Resource-Aware Time Series Imaging Classification for Wireless Link Layer Anomalies

Blaž Bertalančič, Graduate Student Member, IEEE, Marko Meža, Senior Member, IEEE, and Carolina Fortuna

Abstract—The number of end devices that use the last-mile wireless connectivity is dramatically increasing with the rise of smart infrastructures and requires reliable functioning to support smooth and efficient business processes. To efficiently manage such massive wireless networks, more advanced and accurate network monitoring and malfunction detection solutions are required. In this article, we perform a first-time analysis of image-based representation techniques for wireless anomaly detection using recurrence plots (RPs) and Gramian angular fields and propose a new deep learning architecture enabling accurate anomaly detection. We elaborate on the design considerations for developing a resource-aware architecture and propose a new model using time series to image transformation using RPs. We show that the proposed model: 1) outperforms the one based on Gramian angular fields by up to 14% points; 2) outperforms classical ML models using dynamic time warping by up to 24% points; 3) outperforms or performs on par with mainstream architectures, such as AlexNet and VGG11 while having <10x their weights and up to ≈8% of their computational complexity; and d) outperforms the state of the art in the respective application area by up to 55% points. Finally, we also explain randomly chosen examples how the classifier takes decisions.

Index Terms—Anomaly detection, classification, deep learning (DL), Gramian angular field (GAF), imaging, link layer, machine vision, recurrence plot (RP), time series (TS), wireless networks.

I. INTRODUCTION

Wireless networks represent the most convenient last-mile connectivity solution and are used daily by billions of devices, such as phones, tablets, laptops, desktops, and increasingly smart devices, forming the so-called Internet of Things [1]. In tradition, last-mile connectivity issues were resolved by the owner of the access point be it an individual person, a cable, or another operator in case of noncellular operators, the base station in the case of cellular only operators or resetting and reconfiguring equipment when possible. The manual mitigation is mostly due to the fact that the entire network, including the wired-to-wireless converter, used static, preconfigured specialized hardware equipment.

With the increased digitization of society, including cities and infrastructures, such as transportation and energy, the number of end devices that use the last-mile wireless connectivity is dramatically increasing and often requires reliable functioning to support smooth and efficient business processes. Furthermore, the wired-to-wireless converters and the core transport networks are migrating to software controlled virtual functions residing on top of more general-purpose hardware leading to increasingly complex and expensive network operation. For instance, as shown in Fig. 1 and specified in the technical annex of the 3GPP 5G standard, an end-to-end network slice (i.e., virtual network service) can be created in a matter of minutes. The operational slices need to be continuously monitored throughout their lifetime and reconfigured as needed, possibly in a fully automatic manner, by the slice orchestrator in the operations/business support systems (OSS/BSS, respectively) of the network operators. The software and virtualization-driven agility and complexity of the emerging networks requires faster configuration and reaction times. For instance, the automatic selection of a frequency band to be used at a given time and place should be suitably informed by events in the radio spectrum, automatically detected by automatic spectrum sensing, detection, and classification systems [2], while the transmission parameters, such as transmit power and channel number could be selected according to the perceived link quality [3]. Finally, when a link outage or an abnormal network behavior occurs, it should be proactively detected and fixed before it causes significant user dissatisfaction [4] using network monitoring [5] and malfunction detection [6] solutions that automatically report relevant issues that can be mitigated without disrupting the business process.

An anomaly, in general, can be defined from a mathematical perspective as a rare event or an outlier of a distribution. Several machine learning techniques have been developed to detect such outliers and they typically use an unsupervised machine learning approach. For instance, a typical example of such rare, unknown events that cannot be easily characterized a priori is intrusion detection. Most such recent anomaly detections are performed on multivariate time series (TS)
data [7]–[10] using unsupervised deep learning (DL) algorithms in the form of autoencoder networks.

An anomaly can also be defined from an application perspective as being an event that causes inconvenience to the respective application. The respective event does not necessarily need to be rare from a mathematical point of view, and its characterization is often known. For instance, Sheth et al. [6] define and identify anomalies from the IEEE 802.11 physical layer perspective, namely, hidden terminal, capture effect, noise, and signal strength variation anomalies, whereas Gupta et al. [11] define anomalies from a multihop networking perspective, such as a black hole, sinkhole, selective forwarding, and flooding. More recently, starting from observable symptoms of link measurements, namely, the changes in the expected received signal, Cerar et al. [4] proposed four types of link-layer anomalies, namely, sudden degradation (SuddenD), sudden degradation with recovery (SuddenR), instantaneous degradation (InstaD), and slow degradation (SlowD). They were the first to evaluate different TS representations, such as fast Fourier transform (FFT), histogram, aggregates, and compressed autoencoded in an attempt to develop a classifier for the defined anomalies. However, their approach only used classical, non-DL methods. More recently, Bertalanić et al. [12] investigated the same problem by training a DL model on raw TS.

Inspired by the breakthroughs in image recognition [13] and object recognition [14] that have been going on over the last decade, and attempts from various fields of science in formulating domain-specific problems as image problems to benefit from these achievements [15], [16], we also endeavor in the first attempt to investigate image base transformations for supervised anomaly detection for wireless links as defined in [4]. In this article, we propose a new approach for anomaly classification in wireless networks based on a TS to image transformation and DL. The contributions of this article are as follows.

1) We perform a first-time analysis of image-based representation for wireless link layer anomaly detection. For this, we consider the four anomaly shapes defined in [4], the recently introduced Gramian angular fields [17] (GAF) and recurrence plots (RP) [18].

2) We elaborate on the design methodology to develop a new resource-aware deep neural network architecture for the classification of the four types of anomalies. We propose an RP-based model that outperforms the GAFs image-based models by up to 14% points.

3) We also show the potential of the proposed model using RPs to: 1) outperform classical ML models using dynamic time warping (DTW) by up to 24% points; 2) outperform or performs on par with mainstream architectures, such as AlexNet and VGG11 while having <10× their weights and up to ≈8% of their computational complexity; and 3) outperform the state of the art in the respective application area [4], [12] by up to 56% points.

4) We show that the way the proposed model takes the decision to classify instances can be explained.

This article is organized as follows. We discuss related work in Section II. Section III provides the formal problem statement, while Section IV elaborates on various TS transformations that can be used to generate image representations. Section V introduces the proposed DL architecture, Section VI describes the relevant methodological and experimental details, while Section VII provides thorough analyses of the results. Finally, Section VIII concludes this article.

II. RELATED WORK

For detecting anomalies, models can be based on one of the four main categories of techniques: statistical, nearest neighbor, clustering, or classification [19]. Statistical models rely on the underlying data distributions which are normally not known for data with present anomalies. Techniques using nearest neighbor classification assume that normal patterns can be found in a dense neighborhood, while anomalies are far from it [20], but due to the nature of TS data, this is not always the case. Clustering techniques work on a similar principle. Classification-based techniques have shown superiority over other techniques in terms of learning patterns in the data, but their limitation is that they need labeled samples for training to learn how to discriminate between different patterns.

Collecting a suitable training dataset for standard classification techniques is not always a feasible task. Since it is
A. Classification Problem

We formulate the anomaly detection problem as a classification problem in which given an input tensor $H$, TCS problems as noticed by Wang and Oates [17]. In the same paper, they also introduced two other methods of TS to image transformations called GAFs and Markov transition field. All three approaches have recently been used as an indirect method for classifying TS data using convolutional neural networks and DL networks. There are not many works on using TS to image transform for anomaly detection. All three transformations considered in [17] were also used in solving regression problems [27].

B. Image Transformation for Anomaly Detection

The most recent usage of images for anomaly detection was presented by Choi et al. [28] that used generative adversarial networks to transform multivariate TS into images and to detect and localize anomalies in signals from power plants. Krummenacher et al. [29] did an interesting work on finding anomalies within wheels of train cart wagons by transforming sensor signals into GAF and using convolutional neural networks for detecting defects. One of the first applications of GAF for anomaly detection was attempted by Zhang et al. [30] for anomaly detection in EKG signals. Another interesting use of GAF and DL can be found in [31] for recognizing human actions with signals received from wearable devices. This kind of classification is gaining momentum in the last few years, but as far as we know, no one used this approach for anomaly detection in wireless signals.

In recent years, some work has also been done on anomaly detection and TCS using RPs. One of the first works on TCS was by Silva et al. [32], where they showed that transformation to RP improves classification performance on most of the datasets they used for testing. Hu et al. [33] proposed a framework for anomaly detection in TS using RP transformation. Another anomaly detection work was done by [34] where they used RP transform and CNN on multisensor signals for real-time anomaly detection in the flash butt welding process. In the network domain, an attempt was made by [35] to detect DDoS attacks using RP plots and convolutional neural networks.

III. Problem Formulation

Starting from the example of emerging wireless networks and the role of anomaly detection shown in Fig. 1, we formulate the steps that enable automatic anomaly detection from TS. More specifically, we propose transforming the incoming TS data into images and then training a DL model to recognize anomalies in those respective images, as shown in Fig. 2, and further explained in this section. The model is then able to automatically classify the images depending on whether they contain anomalies or not.

A. Classification Problem

We formulate the anomaly detection problem as a classification problem in which given an input tensor $H$,
There is a function $\Phi$ that maps the input to a set of target classes $C$ as provided in the following:

$$C = \Phi(H).$$  

The cardinality of the set $C$, also denoted as $|C|$, denotes the number of classes to be recognized. Without loss of generality, in the remainder of this article, we focus on two cases. In the first case, we consider a binary classification problem with $|C| = 2$, where the set of target classes is $C = \{ \text{anomalous, normal} \}$. In the second case, we start from the four types of anomalies introduced in [4] and consider a five-class classification problem with $|C| = 5$, where the set of target classes is $C = \{ \text{SuddenD, SuddenR, InstaD, SlowD, normal} \}$. The binary classification problem aims to detect whether an incoming portion of a TS contains or not one of the four anomalies. The five-class problem aims to detect the specific type of anomaly defined in the literature or a normal link.

For the case of the five-class problem, the first anomaly, called SuddenD, is an anomaly where the signal unexpectedly drops to a minimal value and never recovers, as shown in Fig. 3(a). The second anomaly, called SuddenR, has a certain similarity to SuddenD only that the signal after a certain period of time recovers from minimal back to the normal value, as shown in Fig. 4(a). The third anomaly that was defined was InstaD, which shows itself as a spike within a trace. Here, the value of a trace drops to a minimal for a very short, instantaneous, amount of time, and then recovers back to a normal value, as shown in Fig. 5(a). The fourth anomaly that was defined is a SlowD anomaly. This anomaly shows itself in a slightly decreasing slope within a trace, where values slowly but gradually decrease like it can be seen in Fig. 6(a).
B. Time-Series Transformation

We define the transformation $T$ shown in Fig. 2 as a function that transforms the input TS $S$ to the tensor $H$ from (1) as

$$H = T(S).$$

For multidimensional TS of dimension $K$, $S$ takes the form of a $[S]_{K \times N}$ matrix and represents a set of $K$ TS traces with length $N$. In this article, we consider a unidimensional TS where $K = 1$, and thus, $[S]_{1 \times N}$. The transformation function $T$ can be represented by the identity function, in which case, the result $H$ is the same as input TS trace or can represent more complex transformation functions as the ones proposed in this work that transform the TS into images.

IV. TIME-SERIES TO IMAGE TRANSFORMATION

Section III provided general definition of TS transformation. In the following, we provide two distinct ways to transform TS data representation into images that can be used as features for training DL models.

A. Recurrence Plot Transformation

The RP is a technique for nonlinear data analysis that represents a visualization of a square matrix in which elements correspond to those times steps when parts of a TS are the most similar to each other. The RP transformation takes as its input the matrix representation of unidimensional TS data $S_{1 \times N}$, which can be also represented as a vector $s_N$. This can be seen in (3), where $N$ is the length of the TS. According to (4), the absolute value of the difference between two points is computed and represents the distance between the points, which is then subtracted from the predefined threshold $\epsilon$. The result $h$ is then converted to binary values through the use of the Heaviside function $\Theta$ presented in (5). The final result is an image $H$ of size $N \times N$. In special cases, the threshold and Heaviside function may be omitted resulting in a nonbinarized matrix of distances between points in the TS

$$S_{1 \times N} = s_N = (s_1, s_2, \ldots, s_N)$$

$$h = \epsilon - \left[\begin{array}{cccc}
    \|s_N - s_1\| & \|s_N - s_2\| & \cdots & \|s_N - s_N\| \\
    \vdots & \vdots & \ddots & \vdots \\
    \|s_2 - s_1\| & \|s_2 - s_2\| & \cdots & \|s_2 - s_N\| \\
    \|s_1 - s_1\| & \|s_1 - s_2\| & \cdots & \|s_1 - s_N\|
\end{array}\right]$$

$$H = \Theta(h).$$

A trace without an anomaly can be seen in Fig. 7(b). No pattern is apparent, as the arrangement within the image can be labeled as random and correlates with the randomness of the TS representation of the trace. On the other hand, Fig. 3(b) shows an RP of the SuddenD anomaly. The typical representation of this anomaly is a white rectangle in the upper right corner. The lower left corner of this rectangle represents the time sample where the anomaly occurred, while the width of the rectangle is the same as the width of the anomaly of the TS anomaly represented in Fig. 3(a).

Looking at Fig. 4(b), the SuddenR anomaly appears as a cross that has a small white rectangle in its center somewhere along the diagonal from the lower left corner to the upper right corner of the image. The lower left corner of the white square represents the beginning of the anomaly, while the upper right
corner of the white square shows where the recovery occurred. The size of the rectangle depends on the width of the anomaly. The third type of anomaly is shown in Fig. 5(b). The InstaD anomaly is very similar to the SuddenR anomaly, except that it is much narrower, which is due to its short occurrence within a trace. Just like SuddenR, this anomaly can be observed along the diagonal line from the lower left corner to the upper right corner.

Finally, in Fig. 6(b), the SlowD anomaly can be observed. This anomaly does not have a typical representation. In some images, it is seen as an area of higher point density along the diagonal from the bottom left to the top right, some have a higher density at the top and right edge of the image, while some other images may have a similar distribution as traces without an anomaly. As such, this anomaly with RP transform is harder to detect with the naked eye than others.

B. Gramian Angular Field Transformation

The GAF is a transformation of a TS that represents the temporal correlation between points within a TS. The end result is a square image representation of the input TS. This approach consists of two techniques, one is the Gramian angular summation field (GASF) and the other is the Gramian angular difference field (GADF). Both techniques are computed in a similar way. First, the TS needs to be scaled with a min–max normalization and then transformed to a polar coordinate system. The angles \( \phi_N \) from the polar plot are then used to compute GASF and GADF. For the sum field, the angular cosine function of the sum between two points is computed, which is represented in (6), where \( H_S \) represents the GASF transformation. For the GADF representation, the angular sine of the difference between each two points is computed, as shown in (7), where \( H_D \) represents the GADF transform. Both GAF representations look similar to those transformed with RP. The size of the transformed image is \( N \times N \), where \( N \) represents the length of the TS used in the transformation.

\[
H_S = \begin{pmatrix}
\cos(\phi_N + \phi_1) & \cos(\phi_N + \phi_2) & \cdots & \cos(\phi_N + \phi_N) \\
\vdots & \vdots & \ddots & \vdots \\
\cos(\phi_2 + \phi_1) & \cos(\phi_2 + \phi_2) & \cdots & \cos(\phi_2 + \phi_N) \\
\cos(\phi_1 + \phi_1) & \cos(\phi_1 + \phi_2) & \cdots & \cos(\phi_1 + \phi_N)
\end{pmatrix}
\]

\[
H_D = \begin{pmatrix}
\sin(\phi_N - \phi_1) & \sin(\phi_N - \phi_2) & \cdots & \sin(\phi_N - \phi_N) \\
\vdots & \vdots & \ddots & \vdots \\
\sin(\phi_2 - \phi_1) & \sin(\phi_2 - \phi_2) & \cdots & \sin(\phi_2 - \phi_N) \\
\sin(\phi_1 - \phi_1) & \sin(\phi_1 - \phi_2) & \cdots & \sin(\phi_1 - \phi_N)
\end{pmatrix}
\]

A trace without an anomaly can be seen in Fig. 7(c). Similar to the RP transformation, there is no obvious pattern in the images and the arrangement also looks random. On the other hand, it can be seen that in Fig. 3(c) that shows the transformation corresponding to the SuddenD anomaly, a rectangle in the upper right corner of the generated image indicates the anomaly. It can be noticed that the representation is similar to the one generated by the RP and that it is more visible in the GASF plot, rather than the GADF.

Fig. 4(c) shows the GAF transforms for the SuddenR anomaly. The anomaly appears as a small rectangle somewhere along the diagonal of the image. The width of the rectangle is the same as the width of the anomaly within the TS as can be seen by comparing with the width of the TS anomaly represented in Fig. 4(a).

InstaD can be seen in Fig. 5(c). It manifests as a small green cross along the diagonal within the GASF image, while it is harder to spot in GADF. Similar to all other GAF anomaly transformations, this one also bears a resemblance to RP representation, only that it is less distinct.

The final GAF anomaly representation is SlowD, which is shown in Fig. 6(c). This is best observed in GADF, where it can be seen that the highest values are in the upper left corner of the image. This represents that the values at the beginning of the curve trace are much more similar to each other than the values at the end of the trace. No obvious pattern can be observed in the GASF image.

V. PROPOSED DNN CLASSIFICATION MODEL

We define our classification model as function \( \Phi \) shown in Fig. 2 that transforms the input transformed data \( H \) to the set of target classes \( C \) as provided in the following:

\[
C = \Phi(H) = \Psi(Z), \quad \text{where } Z = W \cdot H + B \quad (8)
\]

where \( Z \) takes the input data \( H \) and multiplies it with weights and adds bias from the model to it. The activation function \( \Psi \) is then applied to \( Z \) and the predicted class \( C \) is returned.

We have designed two architectures of convolutional neural networks \( \Phi(H) \), one for binary and the other for the five-class classification. In essence, the same architecture is used for both classification problems, the only difference being the output layer, which is adapted according to the classification type. The network architecture was designed using an iterative process using the following design considerations.

A. Design Considerations

In order to ensure that our architecture is mindful of resources, we considered studies on estimating the energy consumption of ML models [36] that show that the increasing complexity of models, manifested in the number of weights, the type of layers, and their respective parameters affect both their performance and energy efficiency. We followed the following design steps.

1) Reduce the Number of the Layers of the Network: The computational complexity of a network is typically evaluated as the number of floating point operations (FLOPs) needed to make a prediction. This depends on the number of layers \( L \) and the computational complexity of each individual layer \( F_i \) as per (9). During our iterative design process, we considered \( L \in \{5, \ldots, 20\} \)

\[
M_{\text{FLOPS}} = \sum_{i=1}^{L} F_i, \quad (9)
\]
2) Optimize the Convolutional Layers: Convolutional layers represent a sequence of matrices used to extract features from the image. A convolutional layer consists of a set of filters of size $K_r \times K_c$ used to scan an input tensor of size $I_r \times I_c \times C$ with a stride $S_r$. More precisely, the number of all FLOPs per filter $F_{pf}$ is given by (10)

$$F_{pf} = \left( \frac{I_r - K_r + 2P_r}{S_r} + 1 \right) \left( \frac{I_c - K_c + 2P_c}{S_c} + 1 \right) \times (2CK, K_c + 1).$$ (10)

The first term of the equation gives the height of the output tensor, where $I_r$ is the size of the input rows, $K_r$ is the height of the filter, $P_r$ is the padding, and $S_r$ is the size of the stride. The second term represents the same calculation for the width of the output tensor, where the indices in $I_c$, $K_c$, $P_c$, and $S_c$ correspond to the input columns. The last term provides the number of computations per filter for each of the input channels $C$ that represent the depth of the input tensor and the bias.

The number of FLOPs used throughout the convolutional layer is equal to the number of filters times the flops per filter given in (10), i.e., $F_c = (F_{pf} + N_{pf})N_f$. However, in the case where ReLU are used, one additional comparison and multiplication are required to calculate the number of FLOPs used in one epoch $F_{pe}$. We, therefore, added the number of FLOPs used for each filter and the number of instances for each filter and then multiplied by the number of all filters $N_f$

$$F_c = (F_{pf} + (2CK, K_c + 1))N_f.$$ (11)

During our iterative design process, we considered the following.

1) Optimizing the number of filters $N_f \in \{16, 32, \ldots, 128\}$ from (11).

2) Optimizing the kernel size $K_r = K_c \in \{2, 3, \ldots, 7\}$ from (10).

B. Architecture

The proposed architecture shown in Fig. 8 yielded the best performance/resource utilization during the design process. On the left-hand side of Fig. 8, there is an input of size $300 \times 300$ pixels, which is the size of the image resulting from transforming the TS. The image is then fed into the neural network, which starts with four convolutional layers. The first layer uses 128 filters with kernel size $3 \times 3$ pixels. The next three convolutional layers use 64, 32, and 16 filters with a kernel size of $7 \times 7$. After the last convolution layer, max-pooling is applied to the output, reducing the height and width of the output by half. The data are then flattened and fed into the dense layer, consisting of 64 nodes, and connected to the output layer, which is either the size of 1 or 5 depending on the classification type. All hidden layers use the ReLU function that inserts nonlinearity into the model and thus helping with the classification of classes with nonlinear boundaries. Finally, the output layer uses a sigmoid activation function that is suitable also for multiclass classification.

VI. METHODOLOGY AND EXPERIMENTAL DETAILS

To develop and evaluate our model, we follow the methodology presented in Sections VI-A–VI-E. Experimental settings such as the train/test split and the energy consumption of the model are provided within the respective sections.

A. Dataset Generation

For our experimental evaluation, we choose the Rutgers dataset [37] containing real-world measurements, and then we synthetically injected anomalies. These traces were then converted into RP, GASF, and GADF images resulting in three datasets for solving our binary and five-class classification problem.

The Rutgers dataset consists of link traces from 29 nodes at five different noise levels. The dataset contains the raw received signal strength indicator (RSSI), sequence numbers, source node ID, destination node ID, and artificial noise levels. Each RSSI value represents the signal strength of a received packet sent every 100 ms, for a period of 30 s. Thus, we obtain traces with a length of 300 RSSI samples.

To inject the anomalies into the Rutgers dataset, we first filtered out samples that did not exhibit packet loss. This left 2123 samples, of which 33%, the same as in [4], were randomly injected with some kind of anomaly, while the other links were left untouched. The anomalies were randomly injected according to [4] according to the parameters from Table I. This gave us four raw TS datasets, one for each of the four anomalies, which together constitute our final dataset. It consists of 8492 samples, with each anomaly represented by 700 samples, while there are a total of 5692 traces that have no anomaly. These traces are then transformed into RP, GASF, GADF images, and snapshots of raw TS resulting in training instances of size $300 \times 300$. In the RP transformation,
the synthetic anomaly injection method.

| Type   | Links | Affected | Appearance | Persistence |
|--------|-------|----------|------------|-------------|
| SuddenD | once, $[200^\text{th}, 280^\text{th}]$ | for $\infty$ |            |             |
| SuddenR | once, $[25^\text{th}, 275^\text{th}]$ | for $[5, 20]$ |            |             |
| InstaD | 2123  | 33% (700) |            |             |
| SlowD  | once, $[1^\text{st}, 20^\text{th}]$ | for 1 datapoint |             |             |  

\[ \text{RSSI}(x, \text{start}) \leftarrow \text{RSSI}(x) + \min(0, -\text{rand}(0.5, 1.5) \cdot (x - \text{start})) \]

1. \(\text{RSSI}(x, \text{start}) \leftarrow \text{RSSI}(x) + \min(0, -\text{rand}(0.5, 1.5) \cdot (x - \text{start}))\)

With respect to used metrics, we evaluated the models using the standard per class precision, recall, and F1. The precision measures how many instances detected as class A actually belong to class A, expressed as: \(\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}\), while recall measures how many of the instances belonging to class A were actually detected, expressed as: \(\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}\), where TP, FP, and FN stand for true positives, false positives, and false negatives. The F1 score is the harmonic mean between precision and recall, where larger values indicate better classifiers. We did not use accuracy as a metric because of the unbalanced dataset and as such would bias toward a class with higher representation and be misleading.

D. Explainability of the Image Models

To better understand the performance of the proposed model using RPs, it is important to explain its decisions. One way this can be done is by using guided backpropagation [39], which is a way of visualizing what image features CNNs detect. This is done by visualizing gradients with respect to the image where only positive values of gradients are used for backpropagating through ReLU layers, while the negative values are set to zero. Doing this, values in the filters of CNNs that are greater than zero signify pixels with higher importance to the recognition and show which pixels in the image contribute the most to the classification.

E. Resource Consumption

For evaluating the resource consumption of the classifiers, we consider the number of weights of the model, the FLOPs as discussed in Section V-A and the theoretical energy consumption per prediction (TEC). To calculate TEC, we used a formula provided by [40] \(\text{TEC} = \frac{\text{FLOPs}}{\text{FLOPs/Watt}}\), where FLOPs/Watt represent FLOPs per second per watt. Since NVIDIA RTX 2080 Ti was used for conducting these experiments, its theoretical FLOPs/Watt is 53.8 GFLOPS for float32 computations were used for calculating the TEC.

VII. Results

In this section, we evaluate the relative performance of the TS transformations considered in Section IV against a baseline and the DL models proposed in Section V for solving the binary and multi-class classification problems formulated in Section III. We also provide the explanation of the model to provide transparency to how the decisions are made and quantify relative resource consumption. The experimental methodology employed to obtain the results is detailed in Section VI.

A. Performance of Image Transformation Models for Binary Classification

Table II presents the results of the classifiers. The first two lines of Table II list the results of the binary classifier, while the last five lines list the results of the multiclass classifier.
The first column of Table II lists the type of classifier, the second lists the classes while the remaining four columns list the four types of considered input data and the results of the corresponding models using the three selected metrics. According to Table II, the best performing model is based on the RP as also bolded in the corresponding column.

The reason the RP model outperforms the GAF models is down to the way transformations are calculated. As explained in Section VI, while calculating the RP transformation, the thresholding and the binarization were omitted which means that none of the information about the TS sample is lost. The images for GASF and GADF transformation are computed from the angles of polar coordinates, while disregarding the information about the amplitude, which means that some of the information about the TS sample is lost, contributing to a worse classification performance compared with the RP model.

As it can be seen in the first line of the binary classifier results in Table II, the RP model achieves a near-perfect F1 score of 0.99 in detecting anomalous links. This is slightly better than the 0.98 F1 score of the GASF model, while the GADF model performs the worst out of the three with the F1 score of 0.91. What can be observed is that all three image models outperform the TS snapshot model that yields an F1 score of 0.90, which is lower by up to 0.09 compared with the F1 scores of the image-based classification models.

What can also be observed from the first row is that the GADF F1 score is lower by 0.07 compared with the GASF model which shows worse performance in detecting the anomaly which is much higher on average for GASF than for GADF, making the anomalies more prominent for the model to learn. This can best be seen in Figs. 3(c) and 4(c), respectively.

Compared with the TS snapshot model, a significant improvement in performance is achieved in detecting nonanomalous links with image models. Looking at the second line of binary classifier results in Table II, it can be seen that both the RP and GADF models perform with an F1 score of 0.97, which is slightly better than the performance of the GASF model with an F1 score of 0.94. All three models are superior to the TS snapshot model which achieved an F1 score of 0.65.

B. Performance of Image Transformation Models for Multiclass Classification

In the lower part of Table II, the last five lines list the results of the multiclass classifier, where the best performing model is based on the RP as also bolded in the corresponding column. To provide additional insights into the classification decisions, in Fig. 9, we provide explanation maps of a randomly extracted sample for each anomaly from the testing dataset. Lighter pixels show higher importance to recognition, while darker pixels show lesser importance.

The general observations regarding the performance of the image-based models are similar to the ones seen in Section VII-A. In general, the GASF model is better at detecting anomalies than the GADF model, while RP is the best image-based model, and the reasons for both observations
are the same as explained in Section VII-A. Finally, according to the F1 scores, all image models outperform the TS snapshot model, where the F1 scores can be higher by up to 0.24.

As it can be seen in the first line of the multiclass classifier results in Table II, all three imaging models perform very well in predicting the SuddenD anomaly, with the RP and GADF models having an F1 score of 1.00, while the F1 score of GASF is 0.99. All three results are similar to or even slightly better than the baseline TS snapshot model. As it can be observed in Fig. 9(a) for detecting the SuddenD anomaly, the model focuses on the black parts of the image seen in the example Fig. 4(b). This also complies with the synthetic injection approach from Table I, where SuddenD anomaly is observed toward the end of the window of the TS.

Looking at the second row of the multiclass classifier results in Table II, on average, there is a slight drop in performance in SuddenR detection compared with SuddenD. The best performing model is the RP model with an F1 score of 1.00, followed by GADF with an F1 score of 0.98. Considering the F1 score of 0.85 of the TS snapshot model, all three image-based models outperform the TS snapshot model.

In the third row of the multiclass classifier results in Table II, the performance of the InstaD classifier can be observed. Again, with an F1 score of 0.92, the RP model performs the best out of all image models, while the GASF model with an F1 score of 0.90 is a close second. All three image-based models outperform the 0.68 F1 score of the TS snapshot baseline model. Fig. 9(a) and (b) representing the explanation maps for the SuddenD and InstaD anomalies shows that for the respective anomalies, the model focuses on black cross-like areas similar to the typical representations in Figs. 4(b) and 5(b). Since SuddenR and InstaD can randomly occur anywhere along the TS length, as shown in Table I, some activation can also be seen in other parts of the figures, to help determine the anomaly.

According to the fourth line of the multiclass classifier results in Table II, compared with the 0.95 F1 score of the GASF model and the 0.85 F1 score of the GADF model, the RP model is the best performing model with an F1 score of 0.99. Both RP and GASF outperform the 0.89 F1 score of the baseline TS snapshot model, while GADF underperforms. The explanation map of the final anomaly, SlowD, is presented in Fig. 9(d). As it can be seen the most important parts of the images are along the upper and right edge. This shows that the model is mostly focused on the last part of the anomaly trace where the values are lower compared with the beginning of the trace, as seen in Fig. 6(a), and also results in a higher density area in Fig. 6(b) along the top and right edge of the figure.

In the last row of the multiclass classifier results in Table II, the performance results of classifying links without anomaly are presented. Like in all of the previous rows also here the RP model with an F1 score of 0.99 is again, compared with the 0.98 F1 score of the GASF model and the 0.97 F1 score of the GADF model, the best performing image model. All three models outperform or are equal to the baseline model which achieved an F1 score of 0.97.

C. Effect of Anomaly Share in Dataset on Classifier Performance

Since, in real-world settings, the presence of anomalies is sparse, we provide insights into how the percentage of examples in the training dataset affects the quality of the multiclass classifier. For this, we injected anomalies at the following rates 1%, 3%, 10%, 20%, 33%, and 50% in the original Rutgers dataset, according to the parameters from Table I. The datasets were then transformed into RP images, since they produce the best performing model according to Table II, and fed into the proposed DL model from Section V. The results are shown in Fig. 10, where the x-axis represents the percentage of anomalies and the y-axis the average F1 score across five-folds. It can be seen that a dataset with at least 10% of injected anomalies is already sufficient for training a good enough classifier with an average F1 score of 0.975. The average F1 score for the 20%, 33%, and 50% ranged between 0.982 and 0.984.

D. Comparison With Multiclass Classical Machine Learning Algorithms Specialized for Time Series Classification

The results are presented in Table III, where the first column lists the classes while the remaining two columns list the two best performing TS classification specialized ML models and the proposed RP model using the three selected metrics. As it can be observed from Table III both the TS KNN (TS-KNN) model and the TS SVM (TS-SVM) model are capable of correctly classifying the SuddenD anomaly with a perfect F1 score of 1.00, the same as our model. For detecting the remaining classes, our model outperforms the other two with an up to 39% point difference in the F1 score compared with the TS-KNN model and up to 9% points in the F1 score compared with TS-SVM. This shows that the proposed model using TS image transformation can outperform classical ML models that are specialized for TS classification in multiclass recognition of wireless anomalies.

E. Comparison With Well-Known DL Models

The results are presented in Table IV, where the first column lists the classes while the remaining two columns list both
selected DL models and our best performing RP model using the three selected metrics. The first five lines represent the per anomaly type performance, while the last three quantify the models’ resource consumption through the number of weights, required FLOPs, and TEC, as detailed in Section VI-E.

The general observation regarding the performance is that our model with six million trainable weights, works on par with VGG11 consisting of 190 million weights, while it outperforms the AlexNet model with 80 million weights in terms of F1 score. From the first two rows of Table IV, it can be seen that according to the F1 score, all three models had very similar performances. For classification of SuddenD and SuddenR anomalies, AlexNet achieved an F1 score of 0.99 for both classes, while both VGG11 and our model performed with the perfect F1 score of 1.00. The main difference between the models was in the classification of InstaD and SlowD anomalies, where both AlexNet scores the lowest F1 score out of all three models. On the other hand, VGG11 and our model perform very similarly with our model slightly outperforming VGG11 in the classification of InstaD anomaly. These results show that our proposed DL model can work better than some of the well-known DL architectures while having up to 97% fewer weights and needs up to $13 \times$ fewer FLOPs, while the TEC per prediction for all three models is negligible.

### F. Comparison With the State of the Art in the Wireless Link Layer Anomaly Detection

The results for the comparison can be seen in Tables V and VI. In Tables V and VI, the first column lists the classes, while the remaining columns lists classifiers from [4] or [12] and our best performing RP model using the three selected metrics.

From Table V, it can be seen that with the exception of the SuddenD anomaly detection, our model outperforms their ensemble learner in all other classes. This article showed the potential of such models but this result shows that those trained models are not yet ready for use in production because they have trouble performing when put into an ensemble. They made assumptions that only isolated cases of various anomalies can appear on links, which is not entirely aligned with a realistic environment where various anomalies can appear simultaneously. Due to these assumptions, the classifiers put into an ensemble can classify anomalies, which are not their own, as false positive. This goes to show that our model is more robust when it comes to an unseen mix of anomalies, compared with the model from [4].
The classifiers trained by Bertalanić et al. [12] used raw TS data as an input and used extensive grid search of optimal parameters in designing their models. It can be seen that with the exception of SuddenD anomaly, our model outperforms their DL classifier (DLC), logistic regression classifier (LRC), random forest classifier (RFC), and SVM classifier (SVMC) on every anomaly class. In terms of performance, the DLC is the closest to our model. In terms of F1 score, the classification of SuddenR, SlowD, and no anomaly classes is worse by only up to 0.04, while the F1 score for the classification of SlowD is worse by 0.27 compared with our model. These results additionally show the potential of our proposed approach for the classification of wireless anomalies.

VIII. CONCLUSION

In this article, we performed a first-time analysis of image-based representation techniques for wireless anomaly detection using RPs and GAFs for binary and multiclass classification and proposed a new deep neural network architecture that is able to distinguish between the considered wireless link layer anomalies. We also elaborated on the design considerations for the proposed resource-aware model. We evaluated the proposed model from four different perspectives and concluded the following.

First, our results show that the best performing model developed using the RP transformation outperforms: 1) the GAFs image-based models by up to 14% points and 2) the baseline TS snapshot image-based model by up to 32% points for binary classification and by up to 24% points for multiclass classification. We also show how decisions taken by the model can be explained and that a training dataset with 10% injected anomalies is already sufficient to train a good classifier.

Second, we show that compared with classical ML models for multiclass classification using the well-known DTW method, the proposed resource-aware model that used the RP transformation improves performance by up to 24% points.

Third, we demonstrate that compared with more well-known DL architectures, such as AlexNet and VGG11, the proposed model outperforms or performs on par while having < 10× their weights and up to ≈ 8% of their computational complexity.

Finally, compared with the state of the art in the application area that relies on classical machine learning-based ensemble model, raw TS DL model, and three other well-known classical ML algorithms, the proposed model is more robust in detecting unseen anomalies outperforming the ensemble by up to 55% point while also outperforming raw TS DL model and other ML algorithms by up to 27% and 90% points, respectively.

ACKNOWLEDGMENT

The authors would like to thank Dr. Gregor Cerar for his technical support and insightful discussions on the topic. Special acknowledgment goes to the anonymous reviewers who provided constructive and insightful comments that significantly increased the quality of this article.

REFERENCES

[1] J. Davies and C. Fortuna, *The Internet of Things: From Data to Insight*. Hoboken, NJ, USA: Wiley, 2020.
[2] T. Gale, T. Šolec, R.-A. Moşoi, M. Mohorčič, and C. Fortuna, “Automatic detection of wireless transmissions,” *IEEE Access*, vol. 8, pp. 24370–24384, 2020.
[3] G. Cerar, H. Yetgin, M. Mohoric, and C. Fortuna, “Machine learning for wireless link quality estimation: A survey,” *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 696–728, 2nd Quart., 2021.
[4] G. Cerar, H. Yetgin, B. Bertalanic, and C. Fortuna, “Learning to detect anomalous wireless links in IoT networks,” *IEEE Access*, vol. 8, pp. 363–368, 2020.
[5] J. De C. Silva, J. José P. C. Rodrigues, K. Saleem, S. A. Kozlov, and R. A. L. Rábolo, “M4DN.IoT—A networks and devices management platform for Internet of Things,” *IEEE Access*, vol. 7, pp. 53305–53313, 2019.
[6] A. Sheth, C. Doerr, D. Gruenwald, R. Han, and D. Sicker, “MOJO: A distributed physical layer anomaly detection system for 802.11 WLANs,” in *Proc. 4th Int. Conf. Mobile Syst.*, Appl. Services (MobiSys), 2006, pp. 191–204.
[7] V. L. L. Thing, “IEEE 802.11 network anomaly detection and attack classification: A deep learning approach,” in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, San Francisco, CA, USA, Mar. 2017, pp. 1–6.
[8] J. Ran, Y. Ji, and B. Tang, “A semi-supervised learning approach to IEEE 802.11 network anomaly detection,” in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–5.
[9] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, “Autoencoder-based network anomaly detection,” in *Proc. Wireless Telecommun. Symp. (WTS)*, Apr. 2018, pp. 1–5.
[10] T. Luo and S. G. Nagarajan, “Distributed anomaly detection using autoencoder neural networks in WSN for IoT,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[11] S. Gupta, R. Zheng, and A. M. K. Cheng, “ANDES: An anomaly detection system for wireless sensor networks,” in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst.*, Oct. 2007, pp. 1–9.
[12] B. Bertalanic, H. Yetgin, G. Cerar, and C. Fortuna, “A deep learning model for anomalous wireless link detection,” in *Proc. 17th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2021, pp. 265–270.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.
Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.