEE Communications Society | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 13 December 2024; revised 21 April 2025; accepted 1 June 2025.
Date of publication 4 June 2025; date of current version 10 June 2025.

The associate editor coordinating the review of this article and approving it for publication was Y. Liu.

Digital Object Identifier 10.1109/TMLCN.2025.3576727

Dual Self-Attention is What You Need for Model Drift Detection in 6G Networks

MAZENE AMEUR[1] (Member, IEEE), BOUZIANE BRIK[2] (Senior Member, IEEE), AND ADLEN KSENTINI[1] (Senior Member, IEEE)

[1]Communication Systems Department, EURECOM, 06410 Biot, France
[2]Computer Science Department, University of Sharjah, Sharjah, United Arab Emirates

CORRESPONDING AUTHOR: M. AMEUR (mazene.ameur@eurecom.fr)

This work was supported by European Union's Horizon Program through the 6G DAta and ML operations automation via an end-to-end AI framework (6G-DALI) Project under Grant 101192750.

ABSTRACT The advent of 6G networks heralds a transformative shift in communication technology, with Artificial Intelligence (AI) and Machine Learning (ML) forming the backbone of its architecture and operations. However, the dynamic nature of 6G environments renders these models vulnerable to performance degradation due to model drift. Existing drift detection approaches, despite advancements, often fail to address the diverse and complex types of drift encountered in telecommunications, particularly in time-series data. To bridge this gap, we propose, for the first time, a novel drift detection framework featuring a Dual Self-Attention AutoEncoder (DSA-AE) designed to handle all major manifestations of drift in 6G networks, including data, label, and concept drift. This architectural design leverages the autoencoder's reconstruction capabilities to monitor both input features and target variables, effectively detecting data and label drift. Additionally, its dual self-attention mechanisms comprising feature and temporal attention blocks capture spatiotemporal fluctuations, addressing concept drift. Extensive evaluations across three diverse telecommunications datasets (two time-series and one non-time-series) demonstrate that our framework achieves substantial advancements over state-of-the-art methods, delivering over a 13.6% improvement in drift detection accuracy and a remarkable 94.7% reduction in detection latency. By balancing higher accuracy with lower latency, this approach offers a robust and efficient solution for model drift detection in the dynamic and complex landscape of 6G networks.

INDEX TERMS Artificial intelligence, machine learning, model drift detection, data drift, concept drift, self-attention mechanisms, autoencoders, 6G networks, telecommunications.

I. INTRODUCTION

THE 6G revolution is rapidly gaining momentum. Once confined to the realm of theoretical speculation, the development of sixth-generation wireless networks has made significant strides [1]. Driven by relentless innovation and research, the vision of 6G is transforming from a distant aspiration into a tangible reality [2]. In this regard, several European projects (6G-INTENSE,[1] SUNRISE-6G,[2] etc.) have been devoted towards the embodiment of the 6G vision contributing to a plethora of universal standardization organizations including the 3rd Generation Partnership Project (3GPP) and European Telecommunications Standards Institute (ETSI) [3]. The common denominator in all these 6G-related standards and projects is the fact that Artificial Intelligence (AI) and Machine Learning (ML) serve as catalysts for driving innovation and paving the way toward achieving Zero Touch Service Management (ZSM) in future networks [4].

The dynamic nature of 6G networks, with constantly evolving data patterns, user behaviors, and environmental factors, poses significant challenges for embedded AI/ML systems [4]. As conditions shift rapidly, a common challenge known as *"Model Drift"* [6], [7] arises, threatening the stability and accuracy of AI-driven applications. As shown in Figure 1, model drift can manifest in several forms: For instance, drift may appear within data through shifts in a single feature (univariate drift) or across multiple features (multivariate drift) [8]. Another form, called label drift, occurs when changes appear in the target variable (i.e.,

[1]https://6g-intense.eu/
[2]https://sunrise6g.eu/

predictions) [9]. The most challenging type to detect is concept drift, which involves subtle shifts in the underlying relationships between features and target variables, rather than observable changes in data distributions [6], [9]. This makes concept drift much harder to identify compared to data drift, which is often apparent through visible alterations in input data patterns.

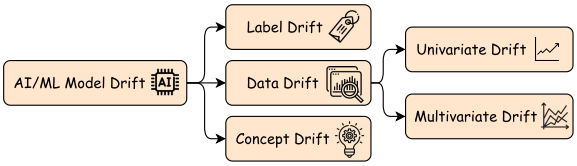


FIGURE 1. Taxonomy of AI/ML model drift forms.

A. MOTIVATION & RESEARCH GAP

A thorough examination of drift detection literature reveals that existing solutions are predominantly designed for supervised settings, relying on ground truth labels to identify model drift [6], [7], [12]. However, this reliance becomes impractical in dynamic environments where labels are scarce or unavailable. Unsupervised approaches, which monitor statistical changes in input data, provide label-independent alternatives but often struggle with concept drift and exhibit high False Positive Rates (FPR) [10], [11]. Recent advancements in Explainable AI (XAI) techniques, such as SHapley Additive exPlanations (SHAP) [16] and Layer-wise Relevance Propagation (LRP) [17], have improved detection accuracy by leveraging feature importance scores [13], [18], [19], [20]. Despite this progress, these methods face significant limitations: their high computational overhead makes them unsuitable for real-time applications, and they perform poorly on time series data, failing to capture univariate temporal dependencies and leading to high false-negative rates [15].

B. NOVELTY

The aforementioned gaps highlight a critical need, particularly for future networks and complex systems. To address this, we propose a novel framework capable of detecting multiple drift types, including univariate and multivariate data drift, label drift, and concept drift. To this end, we introduce a dual self-attention-based autoencoder that utilizes reconstruction error to identify both data and label drifts. Whereas for concept drift, our model incorporates a novel dual self-attention mechanism that fuses feature-based and temporal self-attention blocks, effectively capturing both temporal dependencies and relationships between features and predictions through attention scores. A key innovation of our approach is the use of a single autoencoder for detecting both data and label drift, rather than deploying separate models. This is based on the observation that both types of drift involve similar reconstruction tasks. By seamlessly integrating labels as additional inputs alongside features, we avoid the need for a dedicated label drift autoencoder. This design choice simplifies the architecture, reduces computational overhead, and preserves high detection accuracy. Another key contribution of our work is the introduction of dual self-attention mechanisms tailored for concept drift detection in telecommunications data, with a particular focus on time-series scenarios. Experiments on three open-source telecom datasets, encompassing both time-series and non-time-series formats, demonstrate the effectiveness of our approach, achieving higher detection accuracy while significantly reducing latency compared to existing state-of-the-art methods.

The main contributions of this paper can be summarized as follows:

- We provide an in-depth study of drift detection state-of-the-art, examining a broad spectrum of techniques, tools, standards, and methodologies developed over the past decade to give readers a thorough understanding of the drift detection field's evolution.
- We introduce, for the first time, a novel Dual Self-Attention Autoencoder (DSA-AE) model, designed to enhance drift detection in AI/ML systems.
- Specifically, the model leverages the autoencoder's capability to effectively identify various types of data and label drift, while utilizing inter-feature and temporal self-attention scores to detect the complex spatiotemporal fluctuations that are often the primary drivers of concept drift. This dual approach enables DSA-AE to capture subtle shifts across both data patterns and time dependencies, providing a comprehensive solution for robust drift detection.
- To the best of our knowledge, this work represents one of the initial efforts to adopt a dual self-attention strategy, enabling the detection of subtle, dynamic shifts across both data distribution and temporal dependencies, and offering a more comprehensive and resilient solution for drift detection.
- We validate the effectiveness of our proposed framework using three publicly available real-world telecommunications datasets (two time-series and one non-time-series), thereby ensuring a comprehensive evaluation across diverse data types and practical use cases.
- Empirical evaluation results vividly demonstrate that DSA-AE significantly outperforms existing drift detection methods, achieving up to 94.7% reduction in drift detection latency and over 13.6% improvement in detection accuracy.

C. PAPER ORGANIZATION

The rest of the paper is structured as follows: Section II delves into the latest advancements in drift detection literature, highlighting existing research gaps. In Section III, we introduce our DSA-AE drift detection framework design, system modeling as well as the algorithmic details. Section IV outlines the experimental setup, including the datasets, use

cases, and performance metrics employed for the experiments. Section V then details the results obtained from these experiments. Finally, Section VI wraps up the paper with conclusions and suggestions for future research.

II. LITERATURE REVIEW ANALYSIS

Over the last decade, a wide range of drift detection methods has been developed across numerous fields, such as data mining [23], healthcare [24], cybersecurity [25], and networking [4], [5]. These methods can be broadly categorized based on the availability of ground truth labels during inference: (i) supervised and (ii) unsupervised solutions. In this work, we have categorized recent solutions that leverage the field of XAI into a separate category, which can be classified as unsupervised, for the sake of clarity.

A. SUPERVISED SOLUTIONS

The majority of the existing drift detection solutions in the literature are supervised, as extensively documented in numerous studies [12], [27], [28], [29], [30]. These methods primarily rely on metrics derived from model accuracies or utilize ensemble models to detect performance degradation over time, typically indicated by a decrease in prediction accuracy. The core assumption behind these techniques is that true labels for incoming data are accessible, either immediately or shortly after the data is received. This reliance on labeled data significantly limits the scalability and applicability of supervised drift detection methods, particularly in dynamic environments where labels are sparse, delayed, or expensive to obtain. Consequently, while these methods are effective under ideal conditions, their dependence on the immediate availability of true labels renders them less practical for many real-world scenarios, where the timely and economical acquisition of labels cannot be guaranteed.

B. UNSUPERVISED SOLUTIONS

Unsupervised techniques, which operate without the need for true labels, are a vital component of the drift detection landscape [33], [36], [37]. Among the primary methods employed are statistical-based approaches, which utilize statistical tests or divergence metrics to compare two distributions, typically a reference against a new window [12], [31], [32], [34], [35]. A notable example is the work in [12], where the authors propose a drift detection and adaptation method based on adaptive and sliding window techniques for the Internet of Things field. Overall, these methods are highly versatile, model-agnostic, and require no external resources, leveraging every sample to calculate distances. Despite their advantages, statistical-based methods often struggle to detect true concept drift, particularly in scenarios involving spatiotemporal variations. Such complexities typically fall outside the detection capabilities of traditional sliding window methods, which often suffer from high false-negative rates.

It is also important to highlight a recurring ambiguity in the literature regarding the distinction between concept drift and data drift. In many works, including the aforementioned, techniques designed to detect data drift are labeled as concept drift detection methods. This interchangeable use of terminology, especially in the absence of clear conventions or standards, complicates the classification of drift detection approaches. To uphold scientific integrity, we adopted the original terminology used in the cited works, ensuring consistency with the authors' intent. This approach allowed us to fairly organize and classify the methods, as presented in Table 1.

C. EXPLAINABLE AI-BASED SOLUTIONS

In addition to traditional drift tracking methods, research on XAI-based drift detection methods provides evidence that drifts can be monitored through XAI methods like SHAP. Several studies [18], [19] revealed that drift detection could be explained by tracking features that cause significant changes in the distance between distributions. In [18], the authors use SHAP-based XAI along with a drift suspicion metric to enhance the reliability of drift detection in cybersecurity. Similarly, Koebler et al. [21] propose an explainability-based approach for performance monitoring, using optimal transport to detect distribution shifts and the Shapley method to identify key features. In [22], the authors introduce an application of XAI for diagnosing performance degradation in ML models that continuously learn from user engagement data.

While XAI-based solutions have succeeded in offering transparency into the features driving drift, they face a major limitation that hinders their practical use for drift detection, which is the high computational cost and time needed to compute SHAP values for all features [26]. This issue is compounded in scenarios involving high-dimensional input features and complex Deep Learning (DL) models, due to the exponential complexity of SHAP. As a result, these solutions are less viable for continuous monitoring in dynamic, fast-paced environments like 6G networks. Furthermore, recent studies have highlighted the limitations of XAI-based methods when applied to time series data, particularly in capturing univariate temporal dependencies [15]. These techniques often fail to uncover hidden causes of model errors specific to the model's internal logic or those that emerge uniquely during training on the same observed input. This restricts their effectiveness in providing insights into errors rooted in temporal patterns and dependencies, which are critical for accurate drift detection in time-sensitive applications [14].

D. DRIFT DETECTION TOOLS

Some of the state-of-the-art drift detection tools have made significant strides in addressing the challenges posed by model drift. For instance, Evidently AI[3] is known for its general data drift detection capabilities, while NannyML[4] excels at pinpointing the precise timing of shifts and evaluating their consequent effects on predictive accuracy. A notable

[3]https://www.evidentlyai.com/
[4]https://www.nannyml.com/

TABLE 1. Comparison of drift detection methods between relevant related works and our solution.

	Method	Data Drift	Label Drift	Concept Drift	Time-Series Adaptation	Realtime-Efficiency	For Networking
Supervised	[7]	✓	✗	✓	✗	●	✓
	[8]	✓	✗	✓	✗	●	✓
	[28]	✗	✗	✓	✗	●	✗
	[29]	✓	✗	✓	✗	◐	✗
	[30]	✗	✗	✓	✗	◐	✗
	[31]	✓	✗	✗	✓	●	✗
Unsupervised	[13]	✗	✗	✓	✗	●	✓
	[32]	✗	✗	✓	✓	●	✗
	[33]	✗	✗	✓	✓	●	✓
	[35]	✓	✗	✗	✗	●	✗
	[36]	✗	✗	✓	✗	●	✗
	[37]	✓	✓	✗	✗	◐	✗
	[38]	✗	✗	✓	✗	●	✗
XAI-based	[19]	✓	✗	✓	✗	○	✓
	[21]	✓	✗	✓	✗	○	✗
	[22]	✓	✗	✓	✗	○	✗
Our Method	**DSA-AE**	✓	✓	✓	✓	●	✓

● Adapted ◐ Needs Adjustments ○ Not Adapted ✓ Addressed ✗ Not Addressed

solution from NannyML proposes a concept drift detection method that continuously trains models on incoming serving data and compares the concepts they learn with those of the production model. If a significant difference is detected between the learned concepts, it indicates a potential concept shift. However, this approach has several key limitations. First, it assumes the availability of target labels for the serving data, which is often not the case in many real-world applications. Second, it is not designed to handle the complex intricacies of time-series data, which can limit its applicability in certain domains (e.g., telecommunications). Third, the solution is not open-source and is only available through the paid tier. Another noteworthy tool is Alibi-Detect,[5] an open-source Python library that provides various techniques for drift detection in data. It includes methods such as clustering, classification-based approaches, and statistical tests to identify changes in data distributions.

Overall, while existing tools have made significant progress in detecting data drift, they still fall short when it comes to identifying concept drift. To the best of our knowledge, there is no mature or widely adopted solution that effectively detects concept drift, especially in the context of complex spatiotemporal fluctuations, which are often the primary drivers of such drift.

E. DRIFT DETECTION IN TELECOMMUNICATIONS STANDARDS

In the evolving landscape of Beyond 5G networks (referred to as 5G-Advanced[6] in 3GPP), ensuring the reliability of AI and ML systems is paramount [47]. Standardization efforts have begun to address this need, with organizations such as ETSI taking significant steps to define and classify drift types. For example, ETSI's technical specifications in the context of Traceability for Trustworthy AI [43] provide high-level definitions of various drift types, their causes, and detection methods, with a particular focus on adaptation techniques like transfer and ensemble learning. ETSI TR (i.e., technical report) 104 proposes several performance metrics, including drift detection delay, which we have utilized in our performance evaluations in section V. Previous ETSI ZSM releases [44] partially addressed drift detection, focusing predominantly on data drift without fully detailing methodologies for concept drift. Additionally, current standards frequently oversimplify the drift detection phase by assuming the availability of labels and offering minimal detail on detection methods for such scenarios.

F. SUMMARY

Table 1 presents a summary of existing drift detection methods. It is important to highlight that, despite ongoing confusion in the literature regarding the distinction between concept drift and data drift, we have opted to retain the terminology used in the referenced works for the sake of scientific honesty. While recent research has made considerable progress in identifying specific types of drift individually, nonetheless, the approaches cited in the table exhibit at least one of these notable limitations: *(i) they fail to address all forms of drift within a unified solution, (ii) they lack adaptability to real-time online production systems, and (iii) they are not equipped to handle the specific constraints associated*

[5]https://docs.seldon.io/projects/alibi-detect/en/latest/
[6]https://www.3gpp.org/technologies/ran1-rel18

with time-series data in networking contexts. These constraints include the unique challenges posed by time-series data in networking applications, including the need to capture long-range temporal dependencies, handle high-volume real-time data streams, and maintain robustness under highly dynamic traffic patterns. Telecom data often arrives continuously and at scale, imposing strict requirements on latency and processing efficiency. Moreover, meaningful behavioral patterns such as anomalies or concept shifts frequently span extended time intervals, requiring models capable of capturing complex temporal dynamics. The data is also non-stationary, influenced by evolving user behavior, protocol changes, and shifting threat landscapes. Many existing methods are not well-suited to these conditions, as they are typically developed for general-purpose or static datasets. They often rely on batch retraining, fixed thresholds, or adaptive windowing, which can fail to effectively model the nuanced and persistent temporal characteristics of real-world networking environments.

This paper jointly tackles these challenges through the proposed DSA-AE framework, which serves as the centerpiece of our work. In particular, our dual attention-based drift detection method is designed to explicitly model both short and long-term dependencies in time-series data, enabling it to more effectively detect concept drift driven by complex temporal dynamics. Indeed, our method represents an important step forward, offering a novel approach that could inspire further research and refinement in model drift detection, particularly in data-rich, dynamic domains like networking, where traditional methods fall short.

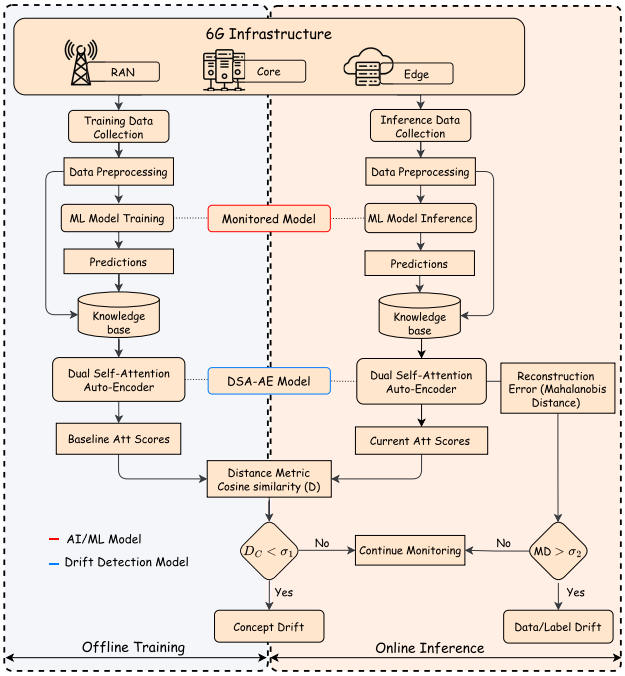


FIGURE 2. The Proposed DSA-AE based Drift Detection Framework. The DSA-AE framework comprises two phases: (i) an Offline Training phase, where input feature and target variable pairs are collected, and the DSE-AE model is trained on the knowledge base data to learn its spatiotemporal properties; and (ii) an Online Inference phase, where the drift detection model DSA-AE will be deployed to detect all forms of drift (data, label, and concept drift) within the new inference data.

III. PROPOSED METHODOLOGY

In this section, we introduce the proposed dual self-attention-based drift detection framework. We begin by exploring the underlying intuitions and insights that shaped our design, followed by a detailed exposition of the technical workflow, system modeling, algorithmic reasoning, as well as the components involved.

A. OVERALL FRAMEWORK WORKFLOW

The overall framework of the proposed DSA-AE drift detection method is illustrated in Figure 2. This framework consists of two main stages: Offline training and Online inference monitoring.

1) OFFLINE TRAINING

During the offline training phase, we begin by collecting a comprehensive dataset from various 6G network sources, including the RAN, Core, and Edge (based on the use case). This raw data undergoes rigorous preprocessing, involving normalization and feature selection, to ensure it is clean, consistent, and tailored to the specific use case and architectural model. Following preprocessing, the prepared dataset is utilized to train the ML model, laying the foundation for robust performance in subsequent stages. A pivotal aspect of this stage is the validation phase, where model predictions are meticulously mapped to the input features at each timestep. This mapping facilitates the creation of a comprehensive knowledge base, allowing for detailed tracking of both input features and the corresponding ML model predictions. Such a tracking mechanism is indispensable for effectively monitoring data drift (variations in input features) and label drift (shifts in predictions) as they evolve over time.

Next, we use the data collected in the knowledge base to train our DSA-AE model, incorporating both predictions and features. This model is designed to reconstruct the input data, including both features and predictions, while minimizing reconstruction error. The DSA-AE model learns to identify deviations from the norm by comparing reconstructed data to the original, leveraging reconstruction error with Mahalanobis Distance (MD) [39], calculating a drift score. This score provides a measure of the deviation between the predicted and actual data later in the Online phase. Additionally, during the validation process, baseline self-attention scores are recorded. These scores act as a reference point for detecting concept drift. By calculating the cosine distance [38] between the baseline attention scores and those observed during inference, the system can pinpoint significant shifts in the data's underlying patterns, signaling potential concept drift.

2) ONLINE MONITORING

In the online inference monitoring phase, after deploying the ML model, incoming inference data undergoes the same preprocessing steps as those applied during training. This includes normalization and feature selection, ensuring consistency in data preparation between the training and inference stages. The preprocessed data is then fed into the deployed ML model to generate predictions. Simultaneously, we capture the mapping between the inference input data and corresponding model outputs to create an inference knowledge base. This knowledge base is then processed by the DSA-AE drift detection model, which attempts to reconstruct the data.

To monitor data and label drift, we compute the MD of the reconstruction error produced by the DSA-AE model. This approach is effective because the MD measures the distance of a point from a multivariate distribution, accounting for correlations between features. If the MD exceeds a predefined threshold, denoted as σ_2, it signals a deviation in the data and/or labels. This prompts the system to flag potential drift and identify the source of the issue.

In addition to data and label drift, the DSA-AE model also tracks concept drift in the online phase. The current weighted attention scores from both self-attention blocks (feature and temporal self-attention) are captured during inference. These scores reflect the relationships between features and temporal patterns within the data. We then compute the cosine distance between these current attention scores and the baseline attention scores established during the validation phase. Cosine distance provides a robust metric for detecting concept drift [40]. If the computed cosine distance (a value close to 1 indicates high similarity and vice-versa with the value -1) falls below a predefined threshold σ_1 the system flags the presence of concept drift.

It is important to note that the thresholds σ_1 and σ_2 may vary across different use cases. These values depend on parameters such as the specific ML model architecture, the nature of the data, and the deployment environment, hence they should be fine-tuned accordingly to optimize drift detection.

B. DSA-AE MODEL ARCHITECTURE

1) AUTOENCODER

Autoencoders have garnered significant attention in the ML landscape for their capacity to learn efficient, compressed representations of data in an unsupervised manner [45]. These neural networks consist of an encoder, which compresses input data into a latent space, and a decoder, which attempts to reconstruct the original input from this compressed representation. Autoencoders have been successfully applied across a range of applications, including image denoising, data compression, and anomaly detection [39], [46].

2) SELF-ATTENTION MECHANISM

On the other hand, the self-attention mechanism, introduced in Transformer architectures [48], has proven to be exceptionally powerful for handling temporal dependencies and long-range interactions within data sequences. Unlike traditional recurrent or convolutional layers, self-attention allows the model to weigh each part of a sequence relative to every other part, regardless of the distance between them. This is particularly beneficial for tasks involving sequential or temporal data, such as time series analysis [41]. By assigning attention weights dynamically, the mechanism enables the model to capture dependencies over long sequences without the limitations of vanishing gradients or limited receptive fields faced by previous architectures.

3) DUAL SELF-ATTENTION AUTOENCODER

Attention-based autoencoders have shown success across various applications in the anomaly detection field, such as monitoring air quality sensors [45] and industrial pump performance [46]. These approaches use self-attention primarily to model temporal dependencies and leverage reconstruction error solely as an anomaly metric. However, the proposed DSA-AE framework goes a step further by utilizing two specialized attention blocks to distinctly capture spatial and temporal patterns within the data. Together, these self-attention blocks provide a comprehensive spatiotemporal perspective on the data as it flows through the autoencoder, effectively capturing the evolving patterns of the monitored time series.

As Figure 3 depicts, the core of our approach lies in leveraging two complementary self-attention blocks, one tailored to decipher the inter-feature spatial dependencies, and another dedicated to capturing the temporal patterns inherent in the data. By computing a weighted attention score that captures both spatial and temporal dependencies, our model establishes a nuanced indicator of concept drift using cosine distance. Furthermore, reconstruction error, assessed via MD, serves as a comprehensive measure for detecting both data and label drift. This integrated approach, combining spatiotemporal pattern recognition with unsupervised drift detection, enables our DSA-AE model to monitor complex drift patterns with precision, making it highly effective for adaptive detection in dynamic data environments like 6G networks.

C. SYSTEM MODEL

In this subsection, we provide an in-depth exploration of the system modeling details for the proposed DSA-AE framework introduced earlier. We begin by detailing the design of the autoencoder and the dual self-attention mechanism, highlighting their role in the system architecture. Additionally, we present the mathematical foundations of the distance metrics employed, specifically the MD with the reconstruction error and cosine similarity with the self-attention scores, which are integral to the drift detection process.

1) INPUT DATA MODELING

The proposed DSA-AE framework is designed to be highly adaptable to various types of input data. In this paper,

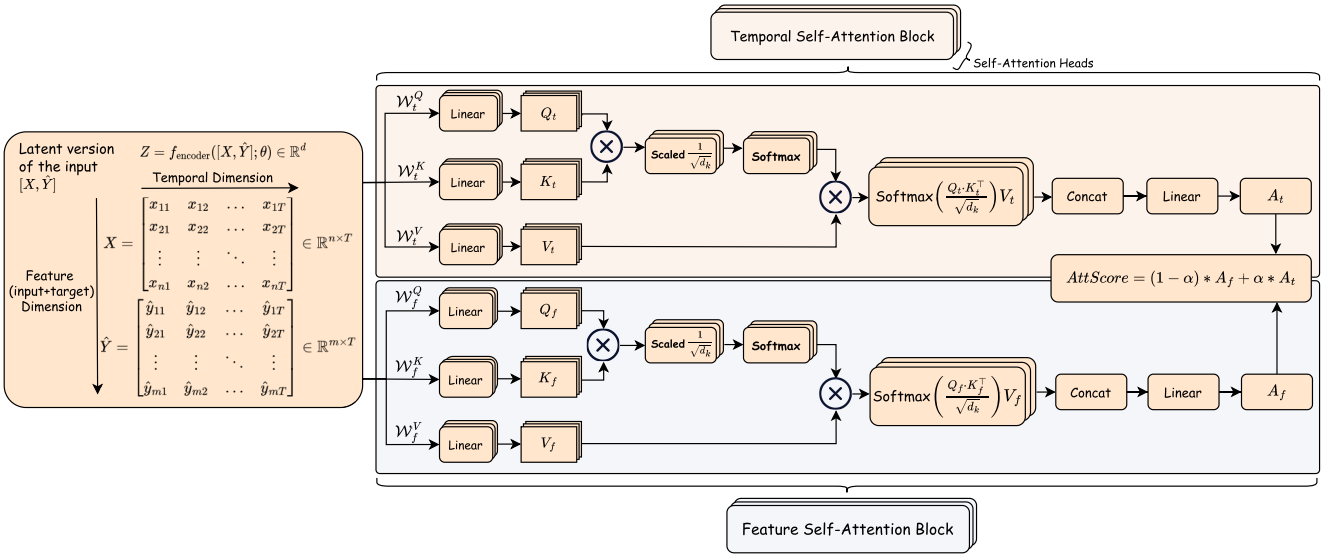


FIGURE 3. **High-level architecture of the proposed dual self-attention autoencoder model.**

we illustrate its application using time series input, which is particularly relevant given that most 5G/6G use cases rely on time-dependent data. Hence, in our modeling, the time series input is represented as a sequence of feature values across multiple time steps, paired with the corresponding predictions from the monitored model. For an input with n features, m target variables, and a total of T time steps, we define the input matrix X and prediction matrix $\hat{Y}$ as:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1T} \\ x_{21} & x_{22} & \dots & x_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nT} \end{bmatrix} \in \mathbb{R}^{n \times T}$$

$$\hat{Y} = \begin{bmatrix} \hat{y}_{11} & \hat{y}_{12} & \dots & \hat{y}_{1T} \\ \hat{y}_{21} & \hat{y}_{22} & \dots & \hat{y}_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{y}_{m1} & \hat{y}_{m2} & \dots & \hat{y}_{mT} \end{bmatrix} \in \mathbb{R}^{m \times T}$$

where each x_{ij} is the value of the i-th feature at time step $t = j$, and each $\hat{y}_{kt}$ is the prediction for the k-th target variable at time step t. By combining X and $\hat{Y}$ as $[X, \hat{Y}]$, we aim to capture both feature and label dynamics over time, making the model sensitive to changes in both the input data and predictions. This combined approach enables the model to simultaneously monitor both data drift and label drift over time. Notably, since all features are passed through the DSA-AE framework and the autoencoder reconstructs the entire input space, the drift detection criteria remain uniform across feature dimensions. As a result, the proposed method can effectively capture both univariate and multivariate data drift, whether the deviation impacts a single feature or multiple features concurrently, by identifying significant changes in the reconstruction error.

2) ENCODING PHASE

In the encoding phase, the autoencoder transforms the combined input matrix $[X, \hat{Y}] \in \mathbb{R}^{(n+m)\times T}$ into a compressed latent representation Z through a series of nonlinear transformations [45], [46]. This encoding process is parameterized by the encoder function f_{encoder}, which reduces the dimensionality of the data while preserving its essential patterns [45], [46]:

$$Z = f_{\text{encoder}}([X, \hat{Y}]; \theta) \in \mathbb{R}^d \tag{1}$$

where θ represents the encoder's parameters, and d is the dimension of the latent space, with $d \ll (n+m) \times T$. The latent representation Z serves as a compressed summary of the original time series data and model predictions, capturing both temporal and inter-feature relationships.

3) DECODING PHASE

After passing through the self-attention blocks (described in detail in the subsequent section), the enhanced latent representation Z' is obtained. The decoder then utilizes Z' to reconstruct the original input data $[X, \hat{Y}]$. This decoding step is represented as:

$$[\hat{X}, \hat{Y}_{\text{reconstructed}}] = f_{\text{decoder}}(Z'; \phi) \tag{2}$$

where f_{decoder} is the decoding function parameterized by ϕ, and $\hat{X} \in \mathbb{R}^{n\times T}$ and $\hat{Y}_{\text{reconstructed}} \in \mathbb{R}^{m\times T}$ are the reconstructed versions of the input features and monitored model's predictions, respectively [45].

4) RECONSTRUCTION ERROR AND LOSS FUNCTION

To evaluate the performance of the autoencoder and identify potential data and/or label drifts, we compute the reconstruction error between the original inputs and the reconstructed

outputs. For each feature value x_{ij} in X and each prediction $\hat{y}_{kt}$ in $\hat{Y}$, we calculate the reconstruction error as follows:

- **Input Reconstruction Error**: The reconstruction error for each feature at each time step is given by:

$$E_{X,ij} = |x_{ij} - \hat{x}_{ij}| \tag{3}$$

- **Prediction Reconstruction Error**: The error between each original prediction $\hat{y}_{kt}$ and its reconstruction $\hat{y}_{\text{reconstructed},kt}$ is calculated as:

$$E_{\hat{Y},kt} = |\hat{y}_{kt} - \hat{y}_{\text{reconstructed},kt}| \tag{4}$$

The total reconstruction loss serves as a key indicator of model performance and is subsequently used, in conjunction with the MD, to detect data and label drift. This reconstruction error is computed by aggregating the absolute differences between the original and reconstructed values across all features and time steps:

$$\begin{aligned}\mathcal{L}_{\text{reconstruction}} &= \sum_{i=1}^{n}\sum_{j=1}^{T} E_{X,ij} + \sum_{k=1}^{m}\sum_{t=1}^{T} E_{\hat{Y},kt} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{T} |x_{ij} - \hat{x}_{ij}| \\ &\quad + \sum_{k=1}^{m}\sum_{t=1}^{T} |\hat{y}_{kt} - \hat{y}_{\text{reconstructed},kt}|\end{aligned} \tag{5}$$

Overall, the reconstruction error provides the foundation for identifying data and label drift even in the time series data, enabling us to monitor changes in data patterns and model behavior robustly [39], [44], [46].

5) MAHALANOBIS DISTANCE (MD) FOR DATA/LABEL DRIFT DETECTION

In the proposed DSA-AE approach, we leverage the MD [39] to capture correlations between features and the covariance structure of the input data, specifically both features and labels. The MD is particularly suitable because it accounts for correlations and variable scales in multivariate data, addressing limitations of simpler metrics like Euclidean distance, which assumes independent features and equal variance [42]. This property makes it an invaluable tool in scenarios involving multivariate data and correlated errors. We employ MD as a baseline threshold to detect both data and label drift. After the autoencoder is fully trained during the offline training phase, its reconstruction error serves as an indicator of potential drift during inference. Any significant deviation from the expected reconstruction error, as measured by the MD, signals a change in the data distribution or label relationships. This approach effectively identifies distributional changes without requiring a temporal comparison of learned representations, unlike the method used for concept drift detection.

The MD equation can be defined as [39], [42]:

$$\text{MD}(\mathcal{L}_t, \boldsymbol{\mu}_{\mathcal{L}}, \Sigma_{\mathcal{L}}) = \sqrt{(\mathcal{L}_t - \boldsymbol{\mu}_{\mathcal{L}})^T \Sigma_{\mathcal{L}}^{-1} (\mathcal{L}_t - \boldsymbol{\mu}_{\mathcal{L}})}, \tag{6}$$

where:

- $\mathcal{L}_t$ represents the reconstruction errors for input features X and predictions $\hat{Y}$ at instance t:

$$\mathcal{L}_t = \begin{bmatrix}\mathbf{E}_{X,ij} \\ \mathbf{E}_{\hat{Y},kt}\end{bmatrix} = \begin{bmatrix}|x_{ij} - \hat{x}_{ij}| \\ |\hat{y}_{kt} - \hat{y}_{\text{reconstructed},kt}|\end{bmatrix}.$$

- $\boldsymbol{\mu}_{\mathcal{L}}$ and $\Sigma_{\mathcal{L}}$ are the mean vector and covariance matrix of the reconstruction errors, respectively, estimated from the baseline dataset in the validation phase (as illustrated in Figure 2).

Algorithm 1 provides an overview of the data/label drift detection algorithm, summarizing the previous steps. A drift is flagged when this distance exceeds a predefined threshold, σ_2, signaling a potential shift in data or label distribution.

Algorithm 1 Data and Label Drift Detection

1: **Input:** $X \in \mathbb{R}^{N\times d}$, N is the sequence length and d is the number of features.
2: Initialize the autoencoder f_{enc} and f_{dec}.
3: **Encoding**

$$Z = f_{\text{enc}}(X)$$

4: **Decoding**

$$\hat{X} = f_{\text{dec}}(Z)$$

5: **Reconstruction Error Calculation**

$$\begin{aligned}\mathcal{L}_{\text{reconstruction}} &= \sum_{i=1}^{n}\sum_{j=1}^{T} |x_{ij} - \hat{x}_{ij}| \\ &\quad + \sum_{k=1}^{m}\sum_{t=1}^{T} |\hat{y}_{kt} - \hat{y}_{\text{reconstructed},kt}|\end{aligned}$$

6: **Mahalanobis Distance Calculation**

$$\text{MD}(\mathcal{L}_t, \boldsymbol{\mu}_{\mathcal{L}}, \Sigma_{\mathcal{L}}) = \sqrt{(\mathcal{L}_t - \boldsymbol{\mu}_{\mathcal{L}})^T \Sigma_{\mathcal{L}}^{-1} (\mathcal{L}_t - \boldsymbol{\mu}_{\mathcal{L}})}$$

7: **Drift Detection**
8: If $\text{MD}(\mathcal{L}_t, \boldsymbol{\mu}_{\mathcal{L}}, \Sigma_{\mathcal{L}}) > \sigma_2$, flag data or label drift.

D. DUAL SELF-ATTENTION MODELING

In the latent space, we employ two distinct self-attention blocks: **Feature Self-Attention** and **Temporal Self-Attention** as illustrated in Figure 3 and detailed in Figure 4. The feature self-attention block focuses on identifying dependencies between different features at each time step, enabling the model to capture correlations and interactions across features. Mathematically, feature self-attention is computed over the latent vectors for each time step, producing feature attention scores that weigh the importance of each feature with respect to others in the latent space. On the other hand, the temporal self-attention block captures dependencies across time steps for each feature, allowing the model to recognize patterns and shifts over time. This is critical for modeling time series data, as it enables the model to capture long-term temporal dependencies and detect subtle changes

in data trends. As showcased in Figure 4, the attention scores from these blocks are averaged to provide an indicator for concept drift.

1) TEMPORAL SELF-ATTENTION BLOCK

The Temporal Self-Attention Block is designed to capture the dependencies over time in the latent space. We define the latent space embedding matrix $Z \in \mathbb{R}^{N\times d}$, where N is the sequence length and d is the feature dimension.

a: LINEAR TRANSFORMATIONS FOR QUERIES, KEYS, AND VALUES

First, we apply linear transformations to the latent matrix Z to generate the Queries (Q_t), Keys (K_t), and Values (V_t) matrices for the temporal attention mechanism [48]:

$$Q_t = ZW_t^Q, \quad K_t = ZW_t^K, \quad V_t = ZW_t^V$$

where $W_t^Q, W_t^K, W_t^V \in \mathbb{R}^{d\times d_k}$ are learned weight matrices, and d_k is the dimensionality of the queries and keys.

b: SCALED DOT-PRODUCT ATTENTION

Next, we compute the attention scores by taking the dot product between Q_t and the transpose of K_t, and then scaling by the square root of d_k [46], [48]:

$$\text{Attention scores}_t = \frac{Q_t K_t^\top}{\sqrt{d_k}} \tag{7}$$

This scaling step is necessary to prevent the dot product values from growing too large as the dimensionality d_k increases. By normalizing the dot product with $\sqrt{d_k}$, we ensure more stable and effective learning [40], [46].

Next, we apply the softmax function to normalize these attention scores [48]:

$$\alpha_t = \text{softmax}\left(\frac{Q_t K_t^\top}{\sqrt{d_k}}\right) \tag{8}$$

The temporal attention output A_t is obtained by multiplying the attention weights α_t with the value matrix V_t:

$$A_t = \alpha_t V_t \tag{9}$$

c: MULTI-HEAD ATTENTION

As we use multi-head attention, we split Q_t, K_t, and V_t into multiple heads, compute the attention for each head independently, concatenate the heads, and apply a linear projection to combine the outputs [48]:

$$A_t = \text{concat}(head_1, \ldots, head_h)W_t^O \tag{10}$$

where $W_t^O \in \mathbb{R}^{d_h\times d}$ is a learned projection matrix.

2) FEATURE SELF-ATTENTION BLOCK

The second component, named the Feature Self-Attention Block, captures the dependencies across features in the latent space. Similar to the temporal block, we first apply linear transformations to Z to generate queries (Q_f), keys (K_f), and values (V_f) matrices for feature attention. After that, we compute the attention scores for features by taking the scaled dot product of Q_f and the transpose of K_f, finishing with the softmax and concatenation of the different heads (eq. 10). Notably, the feature self-attention scores are calculated as:

$$\alpha_f = \text{softmax}\left(\frac{Q_f K_f^\top}{\sqrt{d_k}}\right) \tag{11}$$

$$A_f = \alpha_f V_f \tag{12}$$

3) COMBINING TEMPORAL AND FEATURE ATTENTION OUTPUTS

The resulting attention scores, A_t for temporal and A_f for feature attention, represent each block's focus on different data aspects. These scores are then combined into a unified attention score (AttScore) through a weighted sum, which allows the model to balance the importance of temporal and feature dependencies:

$$\text{AttScore} = (1-\alpha)A_f + \alpha A_t \tag{13}$$

For instance, if we wish to give more weight to the temporal attention (A_t) compared to the feature attention (A_f), we can set α to a value in the range $0.5 < \alpha < 1.0$; the other way around applies if we set α to a value in the range $0 < \alpha < 0.5$.

4) COSINE SIMILARITY FOR CONCEPT DRIFT DETECTION

For concept drift detection, we leverage DSA-AE's weighted self-attention scores with cosine similarity distance [38] to construct an accurate concept drift metric. The latter allows for a more meaningful comparison of attention weights by focusing on the relative importance of different features. This helps the model focus on the most relevant features or time steps without being influenced by their absolute scale, making it a robust metric for assessing the significance of various parts of the input during attention-based tasks [40].

Let $\text{A}_{\text{baseline},t}$ represent the attention scores for the model after the offline training phase (as depicted in Figure 2), which we refer to as the baseline model. This model is trained on the available data and is used as a reference for drift detection. On the other hand, $\text{A}_{\text{inference},t}$ represents the attention scores for the same DSA-AE model after it has been deployed for inference, which we refer to as the inference model. Essentially, both terms refer to the same model, but at different stages, before (baseline) and after (inference) deployment. These attention scores are obtained from both the temporal and feature self-attention blocks. To track the directional changes in feature dependencies, we compute the cosine similarity between $\text{A}_{\text{baseline},t}$ and $\text{A}_{\text{inference},t}$.

The cosine similarity distance is defined as [38]:

$$\text{D}_{Cosine} = \frac{\text{A}_{\text{baseline},t}^\top \text{A}_{\text{inference},t}}{\|\text{A}_{\text{baseline},t}\|\,\|\text{A}_{\text{inference},t}\|} \tag{14}$$

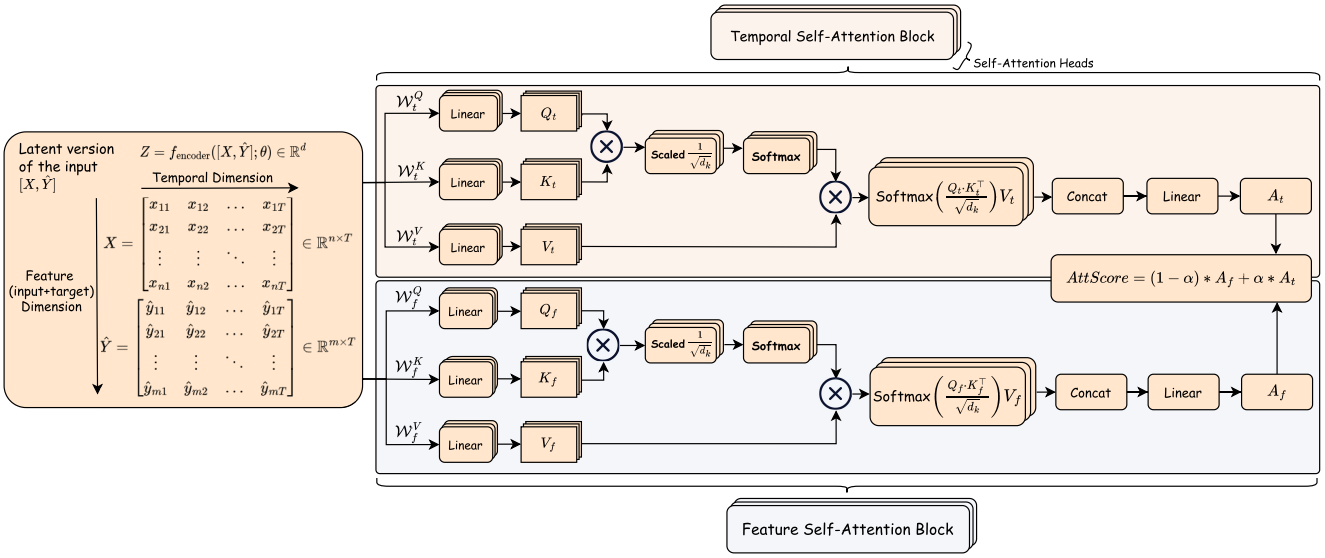


FIGURE 4. The Proposed Dual Self-Attention Mechanism Architecture. The DSA mechanism comprises two self-attention blocks: one focuses on capturing temporal fluctuations in the data, while the other is tailored to learn the spatial dependencies between input and target features.

Expanding the norms, this can also be expressed as [38], [40]:

$$\mathrm{D}_{Cosine} = \frac{\mathrm{A}_{\text{baseline},t}^{\top} \mathrm{A}_{\text{inference},t}}{\sqrt{\mathrm{A}_{\text{baseline},t}^{\top} \mathrm{A}_{\text{baseline},t}} \cdot \sqrt{\mathrm{A}_{\text{inference},t}^{\top} \mathrm{A}_{\text{inference},t}}} \tag{15}$$

This measure quantifies the angle between the two vectors, capturing shifts in the focus of the attention mechanism. A small cosine similarity value (D_{cosine}) indicates a significant change in the model's attention patterns, suggesting a potential shift in feature relationships, also known as concept drift.

To this end, Algorithm 2 summarizes the concept drift detection process. To identify concept drift, the current attention score $\mathrm{A}_{\text{inference},t}$ is compared with a baseline attention score $\mathrm{A}_{\text{baseline},t}$, recorded during the model's validation phase, using the cosine similarity (eq. 15). This similarity measure is particularly sensitive to directional changes in the attention vectors, making it ideal for detecting shifts in inter-feature dependencies without sensitivity to magnitude. If the cosine similarity between the current and baseline attention scores falls below a specified threshold σ_1, a concept drift is flagged. This method ensures that the model remains responsive to subtle shifts in feature relationships, thus maintaining robustness in dynamic environments where spatiotemporal patterns may evolve.

E. COMPLEXITY & THRESHOLDS ANALYSIS

Overall, the proposed algorithms do not involve iterative optimization during inference and therefore do not raise convergence issues at detection time. Algorithm 1 performs data and label drift detection by executing a forward propagation through the encoder-decoder architecture to obtain reconstruction errors, followed by MD evaluation. Algorithm 2 uses temporal and feature self-attention to extract spatiotemporal patterns, with concept drift detected through cosine similarity against baseline attention scores. The computational complexity of our approach is near-optimal for the tasks involved. Reconstruction and Mahalanobis computations scale with $\mathcal{O}(Nd + d^2)$, offering efficiency with respect to both data dimensionality and sample size. Attention-based operations scale with $\mathcal{O}(N^2 d)$, consistent with the theoretical lower bounds of attention mechanisms. The model is trained offline, and inference is lightweight, involving a single forward pass per input window. This design ensures reliable and scalable drift detection with minimal runtime overhead. Further complexity optimization remains an avenue for future work.

The thresholds for detecting drift in the proposed model are dynamically determined during the offline training phase based on the type of drift being monitored. For data and label drift, the threshold σ_2 is determined using the reconstruction error, which is calculated by comparing the original and reconstructed data. The MD is then used to assess the deviation from the expected model behavior, with the threshold adjusted to balance false positives and false negatives. For concept drift, the threshold is determined using the cosine similarity between the baseline and current attention scores, derived from the Temporal and Feature Self-Attention mechanisms. If the cosine similarity falls below a predefined threshold σ_1, concept drift is flagged. This threshold is influenced by the variability in attention scores over time, and it is calibrated to minimize the risk of detecting spurious concept drift while maintaining sensitivity to actual changes in the underlying data distribution. The thresholds for all types of

Algorithm 2 Concept Drift Detection Using DSA Mechanism

Input: $X \in \mathbb{R}^{N\times d}$, baseline attention scores $\alpha_{\text{baseline},t}$ from validation.

Apply encoding transformations to obtain latent representation Z:

$$Z = f_{\text{enc}}(X)$$

Compute **Temporal Self-Attention** queries, keys, and values:

$$Q_t = ZW_t^Q, \quad K_t = ZW_t^K, \quad V_t = ZW_t^V$$

Compute temporal attention output:

$$\alpha_t = \text{softmax}\left(\frac{Q_t K_t^\top}{\sqrt{d_k}}\right), \quad A_t = \alpha_t V_t$$

Compute **Feature Self-Attention** queries, keys, and values:

$$Q_f = ZW_f^Q, \quad K_f = ZW_f^K, \quad V_f = ZW_f^V$$

Compute feature attention output:

$$\alpha_f = \text{softmax}\left(\frac{Q_f K_f^\top}{\sqrt{d_k}}\right), \quad A_f = \alpha_f V_f$$

Compute the weighted attention score:

$$\text{AttScore}_t = \alpha A_t + (1-\alpha)A_f$$

Compute the **cosine similarity** baseline and current attention scores:

$$\text{D}_{Cosine} = \frac{A_{\text{base},t}^\top A_{\text{inf},t}}{\sqrt{A_{\text{base},t}^\top A_{\text{base},t}}\sqrt{A_{\text{inf},t}^\top A_{\text{inf},t}}}$$

Drift Detection

If $\text{D}_{cosine} < \sigma_1$, flag concept drift.

drift are sensitive to factors such as window size, training data characteristics, and the trade-off between detection accuracy and computational efficiency. In our approach, threshold selection follows an expert-driven and systematic process, determined through careful evaluation of drift detection performance with a particular focus on optimizing the F1 score to balance precision and sensitivity. This tuning process also considers the specific requirements of the use case and the characteristics of the dataset, enabling a practical trade-off between detecting true drifts and minimizing false positives. As a result, the selected thresholds are well-aligned with the operational goals and constraints of the application.

IV. EXPERIMENTS EVALUATION

This section presents a comprehensive overview of our experimental evaluation. We start by describing datasets used in the study, followed by a summary of the ML/DL models employed to assess the effectiveness of our drift detection scheme. Finally, we outline the experimental setup and the evaluation metrics applied in the analysis.

A. DATASETS

To carry out our experiments and enhance their realism and applicability to real-world production environments, we utilize three distinct, publicly available datasets from the telecommunications landscape. For diversity, two of the datasets are time series data. The datasets used in our study are as follows:

- **Milano Dataset [49]**: Collected by Telecom Italia over the span of a year, this dataset contains information on user connectivity events. Our experiments focused on the volume of data exchanged with users. We included the weekday to account for traffic variability across different days, and the hour to capture the differences between daytime and nighttime traffic. The internet data metric is proportional to the traffic volume, enabling realistic analysis while concealing actual traffic values.
- **Ireland Dataset [50]**: This realistic 5G trace dataset was collected from a major Irish mobile operator spanning several days in 2019 and 2020. It includes two mobility patterns (static and car) for three application traces, including file download and two streaming applications, Amazon Prime and Netflix. The dataset comprises client-side cellular Key Performance Indicators (KPIs), which include channel-related metrics, context-related metrics, cell-related metrics, and throughput information. For our experiments, we preprocessed the dataset to focus on throughput-related KPI features in both static and drive modes, with timesteps recorded by day, hour, and minute, and additional features such as Longitude, Latitude, Speed, CellID, NetworkMode, RSRP (Reference Signal Received Power), RSRQ (Reference Signal Received Quality), SNR (Signal-to-Noise Ratio), CQI (Channel Quality Indicator), RSSI (Received Signal Strength Indicator), DL_bitrate, UL_bitrate, and State.
- **EURECOM Dataset [51]**: This dataset, obtained from EURECOM's 5G facility, utilizes the OpenAirInterface (OAI)[7] Core network to monitor the performance of the Access and Mobility Management Function (AMF), a pivotal control plane component. The AMF manages essential functions including registration, connection, mobility management, and access authentication and authorization. This dataset evaluates AMF resource usage by measuring CPU and RAM usage, registration times, and the allocation of resources per request.

The three datasets are divided chronologically into training and test sets, as shown in Table 2. The training set comprises 80% of the data, while the remaining 20% is reserved for testing. Finally, the latter also displays the drift percentage for each dataset utilized to conduct the trials.

[7] https://openairinterface.org/oai-5g-core-network-project/

TABLE 2. Summary of datasets characteristics, including training and test set sizes, feature dimensions, and drift percentage.

Dataset	Train	Test	Dimension	Drift (%)
Milano	7026	1758	6	23.4
Ireland	1500	624	26	31.5
EURECOM	17130	4283	15	14.6

B. AI/ML USE CASES

To assess the effectiveness of our proposed DSA-AE drift detection framework, we conducted an extensive series of experiments across three use cases, carefully selecting benchmarks appropriate to the nature of each dataset. Specifically, we employed Long Short-Term Memory (LSTM) models for the time-series use cases (i.e., Milano and Ireland datasets), and a Feedforward Neural Network (FNN) model for the non-time-series use case (i.e., EURECOM dataset). This distinction between LSTM and FNN benchmarks reflects our intent to evaluate the framework comprehensively across both temporal and non-temporal data scenarios, ensuring a fair and representative comparison. The use cases explored in these trials include:

- **Traffic Forecasting**: We trained an LSTM model for the Milano dataset to forecast traffic volume, using weekday and hour information to account for temporal traffic variations.
- **Throughput Forecasting**: Similarly, we used an LSTM model on the Ireland dataset to predict download and upload bitrates across static and mobile states.
- **Core Network Resource Usage Prediction**: For EURECOM dataset, we trained a FNN model to predict CPU and RAM usage for the AMF in the core network, ensuring efficient handling of UE registration requests without breaching SLAs.

C. EXPERIMENT SETUP & EVALUATION METRICS

In our experimental setup, we implemented the DSA-AE model using PyTorch 2.0.1. The DSA-AE architecture consists of an encoder, a dual self-attention mechanism, and a decoder. We set the input dimension to match the number of features in the training data, with hidden and latent dimensions of 64 and 32, respectively. The model was trained using mean squared error loss and optimized with the Adam optimizer at a learning rate of 0.001. The training process spanned 50 epochs, with the model showing progressive improvement in minimizing the loss across training batches. This setup ensures robust and efficient drift detection in dynamic environments. The experiments were conducted on a machine equipped with an AMD 8-Core 3.2 GHz CPU, 16 GB of RAM, and an NVIDIA GeForce RTX 3050 Ti GPU. The AI/ML models (two LSTMs and a FNN) were implemented using TensorFlow 2.17.0 in a Python 3.9.7 environment with CUDA support for GPU acceleration. Lastly, we leveraged a mixture of conventional drift detection metrics, including FPR, recall, precision, and F1 scores [28], [52], along with standardized metrics (e.g., drift detection delay in ETSI [53]), for a more comprehensive and diverse evaluation.

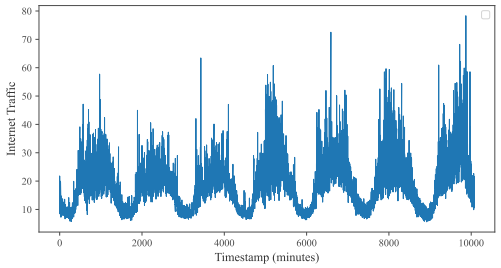


FIGURE 5. Internet traffic patterns in milano dataset - time series.

V. EXPERIMENTAL RESULTS

In this section, we present the results of three experiments. We begin by analyzing the data patterns across the three datasets. Following that, we extensively test the proposed DSA-AE drift detection method across all drift forms, including data, label, and concept drift, with varying parameters. We then provide a comparative analysis of our DSA-AE method with three existing state-of-the-art drift detection methods. Finally, we conclude with a discussion of the findings, giving insights into key lessons learned.

A. ANALYSIS OF TEMPORAL PATTERNS

The temporal analysis of our datasets reveals distinct behavioral patterns across different network scenarios. The Milano dataset (illustrated in Figure 5) exhibits clear temporal dependencies, characterized by recurring patterns and systematic variations over time, particularly evident in feature distributions across different time periods. Similarly, the Ireland dataset demonstrates strong temporal correlations across all three applications as showcased in Figure 6, with distinctive cyclic patterns and time-based dependencies in network usage metrics. In contrast, the AMF dataset depicted in Figure 7 shows notably different characteristics, lacking significant temporal dependencies where the data points appear more stochastic and event-driven rather than time-dependent. This diversity in temporal characteristics across our experimental datasets provides an ideal testing ground for our proposed DSA-AE drift detection method.

B. AI MODELS VALIDATION

Figure 8 presents the predictive performance of the AI models using the different datasets for each use case. In Figure 8a, we can observe the actual versus predicted internet usage values for the Milano dataset, which is time series data. The predicted values closely follow the actual values over time, demonstrating the LSTM model's accuracy in capturing the temporal dependencies inherent in internet usage data. Similarly, Figure 8b shows that the LSTM model accurately predicts the download bitrate for the Ireland dataset. Where the model's predictions align closely with the actual data points, indicating that the LSTM effectively captures patterns

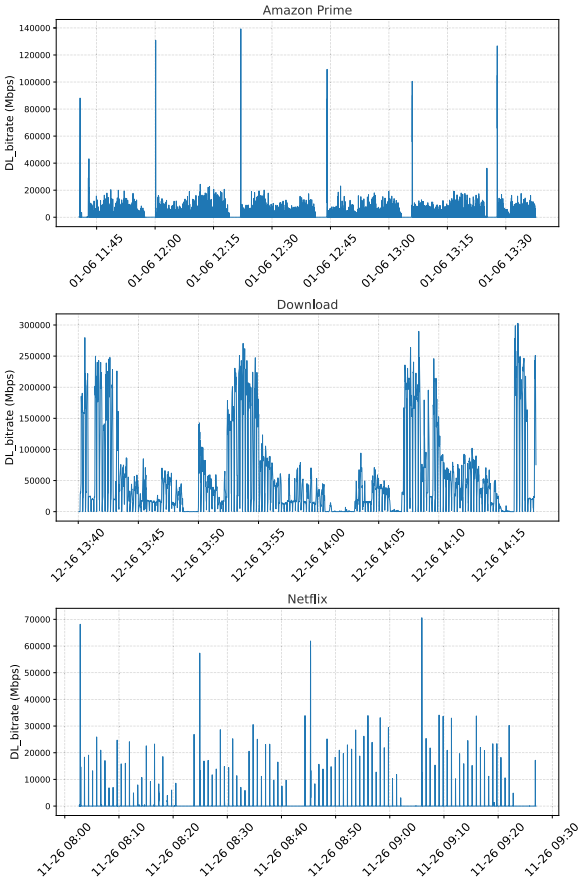


FIGURE 6. Throughput patterns of three applications of ireland dataset - time series.

in the time-dependent features of network throughput. On the other hand, Figure 8c displays the results of the FNN model predicting CPU usage in the EURECOM dataset, which is non-time-series data. The FNN model successfully tracks CPU usage trends, aligning well with actual values and further demonstrating its suitability for structured, non-temporal data.

C. LIMITATION OF XAI-BASED CONCEPT DRIFT DETECTORS WITH TIME SERIES DATA

To further investigate the claims in section II-C regarding the limitations of XAI drift detection for time series data, we conducted a series of experiments using two widely adopted XAI-based methods: SHAP and LRP.

Figure 9 illustrates a bar chart of SHAP values over different time steps in the Milano time series data prediction. The key observation is the progressive increase in SHAP values towards the prediction point, starting low and peaking at the most recent time step. Moreover, Figure 10 depicts SHAP scores for a sequence of time steps in the Ireland dataset, highlighting each step's contribution to model predictions. Time steps 0 to 7 show low SHAP values, indicating minimal impact, while time steps 8 to 11 (i.e., latest timesteps) have

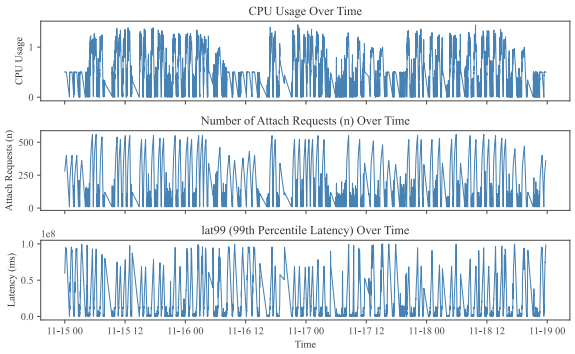


FIGURE 7. AMF resource usage of EURECOM dataset.

increasingly negative values, with the most significant impact at step 11, emphasizing the latter part of the sequence.

To explore whether the SHAP findings can be generalized, we extended our evaluation to another XAI method (i.e., LRP) and tested our time series models using this technique. Figures 11 and 12 illustrate the LRP relevance scores of LSTM model use cases across two time series datasets (Milano and Ireland). As can be clearly observed, the relevance scores exhibit similar patterns to SHAP when dealing with time series data, where the XAI method focuses most often on the last part of the sequence as an indicator of the next timestep. The LRP results confirm the results of SHAP where ML Model tends always in each sequence to focus on the last timesteps to predict the next value at t+1. The extensive results we got from SHAP and LRP experiments can be justified by the fact that XAI-based methods were not originally designed to detect drifts and were primarily designed for NLP and text data, rather than time series data [14], [15].

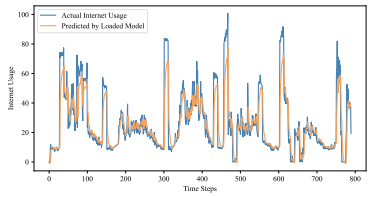
D. THE PROPOSED DSA-AE DRIFT DETECTION VALIDATION

In this subsection, we evaluate the effectiveness of our DSA-AE model in detecting various types of drift. First, we assess the model's capability to detect data and label drift. Finally, we examine the model's ability to identify concept drift, utilizing the dual self-attention mechanism.

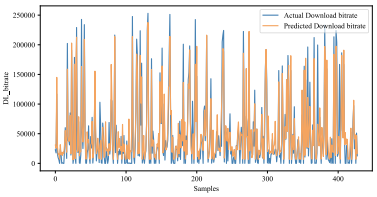
1) DATA DRIFT DETECTION

In this experiment, we aim to test the effectiveness of the proposed DSA-AE method for detecting data drift, using a traffic forecasting use case with the Milano dataset. During inference, we introduced a simulated drift (23.4% as illustrated in Table 2) by exposing the LSTM model to abnormal internet traffic patterns and evaluated whether the DSA-AE drift detector could successfully identify this drift.

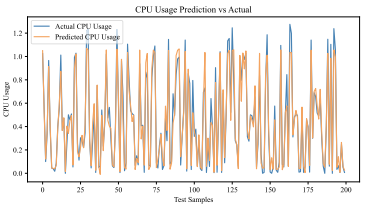
Notably, Figure 13 demonstrates the DSA-AE's performance in data drift detection across three subplots. The top subplot shows the original and drifted data distributions used to generate a drift event, with the original traffic patterns (blue) displaying relatively stable behavior and the drifted



(a) LSTM Model Validation: Internet Traffic Predicted vs Actual - Milano dataset.



(b) LSTM Model Validation: Download bitrate Predicted vs Actual - Ireland dataset.



(c) FNN Model Validation: CPU Usage Predicted vs Actual - EURECOM dataset.

FIGURE 8. LSTM and FNN model validation results across different datasets and use cases.

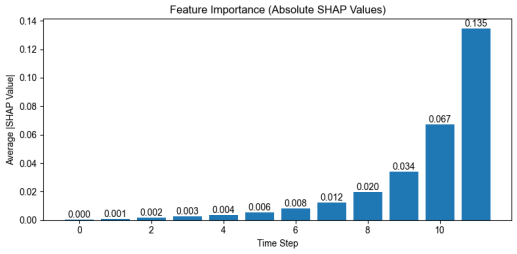


FIGURE 9. Average SHAP importance scores of LSTM model with Milano time series dataset.

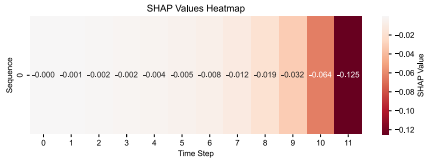


FIGURE 10. Heatmap of SHAP values of one Sequence from throughput predictions use case of ireland dataset.

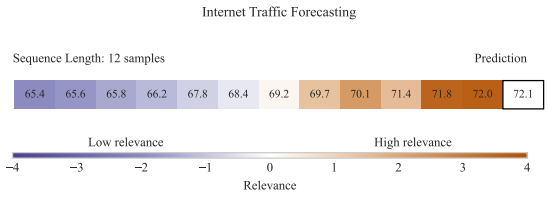


FIGURE 11. LRP Relevance Scores of one sequence using traffic forecasting use case with Milano dataset.

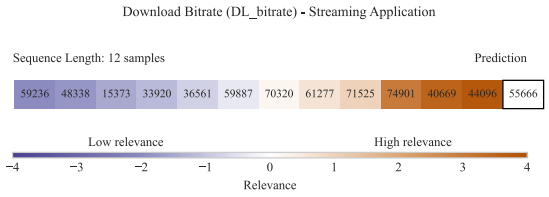


FIGURE 12. LRP Relevance Scores of one sequence using Bitrate predictions use case with Ireland dataset.

data (red) exhibiting significant deviations, clearly indicating abnormal traffic behavior. The middle subplot shows the DSA-AE model's detection response using MD. A drift

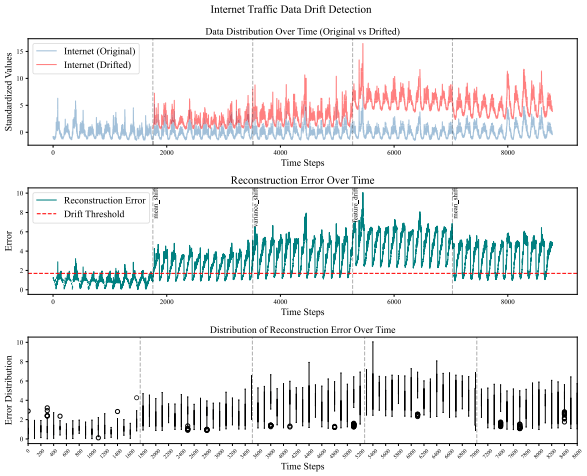


FIGURE 13. DSA-AE Data Drift Detection Harnessing the reconstruction error.

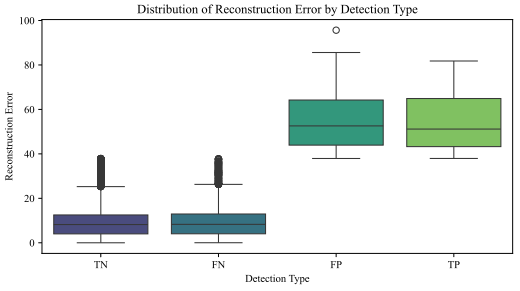


FIGURE 14. True Positive, False Positive, False Negative based on the change of the reconstruction error.

threshold at σ_2 separates normal from anomalous behavior. This threshold is not fixed but varies depending on the characteristics of the dataset (i.e., use case), as the reconstruction error is directly calculated by comparing the input data with the reconstructed output. Consequently, the threshold is inherently dependent on the training data used in each case, adapting to the underlying distribution and variability of that data. For the traffic forecasting use case with the Milano dataset, the reconstruction error remains consistently below the threshold prior to the drift event ($t <$

2000), indicating system stability. As drift occurs, the error surpasses the threshold, effectively signaling the onset of the drift. The lower subplot further illustrates this behavior through box plots, capturing the distribution of reconstruction errors over time and highlighting the shift in data characteristics. Pre-drift, the error distribution is tight with low variance, reflecting normalcy. Post-drift, the distribution widens, median values rise, and outliers increase, indicating a shift in error characteristics. These results demonstrate the DSA-AE's effectiveness in detecting data drift, highlighting its sensitivity and robustness.

To gain deeper insights into the accuracy of the proposed reconstruction error metric within our framework, we conduct further analysis. Hence, Figure 14 shows the boxplot distribution of reconstruction error across four detection types: True Negative (TN), False Negative (FN), False Positive (FP), and True Positive (TP). In this plot, we observe that TN and FN instances have significantly lower reconstruction errors, with their interquartile ranges concentrated around lower values. This suggests that the DSA-AE model reconstructs these cases with relatively high accuracy.

2) LABEL DRIFT DETECTION

In this experiment, we study the effectiveness of the proposed DSA-AE drift detection method for identifying label drift. As detailed previously, our framework captures the predictions of the monitored model alongside the input features and stores them in the knowledge base. This architectural design enables comprehensive monitoring of both data and label drift through attention-based reconstruction error analysis. To evaluate the method's capability in detecting label drift, we simulated drift conditions in one of the prediction sequences of the LSTM traffic forecasting model. Figure 15 presents a heatmap visualization of the reconstruction error intensity over time, providing insights into the temporal dynamics of label drift detection. The heatmap reveals distinct patterns of reconstruction error intensities across different time steps. A pronounced band of high-intensity errors (deep red) at time steps 5-7, with values exceeding 22.5, indicates significant label drift. Indeed, the DSA-AE model effectively detects label drift through these reconstruction error intensities.

3) CONCEPT DRIFT DETECTION

One of the major contributions of this paper is addressing the limitations of XAI-based concept drift detection methods. To manage the complex spatiotemporal dependencies commonly found in network data, we harnessed a dual self-attention mechanism specifically to deal with this, as detailed in section III-D.4. In this regard, to assess the effectiveness of the proposed DSA-AE model for concept drift detection in a time series context, we conducted an experiment where we exposed the throughput prediction LSTM model to concept drift with complex spatiotemporal properties.

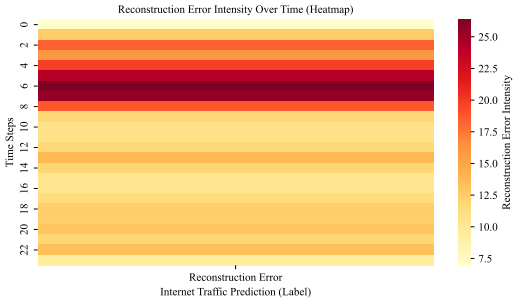


FIGURE 15. DSA-AE Reconstruction error Heatmap after Label Drift.

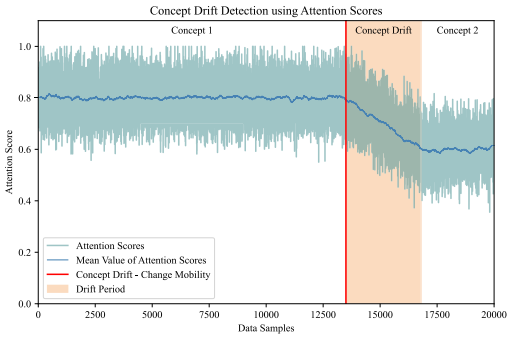


FIGURE 16. Concept Drift Detection using attention scores of the proposed DSA-AE framework.

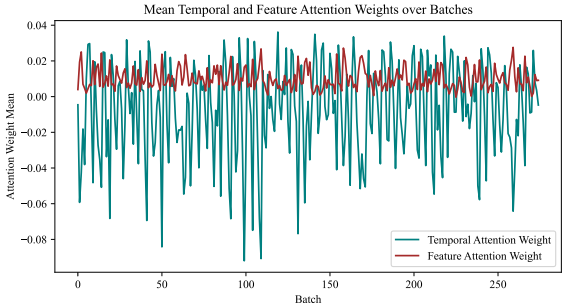


FIGURE 17. Comparison of Temporal Attention and Feature Attention weight distributions of the Ireland dataset (time series).

To achieve this, we leveraged the diversity of the Irish Mobile Operator dataset. In the offline training phase, we trained the LSTM model on data from a streaming application (Netflix) that exhibits specific temporal properties, namely (i) a 2-second interval between observations after preprocessing and normalization process as well as (ii) a static mobility pattern. In the online inference phase, we then introduced data from a different application, a download service, which has similar feature dimensions but different temporal and mobility characteristics. This dataset featured a longer observation interval of 30 seconds and a shift in mobility patterns from static to driving mode, introducing novel

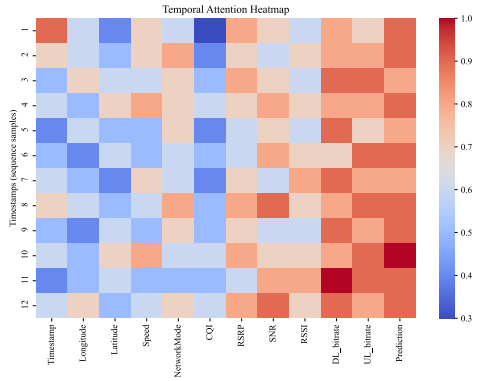


FIGURE 18. Temporal Attention Heatmap of different features (Ireland dataset).

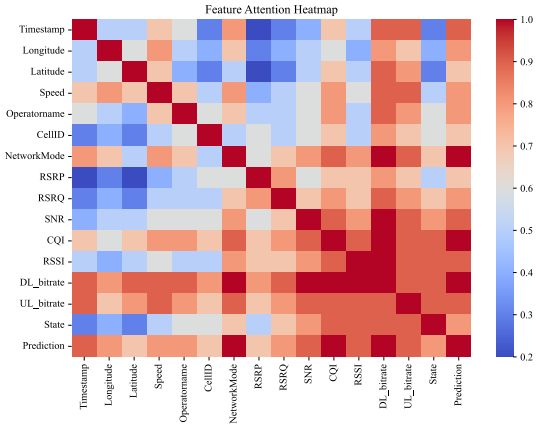


FIGURE 19. Inter-Feature Attention Heatmap of different features (Ireland dataset).

spatiotemporal properties that the model had not encountered during training. This type of complex concept drift is particularly challenging to detect. Because the input data and predictions remain within similar distributions, as a result, traditional data and label drift detectors fail to flag it.

Figure 16 illustrates the weighted attention scores over time. When the concept drift described above is introduced to the proposed DSA-AE model, the latter effectively responds by adjusting the attention score values, indicating the presence of a concept drift even if the data distribution is not changing. This variation in attention scores highlights the effectiveness of our dual self-attention mechanism in detecting hidden spatiotemporal drifts, often overlooked by traditional drift detectors.

To gain more details into the inner workings of the proposed dual self-attention, Figure 17 presents feature and temporal attention scores over time, illustrating the model's sensitivity to spatiotemporal variations in the Ireland dataset. Key observations include notable fluctuations in temporal attention scores, indicating the model's responsiveness to shifts in temporal patterns and its capacity to capture these dependencies effectively. Meanwhile, feature attention scores remain relatively stable, suggesting that the model's attention mechanism is less impacted by changes in the feature space, indicating consistent feature stability despite concept drift. Both Figure 16 and Figure 17 illustrate the effectiveness of the DSA-AE model in detecting complex drifts, leveraging its attention mechanism.

The heatmaps in Figure 18 and Figure 19 highlight the model's attention mechanisms. The temporal heatmap (Figure 18) shows how the model assigns varying attention to features like NetworkMode, CQI, DL_bitrate, and UL_bitrate over time, with the Prediction feature consistently receiving attention. This enables the model to capture long-term dependencies, helping detect temporal drifts and complex patterns. The inter-feature heatmap (Figure 19) reveals strong dependencies between the Prediction feature and key network metrics such as RSSI, RSRQ, SNR, DL_bitrate, and UL_bitrate, highlighting their importance in accurate throughput prediction.

E. IMPACT OF ATTENTION WEIGHT α ON DRIFT DETECTION PERFORMANCE

To investigate the influence of the attention weight α (eq. 13) on drift detection performance, we varied α from 0 to 1 and evaluated the F1-based drift detection accuracy across both time-series (Milano, Ireland) and non–time-series (EURECOM) datasets. As illustrated in Figure 20, the time-series datasets exhibit improved performance when more weight is allocated to the temporal self-attention component (typically around $\alpha \approx 0.6$–0.7), reflecting the importance of capturing temporal dependencies. In contrast, the non–time-series dataset benefits from lower values of α (around 0.3–0.4), indicating that feature-level attention plays a more critical role when temporal patterns are less pronounced.

It is also noteworthy that assigning all or nearly all of the attention weight to one component (i.e., α near 0.0 or 1.0) leads to a drop in overall accuracy, underscoring the need for a balanced allocation. Thus, the choice of α should be guided by the nature of the dataset: time-series tasks typically require stronger temporal attention, while non–time-series problems can rely more heavily on feature-level attention. Consequently, tuning α within a moderate range (e.g. performing grid search optimization) is generally advisable to avoid the performance degradation observed at extreme values.

F. COMPARATIVE ANALYSIS

To comprehensively validate the performance of DSA-AE drift detection, we conducted an extensive comparative analysis with several widely used drift detectors, including Adaptive Sliding Window (ADWIN) [28], Early Drift Detection Method (EDDM) [52], and an XAI-based Drift Detector [18]. This evaluation examined drift detection accuracy metrics such as FPR, recall, and precision, as well as an important standardized metric termed drift detection latency (drift detection delay as acknowledged in ETSI

TABLE 3. Comparative analysis of drift detection methods across various datasets, window sizes, and metrics.

Dataset	Window Size	Method	FPR (%)	Precision (%)	Recall (%)	F1 Score (%)	Detection Latency (s)
Milano Dataset	32	ADWIN	14.0	79.5	82.0	80.7	**0.05**
		EDDM	13.5	83.0	85.5	84.2	0.08
		XAI-Based	12.5	85.7	87.2	86.4	150.8
		DSA-AE	**10.5**	**93.4**	**91.5**	**92.1**	1.9
	64	ADWIN	15.2	78.0	80.5	79.2	**0.055**
		EDDM	14.0	82.5	83.8	83.1	0.09
		XAI-Based	13.0	84.0	86.5	85.0	458.2
		DSA-AE	**10.0**	**91.9**	**92.3**	**92.2**	3.2
	128	ADWIN	16.5	77.1	79.0	77.9	**0.06**
		EDDM	14.5	81.8	82.2	82.1	0.1
		XAI-Based	12.8	83.5	85.0	84.3	5897.1
		DSA-AE	**9.5**	**92.5**	**93.2**	**92.8**	6.1
Ireland Dataset	32	ADWIN	14.8	79.2	81.5	80.2	**0.05**
		EDDM	13.2	82.8	84.7	83.7	0.085
		XAI-Based	13.8	85.3	86.5	85.7	421.9
		DSA-AE	**10.2**	**90.8**	**92.1**	**91.4**	1.5
	64	ADWIN	15.4	78.5	80.8	79.6	**0.055**
		EDDM	13.8	82.3	83.5	82.7	0.095
		XAI-Based	12.8	84.2	85.7	84.9	521.8
		DSA-AE	**9.8**	**91.3**	**92.8**	**92.8**	4.2
	128	ADWIN	16.5	76.5	78.5	77.5	**0.06**
		EDDM	14.8	80.5	82.5	81.2	0.11
		XAI-Based	12.5	83.6	84.3	83.6	6028.5
		DSA-AE	**9.2**	**92.4**	**93.5**	**92.8**	8.2
EURECOM Dataset	32	ADWIN	13.5	81.4	83.5	82.4	**0.045**
		EDDM	12.8	84.0	85.8	84.9	0.075
		XAI-Based	**10.7**	**91.2**	**91.5**	**91.3**	229.2
		DSA-AE	10.8	89.5	90.5	90.0	2.8
	64	ADWIN	14.2	79.7	81.2	80.2	**0.05**
		EDDM	13.1	82.5	84.0	83.2	0.085
		XAI-Based	10.5	89.2	90.8	90.1	858.8
		DSA-AE	**9.8**	**92.1**	**92.5**	**92.3**	5.8
	128	ADWIN	15.5	77.2	79.5	78.3	**0.055**
		EDDM	14.0	81.3	82.5	81.7	0.095
		XAI-Based	11.5	88.8	89.1	88.9	7351.0
		DSA-AE	**10.5**	**92.7**	**92.9**	**93.1**	11.8

specification [53]). We performed a grid search optimization (as detailed in section V-E) to identify the optimal α values for each use case. Additionally, we varied the drift window size to gain deeper insights into its effect on detection performance. To simulate real-world conditions where multiple forms of drift (data, label, and concept drift) can occur unpredictably, we introduced random drift across the data, predictions, and spatiotemporal dependencies to better reflect real-world scenarios. We also varied the types of drift, including seasonal, recurring, and gradual drifts, to comprehensively assess the methods' performance while ensuring common experiment parameters (window size, drift form, dataset, etc.) for fairness.

The comparative analysis of the DSA-AE method against state-of-the-art drift detection techniques is presented in Table 3. The results clearly indicate that the proposed DSA-AE significantly outperforms alternative methods, such as ADWIN, EDDM, and XAI-based approaches, across all major drift detection metrics. Notably, DSA-AE achieves the highest levels of precision, recall, and F1 scores across

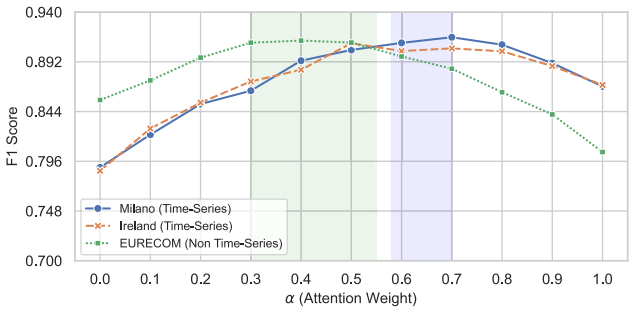


FIGURE 20. Drift Detection Accuracy for the different Datasets across varying Attention Weight α values.

various window sizes (up to 93.5%), demonstrating superior capability in identifying true drift occurrences with minimal FPRs. This performance advantage is particularly significant in time-series datasets (Milano and Ireland), where traditional methods struggle with the univariate nature of the data, leading to lower recall, precision, and F1 scores (around 77%) due to the higher number of false negatives.

Although ADWIN excels in minimal detection delays due to its simple calculation-based approach, this simplicity reduces its accuracy in detecting drift, particularly in larger windows, making it less effective in critical scenarios requiring high reliability. Similarly, EDDM offers slightly better accuracy but still struggles with concept drift, especially in temporally dependent data. In contrast, DSA-AE provides a balanced trade-off, achieving high detection accuracy while maintaining low latency, and prioritizing quality of detection as the primary metric.

While the XAI-based drift detection method is effective with non-time-series datasets (e.g., EURECOM), it exhibits limitations when applied to time-series data such as Milano and Ireland. In these datasets, XAI-based approaches demonstrate higher FPR, as their reliance on feature importance fails to adequately capture univariate dependencies. Conversely, DSA-AE leverages its dual self-attention structure to effectively address these limitations, capturing the intricate patterns in time-series data and delivering consistently high accuracy. Moreover, one notable observation is that XAI-based drift detectors tend to have significant delays in detection. These delays (in hours, sometimes even days) can be even more pronounced when the model is more complex or the window size is larger. The exponential latency is primarily due to the computationally intensive nature of XAI methods, which require considerable time to determine feature importance, a key indicator of model drift.

Besides that, our analysis highlights a trade-off between drift detection latency and accuracy based on window size in the DSA-AE model. Smaller windows, such as 32, allow for faster drift detection but with lower accuracy, as they are limited in capturing long-term dependencies. Conversely, larger windows, like 128, improve drift detection accuracy by capturing extended temporal patterns (gradual drift), though they increase detection latency. Selecting the optimal window size is critical and depends not only on acceptable detection latency, model size, and data's temporal characteristics but also on the type of drift expected. For gradual drift, larger windows are preferable as they better capture subtle, long-term changes, whereas smaller windows may be more effective for detecting sudden shifts. Tailoring the window size based on these factors enables a more balanced and effective approach to drift detection in each use case.

From the results presented in Table 3, we derive the reported improvements of over 13% in drift detection accuracy and 94% in detection latency reduction by comparing the performance of the proposed DSA-AE model to the average results of the benchmarked state-of-the-art methods. For accuracy, we use the F1 score, calculated from precision and recall, across multiple window sizes (32, 64, and 128). In each case, we compute the percentage improvement of DSA-AE over the average performance of the baseline methods. The observed gains in accuracy range from 10.56% to 15.02%, with an average improvement of 13.6%. A similar approach is applied to detection latency, where the reduction ranges from 90.84% to 97.59%, yielding an average improvement of 94.7%. These values form the basis of the performance claims discussed in the comparative analysis.

G. DISCUSSION & LEARNED LESSONS

The extensive analysis and evaluation of our proposed DSA-AE model underscore several important aspects of model performance, architecture design, and the broader implications for drift detection in dynamic environments. Results across three realistic datasets provide several valuable lessons, which we can summarize as follows:

- **Lesson 1: Combining Self-Attention with Autoencoders for Diverse Drift Types.** The numerical results have demonstrated that the synergy between self-attention and autoencoders is highly effective in addressing various types of drift, including data drift, label drift, and concept drift.
- **Lesson 2: Separating Feature and Temporal Self-Attention for Enhanced Spatiotemporal Analysis.** By separating feature-specific and temporal self-attention blocks, our model achieves a more nuanced and accurate handling of spatiotemporal dependencies. This separation allows the model to differentiate between sudden changes in specific features and longer-term trends, refining its ability to identify the underlying factors behind data shifts.
- **Lesson 3: Limitations of XAI Methods in Capturing Temporal Dependencies.** Our experiments with XAI methods highlighted a notable limitation in their effectiveness for univariate time-series data, confirming insights from recent works [15]. In our case, XAI-based drift detection methods struggled to identify drift accurately, as they rely primarily on feature importance metrics. This approach is insufficient for capturing the

sequential nature of time-series data, thereby reinforcing the limitations cited in recent studies.

- **Lesson 4: Impact of Drift Window Size on Detection Accuracy and Latency.** The analysis shows that drift window size significantly impacts detection accuracy and latency. Selecting the optimal size depends on factors such as tolerance for detection delays, model complexity, data's temporal characteristics, and the expected drift type (gradual or sudden). Balancing these factors is crucial for effective drift detection, especially in dynamic 6G network environments.

VI. CONCLUSION

In this study, we introduced a novel Dual Self-Attention Autoencoder drift detection method. Our approach leverages the power of autoencoder reconstruction error with MD to detect data and label drift, and dual self-attention scores with cosine distance to detect concept drift. Our method surpasses the current state-of-the-art and distinguishes itself by addressing the complexities of spatiotemporal time-series data and detecting data, label, and concept drift within the same model. Extensive experiments conducted across three realistic telecommunication datasets, two of which are time series, demonstrate the effectiveness of DSA-AE, outperforming benchmark methods by up to 94% in drift latency and improving drift detection accuracy by over 13%. These results highlight the substantial potential of our proposed model in real-world applications. For future work, we will turn our attention to the challenge of drift adaptation within the continuous learning landscape of 6G networks. This represents a critical next step in advancing the state-of-the-art in this domain.

REFERENCES

[1] I. Ishteyaq, K. Muzaffar, N. Shafi, and M. A. Alathbah, "Unleashing the power of tomorrow: Exploration of next frontier with 6G networks and cutting edge technologies," *IEEE Access*, vol. 12, pp. 29445–29463, 2024.

[2] G. M. Karam, M. Gruber, I. Adam, F. Boutigny, Y. Miche, and S. Mukherjee, "The evolution of networks and management in a 6G world: An inventor's view," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 5395–5407, Dec. 2022.

[3] J. Alanya-Beltran et al., "Advancements in AI for 6G networks: Technology, standardization, and future prospects," in *Proc. Int. Conf. Adv. Comput., Commun. Appl. Inform. (ACCAI)*, May 2024, pp. 1–5.

[4] R. Chataut, M. Nankya, and R. Akl, "6G networks and the AI revolution—Exploring technologies, applications, and emerging challenges," *Sensors*, vol. 24, no. 6, p. 1888, 2024.

[5] B. Yang et al., "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.

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

[7] D. M. Manias, A. Chouman, and A. Shami, "A smodel drift detection and adaptation framework for 5G core networks," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Sep. 2022, pp. 197–202.

[8] *Detecting Data Drift NannyML 0.12.1 Documentation*. Accessed: Nov. 3, 2024. [Online]. Available: https://nannyml.readthedocs.io/en/stable/tutorials/detecting_data_drift.html

[9] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[10] R. Noronha Gemaque, A. F. J. Costa, R. Giusti, and E.-L. Miranda Dos, "An overview of unsupervised drift detection methods," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 10, no. 6, p. e1381, 2020.

[11] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—A survey on monitoring in evolving environments. Part B: Locating and explaining concept drift," *Frontiers Artif. Intell.*, vol. 7, Jul. 2024, Art. no. 1330258.

[12] L. Yang and A. Shami, "A lightweight concept drift detection and adaptation framework for IoT data streams," *IEEE Internet Things Mag.*, vol. 4, no. 2, pp. 96–101, Jun. 2021.

[13] A. Barredo Arrieta et al., "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020.

[14] T. Rojat et al., "Explainable artificial intelligence (XAI) on timeseries data: A survey," 2021, *arXiv:2104.00950*.

[15] C. Fiandrino, E. P. Gmez, P. Fernndez Prez, H. Mohammadalizadeh, M. Fiore, and J. Widmer, "AIChronoLens: Advancing explainability for time series AI forecasting in mobile networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, vol. 70, May 2024, pp. 1521–1530.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.

[17] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: An overview," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Cham, Switzerland: Springer, 2019, pp. 193–209.

[18] Y. Lee, "Explainable artificial intelligence-based model drift detection applicable to unsupervised environments," *Comput., Mater. Continua*, vol. 76, no. 2, p. 1701, 2023.

[19] A. Cossu, F. Spinnato, R. Guidotti, and D. Bacciu, "Drifting explanations in continual learning," *Neurocomputing*, vol. 597, Sep. 2024, Art. no. 127960.

[20] L. Yang, "CADE: Detecting and explaining concept drift samples for security applications," in *Proc. 30th USENIX Secur. Symp. (USENIX Secur.)*, 2021, pp. 1–19.

[21] A. Koebler, T. Decker, M. Lebacher, I. Thon, V. Tresp, and F. Buettner, "Towards explanatory model monitoring," in *Proc. Workshop, XAI Action, Past, Present, Future Appl. (NeurIPS)*, 2023, pp. 1–11.

[22] R. Vishnampet, R. Shenoy, J. Chen, and A. Gupta, "Root causing prediction anomalies using explainable AI," 2024, *arXiv:2403.02439*.

[23] D. M. V. Sato, S. C. De Freitas, J. P. Barddal, and E. E. Scalabrin, "A survey on concept drift in process mining," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–38, Dec. 2022.

[24] Y. Rotalinti, A. Tucker, M. Lonergan, P. Myles, and R. Branson, "Detecting drift in healthcare AI models based on data availability," in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (Communications in Computer and Information Science). Cham, Switzerland: Springer, 2023, pp. 243–258.

[25] A. Kuppa and N.-A. Le-Khac, "Learn to adapt: Robust drift detection in security domain," *Comput. Electr. Eng.*, vol. 102, Sep. 2022, Art. no. 108239.

[26] D. Wang, S. Thunéll, U. Lindberg, L. Jiang, J. Trygg, and M. Tysklind, "Towards better process management in wastewater treatment plants: Process analytics based on SHAP values for tree-based machine learning methods," *J. Environ. Manage.*, vol. 301, Jan. 2022, Art. no. 113941.

[27] I. I. F. Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. M. Bueno, A. A. O. Díaz, and Y. C. Mota, "Online and non-parametric drift detection methods based on Hoeffding's bounds," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 810–823, Mar. 2015.

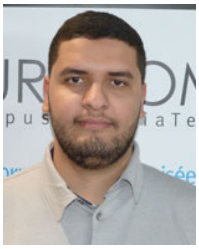
[28] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2007, pp. 443–448.

[29] A. Haque, L. Khan, and M. Baron, "SAND: Semi-supervised adaptive novel class detection and classification over data stream," in *Proc. 13th AAAI Conf.*, 2016, pp. 1–7.

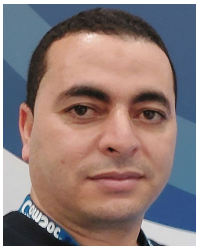
[30] D. Maduskar, D. Sharma, R. Borrison, G. Manca, and M. Dix, "UDDT: An unsupervised drift detection method for industrial time series data," in *Proc. IEEE 2nd Ind. Electron. Soc. Annu. Line Conf. (ONCON)*, Dec. 2023, pp. 1–6.

[31] R. C. Cavalcante, L. L. Minku, and A. L. I. Oliveira, "FEDD: Feature extraction for explicit concept drift detection in time series," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 740–747.

[32] S. Munirathinam, "Drift detection analytics for IoT sensors," *Proc. Comput. Sci.*, vol. 180, pp. 903–912, Jan. 2021.
[33] P. Shen, Y. Ming, H. Li, J. Gao, and W. Zhang, "Unsupervised concept drift detectors: A survey," in *Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*. Cham, Switzerland: Springer, 2023, pp. 1117–1124.
[34] L. Bu, C. Alippi, and D. Zhao, "A pdf-free change detection test based on density difference estimation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 2, pp. 324–334, Feb. 2018, doi: 10.1109/TNNLS.2016.2619909.
[35] T. Cerquitelli, S. Proto, F. Ventura, D. Apiletti, and E. Baralis, "Towards a real-time unsupervised estimation of predictive model degradation," in *Proc. Real-Time Bus. Intell. Anal.*, Aug. 2019, pp. 1–6.
[36] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, "Drift detection in data stream classification without fully labelled instances," in *Proc. IEEE Int. Conf. Evolving Adapt. Intell. Syst. (EAIS)*, Dec. 2015, pp. 1–8.
[37] Ö. Gözüaçik, A. Büyükçakir, H. Bonab, and F. Can, "Unsupervised concept drift detection with a discriminative classifier," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Beijing, China, 2019, pp. 2365–2368.
[38] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Proc. Asian Conf. Comput. Vis.* Berlin, Germany: Springer, 2010, pp. 709–720.
[39] L. Ming, H. Dezhi, and L. Dun, "A method combining improved Mahalanobis distance and adversarial autoencoder to detect abnormal network traffic," in *Proc. Int. Database Engineered Appl. Symp. Conf.*, May 2023, pp. 161–169.
[40] Q. Zhao, W. Yang, and Q. Liao, "AdaSAN: Adaptive cosine similarity self-attention network for gastrointestinal endoscopy image classification," in *Proc. IEEE 18th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2021, pp. 1855–1859.
[41] C. A. Siebra, M. Kurpicz-Briki, and K. Wac, "Transformers in health: A systematic review on architectures for longitudinal data analysis," *Artif. Intell. Rev.*, vol. 57, no. 2, p. 32, 2024.
[42] L. Gjorgiev and S. Gievska, "Time series anomaly detection with variational autoencoder using Mahalanobis distance," in *ICT Innovations 2020. Machine Learning and Applications* (Communications in Computer and Information Science). Cham, Switzerland: Springer, 2020, pp. 42–55.
[43] Securing Artificial Intelligence. *ETSI TR 104 032 V1.1.1 (2024-02)*. Accessed: Aug. 20, 2024. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/104000_104099/104032/01.01.01_60/tr_104032v010101p.pdf
[44] GROUP SPECIFICATION. *ETSI GS ZSM 012 V1.1.1 (2022-12)*. Accessed: Aug. 20, 2024. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/012/01.01.01_60/gs_ZSM012v010101p.pdf
[45] X. Allka, P. Ferrer-Cid, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, "Pattern-based attention recurrent autoencoder for anomaly detection in air quality sensor networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 6, pp. 6372–6381, Nov. 2024.
[46] T. Xie, Q. Xu, C. Jiang, Z. Gao, and X. Wang, "A robust anomaly detection model for pumps based on the spectral residual with self-attention variational autoencoder," *IEEE Trans. Ind. Informat.*, vol. 20, no. 6, pp. 9059–9069, Jun. 2024.
[47] M. Ameur, B. Brik, and A. Ksentini, "Leveraging LLMs to eXplain DRL decisions for transparent 6G network slicing," in *Proc. IEEE 10th Int. Conf. Netw. Softwarization (NetSoft)*, Jun. 2024, pp. 204–212.
[48] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.
[49] G. Barlacchi et al., "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Sci. Data*, vol. 2, no. 1, 2015, Art. no. 150055.
[50] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: A 5G dataset with channel and context metrics," in *Proc. 11th ACM multimedia Syst. Conf.*, 2020, pp. 303–308.
[51] M. Mekki, N. Toumi, and A. Ksentini, "Microservices configurations and the impact on the performance in cloud native environments," in *Proc. IEEE 47th Conf. Local Comput. Netw. (LCN)*, Sep. 2022, pp. 239–244.
[52] M. Baena-Garcıa, J. del Campo-ÁVila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Proc. 4th Int. Workshop Knowl. Discovery Data Streams*, vol. 6, 2006, pp. 77–86.
[53] Securing Artificial Intelligence. *ETSI TR 104 032 V1.1.1 (2024-02)*. Accessed: Nov. 15, 2024. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/104000_104099/104032/01.01.01_60/tr_104032v010101p.pdf



MAZENE AMEUR (Member, IEEE) received the master's degree (Hons.) from the University of Laghouat in 2023. He is currently pursuing the Ph.D. degree with the Communication Systems Department, EURECOM. For the master's degree project, he completed an internship with the XLIM Research Institute, France. His research focuses on integration of novel AI systems into next-generation communication networks, specifically 6G networks. Under the supervision of Prof. Adlen Ksentini and Dr. Bouziane Brik, he is actively involved in several prominent European projects, including SUNRISE-6G, 6G-DALI, and 6G-BRICKS.



BOUZIANE BRIK (Senior Member, IEEE) received the Engineering degree (Hons.) in computer science and the M.Sc. and Ph.D. degrees from the Universityof Laghouat, Algeria, in 2010, 2013, and 2017, respectively. He has held post-doctoral positions with the CESI School, University of Troyes; and the EURECOM Research Institute, Nice, France. Currently, he is an Assistant Professor with the Computer Science Department, College of Computing and Informatics, University of Sharjah, United Arab Emirates. Prior to this role, he was an Assistant Professor with the DRIVE Department, Bourgogne University, Dijon, France. His research has focused on resource management and security challenges in 5G network slicing, contributing to multiple H2020 European projects, including MonB5G, 5G-Drones, InDiD, and 5G-INSIGHT. His broader research interests encompass 5G and beyond networks, explainable AI, and the application of machine learning and deep learning techniques in wireless networks.

ADLEN KSENTINI (Senior Member, IEEE) is a Professor with the Communication Systems Department, EURECOM, where he leads the Network Softwarization Group. His work focuses on advancing network softwarization, 5G/6G technologies, and edge computing. Over the years, he has been actively involved in several European research initiatives under H2020 and Horizon Europe, such as 5G!Pagoda, 5G-Transformer, 5G!Drones, MonB5G, ImagineB5G, 6G-BRICKS, 6G-INTENSE, SUNRISE-6G, and AC3. Notably, he is the Technical Manager of 6G-INTENSE, focusing on zero-touch management of 6G resources and applications and AC3, which explores the Cloud-Edge Continuum. His research spans both system architecture and algorithm development, addressing challenges in these domains with tools such as Markov Chains, optimization algorithms, and machine learning (ML). He has also delivered several tutorials at prestigious IEEE international conferences, including IEEE Globecom 2015, IEEE CCNC (2017, 2018, and 2023), IEEE ICC 2017, IEEE/IFIP IM 2017, and IEEE School 2019. His primary research interests include network virtualization, software-defined networking (SDN), and network cloudification, with a particular emphasis on their applications in 5G and 6G networks. He is a member of the OpenAirInterface (OAI) Board Of Directors, where he oversees activities related to the OAI 5G Core Network and O-RAN management (O1, E2) for OAI RAN projects.