IEEE Open Journal of Signal Processing

Received 1 August 2023; revised 30 October 2023; accepted 3 November 2023. Date of publication 16 November 2023; date of current version 5 December 2023. The review of this article was arranged by Associate Editor Lotfi Chaari.

Digital Object Identifier 10.1109/OJSP.2023.3333756

A Tiny Transformer-Based Anomaly Detection Framework for IoT Solutions

LUCA BARBIERI [1] (Member, IEEE), MATTIA BRAMBILLA [1] (Member, IEEE), MARIO STEFANUTTI [2], CIRO ROMANO [2], NICCOLÒ DE CARLO [2], AND MANUEL ROVERI [1] (Senior Member, IEEE)

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy
[2]Sensoworks, 00192 Rome, Italy

CORRESPONDING AUTHOR: LUCA BARBIERI (e-mail: luca1.barbieri@polimi.it).

This work was supported in part by PNRR-PE-AI FAIR Project funded by the NextGeneration EU program.

ABSTRACT The widespread proliferation of Internet of Things (IoT) devices has pushed for the development of novel transformer-based Anomaly Detection (AD) tools for an accurate monitoring of functionalities in industrial systems. Despite their outstanding performances, transformer models often rely on large Neural Networks (NNs) that are difficult to be executed by IoT devices due to their energy/computing constraints. This paper focuses on introducing tiny transformer-based AD tools to make them viable solutions for on-device AD. Starting from the state-of-the-art Anomaly Transformer (AT) model, which has been shown to provide accurate AD functionalities but it is characterized by high computational and memory demand, we propose a tiny AD framework that finds an optimized configuration of the AT model and uses it for devising a compressed version compatible with resource-constrained IoT systems. A knowledge distillation tool is developed to obtain a highly compressed AT model without degrading the AD performance. The proposed framework is firstly analyzed on four widely-adopted AD datasets and then assessed using data extracted from a real-world monitoring facility. The results show that the tiny AD tool provides a compressed AT model with a staggering 99.93% reduction in the number of trainable parameters compared to the original implementation (from 4.8 million to 3300 or 1400 according to the input dataset), without significantly compromising the accuracy in AD. Moreover, the compressed model substantially outperforms a popular Recurrent Neural Network (RNN)-based AD tool having a similar number of trainable weights as well as a conventional One-Class Support Vector Machine (OCSVM) algorithm.

INDEX TERMS Anomaly detection, machine learning, self-attention, knowledge distillation, Internet of Things, transformer, compression.

I. INTRODUCTION

The recent integration of Internet of Things (IoT) sensor networks within everyday applications has enabled the collection of large volumes of time series data, fostering the development of accurate monitoring and intelligent control systems [1], [2], [3], [4]. This is not only limited to industrial IoT setups but applies to vehicular systems as well [5], [6]. Regardless of the technological implementations and system architectures, a key task of IoT networks is to acquire data for monitoring and raising alarms or alerts when needed [7]. In this context, Anomaly Detection (AD) tools are fundamental to discover unusual or anomalous patterns as well as abrupt changes in data, possibly indicating failures or malfunctions in the system being monitored [8], [9], [10]. Ideally, AD tools should also operate in real-time so as to provide up-to-date information, rapidly forward alert messages, and consequently allow a reaction to anomalous events in due time. Besides, all these functionalities should be general enough to be seamlessly applied to multiple IoT scenarios [11].

On one hand, given the ever-increasing generation and acquisition of time-series data from IoT devices, conventional AD procedures based on historical and statistical analysis may reveal to be inappropriate due to their underlying limiting statistical and modeling assumptions [12]. On the other hand, Machine Learning (ML) techniques are being increasingly used to learn descriptive relations from the collected data, or even hidden relationships among them, that facilitate the

AD process. Well-known ML-based AD tools rely on Support Vector Machines (SVMs) and Decision Trees (DTs) algorithms that aim at detecting anomalies by building a specific classifier [13], [14], [15] or by representing the time-series data in a tree structure [16], [17], [18], respectively. Other widely-used techniques include Isolation Forest (IF) [19], [20], Local Outlier Factor (LOF) [21], [22], and K-Nearest Neighbor (K-NN) [23], [24].

More recently, Neural Networks (NNs) have been increasingly applied for solving AD tasks due to their outstanding ability of capturing highly non-linear and complex relationships from data, consequently facilitating the discovery of anomalies [25], [26]. Common architectures employed in this context include Convolutional Neural Networks (CNNs) [27], Recurrent Neural Networks (RNNs) [28], Graph Neural Networks (GNNs) [29], [30], transformers [31], [32], or any combination thereof (for some examples we refer to [33]). These techniques typically target reconstructing the input time series at the output and discern the anomalies based on the reconstruction error. Indeed, normal data points should be reconstructed quite well while anomalous time series should lead to high reconstruction errors at the output indicating a possible anomaly. Besides selecting a suitable ML model for the considered AD problem, another important aspect to take into account is the learning paradigm under which the models should be trained. Supervised, semi-supervised, and unsupervised paradigms are the most utilized for AD [26]. In this paper, we specifically focus on unsupervised methods as they are able to automatically discern anomalies without any external supervision or labeled data [26], [34].

Besides selecting a suitable ML model and learning paradigm, the integration of data-driven AD strategies into IoT setups faces additional challenges. Indeed, IoT devices are typically characterized by low power consumption and limited computational capabilities, preventing the adoption of large ML models. Therefore, to harness the excellent performances of ML-based AD tools in IoT systems, novel strategies should be developed to optimize and/or compress the NNs so that they can be executed on resource-constrained devices. This paper tries to move in this direction by developing a tiny AD framework providing highly accurate learning-based AD strategies compatible with the (limited) computational and memory resources of IoT devices without sacrificing the AD capabilities.

II. RELATED WORKS AND CONTRIBUTIONS

This section discusses the related works and details the main contributions of the paper. We start by reviewing prior art on ML-based unsupervised AD methods (Section II-A), followed by a discussion on the compression strategies adopted to reduce the computational/memory complexity of large transformer models (Section II-B). Lastly, Section II-C highlights the main contributions of the paper.

A. UNSUPERVISED ANOMALY DETECTION

Common approaches targeted at solving the AD task in a fully unsupervised manner rely on RNNs that learn the temporal dependency across multi-dimensional time series. For example, the authors of [35] propose a Long Short Term Memory (LSTM)-based Variational AutoEncoder (VAE) to reconstruct time series and learn their posterior distributions. Authors in [28] develop OmniAnomaly, a stochastic RNN framework that aims at producing accurate anomaly scores based on reconstruction probabilities. Similarly, InterFusion [36] proposes a hierarchical VAE to faithfully model the relationships among the time series and exploit their representation to perform AD, whereas a temporal one-class classification model introducing dilated RNNs is designed and proposed in [37]. Other ML approaches employ Generative Adversarial Networks (GANs), where adversarially-generated time series are used to improve the discovery of anomalies [38], or a fusion of GANs with LSTMs [39]. More recently, new techniques based on the transformer architecture have also been introduced thanks to the increasing traction of the self-attention mechanism [32], [40], [41], [42], [43], [44], [45], [46], [47]. Specifically, in [40] the authors propose a graph learning with transformer for anomaly detection (GTA) that jointly learns a graph structure and models the temporal dependencies of the time series through a transformer-based module, whereas TranAD, introduced in [32], improves the accuracy of AD while reducing training times. In [41], the authors propose a root-square sparse transformer together with a dynamically-adjusted learning strategy to address concept drift in AD setups, while [42] develops ITran which incorporates knowledge about inductive biases to make the solution effective even when a relatively small amount of training data is available. On the other hand, [43] combines GANs with dilated convolutions to improve the generalization capability of the developed transformer-based AD tool. Other strategies instead focus on combining transformers with VAE to provide more robust AD methods [45], [46]. Lastly, the Anomaly Transformer (AT) introduces a novel association discrepancy metric and redesigns the self-attention mechanisms to work directly on time series data [47]. Despite their competitiveness, transformers entail an excessive number of trainable parameters posing a major challenge for their implementation on IoT devices which are typically characterized by reduced memory and computational abilities.

B. COMPRESSION STRATEGIES FOR TRANSFORMERS

To reduce the computational burden of large transformer models, compression strategies have been developed throughout the years [48], [49]. Most common approaches rely on quantization and pruning operations applied to the ML model, where the former aims at representing the NN weights by using a lower number of bits [50], while the latter removes redundant model parameters, possibly taking into account also the structure of the multi-headed self-attention mechanism [51]. Another approach for transformer compression is knowledge distillation, where a large pre-trained model (called teacher) is used as a reference for training a much smaller model (called student) [52]. Knowledge distillation strategies are typically characterized based on the number of teachers and/or students

considered during the distillation process [53], [54] or based on what information is distilled (e.g., the intermediate outputs, soft labels, and so forth) [55], [56]. Besides the aforementioned strategies, weight sharing can be also utilized to reduce transformer complexity. Under these methods, some trainable parameters are shared between different layers to reduce model complexity [57], [58]. Finally, methods based on matrix decomposition (see e.g., [59], [60]) have been used to factorize large weight matrices into smaller representations. All these techniques have shown remarkable compression capabilities specifically for Natural Language Processing (NLP) tasks where encoder-decoder architectures are largely utilized. However, their use is largely underexplored when it comes to AD tasks. Based on these considerations, in the paper we propose novel solutions for obtaining highly compressed transformer-based AD tools specifically formulated for time-series data. Note that other methods (see e.g., [61] for a review) have been developed to obtain data-driven AD strategies for time-series data compatible with resource-constrained devices. Nevertheless, to the best of our knowledge, this is the first work that considers transformer models which poses additional challenges due to their extremely large model footprint.

C. CONTRIBUTIONS

In this paper, we focus on providing highly compressed transformer-based AD algorithms that can be employed in resource-constrained IoT setups. The goal is to obtain lightweight ML models that can support highly accurate AD functionalities over heterogeneous and complex time series data. To this end, we propose a tiny AD framework that is responsible for optimizing a large AD tool based on the self-attention mechanism, namely AT, and use it for producing a substantially compressed version able to be executed on embedded devices. After optimizing the large AT model, a knowledge distillation policy is developed where the optimized AT algorithm is used to obtain a lighter student ML architecture characterized by a substantially lower number of trainable parameters. This process is done through distillation by matching the representations provided by the large teacher model with the ones attained by the smaller student model. Overall, the developed tiny AD framework is shown to produce transformer-based AD models supporting highly-accurate AD functionalities while requiring minor modifications compared to the original training process of AT, allowing for straightforward implementations. To summarize, the detailed contributions are as follows:

- we propose a tiny AD framework. The framework is responsible for optimizing large AT models and using them to produce highly compressed AD tools that can be integrated into resource-constrained devices;
- we develop a knowledge distillation strategy for compressing AT and making it suitable for on-device AD. The proposed distillation tool is general enough to be applied even when the student and teacher have a different number of layers and self-attention map dimensions without increasing the number of trainable parameters;
- we extensively validate and compare the performances of the model obtained by the framework over AD datasets used in the literature as well as using time-series data collected in a real-world bridge infrastructure monitoring use case;
- we compare the compressed model produced after applying the developed framework with a conventional One-Class Support Vector Machine (OCSVM) algorithm as well as a state-of-the-art RNN AD tool. For a fair comparison, the RNN-based AD strategy is configured so as to have roughly the same number of trainable weights of the compressed model.

Experimental results show that the proposed technique is able to provide a substantially compressed AT model, with a remarkable 99.93% less trainable parameters compared to the original implementation, with negligible performance loss. Indeed, for the considered AD datasets, the F1 score obtained by the distilled model is slightly lower (less than 2%) with respect to the original AT method. Similarly, the F1 score obtained by the original AT and the distilled version using the real-world monitoring time series data is nearly identical. The resulting model obtained after applying the proposed AD framework enables a highly-accurate discovery of anomalies and outperforms the state-of-the-art RNN AD method when the two networks have similar number of trainable weights. Numerical results also show that the compressed AT model substantially outperforms the conventional OCSVM AD strategy. Lastly, the analysis indicates that the distilled model is suitable for integration into resource-constrained environments, such as IoT or embedded systems, thanks to the reduced model footprint, i.e., number of parameters [62].

The remainder of this paper is organized as follows. Section III reviews the AT model and its training process. Section IV details the proposed tiny AD framework. Section V highlights the numerical results characterizing the distilled AT model using 4 widely-adopted AD datasets, while Section VI concentrates on the assessment of the proposed technique considering a real-world AD scenario. Finally, Section VII draws some conclusions.

III. TRANSFORMER-BASED ANOMALY DETECTION

This section briefly describes the AT method. At first, we detail the model architecture (depicted in Fig. 1) together with the multi-head anomaly-attention mechanism (Section III-A). Then, we present the learning strategy used for optimizing the model parameters and the corresponding inference stage (Section III-B).

A. ARCHITECTURE AND MULTI-HEAD ANOMALY-ATTENTION MECHANISM

Given a d-dimensional time-series $\mathbf{X} \in \mathbb{R}^{N \times d}$ of length N, the AT performs AD by reconstructing the original time series at the output. The AT architecture is composed by L layers,

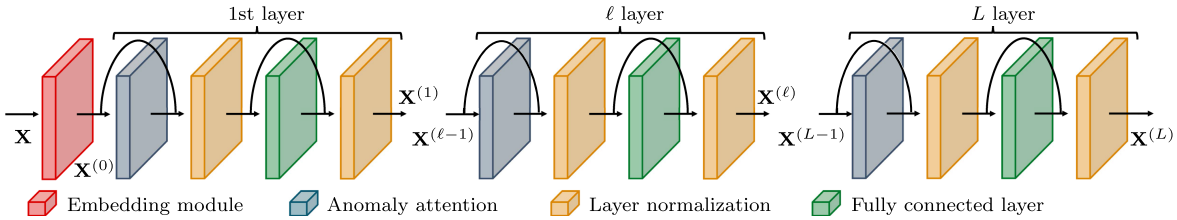


FIGURE 1. Anomaly Transformer architecture: the input $\mathbf{X}$ is sequentially processed by each layer of the architecture to extract rich features for AD by learning to reconstruct the time series at the output. The anomaly-attention module is responsible for learning the prior and series association to facilitate the discovery of anomalous data [47].

where the outputs $\mathbf{X}^{(\ell)}$ at layer ℓ, with $1 \leq \ell \leq L$, are computed as

$$\mathbf{X}^{(\ell)} = f_{\text{LN}}\left(\mathbf{Z}^{(\ell)}\mathbf{W}_Z + \mathbf{Z}^{(\ell)}\right), \tag{1}$$

being $f_{\text{LN}}(\cdot)$ the processing done by a layer normalization operation and $\mathbf{W}_Z$ are the weights associated to a fully connected layer that operates on the intermediate representation $\mathbf{Z}^{(\ell)}$ of layer ℓ, which is defined as

$$\mathbf{Z}^{(\ell)} = f_{\text{LN}}\left(f_{\text{AT}}(\mathbf{X}^{(\ell-1)}) + \mathbf{X}^{(\ell-1)}\right), \tag{2}$$

where $\mathbf{X}^{(\ell-1)}$ are the outputs of layer $\ell - 1$ and $f_{\text{AT}}(\cdot)$ denotes the processing done by the anomaly-attention mechanism. Note that for $\ell = 0$, the input time-series $\mathbf{X}$ is processed by an embedding function that initially converts $\mathbf{X}$ into a sequence of tokens and then sums the results to the output given by a positional encoding function to obtain the input sequence $\mathbf{X}^{(0)} \in \mathbb{R}^{N \times d_m}$, being d_m the dimension of the self-attention map, similarly to what is carried out in traditional self-attention procedures [63]. On the other hand, for $\ell = L$ we have $\mathbf{X}^{(L)} = \mathbf{X}_R$, i.e., the output of the model is the reconstructed time series $\mathbf{X}_R$ of the input $\mathbf{X}$.

To improve the accuracy of the architecture, a multi-head anomaly-attention mechanism is proposed to learn a robust normal-abnormal association criteria. More specifically, two additional learnable rules are introduced, namely the prior and series association using self-attention. The former is designed to learn the relative temporal distance of the time series samples, while the latter learns the association across different time series.

The multi-head anomaly-attention module is depicted in Fig. 2 and works as follows. At first, the output $\mathbf{X}^{(\ell-1)}$ of the $(\ell-1)$-th layer is reorganized into disjoint matrices $\{\mathbf{X}_h^{(\ell-1)} \in \mathbb{R}^{N \times (d_m/N_h)}\}_{h=1}^{N_h}$. Each matrix $\mathbf{X}_h^{(\ell-1)}$ is then fed to four fully connected layers to obtain queries, keys, values, and scale matrices, denoted with $\mathbf{Q}_h^{(\ell)}$, $\mathbf{K}_h^{(\ell)}$, $\mathbf{V}_h^{(\ell)} \in \mathbb{R}^{N \times (d_m/N_h)}$ and $\mathbf{\Sigma}_h^{(\ell)} \in \mathbb{R}^{N \times N_h}$, respectively, and defined as

$$\mathbf{Q}_h^{(\ell)} = \mathbf{X}_h^{(\ell-1)}\mathbf{W}_Q^{(\ell)}, \tag{3}$$

$$\mathbf{K}_h^{(\ell)} = \mathbf{X}_h^{(\ell-1)}\mathbf{W}_K^{(\ell)}, \tag{4}$$

$$\mathbf{V}_h^{(\ell)} = \mathbf{X}_h^{(\ell-1)}\mathbf{W}_V^{(\ell)}, \tag{5}$$

$$\mathbf{\Sigma}_h^{(\ell)} = \mathbf{X}_h^{(\ell-1)}\mathbf{W}_\Sigma^{(\ell)}, \tag{6}$$

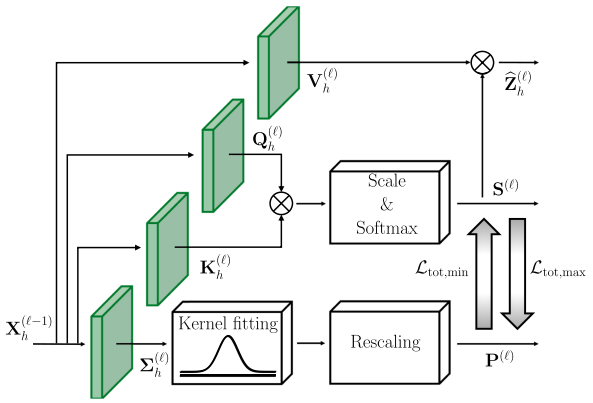


FIGURE 2. Multi-headed anomaly attention module at layer ℓ and for the h-th head. The outputs consist in the prior $\mathbf{P}^{(\ell)}$ and series $\mathbf{S}^{(\ell)}$ association matrices together with the intermediate representation of the time-series $\widehat{\mathbf{Z}}^{(\ell)}$.

being $\mathbf{W}_Q^{(\ell)}$, $\mathbf{W}_K^{(\ell)}$, $\mathbf{W}_V^{(\ell)} \in \mathbb{R}^{(d_m/N_h)\times(d_m/N_h)}$ and $\mathbf{W}_\Sigma^{(\ell)} \in \mathbb{R}^{(d_m/N_h)\times N_h}$ the associated learnable weights. Next, the queries and keys matrices for the h-th head are used to compute the series association matrix $\mathbf{S}^{(\ell)} \in \mathbb{R}^{N \times N}$ as

$$\mathbf{S}^{(\ell)} = \text{Softmax}\left(\frac{\mathbf{Q}_h^{(\ell)}\mathbf{K}_h^{(\ell)^{\text{T}}}}{\sqrt{d_m}}\right), \tag{7}$$

while the entries of the prior association matrix $\mathbf{P}^{(\ell)} \in \mathbb{R}^{N \times N}$ are evaluated by fitting a Gaussian kernel to the pairwise distances among the indices of the time series as follows

$$[\mathbf{P}^{(\ell)}]_{i,j} = \frac{1}{\sqrt{2\pi\sigma_{i,\ell,h}^2}} \exp\left(-\frac{(i-j)^2}{2\sigma_{i,\ell,h}^2}\right). \tag{8}$$

To obtain a proper distribution, $\mathbf{P}^{(\ell)}$ is normalized, i.e.,

$$\mathbf{P}^{(\ell)} = \mathbf{P}^{(\ell)} \oslash (\mathbf{P}^{(\ell)}\mathbf{1}_N), \tag{9}$$

where symbol $\oslash$ denotes the row-wise division. Lastly, the intermediate representation of the reconstructed time series for the h-th head $\widehat{\mathbf{Z}}_h^{(\ell)}$ is computed as

$$\widehat{\mathbf{Z}}_h^{(\ell)} = \mathbf{S}^{(\ell)}\mathbf{V}_h^{(\ell)}. \tag{10}$$

This process is repeated for all N_h heads and the results are concatenated together to obtain the final intermediate representation for the ℓ-th layer

$$\widehat{\mathbf{Z}}^{(\ell)} = \left[\widehat{\mathbf{Z}}_1^{(\ell)} \cdots \widehat{\mathbf{Z}}_{N_h}^{(\ell)}\right]. \tag{11}$$

B. OPTIMIZATION STRATEGY AND INFERENCE PROCESS

The main goal of the AT model is to reconstruct the time series at the output by minimizing the reconstruction loss

$$\mathcal{L}_{\text{rec}} = \|\mathbf{X} - \mathbf{X}_R\|^2, \tag{12}$$

where $\mathbf{X}_R$ is the models' output. Additionally, a symmetrized Kullback-Leibler (KL) divergence [64], representing an association discrepancy, is used to learn a robust normal-abnormal discerning rule by optimizing over the following loss term

$$\mathcal{L}_{\text{sKL}} = \frac{1}{L}\sum_{\ell=1}^{L}\Big(\text{KL}(\mathbf{P}^{(\ell)}\|\mathbf{S}^{(\ell)}) + \text{KL}(\mathbf{S}^{(\ell)}\|\mathbf{P}^{(\ell)})\Big), \tag{13}$$

where $\text{KL}(\cdot\|\cdot)$ denotes the KL divergence between two discrete distributions. Note that $\mathcal{L}_{\text{sKL}}$ is computed separately for each row of the matrices $\mathbf{P}^{(\ell)}$ and $\mathbf{S}^{(\ell)}$ as each row is assumed to model a separate distribution. The total loss is then evaluated as

$$\mathcal{L}_{\text{tot,min}} = \mathcal{L}_{\text{rec}} - \lambda\mathcal{L}_{\text{sKL}}, \tag{14}$$

which is used to updated the NN weights.

Using only (14) for training the model makes the prior association not useful for discerning anomalies. Indeed, the maximization of the association discrepancy in (14) leads to Gaussian kernels in (8) with extremely reduced standard deviation [65]. To overcome this problem, AT introduces a min-max learning strategy, where the prior association is initially optimized to be as close as possible to the series association by minimizing (14). During this phase, the series association is kept constant and not backpropagated. Then, the series association is updated so that the association discrepancy in (13) is maximized, leading to an higher ability of the model to recognize anomalous patterns in the time series data. More specifically, keeping constant the prior association, the series association is updated considering the following loss

$$\mathcal{L}_{\text{tot,max}} = \mathcal{L}_{\text{rec}} + \lambda\mathcal{L}_{\text{sKL}}. \tag{15}$$

Additionally, an early stopping criterion is used to prevent the model to overfit. In particular, the training procedure is terminated when the losses (14) and (15) stop decreasing for more than a pre-fixed number of consecutive epochs.

Upon completion of the training process, AD is performed based on the computation of an Anomaly Score (AS) that incorporates both the reconstruction quality and the value of the association discrepancy. Specifically, given a new time series $\overline{\mathbf{X}} \in \mathbb{R}^{N\times d}$, the AS for each point of the time series is evaluated as

$$\text{AS} = f_{\text{SM}}(\mathcal{L}_{\text{sKL}}) \odot \left\|\overline{\mathbf{X}} - \overline{\mathbf{X}}_R\right\|^2, \tag{16}$$

where $f_{\text{SM}}(\cdot)$ and $\odot$ denote the softmax operation and element-wise multiplication, respectively. Then, a point is flagged as an anomaly if the AS is greater than a pre-defined threshold δ_{th}.

IV. TINY ANOMALY DETECTION FRAMEWORK

This section describes the proposed tiny AD framework, which is responsible for producing a highly compressed version of AT. The procedure firstly optimizes an (uncompressed) AT model and then uses a distillation method to incorporate the knowledge acquired by the optimized model into the distilled one characterized by much lower computational complexity. Given the limited computational and/or memory resources of embedded systems, such as microcontrollers or IoT devices, the use of the original (optimized) AT architecture might not be possible in many cases. Indeed, AT relies on a modified self-attention mechanism that requires $3d_m^2$ trainable parameters associated with the learnable weights of the queries, keys, and value matrices, while the scale matrix requires $d_m N_h$ trainable parameters. All these parameters then need to be multiplied by the number of layers L of the architecture. Considering that the original implementation of AT [47] sets $d_m = 512$, $N_h = 8$, and $L = 3$ layers, the resulting memory footprint is not compatible with resource-constrained devices. Therefore, in what follows, we detail the main inner workings of the proposed tiny AD framework and how it can provide highly-accurate models suitable to be executed on resource-constrained devices.

At first, the developed AD tool is responsible for pre-training a, possibly large, AT model according to the optimization strategies presented in Section III-B so as to maximize its AD performances. Then, it instantiates a new AT network with much lower computational complexity (i.e., by limiting its number of layers L, its number of heads N_h, or by reducing its self-attention map dimension d_m). Training from scratch the compressed model may lead to sub-optimal performances due to its limited expressive capabilities. To overcome this shortcoming, the tiny AD framework integrates a knowledge distillation tool so that the knowledge of the optimized AT network, also referred to as teacher, can be incorporated into the compressed AT model, referred to as student.

Knowledge distillation strategies make use of intermediate or final model outputs to embed the knowledge acquired by a, possibly large, ML model into a much smaller NN. The choice about the specific outputs to be distilled is typically made based on the architecture at hand. In our case, we foresee three possible choices: (*i*) distilling the feature encodings provided by the embedding module, i.e., $\mathbf{X}^{(0)}$; (*ii*) distilling the prior and series association matrices, i.e., $\mathbf{S}^{(\ell)}$ and $\mathbf{P}^{(\ell)}$; (*iii*) distilling the intermediate and final outputs of AT, namely $\mathbf{X}^{(\ell)}$. Clearly, one could also combine the aforementioned strategies at the expense of larger computational complexity. The first choice may not capture the complex relationships required to reconstruct the time-series at the output as only the feature encodings between teacher and student models are distilled. Therefore, the reconstructed time-series at the output provided by the student may differ substantially from the one provided by the teacher, possibly leading to poor reconstruction results

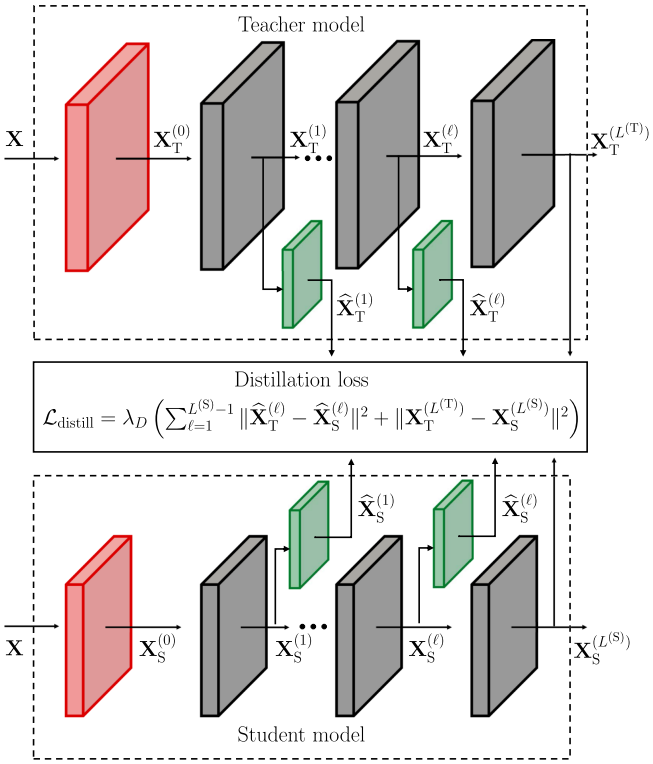


FIGURE 3. Proposed distillation tool: the knowledge of the teacher (top) is incorporated into the student (bottom) by minimizing the difference among the outputs at different layers provided by the two models.

and inefficient AD. On the other hand, the second choice may interfere with the min-max learning strategy of AT as the prior and series associations are alternatively optimized, as detailed in Section III-B. This will make the convergence of the distilled AT model difficult, possibly leading to degenerate prior/series association matrices with a subsequent decrease in AD performance. From these considerations, the strategy adopted in this paper for the distillation process is the third one as it only constrains the intermediate and final outputs of teacher and student AT to be close to each other without explicitly enforcing the prior and series associations of the two models to be equal. This also guarantees that the reconstructed time-series provided by the student closely matches the one provided by the teacher.

According to the previous discussion, the goal of the developed distillation algorithm is to match the intermediate and final outputs of the teacher and student, as highlighted in Fig. 3. The teacher is configured to have $L^{(\mathrm{T})}$ layers, $N_h^{(\mathrm{T})}$ heads, and a self-attention map dimensions of $d_m^{(\mathrm{T})}$. Similarly, the student has $L^{(\mathrm{S})} \leq L^{(\mathrm{T})}$ layers, $N_h^{(\mathrm{S})} \leq N_h^{(\mathrm{T})}$ heads, and $d_m^{(\mathrm{S})} \leq d_m^{(\mathrm{T})}$. Starting from the input time-series $\mathbf{X}$, the teacher computes the outputs at each layer $\{\mathbf{X}_{\mathrm{T}}^{(\ell)}\}_{\ell=1}^{L^{(\mathrm{T})}}$ according to (1)-(2). In a similar manner, $\mathbf{X}$ is also used by the student model to compute the outputs $\{\mathbf{X}_{\mathrm{S}}^{(\ell)}\}_{\ell=1}^{L^{(\mathrm{S})}}$. Then, fully connected layers, exemplified by the green rectangles in Fig. 3, are used to upscale/downscale the outputs of the two architectures so that they have the same dimensions. This leads to the new representations $\{\widehat{\mathbf{X}}_{\mathrm{T}}^{(\ell)} \in \mathbb{R}^{N\times d}\}_{\ell=1}^{L^{(\mathrm{T})}-1}$ and $\{\widehat{\mathbf{X}}_{\mathrm{S}}^{(\ell)} \in \mathbb{R}^{N\times d}\}_{\ell=1}^{L^{(\mathrm{S})}-1}$ for the teacher and student models, respectively. Note that the fully connected layers reuse the weights provided by the last fully connected layer of the models. By doing so, the trainable parameters of both models remain the same.

The knowledge distillation process relies on a modified loss function compared to the ones presented in Section II-I-B. In particular, the student is initially updated following the same min-max optimization strategy described before in (14), (15). Then, a distillation loss term is added to guide the training from the teacher to the student such that the discrepancy between the intermediate outputs of the two models is minimized, as highlighted in Fig. 3. Specifically, this loss is evaluated as

$$\mathcal{L}_{\mathrm{dist}} = \lambda_D \left(\sum_{\ell=1}^{L^{(\mathrm{S})}-1} \left\| \widehat{\mathbf{X}}_{\mathrm{T}}^{(\ell)} - \widehat{\mathbf{X}}_{\mathrm{S}}^{(\ell)} \right\|^2 + \left\| \mathbf{X}_{\mathrm{T}}^{(L^{(\mathrm{T})})} - \mathbf{X}_{\mathrm{S}}^{(L^{(\mathrm{S})})} \right\|^2 \right). \tag{17}$$

The value of λ_D is chosen so that the student is able to learn a robust normal-abnormal discerning rule while also incorporating the knowledge provided by the teacher.

V. NUMERICAL RESULTS ON LITERATURE DATASETS

In this section, we evaluate the performances of the proposed tiny AD framework. Section V-A details the main simulation parameters, while Sections V-B and V-C study the AD accuracy of different student model configurations and the impact of various distillation loss functions, respectively. Finally, Section V-D compares the performances of the model produced by the developed tiny AD framework with state-of-the-art and conventional baselines.

A. SIMULATION PARAMETERS

For evaluation purposes, we consider the following widely-adopted AD datasets:

- Server Machine Dataset (SMD): a 5-week long dataset collected from a large internet company containing information about 28 different machines [28];
- Soil Moisture Active Passive satellite (SMAP) and Mars Science Laboratory rover (MSL): two datasets published by NASA about telemetry of an aircraft system [66];
- Pooled Service Metrics (PSM): a dataset by eBay related to several application server nodes [67].

The main characteristics of the datasets detailing the training/validation/testing split percentages, the dimension d of the time series as well as the percentage of anomalies present in the testing split are summarized in Table 1.

The experiments focus on the comparison between the detection abilities provided by the optimized model produced by the developed tiny AD tool and the ones attained by the compressed model after distillation. In particular, the optimized AT employs $L^{(\mathrm{T})} = 3$ layers, $N_h^{(\mathrm{T})} = 8$ heads, and a self-attention map dimension $d_m^{(\mathrm{T})} = 512$, leading to ∼4.8 million trainable parameters. The optimized model is trained

TABLE 1. Statistics of Literature Datasets

Dataset	Dim.	Training [%]	Validation [%]	Testing [%]	Anomalies [%]
SMD	38	40	10	50	4.2
SMAP	25	55.4	2.7	41.9	12.8
MSL	55	35.3	8.8	55.9	10.5
PSM	25	48.1	12	39.9	27.8

for 20 epochs using a batch size of 64 examples. Unless stated otherwise, each example refers to a windowed time-series comprising $N = 100$ data points. The NN weights are learned via the Adam optimizer configured to have a linear decaying learning rate with an initial value of 0.0001 and momentum parameters of 0.9 and 0.999 while considering $\lambda = 3$ in (14) and (15). The number of epochs used for the early stopping criterion is set to 5. This model is then exploited as teacher for the distillation process.

As performance metrics, we use the standard measure of precision, recall, and F1 score for classification tasks, which are defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{18}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{19}$$

$$\text{F1} = 2\,\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{20}$$

where TP, FP, TN, and FN denote the number of true positives, false positives, true negatives, and false negatives, respectively. Additionally, we also consider the Receiver Operating Characteristics (ROC) and the Area Under the Curve (AUC) value to comprehensively characterize all the methods.

B. IMPACT OF STUDENT MODEL CONFIGURATIONS

This section studies how different student model configurations affect the AD performances. This analysis allows us to analyze the trade-off between the compression ratio of the student model and its AD capabilities and subsequently choose the best configuration. To do so, we pre-train the teacher using the parameters highlighted before and distill its knowledge considering students with $L^{(\text{S})}$ ranging from 1 up to 3 and with $d_m^{(\text{S})}$ ranging from 16 up to 256, while setting $N_h^{(\text{S})} = N_h^{(\text{T})}$. The distillation process utilizes the loss defined in (17). The student models are trained using the same configuration parameters of the teacher with $\lambda_D = 10$. During the inference process, we select the threshold for flagging an anomaly following the approach presented in [47] while also using the adjustment strategy proposed in [68]. Note that the thresholds are optimized separately for each dataset and for each student configuration.

The results of the analysis are highlighted in Table 2 which reports the precision, recall and F1 scores for each literature dataset separately. Additionally, the same table highlights the compression ratio achieved by the student model compared to the teacher. Note that the compression ratio varies slightly across the datasets due to the different dimensions of the time-series, thus we only report it for the SMD dataset. Numerical results show that the student is able to provide accurate AD performances even for large compression ratios. Indeed, the F1 score decreases slightly, i.e., less than 2%, when passing from $L^{(\text{S})} = 3$ and $d_m^{(\text{S})} = 256$ to $L^{(\text{S})} = 1$ and $d_m^{(\text{S})} = 16$. Some configurations provide less accurate results compared to others. For example, the combination $d_m^{(\text{S})} = 16$ and $L^{(\text{S})} = 3$ should be avoided as responsible for low F1 scores across most of the considered datasets. This might indicate that the student AT should be configured with a high enough $d_m^{(\text{S})}$ when $L^{(\text{S})}$ is large to not incur in performance degradation. Overall, the analysis shows that the performance of the student model does not deteriorate too much for all analyzed configurations. Therefore, the best trade-off between model complexity and AD accuracy is achieved when the student is configured with $L^{(\text{S})} = 1$ and $d_m^{(\text{S})} = 16$. Adopting such a configuration allows us to reduce the number of parameters of the student by a staggering 99.91% compared to the original AT without compromising the AD accuracy of the compressed model.

C. IMPACT OF DIFFERENT LOSS FUNCTIONS

Knowledge distillation processes rely on dedicated and handcrafted loss functions to transfer the knowledge between teacher and student models. This section analyzes the impact of the specific loss function used during distillation. To do so, we consider three different loss functions: (*i*) the L2 loss defined in (17), (*ii*) an L1 loss, and (*iii*) a smooth L1 loss. As done in Section IV, the L1 loss is evaluated as

$$\mathcal{L}_{\text{dist}} = \lambda_D \left(\sum_{\ell=1}^{L^{(\text{S})}-1} \left\| \widehat{\mathbf{X}}_{\text{T}}^{(\ell)} - \widehat{\mathbf{X}}_{\text{S}}^{(\ell)} \right\| + \left\| \mathbf{X}_{\text{T}}^{(L^{(\text{T})})} - \mathbf{X}_{\text{S}}^{(L^{(\text{S})})} \right\| \right), \tag{21}$$

On the other hand, the smooth L1 loss is computed as

$$\mathcal{L}_{\text{dist}}^{(\text{SL1})} = \lambda_D \left(\sum_{\ell=1}^{L^{(\text{S})}-1} \mathcal{L}_{\text{s}} \left(\widehat{\mathbf{X}}_{\text{T}}^{(\ell)}, \widehat{\mathbf{X}}_{\text{S}}^{(\ell)} \right) + \mathcal{L}_{\text{s}} \mathbf{X}_{\text{T}}^{(L^{(\text{T})})}, \mathbf{X}_{\text{S}}^{(L^{(\text{S})})} \right), \tag{22}$$

where $\mathcal{L}_s$ is defined in [69]. Note that the parameter β for $\mathcal{L}_s$ is set to 1. According to the previous analysis, we select the student model configuration with $L^{(\text{S})} = 1$, $N_h^{(\text{S})} = 8$ and $d_m^{(\text{S})} = 16$, while the teacher is configured as in Section V-A. The same training configurations detailed in Section V-A are also used here for updating the weights of the student and teacher models, while we set $\lambda_D = 10$ for all the losses. Finally, during inference, we optimize the thresholds according to the policy presented in [47]. This leads to setting $\delta_{\text{th}} = \{0.398, 0.145, 0.157, 0.287\}$ for the SMD,

TABLE 2. Performance Comparison Over Different AD Literature Datasets in Terms of Precision (P), Recall (R), and F1 Score for Different Student Model Configurations

Configuration			SMD			SMAP			MSL			PSM		
$L^{(S)}$	$d_m^{(S)}$	Compr. ratio	P	R	F1	P	R	F1	P	R	F1	P	R	F1
[#]	[#]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
3	256	74.48	88.82	93.29	91.00	93.58	99.23	96.32	91.96	93.36	92.65	97.13	98.75	97.93
2	256	82.72	88.87	94.55	91.62	93.59	99.35	96.39	92.17	97.59	94.80	97.17	98.63	97.89
1	256	90.96	88.74	92.32	90.50	93.60	98.76	96.11	91.39	95.15	93.23	97.21	98.64	97.92
3	64	98.21	89.22	94.11	91.60	93.64	98.59	96.05	92.09	96.70	94.34	97.05	98.79	97.91
2	64	98.74	89.62	95.04	92.25	93.06	91.46	92.25	92.10	96.70	94.35	97.16	98.20	97.68
1	64	99.27	88.57	90.54	89.55	93.68	98.00	95.79	92.02	96.70	94.31	97.03	98.53	97.77
3	16	99.84	86.12	74.72	80.02	93.54	98.51	95.96	91.63	93.07	92.35	97.17	98.86	98.01
2	16	99.88	88.24	88.10	88.17	93.61	98.86	96.16	91.68	93.27	92.47	97.10	98.94	98.01
1	16	99.91	88.30	90.80	89.53	93.66	99.22	96.36	91.77	95.94	93.81	97.12	98.10	97.61

TABLE 3. Performance Comparison Over Different AD Literature Datasets in Terms of Precision (P), Recall (R), and F1 Score for Different Loss Functions

Loss	SMD			SMAP			MSL			PSM		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
L2 loss	88.30	90.80	89.53	93.66	99.22	96.36	91.77	95.94	93.81	97.12	98.10	97.61
L1 loss	87.89	86.78	87.33	93.65	98.99	96.25	91.94	97.50	94.64	97.19	98.12	97.65
Smooth L1 loss	88.29	89.34	88.81	93.65	99.19	96.31	92.03	97.50	94.69	97.19	98.27	97.73

SMAP, MSL and PSM datasets when the L2 loss is considered, while we set $\delta_{\text{th}} = \{0.395, 0.149, 0.152, 0.286\}$ and $\delta_{\text{th}} = \{0.398, 0.145, 0.158, 0.288\}$ for the L1 and smooth L1 losses considering the same datasets, respectively.

Table 3 reports the precision, recall, and F1 scores considering the three aforementioned loss functions. Comparing the results, we can see that the L2 loss is advantageous for the SMD and SMAP datasets, while the smooth L1 has slightly higher performances when MSL and PSM are considered. Nevertheless, the results also highlight that the performances of the proposed approach are not that much affected by the specific loss used for the distillation process. Indeed, the difference among all losses in terms of precision, recall, and F1 score is only marginal. Based on this discussion, the loss function used for the distillation process for the following results is (17).

D. COMPARISON WITH OTHER BASELINES

This section evaluates the performances of the proposed tiny AD framework by comparing it with other baseline approaches. The knowledge distillation process considers as teacher the AT model set as in Section V-A while the student is configured according to the analysis provided in Section V-B. For comparison purposes, we consider a widely-used RNN-based AD method, namely LSTM-VAE [35], implemented so as to roughly have the same number of trainable parameters of the student AT, as well as the conventional OCSVM AD strategy [70]. All models, apart OCSVM, are optimized using the configuration parameters detailed in Section V-A while we set $\lambda_D = 10$ in (17). As far as the inference process is concerned, the threshold δ_{th} is set separately for each dataset according to the policy in [47]. This process leads to $\delta_{\text{th}} = \{0.016, 0.015, 0.012, 0.02\}$ for the SMD, SMAP, MSL, and PSM datasets for the teacher, while for the student we obtain $\delta_{\text{th}} = \{0.398, 0.145, 0.157, 0.287\}$. On the other hand, the thresholds for LSTM-VAE are found via a non-exhaustive search over a grid of possible values.

Table 4 provides a comparison between the models produced by the developed tiny AD tool, LSTM-VAE and OCSVM in terms of precision, recall and F1 metrics. Overall, the distilled AT model is able to closely match the performances of the original (optimized) AT architecture while requiring a substantially lower computational complexity. Indeed, the F1 metric achieved by the student is only slightly lower than the one obtained by the teacher for all datasets. Focusing now on the detection abilities of LSTM-VAE, they are largely inferior compared to the ones achieved by the compressed AT architecture despite having a similar number of trainable parameters. Specifically, LSTM-VAE provides very poor detection abilities for the SMD and SMAP datasets where the F1 score is reduced more than 10% compared to all other methods, while for the MSL and PSM the accuracy

TABLE 4. Performance Comparison Over Different AD Literature Datasets in Terms of Precision (P), Recall (R), and F1 Score

Method	SMD			SMAP			MSL			PSM		
	P [%]	R [%]	F1 [%]	P [%]	R [%]	F1 [%]	P [%]	R [%]	F1 [%]	P [%]	R [%]	F1 [%]
Original AT	89.02	94.67	91.76	93.60	99.29	96.36	92.06	98.06	94.96	97.37	98.12	97.75
Distilled AT (ours)	88.30	90.80	89.53	93.66	99.22	96.36	91.77	95.94	93.81	97.12	98.10	97.61
LSTM-VAE	74.91	81.92	78.26	92.75	55.94	69.78	90.96	86.93	88.90	98.06	88.65	93.12
OCSVM	42.18	74.89	53.96	48.81	50.32	49.55	58.63	84.22	69.13	63.31	82.29	71.56

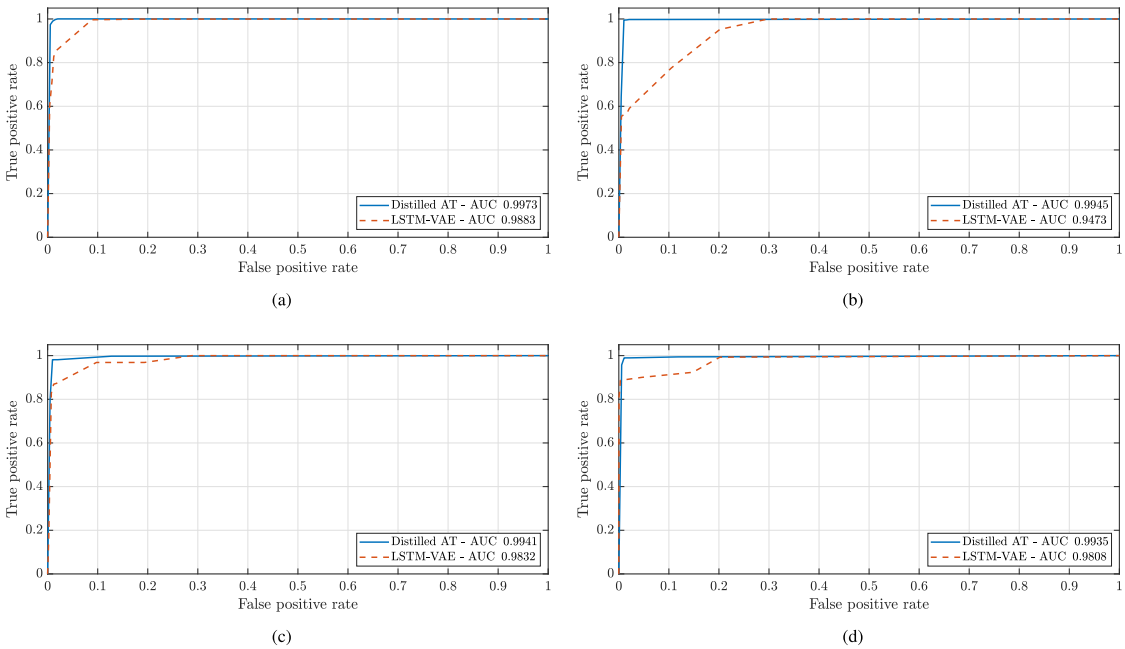


FIGURE 4. ROC curves obtained by the distilled AT model provided by the tiny AD framework and the LSTM-VAE method on different datasets. (a) SMD; (b) SMAP; (c) MSL; and (d) PSM. The corresponding AUC values of the two methods are reported in the legends.

reduction is not so severe (i.e., only a 4–5% reduction on the F1 metric is observed). Similar results are also achieved by OCSVM which is shown to provide AD capabilities far below the ones attained by both LSTM-VAE and the distilled AT model. This suggests how conventional AD strategies, such as OCSVM, might not be adequate for heterogeneous time-series data, and more complex data-driven AD tools are required for improving the performances.

To complement the analysis, we report in Fig. 4 the ROC curves obtained by the distilled AT model produced by the developed AD framework and the LSTM-VAE over all datasets used in the experiments and their corresponding AUC values. This set of results further confirms the findings of the previous analysis: the compressed AT model is able to outperform LSTM-VAE in all cases. The AUC values obtained by the distilled algorithm are superior when compared with the ones attained by LSTM-VAE, especially if the SMAP dataset is considered. Overall, the analysis shows that the proposed tiny AD framework is able to provide a highly-accurate distilled model that closely matches the performances provided by the original (optimized) AT while also outperforming a LSTM-VAE anomaly detector having a similar number of trainable parameters.

VI. CASE STUDY: ANOMALY DETECTION ON A BRIDGE INFRASTRUCTURE

This section analyzes the detection abilities of the models produced by the developed AD tool using time series data acquired from a real-world bridge infrastructure monitoring system deployed in Italy. In the following, we describe the main technical parameters of the dataset (Section VI-A). Next, we evaluate the detection abilities of the teacher model trained under the proposed tiny AD framework by considering different input configurations in order to select the one that provides the best performance (Section VI-B). Lastly, Section VI-C characterizes the AD capabilities of the compressed model which is distilled from the best teacher model selected in Section VI-B according to the framework described in Section IV.

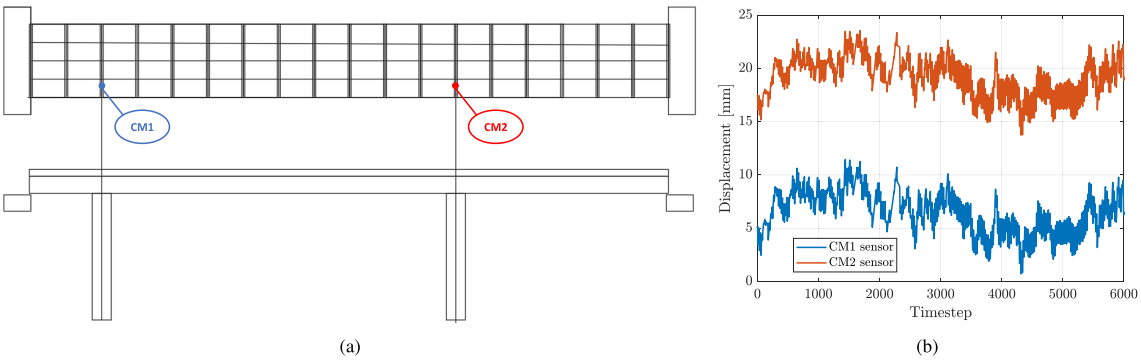


FIGURE 5. AD case study on a real monitoring infrastructure: (a) sketch of the bridge being monitored with the installed IoT sensors; (b) time series data from crack meters in a monitoring period of 9 months.

A. DATASET DESCRIPTION

The case study considers an IoT sensor network comprising two crack-meters monitoring the status of a bridge (illustrated in Fig. 5(a)) by measuring the variation of the displacement across cracks and/or joints over time. Communication protocols, such as MQTT, are used to connect the edge devices to a central software, where raw data are processed. The sampling period is configured so that a recording is generated every two hours. The resulting time series data recorded over 9 consecutive months, namely from December 2021 up to August 2022, comprises 8000 samples and it is shown in Fig. 5(b). We split this dataset into two parts: the first 6000 samples are used used for training, while the remaining 2000 ones constitute the testing dataset.

To assess the performances of the proposed tiny AD framework, we introduce hand-crafted, yet realistic, anomalies in the testing set, while we assume that the training data is free from abnormal points. The following four types of perturbations are introduced in the testing time series:

- Type I - point anomaly: a spike in the time series data;
- Type II - step anomaly: a step function is superimposed to the values of the time series for N_{A} consecutive data points;
- Type III - ascending exponential anomaly: an increasing exponential function is superimposed to the values of the time series data for N_{A} consecutive data points;
- Type IV - descending exponential anomaly: a decreasing exponential function is superimposed to the values of the time series data for N_{A} consecutive data points.

Each type of anomaly is introduced by adding one of the functions $f_{\mathrm{A}}(t)$ provided in Table 5 to the raw sensors data. Specifically, two spikes are added at timesteps $t_{\mathrm{D}} = 900$ and $t_{\mathrm{D}} = 1100$, with $B = 5$, while σ_{T} denotes the empirical standard deviation computed over the training dataset. Note that we consider two different values of σ_{T}, one for each crack-meter. For what concerns the other three anomaly types, i.e., step, increasing, and decreasing exponential, they are introduced in two non-overlapping windows comprising $N_A = 30$ consecutive points each, and using the same values for B and σ_T as before. The window starts at timestep $t_{\mathrm{D1}} = 900$ and ends at $t_{\mathrm{D2}} = 930$ for the first time series, while for the second one, the initial and final timesteps are chosen as $t_{\mathrm{D2}} = 1100$ and $t_{\mathrm{D2}} = 1130$. Additionally, the exponential functions used for simulating the last two anomalies use $b = 8$. The resulting anomalous patterns of the four types of anomalies are highlighted in Fig. 6.

TABLE 5. Definition of Anomaly Functions

Anomaly	$f_{\mathrm{A}}(t)$
Type I	$B\,\sigma_{\mathrm{T}}\,\delta(t - t_{\mathrm{D}})$
Type II	$\begin{cases} B\,\sigma_{\mathrm{T}} & \text{if } t_{\mathrm{D1}} \le t \le t_{\mathrm{D2}} \\ 0 & \text{otherwise} \end{cases}$
Type III	$\begin{cases} B\,\sigma_{\mathrm{T}}\,\dfrac{b^t - 1}{b - 1} & \text{if } t_{\mathrm{D1}} \le t \le t_{\mathrm{D2}} \\ 0 & \text{otherwise} \end{cases}$
Type IV	$\begin{cases} -B\,\sigma_{\mathrm{T}}\,\dfrac{b^{-t} - 1}{b^{-1} - 1} & \text{if } t_{\mathrm{D1}} \le t \le t_{\mathrm{D2}} \\ 0 & \text{otherwise} \end{cases}$

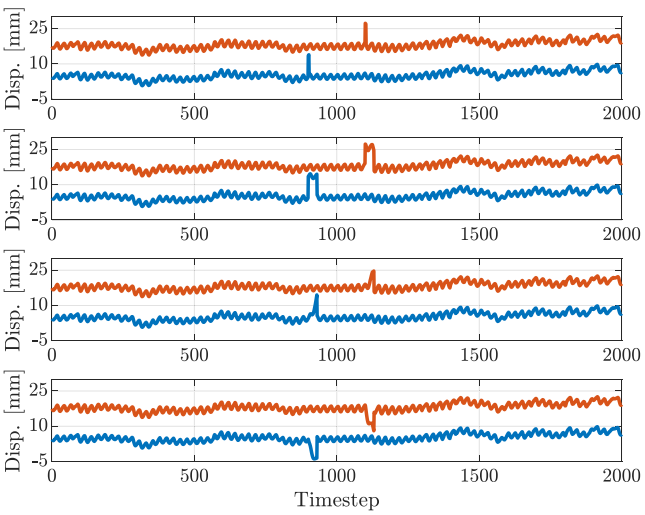


FIGURE 6. Testing time series of crack meters with anomalies. The considered anomalies include (from top to bottom) point, step, ascending, exponential, and descending exponential perturbations.

In the following, we use this database to initially assess the ability of the original AT model (obtained from the AD framework detailed in Section IV) of detecting the four types of anomalies. Then, we will apply the distillation strategy presented in Section IV to compress the AT model and assess its AD performances.

B. ANALYSIS OF THE TEACHER WITH DIFFERENT WINDOWS AND OVERLAPS

This section aims at studying the detection accuracy of the original AT model trained under the proposed AT framework using the raw time series data acquired by the monitoring facility. To achieve this goal, we consider different input configurations for the selection of the best performing (uncompressed) AT model and use it to guide the student's training. Recalling that AT accepts at the input a time series $\mathbf{X}$ with d dimensions and length N, we vary the number of data points N as well as the overlap among adjacent segments and evaluate the performances accordingly. Specifically, we consider two values of N, namely $N = 6$ and $N = 12$, which correspond to windows spanning half a day and one day of recording, and three overlaps, i.e., 0%, (no overlap exists between adjacent segments), 50% and 80%. The original AT model is configured as in Section V, namely it has $L^{(\mathrm{T})} = 3$ layers, $N_h^{(\mathrm{T})} = 8$ heads, and a self-attention map with $d_m^{(\mathrm{T})} = 512$ dimensions. Additionally, the Adam optimizer is used for updating the weights considering a batch size of 64 examples, a learning rate of 0.001, and momentum parameters of 0.9 and 0.999. The model is trained for 200 epochs and it is stopped preemptively when the validation loss does not reduce over 10 consecutive epochs.

Fig. 7 shows the AS obtained by the AT model at the end of the training process for the testing dataset containing the point anomaly with $N = 6$ (Fig. 7(a)) and $N = 12$ (Fig. 7(b)), and considering all overlaps. In particular, each figure firstly shows the time series data of the testing dataset at the top, while the following subplots highlight the AS achieved considering 0%, 50%, and 80% overlaps.

The results indicate that the point anomaly can be detected for all windows and overlaps considered in the analysis, even though in some cases the AS is not particularly high (see e.g., the case with $N = 12$ and 50% overlap). Indeed, for all cases, two spikes are present in the area delimited by the light gray boxes, which highlight the regions comprising the anomalies. Nevertheless, for $N = 6$ the model is able to recognize the second anomaly fairly easily as the associated spike is more pronounced. On the other hand, for $N = 12$, the AT is able to reliably detect the first anomaly. Besides choosing an appropriate value for N, also the overlap has an impact on the overall performances. Indeed, for $N = 6$ a high overlap, i.e., 80%, should be preferred to facilitate the AD process, while for $N = 12$ the model seems to provide the highest AS when the adjacent windows have no overlap. Overall, taking into account also the magnitude of the AS across different input configurations, the model trained with $N = 6$ and 80% overlap is the one showing the highest peaks in the neighborhood

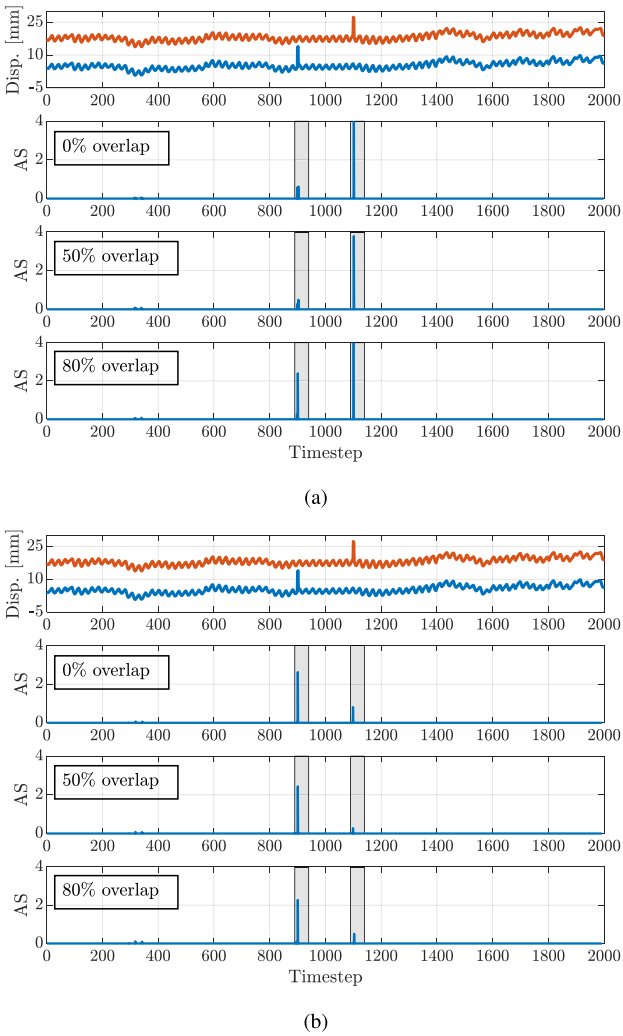


FIGURE 7. Analysis of the detection performance of the original AT for type I anomaly with windows comprising: (a) 6 points and (b) 12 points. From top to bottom, each figure reports the testing time series and the AS obtained considering 0%, 50%, and 80% overlaps. The position of the anomalies is highlighted with a light gray box in all subfigures.

of the point anomalies, therefore, it should be selected for facilitating the AD process.

The detection abilities provided by AT over the testing dataset with the step anomaly, whose results are reported in Fig. 8(a) and 8(b) for $N = 6$ and $N = 12$, respectively, indicate that the AT model is able to recognize the anomalies also in this case. However, $N = 12$ is seen to provide more sparse peaks in the areas delimited by the gray boxes, while $N = 6$ obtains scores with high values that cover more uniformly the area containing the anomalies. The overlaps are also shown to affect more the model trained with $N = 6$ rather than that with $N = 12$. Indeed, 80% overlap should be avoided when $N = 6$ as the peaks of the AS are not particularly high while the results obtained for 0% and 50% overlaps are quite similar. On the other hand, for $N = 12$ different overlaps influence the ASs provided by the model only negligibly. According to the results obtained, also under this case $N = 6$ should

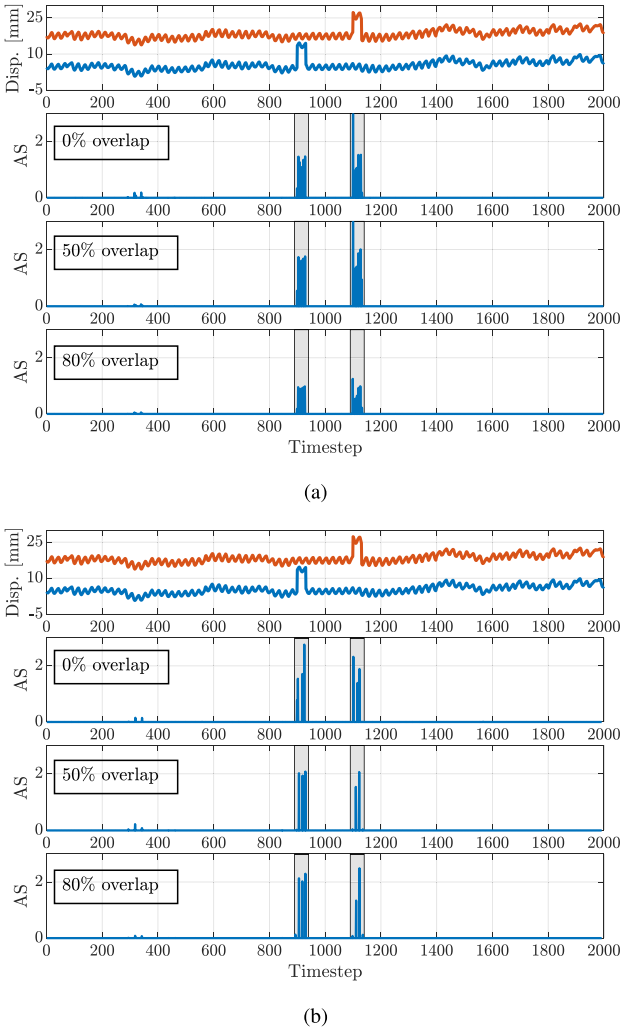


FIGURE 8. Analysis of the detection performance of the original AT for type II anomaly with windows comprising: (a) 6 points and (b) 12 points. From top to bottom, each figure reports the testing time series and the AS obtained considering 0%, 50%, and 80% overlaps. The position of the anomalies is highlighted with a light gray box in all subfigures.

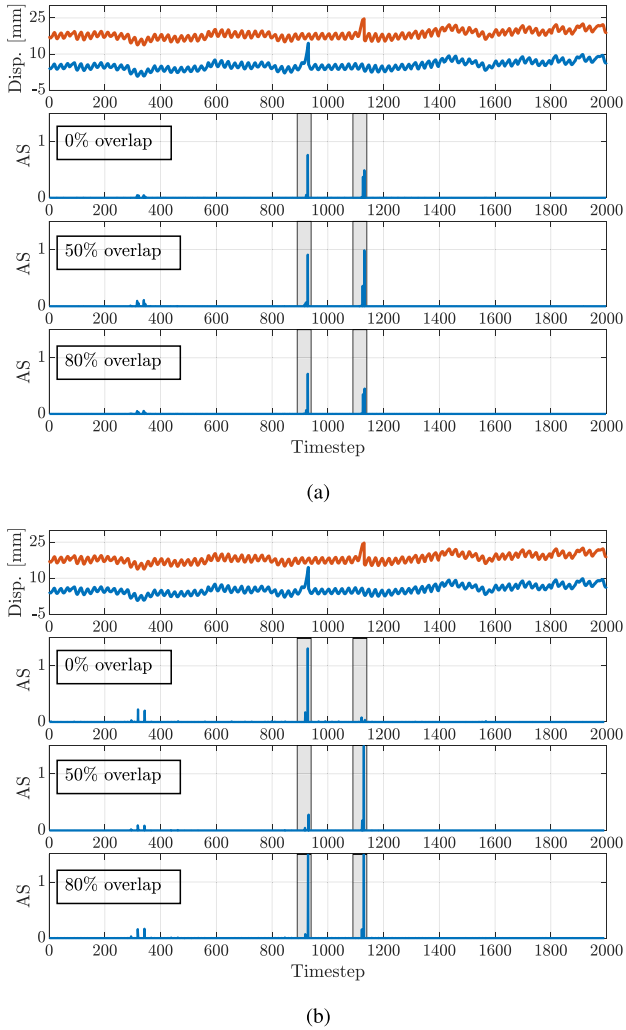


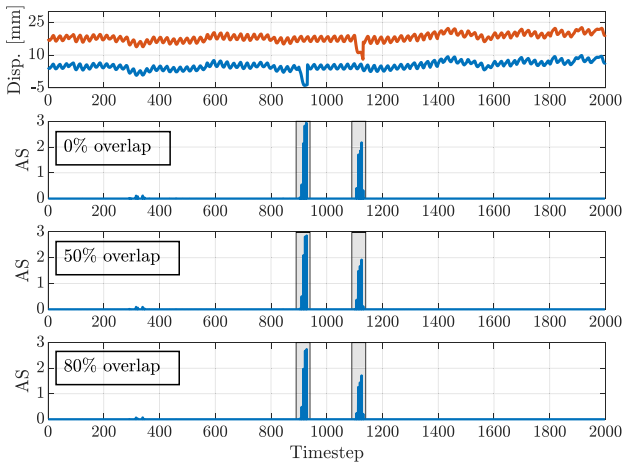
FIGURE 9. Analysis of the detection performance of the original AT for type III anomaly with windows comprising: (a) 6 points and (b) 12 points. From top to bottom, each figure reports the testing time series and the AS obtained considering 0%, 50%, and 80% overlaps. The position of the anomalies is highlighted with a light gray box in all subfigures.

be preferred when compared with $N = 12$ as showing more distributed and higher AS values in the vicinity of the step anomalies, provided the overlap is below 80%.

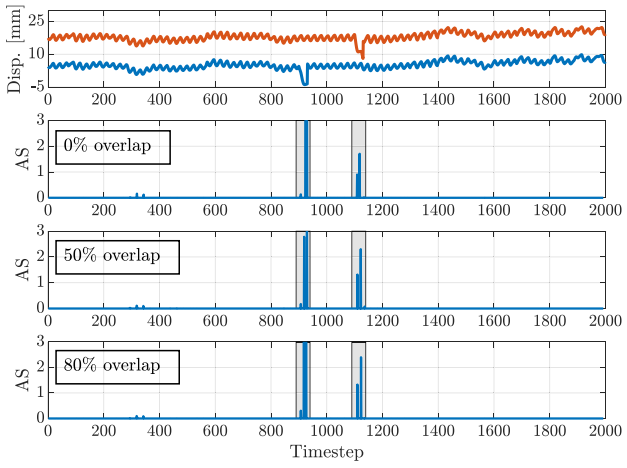
Fig. 9 reports the results achieved by the AT model for the testing dataset containing the ascending exponential anomaly with $N = 6$ (Fig. 9(a)) and $N = 12$ (Fig. 9(b)). The results are in line with the ones found in the previous analysis for the step anomaly: $N = 6$ generally provides AS with peaks more distributed in the neighborhood of the anomalies, while $N = 12$ has fewer peaks but has some spikes with higher magnitude (see e.g., when the overlap is 80%). Interestingly, when no overlap exists, the AS provided by the model trained with $N = 12$ is quite low in the second window, indicating that under this input configuration AT is not able to detect all anomalies. Overall, the best-performing input configuration is $N = 6$ with 50% overlap, as it shows an AS covering most of the anomalies in the testing dataset.

The results considering the last type of anomaly, i.e., the descending exponential, are presented in Fig. 10(a) for $N = 6$ and in Fig. 10(b) for $N = 12$. Compared to the previous type, this anomaly seems to be easier to be detected as the AS provided by the model under all input configurations shows a large number of peaks in the light gray boxes depicted in the figures. Again, using a window of $N = 6$ allows the detection of more anomalous points compared to the case of $N = 12$. Similarly as before, $N = 6$ should be selected in conjunction with 0% or 50% overlap as the AS magnitude is higher in the neighborhood of the anomalies, while when $N = 12$, the overlap should be selected between 50% and 80% to improve the detection of the second anomaly. Considering the different input configurations, $N = 6$ is shown again to provide more accurate detection results compared with $N = 12$, provided that the overlap is below 80%.

To finalize the analysis on the impact of the different input configurations for the (uncompressed) AT model, we report



(a)



(b)

FIGURE 10. Analysis of the detection performance of the original AT for type IV anomaly with windows comprising: (a) 6 points and (b) 12 points. From top to bottom, each figure reports the testing time series and the AS obtained considering 0%, 50%, and 80% overlaps. The position of the anomalies is highlighted with a light gray box in all subfigures.

in Table 6 the number of correct predictions together with the number of false positives obtained by AT considering all the combinations studied before. The results are obtained by numerically searching for the detection threshold δ_{th} that gives the highest and lowest number of correct predictions and false positives, respectively.

For the point anomaly, the performances are not affected by the specific choice of the window and the overlap. However, this does not hold when considering different types of anomalies. Generally, a window of $N = 6$ is seen to provide the highest number of correct predictions while also having a slightly higher number of false positives compared with $N = 12$. Therefore, $N = 6$ should be selected to achieve the highest AD accuracy. Focusing also on the performances for different overlaps, the results show that 80% should be avoided as responsible for low accuracy. On the other hand, 0% overlap provides the least amount of false positives but

TABLE 6. Detection Accuracy and False Positive Analysis for the Original AT Model

Anomaly	Window size N [#]	Overlap [%]	δ_{th} [#]	FP [#]	TP [#]
Type I	6	0	0.58	0	2
		50	0.47	0	2
		80	2.31	0	2
	12	0	0.79	0	2
		50	0.26	0	2
		80	0.48	0	2
Type II	6	0	0.17	1	11
		50	0.53	0	10
		80	0.14	4	11
	12	0	0.75	0	6
		50	1.51	0	5
		80	1.32	0	5
Type III	6	0	0.05	2	4
		50	0.03	4	7
		80	0.07	5	6
	12	0	0.17	3	2
		50	0.05	8	3
		80	0.16	2	3
Type IV	6	0	0.31	2	9
		50	0.12	3	10
		80	0.03	3	10
	12	0	0.69	1	4
		50	0.18	0	4
		80	0.31	1	5

it also detects a lower number of anomalies when compared with the 50% case. Considering all these aspects, we finally select the input configuration with $N = 6$ and 50% overlap and use the resulting model to perform the distillation process as detailed in Section IV.

C. ASSESSMENT OF THE DISTILLED MODEL

This section aims at evaluating the AD performances of the compressed AT model produced by the proposed AD framework after distillation. According to the previous analysis, the optimized AT model is pre-trained using $N = 6$ and 50% overlap and using the same optimization parameters presented before. Regarding the compressed model, it is configured as

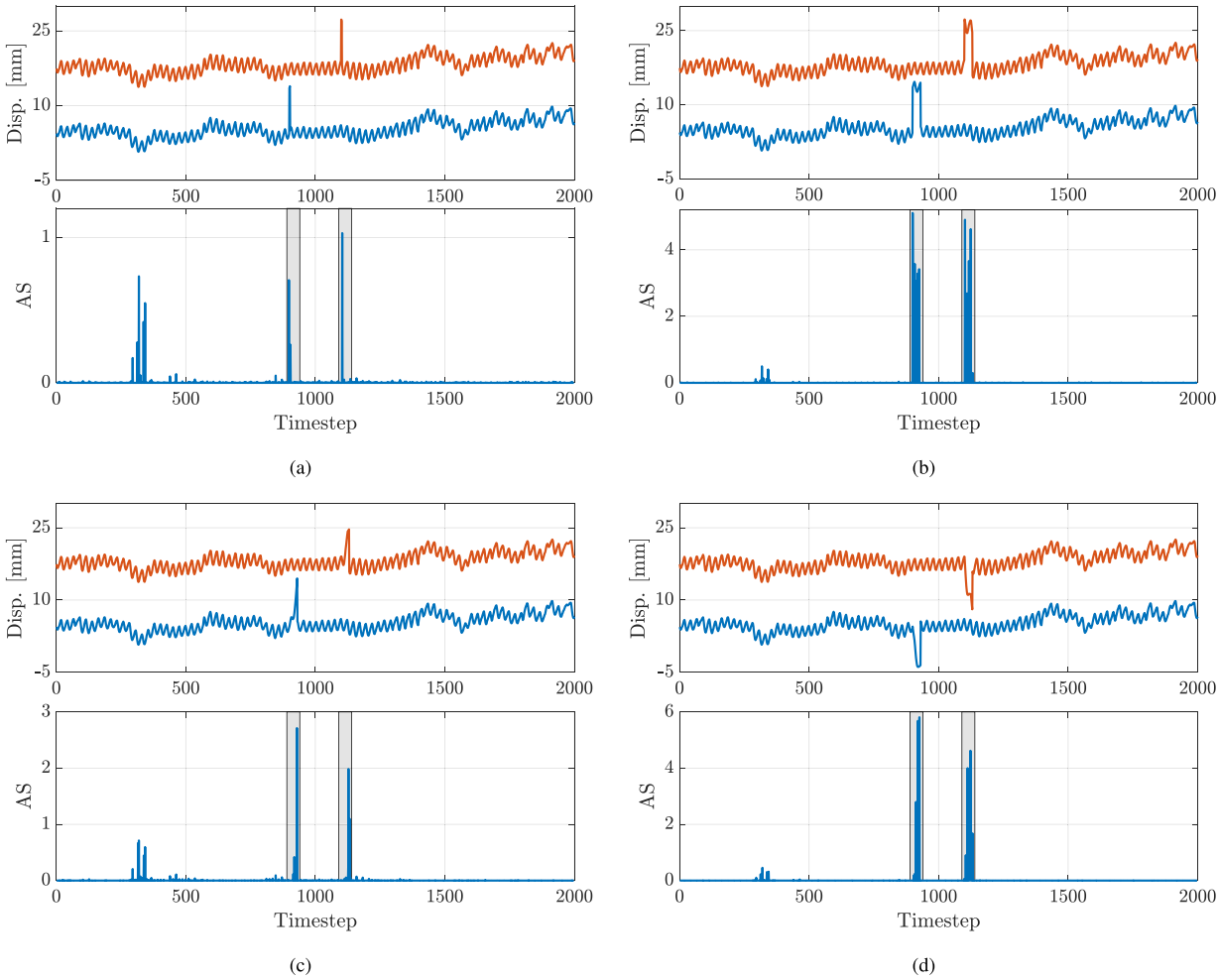


FIGURE 11. Analysis of the detection performance of the distilled AT model considering the following anomaly types: (a) point, (b) step, (c) ascending exponential, and (d) descending exponential. The top part of each figure reports the testing dataset highlighting the specific anomaly type, while the bottom part shows the AS provided by the model.

in Section V, with $L^{(S)} = 1$ layer, $N_h^{(S)} = 8$ heads, and a self-attention map dimension $d_m^{(S)}$ of 16, leading approximately to an overall number of 1400 parameters. It is also trained using a window of $N = 6$ and an overlap of 50%. The distillation process runs for 200 epochs with $\lambda_D = 10$ and it is early stopped if the validation loss does not decrease for more than 5 consecutive epochs. Note that both models do not have access to the simulated anomalies during training (they are added only to the testing database), nor do they use labels to identify anomalies as we consider a fully unsupervised learning setting.

Fig. 11 reports the results obtained by the compressed model over the testing dataset comprising the point (Fig. 11(a)), step (Fig. 11(b)), ascending exponential (Fig. 11(c)) and descending exponential (Fig. 11(d)) anomalies. Each figure shows at the top the testing time series data together with the introduced anomalies, while the bottom highlights the AS obtained by the distilled AT model. To ease the comparison, we also highlight the position of the anomalies with a light gray box. The AS for all figures shows that the model is able to recognize fairly easily all anomaly types. Indeed, several spikes are reported in the positions delimited by the light gray boxes which indicate that the model is confident that the time series data contains anomalous points. This demonstrates that the proposed AD framework integrating the distillation strategy detailed in Section IV is able to provide a lightweight AT model that supports highly-accurate AD capabilities. When comparing the obtained results with the ones achieved by the uncompressed model (Section VI-B), it can be noticed that for some cases high AS values are reported outside of the areas delimited by the light gray boxes (see e.g., the timestep ranging from 300 up to 360 in Fig. 11(a)). This may be caused by the fact that the self-attention map of the compressed AT model is highly reduced making it more difficult for the anomaly attention mechanism to correctly learn the prior and series association and thus they do not fully capture the temporal dependency of the time series. Nevertheless, a careful optimization of the detection threshold δ_{th} may be helpful in suppressing those cases.

To comprehensively characterize the performances of the model obtained by the proposed AD framework after the distillation process, we report in Table 7 the number of correct predictions and false positives obtained by the distilled AT

TABLE 7. Detection Accuracy and False Positive Analysis for the Proposed Compressed AT Model

Anomaly	δ_{th} [#]	FP [#]	TP [#]
Type I	0.68	1	2
Type II	1.90	0	11
Type III	0.38	5	6
Type IV	0.11	4	9

model considering the anomalies previously described. The results are obtained by optimizing the detection threshold δ_{th} in order to maximize the number of correctly detected anomalies while minimizing the number of false positives. The results obtained by the compressed AT model are in line with the ones reported in Table 6 for the case with $N = 6$ and 50% overlap: the number of correct predictions closely matches the one provided by the uncompressed model for all anomaly types. The main difference with respect to the previous case relies on the number of false positives provided by the model trained under the proposed distillation framework. Indeed, a slightly higher number of false positives is shown by the distilled model when compared with the same number provided by the uncompressed AT architecture. This is likely caused by the fact that the student has a much smaller representation capacity compared to the teacher leading it to output a relatively high AS between timesteps 200 and 400. This consequently causes the spikes detected in that region of the testing time series to be flagged as anomalies. Nevertheless, the results still suggest that the model obtained after distillation is capable of closely matching the performances provided by the original AT implementation.

VII. CONCLUSION

This paper explored the problem of accurate AD in IoT setups characterized by devices having limited energy/computing capabilities. Transformer-based AD tools have been demonstrated to provide outstanding performances in detecting anomalies over heterogeneous and streaming time series data. Nevertheless, they generally comprise large and complex NNs, making them unsuitable for being deployed in IoT devices due to energy and/or computing constraints. To overcome such limitations, this paper proposed an effective tiny AD framework based on knowledge distillation. The developed tool aims at initially finding an optimized version of a state-of-the-art AD method, namely AT, and use it as input for the distillation process whose goal is to produce a substantially compressed AT model able to achieve accurate detection abilities.

The proposed framework is firstly assessed using widely adopted AD datasets showing its efficacy in providing a highly-accurate AT model while reducing its trainable parameters by roughly 99.93% (from 4.8 million to 3300 or 1400 depending on the input dataset). Interestingly, the analysis also shows that the compressed model provided by the developed AD tool is able to substantially outperform an RNN-based state-of-the-art AD algorithm when the two have roughly the same computational complexity (i.e., number of trainable parameters) as well as a conventional OCSVM AD strategy. The AD framework is then deployed in a real-world AD scenario where an infrastructure is in charge of monitoring the physical parameters of a bridge via distributed IoT sensors placed on it. Under this scenario, the model produced by the AD strategy after applying the knowledge distillation tool is shown to closely match the performances of the original uncompressed model while only marginally increasing the number of false positives.

The proposed tiny AD tool has been shown to be particularly suitable for dealing with complex and heterogeneous time-series data revealing its potential to be applied to real-world IoT setups. Nevertheless, the developed framework could be further optimized to take into account other constraints, such as latency and reliability, that are likely to be required when adapting the proposed solution to diverse scenarios, which may range from everyday applications to industrial IoT services. In particular, the renovated self-attention mechanism of AT could be modified to introduce sparse computations, allowing to further scale down the inference time. Besides, the optimization of the training pipelines for porting the distillation tool into physical devices is expected to bridge the gap between research and practice. Finally, it could be interesting to explore the integration of edge computing paradigms, including Federated Learning (FL) strategies, to improve the privacy of the proposed tiny AD framework.

REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Inform.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.
[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
[3] C. A. Tokognon, B. Gao, G. Y. Tian, and Y. Yan, "Structural health monitoring framework based on Internet of Things: A survey," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 619–635, Jun. 2017.
[4] C. Alippi and M. Roveri, "The (not) far-away path to smart cyber-physical systems: An information-centric framework," *Computer*, vol. 50, no. 4, pp. 38–47, Apr. 2017.
[5] S. S. Musa, M. Zennaro, M. Libsie, and E. Pietrosemoli, "Convergence of information-centric networks and edge intelligence for IoV: Challenges and future directions," *Future Internet*, vol. 14, no. 7, 2022, Art. 192.
[6] L. Barbieri, S. Savazzi, M. Brambilla, and M. Nicoli, "Decentralized federated learning for extended sensing in 6G connected vehicles," *Veh. Commun.*, vol. 33, 2022, Art. no. 100396.
[7] H. Xie, Z. Yan, Z. Yao, and M. Atiquzzaman, "Data collection for security measurement in wireless sensor networks: A survey," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2205–2224, Apr. 2019.
[8] A. A. Cook, G. Młsłrlł, and Z. Fan, "Anomaly detection for IoT time-series data: A survey," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6481–6494, Jul. 2020.
[9] C. Alippi, S. Ntalampiras, and M. Roveri, "Model-free fault detection and isolation in large-scale cyber-physical systems," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 1, pp. 61–71, Feb. 2017.
[10] C. Alippi, G. Boracchi, and M. Roveri, "Hierarchical change-detection tests," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 2, pp. 246–258, Feb. 2017.

[11] F. Cauteruccio et al., "A framework for anomaly detection and classification in multiple IoT scenarios," *Future Gener. Comput. Syst.*, vol. 114, pp. 322–335, 2021.

[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[13] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Inf. Sci.*, vol. 177, no. 18, pp. 3799–3821, 2007.

[14] V. A. Sotiris, P. W. Tse, and M. G. Pecht, "Anomaly detection through a Bayesian support vector machine," *IEEE Trans. Rel.*, vol. 59, no. 2, pp. 277–286, Jun. 2010.

[15] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proc. ACM SIGKDD Workshop Outlier Detection Description*, 2013, pp. 8–15.

[16] A. P. Muniyandi, R. Rajeswari, and R. Rajaram, "Network anomaly detection by cascading K-means clustering and C4.5 decision tree algorithm," *Procedia Eng.*, vol. 30, pp. 174–182, 2012.

[17] S. Thaseen and C. A. Kumar, "An analysis of supervised tree based classifiers for intrusion detection system," in *Proc. IEEE Int. Conf. Pattern Recognit., Inform. Mobile Eng.*, 2013, pp. 294–299.

[18] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern., Part C. (Appl. Rev.)*, vol. 38, no. 5, pp. 649–659, Sep. 2008.

[19] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. IEEE 8th Int. Conf. Data Mining*, 2008, pp. 413–422.

[20] D. Xu, Y. Wang, Y. Meng, and Z. Zhang, "An improved data anomaly detection method based on isolation forest," in *Proc. IEEE 10th Int. Symp. Comput. Intell. Des.*, 2017, pp. 287–291.

[21] O. Alghushairy, R. Alsini, T. Soule, and X. Ma, "A review of local outlier factor algorithms for outlier detection in Big Data streams," *Big Data Cogn. Comput.*, vol. 5, no. 1, 2021, Art. no. 1.

[22] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," in *Proc. Conf. Res. Adaptive Convergent Syst.*, 2019, pp. 161–168.

[23] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using K-nearest neighbour graph," in *Proc. IEEE 17th Int. Conf. Pattern Recognit.*, 2004, pp. 430–433.

[24] Y. Djenouri, A. Belhadi, J. C.-W. Lin, and A. Cano, "Adapted K-nearest neighbors for detecting anomalies on spatio–temporal traffic flow," *IEEE Access*, vol. 7, pp. 10015–10027, 2019.

[25] S. Omar, A. M. Ngadi, and H. H. Jebur, "Machine learning techniques for anomaly detection: An overview," *Int. J. Comput. Appl.*, vol. 79, pp. 33–41, 2013.

[26] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine learning for anomaly detection: A systematic review," *IEEE Access*, vol. 9, pp. 78658–78700, 2021.

[27] D. Kwon et al., "An empirical study on network anomaly detection using convolutional neural networks," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 1595–1598.

[28] Y. Su et al., "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2828–2837.

[29] Y. Wu, H.-N. Dai, and H. Tang, "Graph neural networks for anomaly detection in Industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9214–9231, Jun. 2022.

[30] Y. Feng, J. Chen, Z. Liu, H. Lv, and J. Wang, "Full graph autoencoder for one-class group anomaly detection of IIoT system," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21886–21898, Nov. 2022.

[31] Q. Wen et al., "Transformers in time series: A survey," 2022, in *Proc. 32th Int. Joint Conf. Artif. Intell.*, Aug. 2023, pp. 6778–6786, doi: 10.24963/ijcai.2023/759.

[32] S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep transformer networks for anomaly detection in multivariate time series data," in *Proc. VLDB*, vol. 15, no. 6, 2022, pp. 1201–1214.

[33] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, Mar. 2021.

[34] Y. Liu, Y. Zhou, K. Yang, and X. Wang, "Unsupervised deep learning for IoT time series," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14285–14306, Aug. 2023.

[35] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1544–1551, Jul. 2018.

[36] Z. Li et al., "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 3220–3230.

[37] L. Shen, Z. Li, and J. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 13016–13026.

[38] B. Zhou et al., "BeatGAN: Anomalous rhythm detection using adversarially generated time series," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4433–4439.

[39] D. Li et al., "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *Proc. Artif. Neural Netw. Mach. Learn.*, 2019, pp. 703–716.

[40] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning graph structures with transformer for multivariate time-series anomaly detection in IoT," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9179–9189, Jun. 2022.

[41] S. Zhang et al., "CAT: Beyond efficient transformer for content-aware anomaly detection in event sequences," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 4541–4550.

[42] X. Cai et al., "ITran: A novel transformer-based approach for industrial anomaly detection and localization," *Eng. Appl. Artif. Intell.*, vol. 125, 2023, Art. no. 106677.

[43] Y. Li, X. Peng, J. Zhang, Z. Li, and M. Wen, "DCT-GAN: Dilated convolutional transformer-based GAN for time series anomaly detection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3632–3644, Apr. 2023.

[44] C. Ding, J. Zhao, and S. Sun, "Concept drift adaptation for time series anomaly detection via transformer," *Neural Process. Lett.*, vol. 55, pp. 2081–2101, 2023.

[45] H. Zhang, Y. Xia, T. Yan, and G. Liu, "Unsupervised anomaly detection in multivariate time series through transformer-based variational autoencoder," in *Proc. IEEE 33rd Chin. Control Decis. Conf.*, 2021, pp. 281–286.

[46] X. Wang et al., "Variational transformer-based anomaly detection approach for multivariate time series," *Measurement*, vol. 191, 2022, Art. no. 110791.

[47] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *Proc. Int. Conf. Learn. Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=LzQQ89U1qm_

[48] K. Han et al., "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 87–110, Jan. 2023.

[49] M. Gupta and P. Agrawal, "Compression of deep learning models for text: A survey," *ACM Trans. Knowl. Discov. Data*, vol. 16, no. 4, pp. 1–55, Jan. 2022.

[50] Z. Yang et al., "Searching for low-bit weights in quantized neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 4091–4102.

[51] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/2c601ad9d2ff9bc8b282670cdd54f69f-Abstract.html

[52] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[53] X. Liu, P. He, W. Chen, and J. Gao, "Improving multi-task deep neural networks via knowledge distillation for natural language understanding," 2019, *arXiv:1904.09482*.

[54] R. Tang et al., "Distilling task-specific knowledge from BERT into simple neural networks," Mar. 2019, *arXiv:1903.12136*.

[55] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for BERT model compression," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, K. Inui, J. Jiang, V. Ng, and X. Wan, Nov. 2019, pp. 4323–4332. [Online]. Available: https://aclanthology.org/D19-1441

[56] J. Ko et al., "Revisiting intermediate layer distillation for compressing language models: An overfitting perspective," in *Proc. Findings Assoc. Comput. Linguistics*, 2023, pp. 158–175.

[57] Z. Lan et al., "ALBERT: A lite BERT for self-supervised learning of language representations," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1eA7AEtvS

[58] M. Dehghani et al., "Universal transformers," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HyzdRiR9Y7

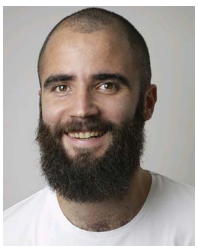
[59] X. Ma et al., "A tensorized transformer for language modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/dc960c46c38bd16e953d97cdeefdbc68-Abstract.html
[60] O. Hrinchuk, V. Khrulkov, L. Mirvakhabova, E. Orlova, and I. Oseledets, "Tensorized Embedding Layers," *Findings Assoc. Comput. Linguistics: EMNLP*, pp. 4847–4860, Nov. 2020, doi: 10.18653/v1/2020.findings-emnlp.436.
[61] Y. Abadade et al., "A comprehensive survey on tinyML," *IEEE Access*, vol. 11, pp. 96892–96922, 2023.
[62] T. S. Ajani, A. L. Imoize, and A. A. Atayero, "An overview of machine learning within embedded and mobile devices–optimizations and applications," *Sensors*, vol. 21, no. 13, 2021, Art. no. 4412.
[63] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[64] A.-K. Seghouane and S.-I. Amari, "The AIC criterion and symmetrizing the Kullback–Leibler divergence," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 97–106, Jan. 2007.
[65] C. M. Bishop, *Pattern Recognition and Machine Learning*, (Information Science and Statistics Series). Berlin, Germany: Springer, 2006.
[66] K. Hundman et al., "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 387–395.
[67] A. Abdulaal, Z. Liu, and T. Lancewicki, "Practical approach to asynchronous multivariate time series anomaly detection and localization," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 2485–2494.
[68] H. Xu et al., "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications," in *Proc. World Wide Web Conf.*, 2018, pp. 187–196.
[69] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
[70] D. M. Tax and R. P. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, pp. 45–66, 2004.

LUCA BARBIERI (Member, IEEE) received the B.Sc. and M.Sc. (*cum laude*) degrees in telecommunication engineering and the Ph.D. degree (*cum laude*) in information technology from Politecnico di Milano, Milan, Italy, in 2017, 2019, and 2023, respectively. He was a Visiting Researcher with the King's Communications, Learning & Information Processing Lab, King's College London, London, U.K., in 2022. He is currently a research Assistant with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. His research interests include machine learning, federated learning, localization methods for vehicular, and industrial networks.

MATTIA BRAMBILLA (Member, IEEE) received the B.Sc. and M.Sc. degrees in telecommunication engineering and the Ph.D. degree (*cum laude*) in information technology from the Politecnico di Milano, Milan, Italy, in 2015, 2017, and 2021, respectively. In 2019, he was a Visiting Researcher with the NATO Centre for Maritime Research and Experimentation, La Spezia, Italy. In 2021, he joined the Faculty of Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano as a Research Fellow. His research interests include signal processing, statistical learning, data fusion for cooperative localization and communication in vehicular, and IoT networks. He was the recipient of the Best Student Paper Award at the 2018 IEEE Statistical Signal Processing Workshop.

MARIO STEFANUTTI with more than 30 years of experience with complex IT infrastructures and systems, gained on major companies like Telecom, Poste, Autostrade, Alitalia, Ericsson, Telespazio, Iridium, and Reply, where he holds roles ranging from Solution Architect, Technical Project Leader, Project Manager, Programmer. He was with Sensoworks at the time this work was carried out. He is currently with GreenVulcano where he is in charge of the R&D Lab and the Technical Services at Corporate Level (90+ people). He is responsible for the strategic definition of technical developments and roadmap of all GreenVulcano products. His research interests include machine learning and IoT.

CIRO ROMANO received the graduation degree in electronic engineering. He is CTO at Sensoworks, shapes the company's tech trajectory, leveraging more than 25 years of ICT expertise. Pioneering IoT, automotive, energy, and multiutilities projects, he's led also research projects spacing in semantic web, federated machine learning, and quantum technologies. Notably, Water4All innovates leak detection prediction in water networks providing a DSS (Decision Support System) to utilies helping them in effectively maintenance activities. He is steering several projects in critical infrastructure monitoring and smart parking solution. He is a luminary in tech, his contributions fuel research and tech evolution, making Sensoworks a force in innovation.



NICCOLÒ DE CARLO is CEO of Sensoworks, embodies a tech maven's journey. He has fifteen years experience in international projects. Co-founding Sensoworks in the early 2020's, he foresight propelled the company to the forefront of IoT innovation. Leading with strategic acumen, he forged pivotal partnerships that solidified Sensoworks' standing in the competitive tech landscape. He instilled a commitment to ethical tech practices, making Sensoworks a beacon of responsible innovation. As Sensoworks continues to thrive under his leadership, he stands as a trailblazer, showcasing the transformative potential of IoT under his guidance.

MANUEL ROVERI (Senior Member, IEEE) received the M.S. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2003, and the Dr.Eng. degree in computer science engineering, in 2003, and the Ph.D. degree in computer engineering from the Politecnico di Milano, Milan, Italy, in 2007. He is currently a Full Professor with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. He holds one patent and has authored or coauthored more than 110 articles in international journals and conference proceedings. His research interests include embedded and edge AI, intelligent embedded, cyber-physical systems, learning in presence of concept-drift, privacy-preserving machine, and deep learning. He was the recipient of the 2018 IEEE Computational Intelligence Magazine Outstanding Paper Award, 2016 IEEE Computational Intelligence Society Outstanding Transactions on Neural Networks and Learning Systems Paper Award Best Regular Paper Award at the INNS Conference on Big Data in 2016, and 2021 Outstanding Associate Editor for IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE. He was the Chair and a Member of several IEEE committees and subcommittees, and an Associated Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, *IEEE Computational Intelligence Magazine*, and *Neural Networks* (Elsevier).

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement