Adversarial Data Augmentation for HMM-based Anomaly Detection

Alberto Castellini, Francesco Masillo*, Davide Azzalini*, Francesco Amigoni, and Alessandro Farinelli

Abstract—In this work, we concentrate on the detection of anomalous behaviors in systems operating in the physical world and for which it is usually not possible to have a complete set of all possible anomalies in advance. We present a data augmentation and retraining approach based on adversarial learning for improving anomaly detection. In particular, we first define a method for generating adversarial examples for anomaly detectors based on Hidden Markov Models (HMMs). Then, we present a data augmentation and retraining technique that uses these adversarial examples to improve anomaly detection performance. Finally, we evaluate our adversarial data augmentation and retraining approach on four datasets showing that it achieves a statistically significant performance improvement and enhances the robustness to adversarial attacks. Key differences from the state-of-the-art on adversarial data augmentation are the focus on multivariate time series (as opposed to images), the context of one-class classification (in contrast to standard multi-class classification), and the use of HMMs (in contrast to neural networks).

Index Terms—Data Augmentation, Anomaly Detection, Adversarial Learning, HMMs, Robotic Systems, Cyber-Physical Systems

1 Introduction

Newly conceived intelligent systems that operate in the physical world are required to reliably work over long periods of time under changing and unpredictable environmental conditions. In robotic applications, for instance, this is referred to as *long-term autonomy* (LTA) [1]. In this context, anomaly detection plays a paramount role because it allows to identify as soon as possible situations that diverge from the desired (i.e., safe/optimal) ones. A key property of anomalies in autonomous robotic systems and Cyber-Physical Systems (CPSs) is that they are often related to (dynamical) *behaviours* of the whole system, rather than to specific components (e.g., a specific sensor or actuator). Hence, they can be detected from observations acquired by multiple sensors and spanning some time intervals, rather than from single instantaneous observations. This requires the analysis of *multivariate time series* to identify behavioural anomalies. The literature about detectors for anomalous (dynamical) behaviours is still in its early stages and further efforts are needed to meet the requirements of autonomous robotic systems and CPSs.

In this work we focus on *Hidden Markov Models* (HMMs) [2] as they represent a powerful and widely used mathematical model for learning robot behaviour [3] and for encoding noisy time series [4]. Moreover, HMMs are known to be robust to spatio-temporal variations and can successfully model intelligent systems behaviours in several LTA contexts, where similar sequences of actions (tasks) are typically repeated multiple times [5], [6]. In particular, in [7] an online approach has been proposed for detecting anomalous behaviours of robot systems involved in complex LTA scenarios. The methodology uses HMMs to model the nominal (expected) behaviour of a robot and the Hellinger distance [8] to evaluate the dissimilarity between the probability distribution of subsequences of observations (i.e., multivariate sensor time series) in a sliding window and the emission probability of the related HMM hidden states. The advantage of using such a distance measure instead of standard measures (e.g., the likelihood of observation subsequences) is twofold: first, the Hellinger distance is bounded and thus lends itself to simpler interpretation and thresholding; second, it is less noisy, hence more informative and discriminative [7]. For simplicity, in the following we refer to the online algorithm proposed in [7] as *HMM-Hellinger-based Anomaly Detector* (*HHAD*).

A key issue for behavioural anomaly detection in robotics and CPSs is the lack (or paucity) of anomalous examples and the noise that characterizes data in these contexts. To address this issue, in this paper we propose an *adversarial data augmentation and retraining approach for HHAD* (called *HHAD-AUG*). Following the recent promising trends of adversarial example generation and adversarial attack generation for machine learning models [9], we base our data augmentation method on adversarial examples, namely, perturbed time series [10], [11], [12], [13], that have the advantage of not requiring any prior knowledge about the application domain and data conformation. In particular, we generate adversarial examples for nominal points, the only knowledge available at training time, using two algorithms. The first algorithm (called *H-ADV*) uses a loss function[1] based on the Hellinger distance between the observed and the expected data distributions (i.e., model parameters) and the second (called *L-ADV*) uses a loss

- *A. Castellini, F. Masillo, and A. Farinelli are with the Department of Computer Science, University of Verona, Italy. E-mail: alberto.castellini@univr.it, francesco.masillo@studenti.univr.it, alessandro.farinelli@univr.it*
- *D. Azzalini and F. Amigoni are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy. E-mail: davide.azzalini@polimi.it, francesco.amigoni@polimi.it*

* *F. Masillo and D. Azzalini contributed equally to this work.*

1. Throughout the manuscript we use term "loss" as a synonym of "anomaly score".

function based on sample likelihood. We mathematically derive the gradients of both loss functions and present a procedure to use them to augment the original dataset. We then use adversarial data augmentation to improve the detection performance of HHAD (in terms of F1-score) when limited amounts of training data is available, but we show that the methodology we propose achieves performance improvements also when models are trained on large datasets. Notice that we focus on anomaly detection as a one-class classification problem, in which examples of the anomalous class are not considered in the learning phase, hence the only way to perform data augmentation is to generate new nominal examples.

In contrast to the literature on generating adversarial examples, we focus on time series rather than images and we consider HMMs rather than deep neural networks. Recently, an approach for performing adversarial attacks on (univariate) time series classifiers was proposed [12], but it is based on neural network classifiers, hence it requires large datasets. Our method employs the definition of adversarial attacks for time series used in [12] but it is oriented to HMM-based anomaly detectors, which achieve strong results on multivariate time series also on small datasets [7]. Moreover, the proposed method focuses on anomaly detection as one-class classification, a key difference with respect to [12].

We evaluate our data augmentation and retraining approach on four public datasets, three known real-world benchmarks for anomaly detection in robotic systems and CPSs, i.e., Tennessee Eastman [14], SWaT [15], and ALFA [16], and one created by the authors using aquatic drones developed in a EU H2020 project, i.e., INTCATCH [17]. The experimental evaluation of the proposed approach shows that *(i)* H-ADV and L-ADV can generate meaningful adversarial examples for HHAD; *(ii)* HHAD-AUG can employ these new examples to significantly improve the performance of HHAD; *(iii)* using examples from both H-ADV and L-ADV outperforms state-of-the-art augmentation methods; *(iv)* using examples generated by H-ADV is better than using examples generated by L-ADV (i.e., the former wins on three datasets and ties on one), hence we consider H-ADV as the best method to generate adversarial examples for data augmentation of HHAD; *(vi)* the low computational complexity of H-ADV and the high parallelizability of L-ADV allow for a fast data augmentation and retraining of HHAD. The generated examples are guaranteed to have small distance from the original examples, according to the definition of adversarial examples for time series [12].

In summary, the main contributions of this work to the state-of-the-art are the following:

- we propose an algorithm able to generate adversarial examples for a one-class anomaly detector based on HMMs and working with multivariate time series (Section 4.1);
- we propose an algorithm for data augmentation based on adversarial examples which improves the performance of the anomaly detector (Section 4.2);
- we evaluate, with positive results, both adversarial generation and data augmentation on four datasets of multivariate sensor signals acquired from autonomous robots and industrial CPSs (Section 5).

2 Related Work

Three research topics are mainly related to our work: data augmentation, adversarial example generation, and anomaly detection in autonomous robots and CPSs.

Data Augmentation. Data augmentation techniques are used to increase the amount of available data by adding slightly modified copies of already existing data with the aim of regularizing and reducing overfitting when training a machine learning model. In [18], data augmentation methods for images are surveyed. The main goal of data augmentation, both for images and for other types of data, is to prevent class imbalance and model overfitting due to data limitations, by adding synthetic examples to available datasets [19].

Time series data augmentation is not an established practice. The majority of the state-of-the-art approaches for time series do not use data augmentation, and the first surveys on these techniques have been presented only recently [20], [21]. Methods related to adversarial training are still not considered in the literature related to time series data augmentation. Most data augmentation techniques for time series [20], [21], [22], [23], [24] are instead based on random transformations, such as, addition of random noise, slicing, cropping, scaling, random warping in the time dimension, and frequency warping.

Adversarial data augmentation, also called adversarial training [25], is the process of augmenting a dataset using adversarial examples (a.k.a. adversarial attacks) to achieve two main goals, namely, making classifiers more robust to adversarial attacks and reducing their test error on clean inputs, i.e., improving the accuracy of the classifier on the test set. This practice has been very recently applied to image classifiers [18], where adversarial data augmentation has been used to improve generalization to unseen domains. In [26], an iterative procedure is proposed to augment a dataset with examples from a fictitious target domain that is hard under the current model. The methodology is inspired by recent developments in distributionally robust optimization and adversarial training. For instance, in [27], a good performance under adversarial input perturbations is guaranteed by considering in the learning optimization problem a Lagrangian penalty for perturbing the data distribution in a Wasserstein (a distance over probability distribution) ball. In [28], it is proposed a novel regularization term for adversarial data augmentation in deep neural networks for image classification. The methodology extends [26] and reduces to it when the maximum-entropy term is discarded. These approaches show that adversarial training can effectively help when searching for augmentations [18].

Our method applies these principles to a specific one-class classification problem (i.e., anomaly detection), for time series data instead of images. We apply, in particular, adversarial data augmentation to the HMM-based detector presented in [7]. The adversarial-based strategy that we use to generate synthetic examples makes our methodology deeply different from traditional time series augmentation methods, since we do not need prior knowledge about the application domain to generate data transformations. Furthermore, our strategy allows to improve detection performance on both the original test set and the adversarial

attacks generated from the test set, hence it improves also the robustness to adversarial attacks.

Adversarial Example Generation. Adversarial examples for classification models are investigated in [29] and the analysis is specialized to neural networks for image classification in [30], where authors notice that "imperceptible non-random perturbations applied to a test image can change the network prediction". An optimization procedure called box-constrained Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) is proposed in [30] to compute adversarial perturbations of images given network parameters. To overcome some time-complexity issues of this method, another approach called Fast Gradient Sign Method (FGSM) has been proposed [9]. It produces sub-optimal adversarial examples, in terms of distance from the original example, but being very fast it has quickly become popular and has inspired other approaches. Two methods that produce smaller perturbations than FGSM while ensuring good efficiency are Deepfool [31] and the Carlini-Wagner method [32], that use iterative procedures based on local linearization of the classifier function. A theoretical framework for analyzing the robustness of classifiers to adversarial perturbations is proposed in [33]. Adversarial training is used as a regularization method for supervised and semi-supervised learning of neural networks in [34]. A method for generating universal adversarial perturbations is presented in [35]. Surveys on adversarial attack methods are proposed in [36], [37].

Methodologies for generating adversarial attacks on time series are proposed in [10], [11], [12], [13]. A strategy based on Adversarial Transformation Networks (ATNs) is used, in particular, in [11], [12] to generate adversarial attacks on a target classifier of time series via a student model trained using standard model distillation techniques. The target classifier can be a fully convolutional neural network or a 1-nearest neighbor classifier with Dynamic Time Warping. The ATN takes in input a time series and its gradient with respect to the softmax-scaled logits of the target class predicted by the attacked classifier, and returns a perturbed time series that represents a possible adversarial example. If the classifier being attacked is unknown (i.e., black-box attack) or it is 1-nearest neighbor with Dynamic Time Warping (i.e., white-box attack on a non derivable classifier), then the gradient cannot be computed. In these cases, the attack is performed on the student model which is a neural network that imitates the classifier and it is derivable. In [13], ATNs are extended with autoencoders to attack multivariate time series classification models.

In our work, we consider the same definition of adversarial attacks used in [11], [12], [13] but the approach we propose has a different objective and uses a different methodology. We propose a data augmentation technique based on adversarial examples for improving HMM-based anomaly detectors that work on multivariate time series, while [11], [12], [13] propose new methodologies for generating adversarial attacks on time series. We derive the gradient of the specific loss function of the anomaly detector, hence our method generates adversarial examples directly on the detector, not on a neural network that approximates the detector. Also the target model is different: in our case it is an anomaly detector trained using only nominal examples, while in the literature examples of all classes are considered to be available both in the training phase and in the adversarial attack generation phase. The generation of adversarial attacks has been studied also in the context of Natural Language Processing [38], [39], where classification models are sometimes similar to those used for time series classification. However, to the best of our knowledge all the methodologies proposed so far work with deep neural network models.

Anomaly Detection in Autonomous Robots and CPS. Anomaly detection approaches for robotic systems and CPSs can be divided into three main categories: model-based, knowledge-based, and data-driven. Online data-driven methods are getting more and more popular for autonomous robots and modern industrial CPSs. Only few methods deal with anomalies in system *behaviours*. Chandola et al. [40] refer to this type of anomalies as contextual faults since they are originated by observations in specific contexts. In other words, an observation can be considered anomalous in a specific context but nominal in another context [41], [42]. Recently, an approach has been proposed for detecting anomalous process behaviours generated by stealthy-attacks in CPSs and, in particular, in industrial control systems [43]. Other approaches for the same application domain have been recently presented [44], [45], some of them based on deep learning [46], [47], [48], [49], [50], [51], [52]. The anomaly detector that we aim to improve by means of data augmentation, called HHAD in this paper, considers contextual anomalies (namely, observations that are infrequent in specific contexts) but differs from other approaches in the literature because it represents contexts as maximally frequent HMM states in a time window instead of as sets of past observations [7]. The approaches that most resemble HHAD are [53] and [54], in which HMMs are trained using multimodal sensory signals for detecting anomalies in assistive robots. At run time, the trained HMMs provide likelihood scores for data inside a window, which are compared to an adaptive detection threshold to identify putative anomalies. HHAD substitutes the likelihood estimation used in these approaches with the Hellinger distance which is a more informative and interpretable measure. While other probabilistic distances [55] have been recently proposed for anomaly and change detection, we focus on the Hellinger distance since it is part of the original HHAD approach [7] and, as discussed there, it is bounded and little noisy.

3 Background and Notation

In this section, we informally define the problem addressed in this paper, the HMMs and Hellinger distance, and we describe the HHAD algorithm. The main strategies for adversarial example generation and data augmentation are finally presented.

3.1 Problem Definition

Given the online one-class HMM-based anomaly detector HHAD, introduced in [7], and trained on a dataset of nominal multivariate time series, our goal is to improve its

performance by augmenting the training set with adversarial examples. Moreover, we aim at improving the robustness of the detector to adversarial attacks.

3.2 Hidden Markov Models

We use HMMs [2] as a probabilistic model for the system that generated a given d-dimensional time series $\boldsymbol{X} = \langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \rangle$ where $\boldsymbol{x}_t = [x_t^1, \ldots, x_t^d], t \in \{1, \ldots, n\}$, is the multivariate observation $\boldsymbol{x}_t$ at time t. An HMM is a statistical model in which the system being modeled is assumed to be a Markov process with K hidden states. The mathematical notation $\lambda = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}\}$ is used to represent an HMM, where $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^K$ is the set of initial state probabilities, $\boldsymbol{A} = \{a_{ij}\}_{i,j=1}^K$ is the set of state transition probabilities (i.e., a_{ij} is the probability to move from state s_i to state s_j), and $\boldsymbol{B} = \{b_i(\boldsymbol{x}_t)\}_{i=1}^K$ is the set of the probability distributions over observations in each state (emission probabilities). In our setting, we assume a multivariate Gaussian distribution for the emission probabilities, which means that $\boldsymbol{B} = \{\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_{i=1}^K$, where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean and the covariance matrix for state s_i, respectively. To find the parameters of the HMM λ that maximize the fit (likelihood) to an observed (sub)sequence $\boldsymbol{X}$ we use the *Baum-Welch* algorithm, while to compute the optimal HMM state sequence (known as Viterbi path) that best explains a given observed (sub)sequence $\boldsymbol{X}$ we use the *Viterbi* algorithm. The number of hidden states and the covariance type of an HMM can be found by minimizing the Bayesian Information Criterion (BIC), which finds a trade-off between maximizing the likelihood of the training data with respect to. the model and minimizing the number of parameters required (i.e., the number of hidden states) [56].

3.3 Hellinger Distance

The Hellinger distance [8] is a $[0, 1]$-bounded function that quantifies the similarity between two probability density functions $p_1(\boldsymbol{x})$ and $p_2(\boldsymbol{x})$. In the case of two multivariate Gaussian distributions $p_1(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})$ and $p_2(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$, the Hellinger distance can be computed in closed form as:

$$H^2(p_1, p_2) = 1 - \frac{det(\boldsymbol{\Sigma_1})^{1/4} det(\boldsymbol{\Sigma_2})^{1/4}}{det\left(\frac{\boldsymbol{\Sigma_1}+\boldsymbol{\Sigma_2}}{2}\right)^{1/2}} \cdot exp\left\{-\frac{1}{8}(\boldsymbol{\mu_1}-\boldsymbol{\mu_2})^T \left(\frac{\boldsymbol{\Sigma_1}+\boldsymbol{\Sigma_2}}{2}\right)^{-1} (\boldsymbol{\mu_1}-\boldsymbol{\mu_2})\right\}. \quad (1)$$

3.4 The Online Anomaly Detector HHAD

The nominal behaviour of the system whose anomalies should be detected is modeled as an HMM λ^N that is trained using the Baum-Welch algorithm from a sequence $\boldsymbol{X}$ of observations collected during the nominal functioning of the system. The number of hidden states and the covariance type are selected by minimizing the BIC. Online anomaly detection at time step t is performed by means of a sliding window $\boldsymbol{W}_t = \langle \boldsymbol{x}_{t-w+1}, \ldots, \boldsymbol{x}_t \rangle$ of length w (columns) and with observations that contain d variables (rows), hence $\boldsymbol{W}_t \in \mathbb{R}^{d \times w}$. For each example $\boldsymbol{W}_t$, an anomaly score is computed and, when the score exceeds a predefined threshold τ, the behaviour is considered anomalous. The score is the Hellinger distance between the estimated distribution of the observations corresponding to the state $\hat{s}_t$ occurring most frequently in the Viterbi path $S_t = \langle s_{t-w+1}, \ldots, s_t \rangle$ of window $\boldsymbol{W}_t$ and the emission probability of state $\hat{s}_t$ of λ^N. See details in [7].

Algorithm 1: Anomaly detection (HHAD) [7]

Input: $\boldsymbol{W}_t \leftarrow \langle \boldsymbol{x}_{t-w+1}, \ldots, \boldsymbol{x}_t \rangle,\ t \in \{w, \ldots, n\}$
$K \leftarrow$ # hidden states, $w \leftarrow$ window size,
$\lambda^N \leftarrow$ Baum-Welch($\boldsymbol{X}$, K), $\tau \leftarrow$ threshold
Output: $a_t \in \{0, 1\}$
1 $\boldsymbol{S}_t \leftarrow$ Viterbi($\lambda^N, \boldsymbol{W}_t$)
2 $\hat{s}_t \leftarrow$ most frequent state in $\boldsymbol{S}_t$
3 $\boldsymbol{M} \leftarrow \{\boldsymbol{x}_j \in \boldsymbol{W}_t \ :\ s_j = \hat{s}_t\}$
4 $\boldsymbol{\mu} \leftarrow E[\boldsymbol{M}]$
5 $\boldsymbol{\Sigma} \leftarrow E[(\boldsymbol{M} - \boldsymbol{\mu})(\boldsymbol{M} - \boldsymbol{\mu})^T]$
6 **if** $H^2(b^N_{\hat{s}_t}, \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) > \tau$ **then**
7 $\quad a_t = 1$
8 **else**
9 $\quad a_t = 0$
10 **end**
11 **return** a_t

HHAD, formalized in Algorithm 1, receives the current window of multivariate data $\boldsymbol{W}_t$, the nominal HMM λ^N with its number of hidden states K, the window size w and a threshold $\tau \in \mathbb{R}^+$. The returned value is a label $a_t \in \{0, 1\}$ representing the class of $\boldsymbol{W}_t$, namely, $a_t = 0$ for nominal examples or $a_t = 1$ for anomalous examples. Algorithm 1 is run for each window $\boldsymbol{W}_t$ that has to be classified. It first computes the Viterbi path of the multivariate time series in $\boldsymbol{W}_t$ (line 1). For the state $\hat{s}_t$ occurring most frequently in the Viterbi path (line 2) a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is fit through maximum likelihood to the corresponding data observed in the window (lines 3-5). Then the Hellinger distance is computed (using Equation 1) between $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and the emission probability of state $\hat{s}_t$ in λ^N (line 6). If the distance is larger than τ, then an anomaly is reported (line 7). Otherwise, label 0 is returned (line 9). Usually this algorithm is run with a sliding window $\boldsymbol{W}_t$ which shifts of one time instant at a time.

3.5 Adversarial Example Generation

A formal definition of (minimal)[2] adversarial *perturbation* for a q-dimensional object $\boldsymbol{\xi}$ is provided in [31] as the minimal perturbation $\boldsymbol{\eta}$ that is sufficient to change the label $\hat{y} = f(\boldsymbol{\xi})$ estimated by a classifier $f : \mathbb{R}^q \rightarrow \{1, \ldots, k\}, q \in \mathbb{N}$ for example $\boldsymbol{\xi}$. The L_p-distance of such a minimal perturbation is therefore:

$$\Delta(\boldsymbol{\xi}; f) \doteq \min_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_p \quad \text{subject to } f(\boldsymbol{\xi} + \boldsymbol{\eta}) \neq f(\boldsymbol{\xi}). \quad (2)$$

This definition requires a distance metric $L_p = \|\cdot\|_p$ to quantify the similarity of the adversarial example to the original one. The most used distances are L_0, i.e., the number of coordinates changed by the perturbation, L_2, the

2. In [31] this is called adversarial perturbation but we specify minimal because adversarial examples can also be related to non-minimal perturbations.

standard Euclidean distance between the original and the perturbed example, and L_∞, the maximum change among all coordinates [32]. Different distances L_p have different impacts on the classification of the perturbed example. The usage of a distance metric that reliably captures the ground truth similarity is fundamental to generate good adversarial examples[3]. In the context of time series data, the ground truth of an example cannot be provided by human perception, hence it must be known in advance to evaluate the perturbed examples. Adversarial examples are then defined as slight perturbations of original examples that produce a misclassification with respect to the ground truth. Since the ground truth is not available for all examples, we use the definition of [11], [12], [13], namely, the label predicted by the classifier is assumed to be the ground truth and adversarial examples are defined as examples whose predicted class label is different from the ground truth label.

Fast Gradient Sign Method (FGSM) [9] is the foundation for the approach we propose in this paper. It is *untargeted*, since it does not allow to specify the target label, and optimized for the L_∞ distance metric. This method does not guarantee to return the closest adversarial examples but it is faster than L-BFGS [30]. Given an example $\boldsymbol{\xi}$, FGSM searches for an example $\boldsymbol{\xi}' = \boldsymbol{\xi} + \epsilon \cdot sign(\nabla_{\boldsymbol{\xi}} loss_f(\boldsymbol{\xi}, y))$, where $\nabla_{\boldsymbol{\xi}}$ is the gradient function over variable $\boldsymbol{\xi}$. Given the parameters $\boldsymbol{\theta}$ of the classifier f, an example $\boldsymbol{\xi}$, its original class y in the training set, and the cost function used to train the classifier $loss_f$, an adversarial example is obtained by linearizing the loss function around the current value of $\boldsymbol{\theta}$ and moving (in $\mathbb{R}^q$) in the direction that maximises the loss of f. The time performance can be improved by efficiently computing the gradient using backpropagation. The adversarial example generation we propose in this paper (Section 4.1) starts from examples $\boldsymbol{\xi}$ that are time series $\boldsymbol{X}$ and generates new examples $\boldsymbol{\xi'}$ that are new time series $\boldsymbol{X'}$, with the goal of having $\boldsymbol{X'}$ classified differently from $\boldsymbol{X}$ by the HHAD classifier f.

3.6 Data Augmentation

The data augmentation problem consists in extending a dataset by adding new examples to improve the performance of a model trained with that dataset. Standard approaches for time series data augmentation have been discussed in Section 2. Since usually new examples are generated from the original ones, the problem can be formalized as the generation, from original examples $\boldsymbol{\xi}$, of perturbed examples $\boldsymbol{\xi} + \boldsymbol{\eta}$ that maximize the performance of model f. In the supplementary material (Section 3.1) we describe four methods for time series data augmentation [21] that we use as baselines to evaluate the performance of our method. They are, Random Data Augmentation (R-AUG), Drift Data Augmentation (D-AUG), Gaussian Data Augmentation (G-AUG), and Synthetic Minority Over-sampling Technique (SMOTE) [19] (S-AUG). We select these methods as baselines because they are approaches that do not require specific domain knowledge, similarly to our adversarial example generation method.

3. In [32], authors refer to human perceptual similarity but this concept can be extended to the similarity in the ground truth when the compared examples are not images.

4 Proposed Methods

We present two approaches for adversarial example generation on HHAD. The first uses a loss function based on the Hellinger distance; the second a loss function based on the likelihood. Then, we introduce our adversarial data augmentation methodology.

4.1 Adversarial Example Generation: H-ADV and L-ADV

Our approach for adversarial example generation is outlined in Figure 1 and formalized in Algorithm 2 (Hellinger-based loss) and Algorithm 3 (Loglikelihood-based loss). As shown in Figure 1, the main idea is to take an example[4] $\boldsymbol{X}$, i.e., a slice of the multivariate time series $\boldsymbol{X}$, and to pass it to the adversarial example generator (Algorithms 2 or 3, called H-ADV and L-ADV, respectively, in the following) which uses some elements of HHAD (i.e., λ^N and τ) to generate the perturbation $\boldsymbol{X} + \boldsymbol{\eta}$. This perturbation is called adversarial if it actually changes the class of $\boldsymbol{X}$, as done in the literature [11], [12], [13].

H-ADV. Algorithm 2 describes H-ADV, which uses a loss function based on the Hellinger distance. It receives five inputs, namely, *i)* a nominal example[5], i.e., a slice of a multivariate time series $\boldsymbol{X} \in \mathbb{R}^{d \times w}$, where w is the length of the considered window and $\boldsymbol{x}_t$ a d-dimensional observation at time t, *ii)* the nominal HMM λ^N used by HHAD, *iii)* a parameter $\epsilon \in \mathbb{R}^+$ representing the maximum perturbation size for each element of $\boldsymbol{X}$ (which should be kept as small as possible), *iv)* the number of steps $c \in \mathbb{N}$ in which the interval $[0, \epsilon]$ is divided, *v)* the threshold τ for the Hellinger distance used by HHAD. The algorithm returns an example $\boldsymbol{X'} = \boldsymbol{X} + \boldsymbol{\eta} \in \mathbb{R}^{d \times w}$ close to $\boldsymbol{X}$ (i.e., inside the hypercube with side 2ϵ centered in $\boldsymbol{X}$) and perturbed in a direction which facilitates the change of class, i.e., $f(\boldsymbol{X'}) \neq f(\boldsymbol{X})$. Remember that HHAD is a function $f : \mathbb{R}^{d \times w} \rightarrow \{0, 1\}$, where 0 is the class for nominal behaviours and 1 that for anomalous behaviours (see Algorithm 1). We do not consider adversarial examples generated from anomalous points since our goal is to augment the training set, which contains only nominal data.

Given the example $\boldsymbol{X}$, whose class is assumed to be 0 (i.e., nominal example), Algorithm 2 first computes the direction of the perturbation as the sign of the maximum loss increment $sign(\nabla_{\boldsymbol{X}} H^2(b^N_{\hat{s}_t}, \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})))$ (lines 1-7), following the strategy of FGSM. The classifier f is HHAD, hence it combines the application of the Viterbi algorithm and the threshold on the Hellinger distance between data distribution in $\boldsymbol{X}$ and distribution of the HMM emission model, to determine the class of the example. The loss function has in general a complex form in this case, because it depends on the maximally frequent state in $\boldsymbol{X}$. The main obstacle in the computation of the derivative of the loss function is related to the solution of the inverse Viterbi problem, which is NP-complete [57]. Intuitively, it is very complex to identify the point $\boldsymbol{X'}$ of maximum increase of the loss function in the ϵ-neighbourhood of $\boldsymbol{X}$ because the reference emission

4. With a slight abuse of notation, $\boldsymbol{X}$ represents both a multivariate time series and a generic slice (window) of the time series; the latter is sometimes also called $\boldsymbol{W}$ or $\boldsymbol{W}_t$ for a generic time t (see Section 3).

5. The algorithm assumes $\boldsymbol{X}$ to be an example of the training set which we want to augment, hence $\boldsymbol{X}$ is nominal.

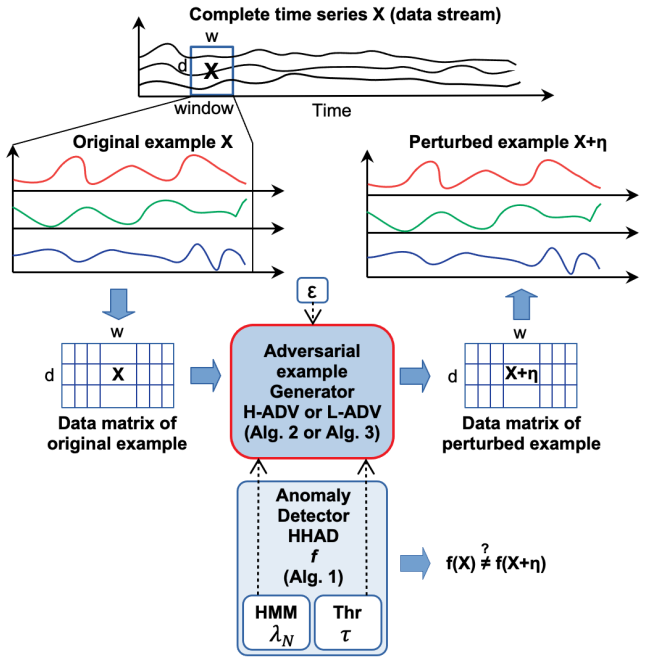

Fig. 1: Overview of the adversarial generation process.

Algorithm 2: Adversarial example generation based on Hellinger distance (H-ADV)

Input: $\boldsymbol{X} \in \mathbb{R}^{d \times w}$ ← original nominal example
λ^N ← nominal HMM, $c \in \mathbb{N}$ ← # steps per state
$\tau \in [0,1]$ ← H^2 threshold, ϵ ← max perturbation size
Output: $\boldsymbol{X}' \in \mathbb{R}^{d \times w}$: perturbed example

1 $\boldsymbol{S}_t \leftarrow$ Viterbi$(\lambda^N, \boldsymbol{X})$
2 $\hat{s}_t =$ most frequent state in $\boldsymbol{X}$
3 $\boldsymbol{Y} \leftarrow \{\boldsymbol{x}_j \in \boldsymbol{X} \ : \ s_j = \hat{s}_t\}$
4 $\boldsymbol{\mu} \leftarrow E[\boldsymbol{Y}]$
5 $\boldsymbol{\Sigma} \leftarrow E[(\boldsymbol{Y} - \boldsymbol{\mu})(\boldsymbol{Y} - \boldsymbol{\mu})^T]$
6 $\boldsymbol{g} = sign(\nabla_{\boldsymbol{X}} H^2(b^N_{\hat{s}_t}, \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}))$ // gradient in $\boldsymbol{X}$
7 $h_{max} = H^2(b^N_{\hat{s}_t}, \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}))$ // Hellinger distance of $\boldsymbol{X}$
8 $\boldsymbol{X}' = \boldsymbol{X}$
9 **repeat**
10 cont=False
11 $\boldsymbol{X}' = \boldsymbol{X}' + \frac{\epsilon}{c} \cdot \boldsymbol{g}$ // $\boldsymbol{X}$ update
12 $\boldsymbol{S}'_t \leftarrow$ Viterbi$(\lambda^N, \boldsymbol{X}')$
13 $\hat{s}'_t$ most frequent state for $\boldsymbol{S}'_t$
14 $\boldsymbol{Y}' \leftarrow \{\boldsymbol{x}_j \in \boldsymbol{X}' \ : \ s_j = \hat{s}'_t\}$
15 $\boldsymbol{\mu}' \leftarrow E[\boldsymbol{Y}']$
16 $\boldsymbol{\Sigma}' \leftarrow E[(\boldsymbol{Y}' - \boldsymbol{\mu}')(\boldsymbol{Y}' - \boldsymbol{\mu}')^T]$
17 $h' = H^2(b^N_{\hat{s}'_t}, \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\Sigma}'))$
18 **if** $h' > h_{max}$ **then**
19 $h_{max} = h'$ // update max Hellinger
20 cont=True
21 **end**
22 **if** $\hat{s}_t \neq \hat{s}'_t$ **then**
23 // New most frequent state
24 $\boldsymbol{g} = sign(\nabla_{\boldsymbol{X}'} H^2(b^N_{\hat{s}'_t}, \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\Sigma}'))$ // update $\boldsymbol{g}$
25 $\hat{s}_t = \hat{s}'_t$
26 **end**
27 **until** $(\|\boldsymbol{X}' - \boldsymbol{X}\|_1 \geq d \cdot \epsilon) \vee$ (cont==False) $\vee$ $(h_{max} > \tau)$
28 **return** $\boldsymbol{X}'$

model $b^N_{\hat{s}_t}$ in that neighbourhood can change if the most frequent hidden state $\hat{s}_t$ changes. To overcome this issue, we compute the gradient in $\boldsymbol{X}$ and assume it does not change in a small $\frac{\epsilon}{c}$-neighbourhood of $\boldsymbol{X}$. Hence, we move in this neighbourhood following the direction of the gradient in $\boldsymbol{X}$ and then we iterate this procedure from the new point $\boldsymbol{X}'$ reached from $\boldsymbol{X}$ (lines 9-27). Namely, in $\boldsymbol{X}'$ we re-compute the gradient of the loss in $\boldsymbol{X}'$ and we move according to it. The algorithm in this way adapts the gradient of the loss function to the reference emission model that can change inside the ϵ-hypercube with side 2ϵ centered in $\boldsymbol{X}$. The gradient of this loss function can be expressed in closed form when the covariance matrix is diagonal (see details in Section 2.1 of the supplementary material). The process is iterated until the border of the hypercube is reached, or the Hellinger distance of the perturbed example starts to decrease, or the Hellinger distance exceeds the threshold τ (i.e., is $\boldsymbol{X}'$ classified as anomalous). Not all the perturbed examples change their class; only the perturbed examples that change their class[6] are *adversarial* examples.

Figure 2 provides a graphical overview of the strategy implemented by Algorithm 2. The algorithm computes the final perturbed $\boldsymbol{X}'$ by iteratively performing two macro-steps. First, it moves in the direction of the gradient of the loss function. Second, if the reference emission model changes in the path, then it recomputes the gradient based on the new emission model. The point is perturbed until it reaches the border of the hypercube or the decision boundary of the anomaly detector (i.e., a point in which the example is classified as an anomaly). In the picture the most frequent state in example $\boldsymbol{X}$ is $\hat{s}_t = 1$, hence the first perturbation is computed according to gradient $\nabla_{\boldsymbol{X}} H^2(b^N_{\hat{s}_t=1}, \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}))$, that uses the emission model of the first hidden state as a reference. Then, a change of the most frequent state to $\hat{s}_t = 2$ occurs in $\boldsymbol{X}^{(1)}$, hence, the gradient is there recomputed according to the parameters of the emission model of that state, namely, $\nabla_{\boldsymbol{X}} H^2(b^N_{\hat{s}_t=2}, \mathcal{N}(\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}))$ where $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\Sigma}^{(1)}$ are computed in $\boldsymbol{X}^{(1)}$, and that gradient is followed from $\boldsymbol{X}^{(1)}$ until the most frequent state changes again in $\boldsymbol{X}^{(2)}$ to $\hat{s}_t = 3$. Again, the gradient is recomputed according to the emission model of state $\hat{s}_t = 3$ and it is followed until the decision boundary is reached in $\boldsymbol{X}^{(3)}$. That point represents the final perturbation of $\boldsymbol{X}$, which is an adversarial example since $\boldsymbol{X}'$ is classified as anomalous.

L-ADV. The second algorithm for adversarial example generation that we propose is applied to the same detector HHAD but it generates adversarial examples following the gradient of the likelihood of the example $\boldsymbol{X}$ instead of the gradient of the Hellinger distance. This algorithm for adversarial example generation is called L-ADV and is formalized in Algorithm 3. It avoids the mathematical calculation of the gradient of the loss function needed for the case of the Hellinger distance. With the likelihood, in fact, the loss does not depend on the maximally frequent state in the window, but it can be computed using the standard forward algorithm [58]. The gradient of the likelihood can be recursively calculated. Note that Algorithm 3 does not require the $\frac{\epsilon}{c}$ steps of the case of the Hellinger distance, because it does not matter if the maximally frequent state

6. Since original examples from the training set are nominal, the change of class makes adversarial examples anomalous.

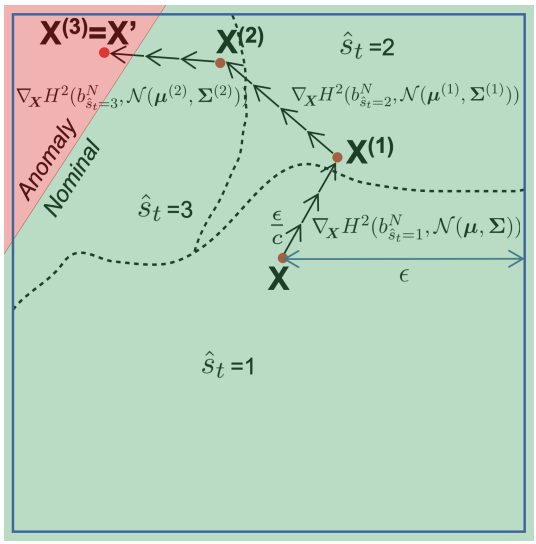

Fig. 2: Iterative gradient ascent strategy performed by the adversarial generation algorithm H-ADV.

in the window changes or not. Given an observation $\boldsymbol{X}$, first it computes the sign of the gradient of the likelihood of the example given the nominal HMM (line 1), namely $sign(\nabla_{\boldsymbol{X}} P(\boldsymbol{X}, \lambda^N))$ (see details in Section 2.2 of the supplementary material). Then it moves the example in the direction of the gradient for a step ϵ in each dimension (line 2). The algorithm returns the perturbed example $\boldsymbol{X}'$. Notice that the perturbed example $\boldsymbol{X}'$ is an adversarial example only if the Hellinger distance between the maximally frequent observations in $\boldsymbol{X}'$ and the emission model of the maximally frequent state in $\boldsymbol{X}'$ is larger than threshold τ, namely, only if HHAD classifies $\boldsymbol{X}'$ as anomalous according to its parameters λ^N and τ.

Algorithm 3: Adversarial example generation based on likelihood (L-ADV)

Input: $\boldsymbol{X} \in \mathbb{R}^{d \times w}$ ← original nominal example
λ^N ← nominal HMM
ϵ ← maximum perturbation size
Output: $\boldsymbol{X}' \in \mathbb{R}^{d \times w}$: perturbed example
1 $\boldsymbol{g} = sign(\nabla_{\boldsymbol{X}} P(\boldsymbol{X}, \lambda^N))$ // likelihood gradient in $\boldsymbol{X}$
2 $\boldsymbol{X}' = \boldsymbol{X} - \epsilon \cdot \boldsymbol{g}$ // $\boldsymbol{X}$ update
3 **return** $\boldsymbol{X}'$

Complexity analysis. The time complexity of Algorithm 2 is $O(c \cdot (wK^2 + wd))$, where c is the number of steps per state, $O(wK^2)$ is the computational cost of the Viterbi algorithm, and $O(wd)$ is the cost of performing a gradient step (cost $O(1)$) on each dimension and time step of the window. The time complexity of Algorithm 3 is $O(K^2 w^2 d)$. Being the gradient of the HMM likelihood not computable in closed form but only recursively, its time complexity is $O(wK^2)$, which is considerably higher than the complexity of the computation of the gradient of the Hellinger distance, which is $O(1)$ when computable in closed form. As a consequence, the computational complexity of Algorithm 3 is quadratic in the window length, which results in a considerable increase of the running time (i.e., it is $\approx$ 400 times slower than Algorithm 2). Fortunately, Algorithm 3 is quite parallizable, in fact, after re-implementing it in `Cython` we managed to achieve a running time similar to that of Algorithm 2 (see Section 5). The complexities of the proposed algorithms are summarized in the supplementary material (Section 1 and Table 1).

Remark. As discussed later, H-ADV empirically shows the best performance but its gradients are not derivable in closed form for HMM with non-diagonal covariance matrices of emission distributions. L-ADV should be employed in these cases.

4.2 Data Augmentation and Retraining: HHAD-AUG

Adversarial generation procedures H-ADV and L-ADV are here integrated in a technique for data augmentation and retraining called HHAD-AUG. Since the two adversarial generation procedures are used in a mutually exclusive way we refer to the data augmentation using H-ADV as H-AUG and to the data augmentation using L-ADV as L-AUG. Algorithm 4 formalizes the proposed approach. It aims at improving the performance of the anomaly detector HHAD and its robustness to adversarial attacks. Inputs are the nominal time series $\boldsymbol{X}$ used to train HMM λ^N, the original HMM λ^N (having K hidden states), the window size w, the loss function $loss_f$ (i.e., based on Hellinger distance or on likelihood), the threshold τ for the Hellinger distance, the maximum perturbation size ϵ, the number of steps c in which ϵ is split during adversarial generation, and the number of times m that adversarial examples are generated on the training set. Outputs are the augmented training set of nominal examples $\hat{\boldsymbol{\mathcal{W}}}$, the augmented nominal HMM $\hat{\lambda}^N$ (trained on $\hat{\boldsymbol{\mathcal{W}}}$), and the augmented threshold $\hat{\tau}$ learned from $\hat{\lambda}^N$ and $\boldsymbol{\mathcal{W}}$ (the original training set generated from $\boldsymbol{X}$).

The augmented dataset $\hat{\boldsymbol{\mathcal{W}}}$ is first initialized to the set of examples in the training set generated by covering the complete time series $\boldsymbol{X}$ with a sliding window of length w (line 2). Similarly, the augmented nominal HMM is initialized to the original nominal HMM (line 3) and the augmented threshold to the original threshold (line 4). Then the augmentation loop is iterated m times. The steps of this loop are described in the following. For each training example $\boldsymbol{W}_t = \langle \boldsymbol{x}_{t-w+1}, \ldots, \boldsymbol{x}_t \rangle$ in the training set $\boldsymbol{\mathcal{W}}$ (line 7) a perturbed example $\boldsymbol{W}'_t$ is generated using algorithm H-ADV or L-ADV (lines 9 and 11) depending on the loss function $loss_f$ chosen for adversarial generation. The perturbed example $\boldsymbol{W}'_t$ is added to the augmented set $\hat{\boldsymbol{\mathcal{W}}}$ as a nominal example only if it is classified as an anomaly by HHAD (lines 13-15). We use label 0 for nominal and 1 for anomalous examples, hence the output of HHAD can be 0 or 1 and in line 14 $y == 1$ means that the window $\boldsymbol{W}'_t$ has been labeled as an anomalous. When all examples in the training set have been perturbed, the nominal HMM is retrained using the augmented dataset $\hat{\boldsymbol{\mathcal{W}}}$ (line 18) which contains original examples and adversarial examples. The retrain is performed from scratch to avoid any bias in the HMM parameters. The threshold is then updated, only if its value is increased, to the value of the maximum Hellinger distance computed using the augmented HMM on examples in the original dataset $\boldsymbol{\mathcal{W}}$ (lines 19-22 in Algorithm 4).

Algorithm 4: Adversarial data augmentation and retraining (HHAD-AUG)

Input: $\boldsymbol{X} \leftarrow$ nominal d-dimensional time series, $\lambda^N \leftarrow$ Baum-Welch($\boldsymbol{X}$, K), $K \leftarrow$ # hidden states, $m \in \mathbb{N} \leftarrow$ # iterations, $\epsilon \leftarrow$ max perturbation size, $w \leftarrow$ window size, $\tau \leftarrow$ threshold, $loss_f \leftarrow$ loss function (H^2 or LL), $c \in \mathbb{N} \leftarrow$ # steps per state

Output: $\hat{\boldsymbol{\mathcal{W}}}$ augmented set of nominal examples $\hat{\lambda}^N$ aug. nominal HMM, $\hat{\tau}$ aug. threshold

1 $\boldsymbol{\mathcal{W}} = \{\langle \boldsymbol{x}_{t-w+1}, \ldots, \boldsymbol{x}_t \rangle \mid t = w, \ldots, n\}$ // original training set
2 $\hat{\boldsymbol{\mathcal{W}}} = \boldsymbol{\mathcal{W}}$ // initialization of the augmented dataset
3 $\hat{\lambda}^N = \lambda^N$ // initialization of the augmented HMM
4 $\hat{\tau} = \tau$ // initialization of the augmented threshold
5 **foreach** $i = 1, \ldots, m$ **do**
6 **foreach** $t = w, \ldots, n$ **do**
7 $\boldsymbol{W}_t = \langle \boldsymbol{x}_{t-w+1}, \ldots, \boldsymbol{x}_t \rangle$ // select example
8 **if** $(loss_f == H^2)$ **then**
9 $\boldsymbol{W}'_t = \text{H-ADV}(\boldsymbol{W}_t, \hat{\lambda}^N, \epsilon, c, \tau)$
10 **else**
11 $\boldsymbol{W}'_t = \text{L-ADV}(\boldsymbol{W}_t, \hat{\lambda}^N, \epsilon)$
12 **end**
13 $y = \text{HHAD}(\boldsymbol{W}'_t, K, \hat{\lambda}^N, w, \hat{\tau})$
14 **if** *(y==1)* **then**
15 $\hat{\boldsymbol{\mathcal{W}}} = \hat{\boldsymbol{\mathcal{W}}} \cup \boldsymbol{W}'_t$ // add adversarial $\boldsymbol{W}'_t$
16 **end**
17 **end**
18 $\hat{\lambda}^N = Baum\text{-}Welch(\hat{\boldsymbol{\mathcal{W}}}, K)$ // retrain HMM λ^N
19 $\tau' =$ maximum value of H^2 for examples in $\boldsymbol{\mathcal{W}}$ computed using $\hat{\lambda}^N$
20 **if** $(\tau' > \hat{\tau})$ **then**
21 $\hat{\tau} = \tau'$ // update threshold $\hat{\tau}$
22 **end**
23 **end**
24 **return** $\hat{\boldsymbol{\mathcal{W}}}, \hat{\lambda}^N, \hat{\tau}$

Again, we observe that adversarial examples are added to the augmented dataset of nominal behaviours only if they have been classified as anomalies by HHAD. They are added as nominal examples, hence the learning process remains one-class also after data augmentation. This is the main idea of our approach, and it is based on the intuition that the adversarial examples are very close to the original nominal examples, hence we consider them as misclassified by the original detector. Finally, in line 7 we consider only examples $\boldsymbol{W}_t$ from the training set in all m iterations, i.e., adversarial examples are not considered as original examples to generate other examples. This guarantees that the decision boundary is not moved away from the training examples indefinitely.

The generation of adversarial examples from each example in the training set is iterated m times (lines 5-23). Each time the HMM is retrained and the threshold $\hat{\tau}$ updated. The algorithm augments the training set m times, iteratively, each time using adversarial examples based on the current version of the HMM. At the first iteration the HMM is the original one, trained using examples from the training set $\mathcal{W}$. At the second iteration the HMM is retrained using both the examples in the original training set and the adversarial examples generated at the first iteration (all labeled as nominal). At the third iteration the HMM is retrained another time, considering the examples previously generated. Our empirical analysis shown that the main improvement in terms of F1-score of the augmented detector is achieved in the first three iterations, hence we set $m = 3$ in our tests. The updated HMM $\hat{\lambda}^N$ and threshold $\hat{\tau}$ provide an actual performance improvement in terms of anomaly detection accuracy and other measures discussed in the next section.

Complexity analysis. The computational complexity of Algorithm 4 is $O(m \cdot (|\boldsymbol{X}| - w) \cdot ADV)$, where ADV is the complexity of one of the adversarial generation algorithms, i.e., H-ADV or L-ADV. The computational complexity is linear in the number of iterations, in the size of the initial training set, and in the complexity of the adversarial example generation algorithm. The empirical evaluation shows that the proposed approach can scale to realistic scenarios.

5 Experimental Results

Results of application of our approach are presented for four application domains related to robotic and cyber-physical systems. We evaluate the performance improvement (measured as F1-score) of the augmented detectors on different training set sizes and provide insights about the mechanisms that generate the improvement. The adversarial examples generated by H-ADV and L-ADV are shown to be very similar to original examples (for images, we would say that they are indistinguishable) and to yield performance improvement if added to the training set. In fact, perturbations of the same intensity but performed using (non-adversarial) baseline augmentation methods are not able to achieve the same performance improvement. The Python code of the experiments is available[7]. Software and hardware details are provided in Section 4 of the supplementary material.

5.1 Experimental Setting

Given a specific application domain and a related dataset $\boldsymbol{D}$ containing multiple time series for a process of interest (in which each time series has been standardized), our main results show that the proposed data augmentation techniques H-AUG and L-AUG outperform the baseline augmentation techniques R-AUG, D-AUG, G-AUG, and S-AUG. We also show that HHAD performs as or better than state-of-the-art detectors based on vanilla autoencoders (AEs), LSTM autoencorders (LSTM-AEs), and one-class support vector machines (OCSVM) (see details in Section 3.2 of the supplementary material), hence the performance of our augmented detectors H-AUG and L-AUG exceeds not only that of HHAD but also the state-of-the-art.

We assume to have a nominal HMM λ^N with K hidden states (chosen by optimizing the BIC) and trained on a training set of nominal time series $\boldsymbol{X}$ which is part of dataset $\boldsymbol{D}$. Another part of $\boldsymbol{D}$, called $\boldsymbol{T}$ (i.e., test set) in the following, is used to evaluate the performance of the anomaly detection and the data augmentation and retraining algorithms. We also assume a specific window size w and a threshold τ learned on $\boldsymbol{X}$ as described in [7]. With these three elements, namely, λ^N, w, and τ, a complete instance of the original

7. https://github.com/HHADAdversarialAugmentation/adv_data_aug_hmm

anomaly detector HHAD is available. Notice that, when the number of variables in the dataset is high, feature selection or dimensionality reduction could be necessary to obtain good performance of the original anomaly detector.

The main dimensions of analysis that we consider are the *type of loss function* used to generate adversarial examples (i.e., Hellinger distance or likelihood) and the *size of the training set* $|\boldsymbol{X}|$. Table 1 summarizes the parameters of all the experiments described in the following subsections. The number of repetitions *#rep* of each test on different training sets of the same size is set to 30 in all domains. It means that, given a training set size $|\boldsymbol{X}|$ we recompute the performance improvement 30 times, and each time we train the HMM λ^N and the threshold τ on different training sets of size $|\boldsymbol{X}|$.

	Domain			
Par.	**TE**	**SWaT**	**ALFA**	**INTCATCH**
$\lvert\boldsymbol{X}\rvert$	250, 500, 750, 1000 1250, 1500	2500, 5000, 10,000 20,000	250, 500, 750, 1000 1250, 1500	250, 500, 750, 1000 1250, 1500
#rep	30	30	30	30
$\lvert\boldsymbol{T}\rvert$	3201	449,919	1068	6619
d	4 PCs	2 PCs	3 PCs	4 PCs
K	[2,15]	[2,25]	[2,20]	[2,15]
w	100	50	100	100
ϵ	0.05	0.05	0.05	0.05
m	3	3	3	3

TABLE 1: Summary of experimental parameters used in all application domains.

5.2 Performance Measures

Anomaly detection performance of algorithm HHAD is evaluated by *F1-score* [56] (Cohen's κ-score and recall are also reported in Section 3 of the supplementary material) on a test set $\boldsymbol{T}$ which is kept separated from the training set used to learn λ^N and τ. We consider positive the nominal examples and negative the anomalous examples. Hence, true positives are nominal examples correctly classified by HHAD, true negatives are anomalous examples correctly detected by HHAD, and so on. The value of the F1-score must be maximized.

Data augmentation and retraining algorithms H-AUG and L-AUG, using loss functions based on the Hellinger distance and the likelihood, respectively, are evaluated by two measures. First, we compute the improvement of F1-score on the test set induced on HHAD by data augmentation and retraining. This is the *difference between the F1-score on the test set after and before data augmentation and retraining*. To test the statistical significance of this difference we apply algorithm HHAD-AUG to several training sets containing different examples and with different size (see results in the next sections) and then use Student's t-test for testing the null hypothesis that the average performance on test sets are significantly improved. Then we evaluate the improvement of the robustness to adversarial attacks introduced by HHAD-AUG. We measure it as the *difference in percent success rate SR%* between original and augmented HHAD. The *percent success rate* (*SR%*) of a detector is the percentage of perturbed examples (on the test set) that are actually misclassified by the detector. A small *SR%* means that the detector is robust to adversarial attacks generated on the test set. Differences in percent success rate *SR%* between HHAD and HHAD-AUG are shown in Tables 3, 7, 11, and 15 of the supplementary material.

5.3 Tennessee-Eastman Industrial Process (TE)

In this domain a synthetic model of a real industrial chemical process is used to evaluate process control strategies [14].

Domain and dataset description. This domain has become popular in the Industrial Control System (ICS) security community because it allows to test attack and defense approaches on a realistic (although simulated) environment. We used the dataset provided by [43] which contains integrity attacks on both sensors and actuators. We use data related to the stealth attack named SA1. They have 41 variables and 4801 observations. A label is available for each example, namely, 0 for nominal observations and 1 for anomalous observations. The training set is generated taking a slice of sequential nominal observations of length $|\boldsymbol{X}| \in \{250, 500, 750, 1000, 1250, 1500\}$ (see Table 1). The test set is a sequence containing $|\boldsymbol{T}| = 3201$ observations, of which 2400 are nominal and 801 anomalous. The training sets are selected in the interval between time steps 0 and 1599, and the test set in the interval between time steps 1600 and time step 4801. For each training set size, we generate 30 training sets (sampling the original dataset in different positions) and we compute mean and standard deviations of the performance in the test set.

Experimental parameters. For each training set, we reduce the dimensionality to the first 4 principal components (using PCA). The number of hidden states K of the nominal HMM λ^N is then selected by BIC in the interval $[2, 15]$. Diagonal covariance matrix is used. The window length is $w = 100$ and the maximum perturbation size is $\epsilon = 0.05$. The number of iterations of the data augmentation and retraining procedure is $m = 3$ (see Table 1).

Results. Figure 3 shows the main results. The x-axis represents the training set size $|\boldsymbol{X}|$ and the y-axis the F1-score on the test set. The blue solid line is the original detector HHAD with related 95%-confidence interval (shaded area). Notice that this detector is exactly the one that we use to generate adversarial examples in the initial iteration of data augmentation. Dashed lines with other colors represent different data augmentation strategies, namely, orange is H-AUG, green is L-AUG, red is R-AUG, purple is D-AUG, brown is G-AUG, and pink is S-AUG. The average performance improvement achieved by H-AUG is statistically significant for all training set sizes. In particular, the F1-score of the augmented anomaly detector (i.e., HHAD with $\hat{\lambda}^N$ and $\hat{\tau}$) and show that it is higher than that of the original detector (i.e., HHAD with λ^N and τ). L-AUG provides a statistically significant improvement only for $|\boldsymbol{X}| \in \{750, 1000, 15000\}$. A motivation for this is provided below. Interestingly, HHAD augmented by baseline methods do not achieve any significant performance improvement with respect to the original HHAD. Overall, these results show that the proposed adversarial data augmentation strategy is effective in this application domain.

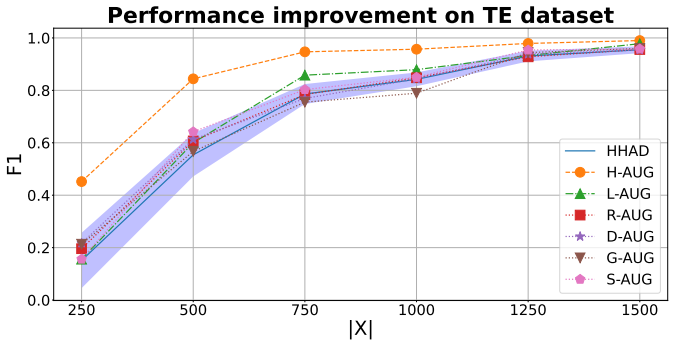

Fig. 3: Average F1-score (over 30 datasets) for the original detector HHAD and augmented detectors H-AUG, L-AUG, R-AUG, D-AUG, G-AUG, S-AUG on different training set sizes in the TE dataset.

Table 2 provides a quantitative evaluation of the performance improvement achieved by each data augmentation method with respect to the original detector HHAD. The first two rows show, respectively, the average F1-score μ_{F1} and related standard deviation σ_{F1} for HHAD. Both statistics are computed over 30 repeats for each training set size. Then, for each data augmentation algorithm we show in the white rows the average F1-score and standard deviation, and in the gray rows the difference of average F1-score Δ_{F1} (i.e., augmented detector minus original detector) and the p-value of the Student's t-test (p-val) for the difference in the average F1-scores. We consider performance improvements statistically significant when the p-value is less than 0.05. These values are highlighted in bold in the table. The improvement of H-AUG is large with small training sets and it decreases as the training set size increases, as expected, hence our methodology could be effectively used in applications with small amounts of data to improve the detection performance. For instance, the average F1-score (μ_{F1}) of the augmented detector H-AUG trained with 500 examples is equivalent to the F1-score of the original detector HHAD trained with 1000 examples.

Role of adversarial examples in data augmentation and retraining. To investigate the role of adversarial examples in data augmentation and retraining, we first observe that, on average, 1.12% of the adversarial examples generated on the training set by H-AUG and 0.15% of those generated by L-AUG are successful (i.e., they change the HHAD classification) in all the m iterations of the related augmentation algorithm. This corresponds to an average of 7.63 adversarial examples added to the training set by H-AUG and 1.31 by L-AUG (these values are averaged over different training set sizes). The F1-score improvement (on the test set) of H-AUG with $|\boldsymbol{X}| = 500$, for instance, is obtained adding only 6.05 adversarial examples on average (over 30 repetitions on different training sets). The small performance improvement of L-AUG on TE is instead probably due to the too low success rate in generating adversarial examples. Further results are available in Section 3.3 of the supplementary material.

Comparison with state-of-the-art anomaly detectors. Figure 4 shows the average F1-score of the original anomaly detector HHAD, the augmented detectors H-AUG and L-AUG, and the three state-of-the-art (non-augmented) detectors AE, LSTM-AE, and OCSVM (see details in Section 3.2 of the supplementary material), for different dataset sizes. Clearly, H-AUG outperforms all methods. L-AUG have slightly better performance than HHAD but worse than H-AUG. The detectors based on artificial neural networks have very bad performance in this case, because of the small size of the dataset and specific data structure. OCSVM starts to increase its performance from 1000 examples but it does not reach the F1-scores of HHAD. Further details are provided in Section 3.3 of the supplementary material.

		Training set size $\|\boldsymbol{X}\|$ (Tennessee Eastman)					
Detector	**Means**	**250**	**500**	**750**	**1000**	**1250**	**1500**
HHAD	μ_{F1}	0.151	0.554	0.786	0.842	0.931	0.955
	σ_{F1}	0.177	0.184	0.146	0.096	0.043	0.028
H-AUG	μ_{F1}	0.452	0.844	0.947	0.957	0.979	0.99
	σ_{F1}	0.244	0.158	0.066	0.051	0.021	0.009
	Δ_{F1}	**0.301**	**0.290**	**0.161**	**0.115**	**0.048**	**0.035**
	p-val	**9.8e-6**	**1.5e-8**	**2.3e-6**	**4.9e-6**	**8.6e-7**	**1.1e-6**
L-AUG	μ_{F1}	0.156	0.600	0.858	0.879	0.933	0.978
	σ_{F1}	0.163	0.177	0.116	0.088	0.045	0.023
	Δ_{F1}	0.005	0.046	**0.077**	**0.037**	0.003	**0.023**
	p-val	0.676	0.051	**0.012**	**0.019**	0.440	**3.4e-4**
R-AUG	μ_{F1}	0.196	0.607	0.784	0.848	0.928	0.956
	σ_{F1}	0.206	0.204	0.146	0.098	0.036	0.026
	Δ_{F1}	0.045	0.053	-0.002	0.006	-0.002	0.001
	p-val	0.199	0.239	0.952	0.794	0.834	0.693
D-AUG	μ_{F1}	0.217	0.613	0.769	0.844	0.938	0.954
	σ_{F1}	0.251	0.211	0.173	0.092	0.046	0.030
	Δ_{F1}	0.066	0.059	-0.017	0.002	0.007	-0.001
	p-val	0.153	0.260	0.643	0.935	0.460	0.982
G-AUG	μ_{F1}	0.213	0.567	0.755	0.789	0.944	0.960
	σ_{F1}	0.309	0.236	0.135	0.082	0.041	0.024
	Δ_{F1}	0.062	0.013	-0.031	**-0.052**	0.013	0.005
	p-val	0.156	0.792	0.273	**0.011**	0.261	0.581
S-AUG	μ_{F1}	0.156	0.642	0.803	0.850	0.954	0.958
	σ_{F1}	0.319	0.249	0.146	0.070	0.034	0.021
	Δ_{F1}	0.005	0.088	0.017	0.008	**0.023**	0.003
	p-val	0.941	0.057	0.601	0.669	**0.035**	0.698

TABLE 2: Average F1-scores on the test set of the original (HHAD) and augmented anomaly detectors (H-AUG, L-AUG, R-AUG, D-AUG, G-AUG, and S-AUG) on different training set sizes in the TE domain. Averages are computed over 30 datasets, for each dataset size. Average F1-score improvements Δ_{F1} with respect to HHAD are also displayed with p-values for testing their statistical significance. Statistically significant performance improvements (p-value < 0.05) are highlighted in bold.

5.4 Secure Water Treatment Testbed (SWaT)

As a second test on an industrial CPS we use SWaT, a scaled-down version of a real-world industrial water treatment plant [15]. We report the description of domain and experiments, and full results in the supplementary material (Section 3.4). Results are successful also in this case. The average performance improvement achieved by both H-AUG and L-AUG with respect to HHAD are statistically significant for all training set sizes. Baseline augmented detectors R-AUG and D-AUG in this case manage to improve the average F1-score on the test set only for the smaller datasets but the improvement is smaller than that achieved by H-AUG. For larger training set sizes these two baseline methods achieve negative or null improvement, while the proposed methods always get significant improvement. G-AUG and S-AUG do not achieve significant improvement with respect to.

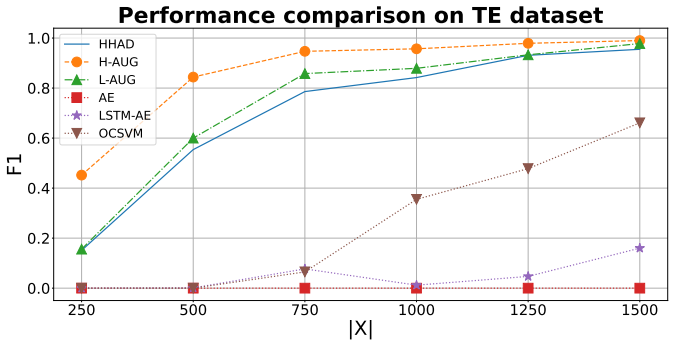

Fig. 4: Average F1-score (over 10 datasets) for the original detector HHAD, augmented detectors H-AUG and L-AUG, and state-of-the-art (non augmented) anomaly detectors AE, LSTM-AE, and OCSVM on different training set sizes in the TE dataset.

HHAD. The improvement of H-AUG keeps almost constant with the increase of the training set size, showing that the F1-score can still increase also starting from relatively large datasets (i.e., 2500 examples). Also in this case the gain is relevant, since the F1-score obtained by the detector augmented with H-ADV on 5000 examples is larger than the F1-score obtained by the original detector HHAD using 20,000 examples, with a "saving" of about 15,000 examples. The average improvement of success rate $\Delta_{SR\%}$ of adversarial attacks on the test set is always negative, meaning that the data augmentation improves the robustness of the detector to adversarial attacks.

5.5 UAV Fault and Anomaly Detection (ALFA)

The third domain is a robotic one related to Unmanned Aerial Vehicles (UAV). The dataset[8] presents several fault types in control surfaces of a fixed-wing UAV for use in Fault Detection and Isolation (FDI) and Anomaly Detection (AD) research [16]. It includes processed data for 47 autonomous flights with 23 sudden full engine failure scenarios and 24 scenarios for seven other types of sudden control surface (actuator) faults, with a total of 66 minutes of flight in normal conditions and 13 minutes of post-fault flight time. The platform used for collecting data is a custom modification of the Carbon Z T-28 model plane. The average performance improvement achieved by H-AUG with respect to HHAD on the ALFA domain is statistically significant for all training set sizes. Figure 5 graphically shows these results. L-AUG has on average better F1-score than HHAD and the difference is statistically significant for all training set sizes except for $|\boldsymbol{X}| = 250$, however the amount of the improvement is less than that achieved by H-AUG. Baseline methods R-AUG, D-AUG G-AUG, and S-AUG do not achieve any significant performance improvement except for a small improvement obtained by D-AUG on $|\boldsymbol{X}| = 1500$. In this case the larger improvement is achieved by H-AUG on small and medium-size datasets, with a maximum improvement of the F1-score of 0.313 for $|\boldsymbol{X}| = 500$. Out of the four, this is certainly the most difficult dataset containing complex behaviours of a real autonomous system, in fact, anomalies are often not recognizable at human inspection. Nevertheless, H-AUG manages to strongly improve the anomaly detection performance, reaching F1-score up to 0.909 with the larger training sets considered in our analysis. Section 3.5 of the supplementary material provides full details about the results of experiments performed on this domain.

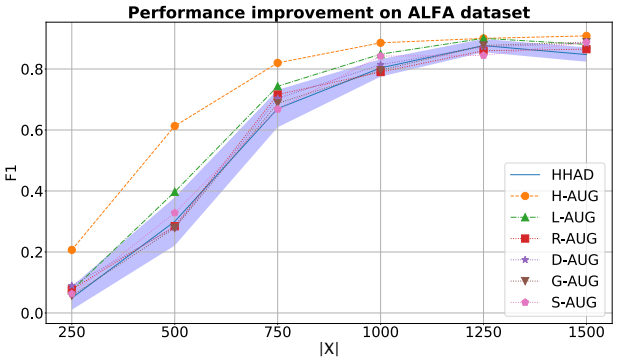

Fig. 5: Average F1-score (over 30 datasets) for the original detector HHAD and augmented detectors H-AUG, L-AUG, R-AUG, D-AUG, G-AUG, and S-AUG on different training set sizes in the ALFA dataset.

5.6 Water Monitoring with ASV (INTCATCH)

The fourth domain is again related to mobile robots but in this case we consider an Autonomous Surface Vessel (ASV). The dataset [17] has been generated by an ASV called Platypus which operates in the context of the INTCATCH Project[9], an EU H2020 project aiming at developing new paradigms for water monitoring in river and lakes. Also in this case the average performance improvement of both H-AUG and L-AUG with respect to HHAD and the baseline data augmentation techniques is statistically significant for small training set sizes while it becomes negligible for larger sizes. The robustness against adversarial attacks on the test set also improves. A detailed description of the domain, the experiments, and the results is reported in Section 3.6 of the supplementary material.

6 Conclusions

Detection of anomalous behaviours in intelligent systems that operate in the physical world, such as autonomous robots and CPSs, requires tools able to represent nominal behaviours and discover patterns that do not match with them in sensor traces. HMMs are a viable and established tool for modeling dynamical behaviours contained in multivariate time series and recent methods use them to detect anomalous behaviours in robotic systems. The approach of adversarial data augmentation presented in this work improves the detection performance of such tools without using any prior knowledge about the form of the nominal time series, as traditional data augmentation requires. The adversarial examples we generate are multivariate time series very similar to the original but able to produce a

8. http://theairlab.org/alfa-dataset

9. https://www.intcatch.eu

significant performance improvement (up to 0.313 of F1 improvement in our empirical tests, see Δ_{F1} of H-AUG on ALFA with training set of 500 samples) if used to augment the training set of our detector. The same examples improve also the robustness of the detector to adversarial attacks (with an increase up to 8.6% in our experiments, see $\Delta_{SR\%}$ of H-AUG on TE with training set of 1000 samples).

This paper paves the way for future work along several directions. We will concentrate on three main extensions. The first concerns the introduction of other time series distance measures in the adversarial example generation strategy. The second is related to the application of the proposed approach to other types of anomaly detectors, such as autoencoders or one-class Support Vector Machines. The third extension focuses on the application of adversarial data augmentation to *active* anomaly detection, which aims to make the system controller actively looking for possible anomalies to detect them more precisely and promptly.

ACKNOWLEDGMENTS

Research partially funded by "Dipartimenti di Eccellenza 2018-2022", Italian Ministry of Education, Universities and Research, and European Union's Horizon 2020 research and innovation programme, grant agreement No 689341. DA is supported by the ABB-Politecnico di Milano Joint Research Center, which provides financial support. This paper is partially supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

REFERENCES

[1] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, "Artificial intelligence for long-term robot autonomy: A survey," *IEEE RA-L*, vol. 3, no. 4, pp. 4023–4030, 2018.

[2] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *P IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[3] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot Auton Syst*, vol. 57, no. 5, pp. 469–483, 2009.

[4] E. Di Lello, M. Klotzbücher, T. De Laet, and H. Bruyninckx, "Bayesian time-series models for continuous fault detection and recognition in industrial robotic tasks," in *Proc. IROS*, 2013, pp. 5827–5833.

[5] A. Castellini, M. Bicego, D. Bloisi, J. Blum, F. Masillo, S. Peignier, and A. Farinelli, "Subspace clustering for situation assessment in aquatic drones: A sensitivity analysis for state-model improvement," *Cybern and Syst*, vol. 50, no. 8, pp. 658–671, 2019.

[6] A. Castellini, M. Bicego, F. Masillo, M. Zuccotto, and A. Farinelli, "Time series segmentation for state-model generation of autonomous aquatic drones: A systematic framework," *Eng Appl Artif Intell*, vol. 90, no. C, 2020.

[7] D. Azzalini, A. Castellini, M. Luperto, A. Farinelli, and F. Amigoni, "HMMs for anomaly detection in autonomous robots," in *Proc. AAMAS*, 2020, p. 105–113.

[8] E. Hellinger, "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen," *Journal für die reine und angewandte Mathematik*, vol. 136, pp. 210–271, 1909.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, 2015.

[10] I. Oregi, J. D. Ser, A. Pérez, and J. A. Lozano, "Adversarial sample crafting for time series classification with elastic similarity measures," in *Proc. IDC*, 2018, pp. 26–39.

[11] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Adversarial attacks on deep neural networks for time series classification," in *Proc. IJCNN*, 2019, pp. 1–8.

[12] F. Karim, S. Majumdar, and H. Darabi, "Adversarial attacks on time series," *IEEE Trans Pattern Anal Mach Intell*, vol. 43, no. 10, pp. 3309–3320, 2020.

[13] S. Harford, F. Karim, and H. Darabi, "Generating adversarial samples on multivariate time series using variational autoencoders," *IEEE/CAA J Autom Sin*, vol. 8, no. 9, pp. 1523–1538, 2021.

[14] T. Larsson, K. Hestetun, E. Hovland, and S. Skogestad, "Self-optimizing control of a large-scale plant: The Tennessee Eastman Process," *Ind Eng Chem Res*, vol. 40, no. 22, pp. 4889–4901, 2001.

[15] A. P. Mathur and N. O. Tippenhauer, "SWaT: a water treatment testbed for research and training on ICS security," in *Proc. International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, 2016, pp. 31–36.

[16] A. Keipour, M. Mousaei, and S. Scherer, "ALFA: A Dataset for UAV Fault and Anomaly Detection," *Int J Robot Res*, vol. 1, p. 1–6, 2020.

[17] A. Castellini, D. Bloisi, J. Blum, F. Masillo, and A. Farinelli, "Multivariate sensor signals collected by aquatic drones involved in water monitoring: A complete dataset," *Data Brief*, vol. 30, p. 105436, 2020.

[18] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J Big Data*, vol. 6, no. 60, pp. 1 – 48, 2019.

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J Artif Intell Res*, vol. 16, no. 1, p. 321–357, 2002.

[20] Q. Wen, L. Sun, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," arXiv:2002.12478, 2020.

[21] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," arXiv:2007.15951, 2020.

[22] A. Salazar, L. Vergara, and G. Safont, "Generative adversarial networks and markov random fields for oversampling very small training sets," *Expert Syst Appl*, vol. 163, p. 113819, 2021.

[23] T. T. Um, F. M. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks," in *Proc. ICMI*, 2017, pp. 216–220.

[24] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data Augmentation for Time Series Classification using Convolutional Neural Networks," in *Proc. ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2016.

[25] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," arXiv:1611.01236, 2016.

[26] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," in *Proc. NeurIPS*, 2018, pp. 5334–5344.

[27] A. Sinha, H. Namkoong, and J. C. Duchi, "Certifying some distributional robustness with principled adversarial training," in *Proc. ICLR*, 2018.

[28] L. Zhao, T. Liu, X. Peng, and D. Metaxas, "Maximum-entropy adversarial data augmentation for improved generalization and robustness," arXiv:2010.08001, 2020.

[29] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. ECML/PKDD*, 2013, pp. 387–402.

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. ICLR*, 2014.

[31] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," arXiv:1511.04599, 2015.

[32] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. SP*, 2017, pp. 39–57.

[33] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," *Mach Learn*, vol. 107, no. 3, pp. 481–508, 2018.

[34] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Trans Pattern Anal Mach Intell*, vol. 41, no. 8, pp. 1979–1993, 2019.

[35] K. R. Mopuri, A. Ganeshan, and R. V. Babu, "Generalizable data-free objective for crafting universal adversarial perturbations," *IEEE Trans Pattern Anal Mach Intell*, vol. 41, no. 10, pp. 2452–2465, 2019.

[36] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," arXiv:1909.08072, 2019.

[37] G. Ortiz-Jimenez, A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "Optimism in the face of adversity: Understanding and improving deep learning through adversarial robustness," arXiv:2010.09624, 2021.
[38] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," in *Proc. EMNLP*, 2017, pp. 2021–2031.
[39] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," in *Proc. EMNLP*, 2018, pp. 2890–2896.
[40] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput Surv*, vol. 41, no. 3, pp. 1–58, 2009.
[41] L. Zhang, J. Zhao, and W. Li, "Online and unsupervised anomaly detection for streaming data using an array of sliding windows and PDDs," *IEEE Trans Cybern*, vol. 1, pp. 1–6, 2019.
[42] E. Khalastchi, M. Kalech, G. Kaminka, and R. Lin, "Online data-driven anomaly detection in autonomous robots," *Knowl Inf Syst*, vol. 43, no. 3, pp. 657–688, 2015.
[43] W. Aoudi, M. Iturbe, and M. Almgren, "Truth will out: Departure-based process-level detection of stealthy attacks on control systems," in *Proc. CCS*, 2018, p. 817–831.
[44] N. Muralidhar, C. Wang, N. Self, M. Momtazpour, K. Nakayama, R. Sharma, and N. Ramakrishnan, "Illiad: Intelligent invariant and anomaly detection in cyber-physical systems," *ACM Trans Intell Syst Technol*, vol. 9, no. 3, p. 35, 2018.
[45] A. Theissler, "Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection," *Knowl-Based Syst*, vol. 123, pp. 163 – 173, 2017.
[46] D. Azzalini, L. Bonali, and F. Amigoni, "A minimally supervised approach based on variational autoencoders for anomaly detection in autonomous robots," *IEEE Robotics Autom. Lett.*, vol. 6, no. 2, pp. 2985–2992, 2021.
[47] Z. Yang, I. S. Bozchalooi, and E. Darve, "Regularized cycle consistent generative adversarial network for anomaly detection," in *Proc. ECAI*, 2020.
[48] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly detection based on convolutional recurrent autoencoder for IoT time series," *IEEE Trans Syst Man Cybern Syst*, vol. 1, pp. 1–11, 2020.
[49] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. D. Yao, "Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities," arXiv:2003.13213, 2020.
[50] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *Proc. ICANN*, 2019, pp. 703–716.
[51] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in *Proc. IJCAI*, 2019, pp. 2725–2732.
[52] Y. H. Yoo, U. H. Kim, and J. H. Kim, "Recurrent reconstructive network for sequential anomaly detection," *IEEE Trans Cybern*, vol. 1, pp. 1–12, 2019.
[53] D. Park, Z. Erickson, T. Bhattacharjee, and C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *Proc. ICRA*, 2016, pp. 407–414.
[54] D. Park, H. Kim, and C. Kemp, "Multimodal anomaly detection for assistive robots," *Auton Robot*, vol. 43, no. 3, pp. 611–629, 2019.
[55] G. Safont, A. Salazar, L. Vergara, E. Gómez, and V. Villanueva, "Probabilistic distance for mixtures of independent component analyzers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 1161–1173, 2018.
[56] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
[57] M. Schnall-Levin, L. Chindelevitch, and B. Berger, "Inverting the Viterbi algorithm: An abstract framework for structure design," in *Proc. ICML*, 2008, p. 904–911.
[58] L. Baum and J. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology," *Bull Amer Math Soc*, vol. 73, no. 3, pp. 360–363, 1967.

Alberto Castellini Alberto Castellini is assistant professor at the University of Verona, Dept. of Computer Science, since 2018. Before he was research fellow at Potsdam University/Max Planck Institute (Germany) and University of Verona. His research interests include predictive modeling, planning under uncertainty, reinforcement learning and data analysis for intelligent systems (robotic/cyber-physical systems). He published in artificial intelligence journals and conferences (IEEE Int. Syst., Eng. App. of Artificial Intelligence, Rob. Aut. Systems, IJCAI, AAMAS, ICAPS, IROS).

Francesco Masillo is a PhD student at University of Verona, Dept. of Computer Science, since 2021. Since Bachelor's Degree, he has been working in the field of machine learning and artificial intelligence with robotic and cyber-physical applications focusing on clustering, predictive models and anomaly detection. He has published in international scientific journals and conferences on artificial intelligence (e.g., Eng. App. of Artificial Intelligence, Cybernetics and Systems, SAC, AAMAS).

Davide Azzalini is PhD student at Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria. His research focuses on anomaly detection and fault detection both on autonomous robots and on time series in general. He has published in international scientific journals and conferences on artificial intelligence and robotics (e.g., AAMAS, IEEE ICRA, IEEE RA-L).

Francesco Amigoni got the Laurea degree in Computer Engineering from the Politecnico di Milano in 1996 and the PhD degree in Computer Engineering and Automatica from the Politecnico di Milano in 2000. From December 1999 to September 2000 he has been a visiting scholar at the Computer Science Department of the Stanford University (USA). He currently is full professor at the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano, where he has previously been assistant professor and then associate professor. His main research interests include: agents and multiagent systems and autonomous mobile robotics.

Alessandro Farinelli is a full professor at University of Verona, Dept. of Computer Science, since December 2019. His research interests comprise theoretical and practical issues related to the development of Artificial Intelligent Systems applied to robotics. In particular, he focuses on statistical data analysis, decentralised optimisation and reinforcement learning for Multi-Agent and Multi-Robot systems. He was principal investigator for several national and international research projects in the broad area of Artificial Intelligence for robotic systems. He co-authored more than 100 peer-reviewed scientific contributions in top international journals and conferences.