POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

DETECTING ANOMALIES IN THE BEHAVIOR OF AUTONOMOUS ROBOTS

Doctoral Dissertation of:
Davide Azzalini

Supervisor:
Prof. Francesco Amigoni

Tutor:
Prof. Luciano Baresi

The Chair of the Doctoral Program:
Prof. Luigi Piroddi

2022 – 34th Cycle

To my family

Acknowledgements

FIRST and foremost, I would like to express my deepest gratitude to Prof. Francesco Amigoni, for giving me the opportunity to pursue a PhD, for being such a good advisor and an inspiring researcher, for helping my professional growth, and for being an example to look at. Thank you also for all the late nights spent revising and submitting papers, thank you for teaching me how to (properly) write, thank you for always being there, and thank you for all the countless and endless exciting conversations about the more disparate topics.

A big thank you goes also to Prof. Letizia Tanca for being the first to believe in me, without you there would be no PhD. Thank you also for all the chats, the trips, and, most importantly, for being a friend.

I sincerely thank ABB S.p.A., for having funded my research, and, in particular, I want to express my gratitude to Prof. Enrico Ragaini, for the many collaborations over these years and for always finding new data for me to analyze. Thank you also to Giulia, Lorenzo, Max, Silvio, and to the other ABB PhD students, Marco and Edoardo.

Thank you to all my coauthors, Matteo, Alessandro, Alberto, Francesco, Elisa, Mirjana and Emanuele, it has been an honor and an immense pleasure to work with you.

Thank you to my fellow PoliTong-ers, that has surely been and always will be one of the best periods of my life.

Thank you to my colleagues-friends: Paul, Lucy, Angela, Rab, Barba, Emilia, Chiara, Luna, Tommaso. I am not sure wheter the secret at the base of our friendship has been not having ever worked together, in any case, it certainly worked! I hold each one of you very dear to my heart.

A big thank you to Benedetta, my new PhD sister. The thing that I certainly missed and suffered the most during these three years has been not having someone to work with every day on the same things and in close contact. Even though when you arrived my PhD was almost over, it has been amazing working with you.

Thank you to my Technology for Information Systems students for always being curious, involved, interested and (almost too) polite. Thank you also for being more and more every year, I will pretend it is because of the fascinating contents and the exceptional quality of the teaching, rather than my way too generous grading of your exams.

Thank you to all my master thesis students, you are too many to mention individually. Thank you for trusting me even at times I would not even trust myself. I have fond memories of each one of you.

Thank you to all the students who have been part of the TIS Lab (and, in general, the south wing of the third floor of building 21), thank you for all the lunches, the coffee breaks, and the nights out together.

I thank my family for the immense and unconditional love they have always given me and for their never-ending support, even when I was always away and busy, trying to cope with PhD duties. Thank you also for your countless requests for IT support, which have taught me that there is no too humble task for a PhD student. From now on you will (hopefully) have an (actual) Doctor addressing your *very basic* IT issues. You're welcome.

And finally, thank you A., you know who you are. Thank you for seven years of (mostly) great memories. But especially, thank you for making that decision I never could have made, now I understand it was for the best.

Milano, April 2022

Davide Azzalini

Abstract

DETECTION of anomalies and faults is a key element for long-term robot autonomy, because, together with subsequent diagnosis and recovery, it allows to reach the required levels of robustness and persistency. A fault which is not promptly detected and addressed, in fact, may result in the robot damaging itself or, even worse, in harming surrounding people. In this thesis, multiple approaches for detecting anomalous behaviors in autonomous robots starting from data collected during their routine operations are proposed. The main idea is to model the nominal (expected) behavior of a robot and to evaluate how far the observed behavior is from the nominal one. The first approach we propose uses Hidden Markov Models (HMMs) to learn the robot's behavior under normal circumstances and detects anomalies by computing variants of the Hellinger distance between the distribution of observations made in a sliding window and the corresponding nominal emission probability distribution (online anomaly detection), or between two HMMs representing nominal and observed behaviors (offline anomaly detection). We then present a data augmentation and retraining technique based on adversarial learning for improving anomaly detection performance of our HMM-based approach when few nominal examples are available. In particular, we first define a methodology for generating adversarial examples for anomaly detectors based on HMMs; then, we present a data augmentation and retraining technique using these adversarial examples to improve anomaly detection performance and robustness to adversarial attacks. The second approach we introduce is a new deep learning-based minimally supervised method which

employs a new Variational Auto-Encoder (VAE) architecture and a new incremental training method that, unlike most existing approaches, requires only very few labeled nominal executions to be trained. Also in this case we present both an online and an offline technique. We then propose an adaptation of the VAE-based approach to allow individual robots in a multi-robot systems to detect anomalies in one another. Particular attention is devoted to ensuring that the proposed methods can be easily applicable in different practical settings. Accordingly, all the approaches proposed in this thesis are designed not to make any limiting assumption on how anomalies look like and to work with small amounts of (labeled) training examples. We show how the methods proposed in this thesis positively compare against state-of-the-art anomaly detectors commonly used in robotics in a variety of application domains involving different robotic platforms required to operate for long periods of time without interruption.

Summary

Il rilevamento di anomalie e guasti e' un elemento chiave per fare in modo che i sistemi robotici possano operare in autonomia per lunghi periodi di tempo, in quanto, insieme alla successiva diagnosi (capire la natura dell'anomalia) e al ripristino ad uno stato pre-anomalo, cio' consente di raggiungere i livelli di robustezza e persistenza desiderati. Un guasto che non viene tempestivamente rilevato e risolto, infatti, puo' comportare il danneggiamento del robot o, peggio ancora, minacciare la sicurezza di eventuali persone circostanti. In questa tesi vengono proposti diversi approcci per rilevare comportamenti anomali in robot autonomi a partire dai dati raccolti durante le loro operazioni di routine. L'idea principale e' quella di modellare il comportamento nominale (atteso) di un robot e poi valutare quanto il comportamento osservato a runtime sia lontano da quello appreso. Il primo approccio proposto utilizza gli Hidden Markov Model (HMM) per apprendere il comportamento del robot in circostanze di funzionamento normale e rileva eventuali anomalie calcolando una variante della distanza di Hellinger tra la distribuzione delle osservazioni fatte in una finestra scorrevole e la corrispondente distribuzione di probabilita' di emissione dell'HMM (anomaly detection online) o tra due HMM che rappresentano i comportamenti nominali e osservati (anomaly detection offline). Viene poi presentata una tecnica di data augmentation e riaddestramento basata sul concetto di adversarial learning che ha il fine di migliorare le prestazioni di rilevamento delle anomalie dell'approccio basato su HMM in condizioni di scarsita' di dati nominali. In particolare, viene definita una metodologia per la generazione di esempi adversarial apposi-

tamente per algoritmi di anomaly detection basati su HMM. Viene inoltre introdotta una procedura per utilizzare tali esempi adversarial per il riaddestramento dell'HMM e mostrato come cio' conduca a un miglioramento nelle prestazioni di rilevamento delle anomalie. Il secondo approccio che che viene introdotto e' un nuovo metodo di deep learning a "supervisione minima" che impiega una nuova architettura di Variational Auto-Encoder (VAE) e un nuovo metodo di addestramento incrementale che, a differenza della maggior parte degli approcci esistenti, richiede una quantita' minima di esecuzioni nominali durante l'addestramento. Anche in questo caso, viene proposta sia una tecnica online che una offline per il rilevamento delle anomalie. Come ulteriore contributo, viene presentato un adattamento dell'approccio basato su VAE per consentire a singoli robot in un sistema multi-robot di tipo swarm di rilevare anomalie l'uno nell'altro. Particolare attenzione e' dedicata a garantire che i metodi proposti possano essere facilmente applicabili in diversi contesti pratici. Di conseguenza, tutti gli approcci proposti in questa tesi sono progettati per non fare alcuna ipotesi limitante sull'aspetto delle anomalie considerate e sono predisposti per funzionare con piccole quantita' di esempi nominali. Attraverso un'estesa serie di esperimenti, viene mostrato come i metodi proposti in questa tesi siano competitivi rispetto ad altri metodi per l'anomaly detection in sistemi robotici. Tali esperimenti coinvolgono molteplici scenari in cui e' richiesto che i robot dislocati siano in grado di operare per lunghi periodi di tempo senza interruzioni.

Contents

CHAPTER 1

Introduction

1.1 Overview

Autonomous robots are increasingly becoming part of human everyday life. Recent reports[1] by the International Federation of Robotics describe sales value of professional service robots increased by 37% to 11.2 billion USD worldwide in 2019. For the same year, sales value of personal and domestic service robots (vacuuming and floor cleaning robots, lawn-mowing robots or entertainment robots) has increased up 20% to 5.7 billion USD worldwide. A similar increase has been registered also for assistance robots for elderly or disabled persons, for which the estimated sales value increased by 17% to 91 million USD in 2019. The insurgence of the COVID-19 pandemic has since further boosted the market causing, in addition to the already existing surge, a high demand for robotic disinfection solutions, robotic logistics solutions in factories and warehouses, and robots for home delivery.

From driverless cars to assistive robots for elderly people, these systems are leaving the factories and entering unconstrained scenarios with close interaction with humans. Complex and dynamic environments are

[1] https://ifr.org/ifr-press-releases/news/service-robots-record-sales-worldwide-up-32

characterized by large degrees of uncertainty and pose big challenges to robot designers. One of the key competences required to newly conceived robots is to reliably operate over long periods of time under changing and unpredictable environmental conditions, which is referred to as *long-term autonomy* (LTA) [95].

Exhibiting LTA means that robots are persistent, robust, and able to adapt to changes in their operational environments. As these sophisticated machines are susceptible to different types of faults, *fault detection* (identifying when a fault has occurred), *fault diagnosis* (pinpointing the type of fault and its location) and *recovery* (enacting the right actions required to revert to a non-anomalous state) approaches are a fundamental ingredient of LTA in order to promptly identify anomalies and recover a robot system in time for continuing its operations.

In this thesis we focus on fault detection and propose new methods that learn a model of the nominal (expected) behavior of a robot from data (i.e., sensor logs) and detect faults by evaluating how far the observed behavior is from the nominal one.

Fault detection can be performed at different levels of abstraction, in our case we are interested in identifying anomalies not at the component-level, but in the *overall behavior* of a system. Although component-level fault detection may be suitable when industrial robots (e.g. robotic arms) are considered, as they operate in controlled and very predictable environments, when we consider autonomous mobile robots deployed in complex and dynamic environments, fault detection can no longer be just the identification of which one of the n components constituting a robot stopped working properly. In fact, an anomalous behavior may be the consequence not only of a component's fault, but also the result of the robot's inability to effectively interact with the environment, people nearby, or even other robots. In this thesis we are not interested in detecting anomalies or faults just on the individual components of a robot, but rather in discovering anomalous behaviors in a broader sense, regardless of the cause. As a consequence, the term *anomaly* is preferred over the term *fault*, as it is more general and better reflects the broad spectrum of deviating behaviors we are interested to detect. In fact, unexpected behaviors displayed by a robot may happen that do not necessarily encompass failures and are nevertheless interesting to be detected.

Beyond considering a broad range of possible anomalies, one of the most important aspects of anomaly detection in robotic systems that we address in this thesis is the practical applicability of such methods. A key enabler of learning-based anomaly detection systems in real world set-

tings is undoubtedly the design of techniques that do not require prohibitive amounts of training data to be employed. In this thesis, a lot of attention is devoted to developing methods suitable to be operated also in cases in which data are scarce. All the approaches proposed in this thesis are in fact designed to work with small amounts of (labeled) training examples. Even when we consider deep learning-based approaches, which are notoriously data-hungry, we devise training solutions that minimize the need of labeled examples.

1.2 Contributions

The contributions of this thesis revolve around the development of new approaches for the detection of anomalous behaviors displayed by robotic systems.

A new HMM-based Anomaly Detector

We propose an approach for detecting anomalous behaviors in autonomous robots starting from data collected during their routine operations. The main idea is to model the nominal (expected) behavior of a robot system using Hidden Markov Models (HMMs) and to evaluate how far the observed behavior is from the nominal one using variants of the Hellinger distance adopted for our purposes. We present a method for online anomaly detection that computes the Hellinger distance between the probability distribution of observations made in a sliding window and the corresponding nominal emission probability distribution. We also present a method for offline anomaly detection that computes a variant of the Hellinger distance between two HMMs representing nominal and observed behaviors. The use of the Hellinger distance positively impacts on both detection performance and interpretability of detected anomalies, as shown by results of experiments performed in two real-world application domains. This contribution has been published in [8].

A new Data Augmentation Procedure for HMM-based Anomaly Detection

We also present a data augmentation and retraining technique based on adversarial learning for the HMM-based method just mentioned. In particular, we first define a methodology for generating adversarial examples, then, we present a data augmentation and retraining technique which uses these adversarial examples to improve the anomaly detection performance of the HMM-based method. We evaluate the adversarial data augmenta-

tion and retraining technique on several datasets showing that our method achieves statistically significant performance improvements and enhances the robustness to adversarial attacks.

A new VAE-based Anomaly Detector

We contribute a new Variational AutoEncoder (VAE) architecture able to model very long multivariate sensor logs exploiting a new incremental training method, which induces a progress-based latent space that can be used to detect anomalies both at runtime and offline. While most existing approaches are trained in a semi-supervised fashion and require big batches of nominal observations, our method is trained using unlabeled observations of a robot performing a task, containing both nominal and anomalous executions. Only a very little amount (even just one) of labeled nominal executions is then required to partition the learned latent space into nominal and anomalous regions. Experimental results show that our method outperforms state-of-the-art anomaly detectors commonly used in robotics both in terms of false positive rate and alert delay. This contribution has been published in [7].

A new Anomaly Detector for Robot Swarms

We also show that a variant of the VAE-based method can be employed to detect anomalies in swarms of robots. We adapt our method for being used by robots in a swarm to detect anomalies in one another. We perform experiments considering four different swarm tasks and show how the detection performance of our method is on par with other state of the art anomaly detectors for swarm systems.

1.3 Applications

We apply the proposed methods in several application domains, in which autonomous mobile robots perform tasks requiring LTA. Specifically, we consider:

- An Autonomous Surface Vessel (ASV) performing a water monitoring task. The water drone is called Platypus and has been developed in the context of the INTCATCH Project [28–30], which aims at developing new paradigms for water monitoring in rivers and lakes by harmonizing a range of innovative tools into a single efficient and user-friendly model.

- An assistive robot called Giraff-X, developed as part of the MOVECARE Project [106], for supporting the independence of elderly people living alone at home. This socially assistive autonomous mobile robot moves in domestic environments, which represent a typical context for LTA. The goal of the robot is to provide notifications to the elder and interact with him/her for stimulation by suggesting activities that aim to counteract physical and cognitive decline, as well as isolation.

- A human-sized robot called SCITOS-G5 performing a patrolling task in a small office as part of the STRANDS Project [66].

- A fixed-wing Unmanned Aerial Vehicle (UAV) performing autonomous flight [83].

- A robotic swarm performing four different swarm behaviors which we simulate using the ARGoS simulator [134] and e-puck robots [113]. The behaviors we consider are: dispersion, aggregation, homing, and flocking.

Please note the experiments performed in this thesis have not been implemented on the actual robotic platforms just described, but on datasets produced by them.

Additionally, we consider two other applications which, unlike the ones just introduced, are outside of the robotics domain. We consider them as they are well-known benchmarks for anomaly detection in real systems:

- The Tennessee-Eastman industrial chemical process [97, 99].

- The Secure Water Treatment (SWaT) testbed [51, 110], a scaled version of an industrial water treatment plant.

1.4 Organization of the Thesis

We organize the remainder of this thesis as follows. We present the related work relevant to anomaly detection in robotics in Chapter 2. After a general introduction to anomaly detection in general, a quick overview of anomaly detection on time series is presented. The core of the chapter is the analysis of anomaly detection techniques applied in the robotics domain. We start by presenting methods designed for single-robot systems and classify them according to several dimensions of analysis. We then shift our focus to multi-robot and swarm robotics approaches.

Chapter 3 contains the presentation of a novel method for detecting anomalous behaviors of robotic systems both online (i.e., while robots are operating) and offline. HMMs are used to model the behavior of robots, while, the Hellinger distance [67] (a distance between probability distributions) is used to compute the dissimilarity between the observed behavior and the nominal one. We show the advantage (i.e., more informative and easier to threshold) of using such a distance measure w.r.t. other standard measures by performing experiments on two robot systems operating in real-world settings requiring LTA.

Then, Chapter 4 presents a data augmentation procedure for augmenting the training dataset of the HMM-based anomaly detector presented in Chapter 3. The proposed data augmentation is based on the concept of adversarial attacks, i.e., examples which, although being very similar (indistinguishable) to nominal ones, are misclassified (i.e., in our case, detected as anomalous). If explicitly included in the training set of nominal examples, these adversarial samples generally lead to a performance improvement, especially if the training dataset is small. After reviewing the relevant literature on adversarial attacks and data augmentation, we propose an algorithm able to generate adversarial examples for an anomaly detector based on HMMs and working with multivariate time series. We propose an algorithm for data augmentation and retraining based on adversarial examples and show how our technique improves the performance of the anomaly detector on four different datasets (two of which involving robot systems operating in real-world settings requiring LTA).

In Chapter 5 we present a novel learning paradigm, which we call *minimal supervision*. In particular, after introducing a new VAE architecture able to model very long multivariate sensor logs of a robot performing a task, and a new incremental method for training VAEs, we show how, differently from most approaches for anomaly detection in robotics, it is possible to train our VAE with unlabeled observations, then only few (even just one) labeled nominal executions are required to partition the learned latent space into nominal and anomalous regions. Both online and offline procedures are presented and tested by performing experiments on three robot systems operating in real-world settings requiring LTA. Lastly, we show the influence that different learning approaches have on the shape of the learned VAE latent space and the consequences that this has on the detectability of anomalies.

Chapter 6 extends Chapter 5 by presenting and adaptation of the proposed VAE architecture to detect anomalies in swarms of robots. An extensive experimental campaign is performed by testing different combinations

of faults and tasks.

Finally, Chapter 7 summarizes this work, its main results, and proposes some directions of future work.

CHAPTER 2

Related Work

In this chapter, after an initial general introduction to anomaly detection, a comprehensive analysis of anomaly detection approaches applied to single- and multi-robot systems is presented. Anomaly detection methods not applied specifically to the robotic domain are intentionally left out for the sake of providing a clear and concise picture of the state of the art of anomaly detection in robotic systems. Moreover, being anomaly detection a very active research field with contributions from several areas, they would simply be too many. Obviously, some of the approaches available in the literature and outside of the domain of robotics are still relevant to this thesis as they are based on some of the principles and tools (algorithms) exploited also by the methods proposed in this thesis. These related works (e.g., HMM-based and autoencoder-based anomaly detectors) will be discussed in the next chapters and thoroughly compared to the ones proposed in this thesis.

2.1 An Introduction to Anomaly Detection

Anomaly detection [19, 32, 33, 122, 144] is the task of finding patterns in data that do not conform to the expected behavior. In the scientific literature, these unorthodox patterns are referred to in different ways, among

which *anomalies* and *outliers* are certainly the most common ones; less frequent alternatives are: *discordant observations*, *exceptions*, *aberrations*, *surprises*, *peculiarities*, *contaminants* or *faults* depending on the specific application domain.

Anomaly detection has been a field of interest both within academia and industry for a very long time, suffice to say that the first works in this direction date back to as early as the late XIX Century [45]. The importance of anomaly detection is rooted in the fact that detected anomalies can lead to significant and often critical insight that can be exploited in a wide variety of application areas. The impact that this actionable information provided by anomaly detection methods can have is often quite substantial, ranging from cost savings (e.g., when energy waste is detected from electricity consumption time series [23]) up to even saving lives (e.g., when cancer is early detected form medical imaging [47]). The application domains in which anomaly detection has been applied to over the years are very varied and include credit card or insurance fraud [137], healthcare [47], cyber-physical systems [52], surveillance [118], intrusion detection in computer networks [3] and many others [19, 32, 33, 122, 144]. Some methods have also been developed specifically for the robotic domain [85] (more on this in Sections 2.2 and 2.3).

From a technological standpoint, many different methods and algorithms have been proposed to address anomaly detection tasks. Depending on the data modality on which they work, different approaches need generally to be adopted, examples of modalities are: structured i.i.d. tabular data, videos, images, text, graphs, etc. Some anomaly detectors that work simultaneously on multiple modalities have been proposed [128], these generally extract meaningful features from each modality independently and then fuse them together to perform anomaly detection as a downstream task. Although these multimodal methods have a big potential in robotics (as a robot generally senses and collects multimodal information), the modality which is of most interest to this thesis is temporal data, also referred to as *time series*, as the focus is on detecting anomalies in the robot itself (by analyzing its sensor streams) rather than in its surroundings. It should also be noted that the majority of anomaly detectors in robotics rely on time series data.

Time Series Anomaly Detection

A *time series* consists in a sequence of observations that have been recorded in an orderly fashion and which are correlated in time [59]. Approaches to detect anomalies in time series can be broadly categorized according to:

(i) the nature of the input data and; *(ii)* the type of treatment that is done to such data. The former refers to the type of input data that the detection method is able to deal with (i.e., a univariate or a multivariate time series) [19]. *Univariate* time series are ordered sequences of real-valued observations, while *multivariate* time series can be thought as ordered sets of k-dimensional vectors, where k is the number of observations available at each timestamp. Regarding the kind of processing that can be applied to the input, univariate or multivariate detection methods can be distinguished. A univariate detection method considers only a single time-dependent dimension, whereas a multivariate detection method is able to deal with more than one dimension at the same time. Note that a detection method may be univariate even if the input series is multivariate, since an individual analysis could be performed separately on each dimension without considering the dependencies that may exist among them. It should also be noted how multivariate approaches represent a more powerful tool as they can model also the fact that each variable could depend not only on its past values but also on the other variables (both at the current time instant as well as in the past). In this thesis we are mainly interested in multivariate approaches as generally a robot has access to multiple proprioceptor sensor streams. Among the fully multivariate methods, HMMs [136] and deep learning-based methods [32, 122, 144] (especially autoencoders [70, 90] paired with recurrent architectures such as LTSMs [72] and GRUs [40]) are the most common choices.

2.2 Anomaly Detection in Single-Robot Systems

When it comes to the domain of robotics, and real systems in general, anomalies are more commonly referred to as faults [85]. However, as said in the introductory chapter, in this thesis the term anomaly is preferred as it is more general and inclusive as it better reflects the broad range of deviating behaviors we are interested to detect. In fact, uncommon behaviors displayed by a robot may happen that do not necessarily encompass failures and are nevertheless interesting to be detected.

While the field of anomaly detection has been very active for more than a century, the study of anomaly detection techniques specifically tailored to autonomous robotics is relatively new [85, 133, 172].

Robots are complex systems consisting of physical (hardware) and virtual (software) components capable of varying degrees of autonomy that operate in diverse and dynamic physical environments, some examples are industrial manipulators, warehouse AMRs, Mars rovers, satellites, and un-

manned aerial and underwater vehicles. Unfortunately, like any physical system, these sophisticated and often expensive machines are susceptible to various types of failures [151]. It is important to detect and react to these faults as they have the potential to affect the robot's efficiency, cause failures, or even jeopardize the safety of the robot or its surroundings [42].

Model-based vs. Knowledge-based vs. Data-driven

Anomaly detectors applied to robots, and real systems in general, can be divided into three main categories depending on the kind of representation medium used: *model-based*, *knowledge-based*, and *data-driven* [85].

Model-based approaches [75] require explicit analytical models (i.e., mathematical equations or logic formulas) of robotic components and therefore need expert knowledge to be built. Domain experts are generally expected to inject vast amounts of domain knowledge in these models as the nominal behavior of each component in the robotic system and the interaction between them need to be specified and modeled analytically. Once the correct behavior of each component has been modeled, these approaches detect anomalies by comparing the expected output to the observed one [140]. The biggest barrier to the adoption of model-based methods is generally the cost deriving from needing domain experts for constructing such analytical models [133], which, when the robotic domain is considered, is exacerbated even more as these models need to take into account the dynamic context of the robots, i.e., the environment and the task at hand, as well as the robot's complexity. Examples of model-based approaches in robotics are [1] and [73]. Some attention to the topic of detecting anomalies in robotic systems has also been devoted by the formal methods community. Formal verification is the application of formal methods to the verification of systems. When used in robotics, formal verification methods tend to be less fit w.r.t. when used for checking traditional software systems, in fact, robotic systems are generally not developed with verification in mind, which can often complicate the later employment of formal verification on such algorithms and systems. Robot controller software, unlike traditional software programs, usually consist of several interacting modules that can be grouped into two categories: high-level modules, taking discrete decisions and planning to achieves complex tasks, and low-level modules, usually governing controllers and actuators. Using exhaustive techniques, such as model checking (i.e., providing a proof of correctness that the software obeys its requirements), on such software in a monolithic way is generally impossible due to the intractable state space. Non-exhaustive techniques, such as runtime verification, are generally bet-

ter alternatives to model checking in robotics. Runtime verification is the area of formal methods that studies the dynamic analysis of execution traces against formal specifications. Typically, the two main activities in runtime verification efforts are the process of creating monitors from specifications, and the algorithms for the evaluation of traces against the generated monitors. Despite the use of formal verification methods in robotics is still limited, some notable example exist [41, 50, 74, 105, 146, 162].

Knowledge-based approaches typically associate each known fault to a detection rule which is triggered when the specific behavior is observed. They idea behind knowledge-based approaches is to mimic the behavior of a human expert by associating symptoms with diagnoses. The obvious drawback is that this kind of approaches are not suitable for the detection of previously unknown anomalies. The two most common families of methods belonging to this category are causal models [37] and expert systems [129].

Data-driven approaches are instead based on (usually probabilistic) descriptions of behaviors or faults that are automatically learnt from previous observations of the system. Their advantage is that they do not need any explicit prior knowledge of the system and of the faults. Online data-driven methods are mostly used for autonomous robots, which compare the system's behavior in real-time, generally in the form of sensor streams, to previously-learnt probabilistic representations to statistically differentiate potential faults from normal behavior. Some approaches use statistical filtering such as Kalman and particle filters [2, 44, 158]. Some works (e.g., [38]) propose supervised machine learning approaches [18] and fault injection to classify data produced in real-time by a robot. In [81, 87], the authors introduce an online multivariate data-driven fault detection approach which uses the Mahalanobis-distance to compare correlated streams of data with previously observed data. In [53], a self-awareness approach is proposed which builds a probabilistic model on the basis of the whole discrete event-based data interchange inside the robot. In [65], self-organizing maps and probabilistic graphical models are used. In [92] the authors try to explicitly model environmental dynamics in the context of localization and navigation with the aim of long-term mobile robot autonomy by means of a spectral model which allows to represent the probability of observing a given environment state by combination of harmonic functions whose parameters relate to that of the hidden processes that cause the environment variations. Other works (e.g., [126] and [128]) train HMMs using multimodal sensory signals for detecting anomalies in assistive robots. Recently, deep learning models [54, 101] have been used to re-address many spatio-

temporal modeling tasks providing improvements over the state-of-the-art methods. Recent examples of works adopting deep neural networks for online anomaly detection in robot systems are [119, 127, 150], which employ various kinds of autoencoders.

Supervised vs. Unsupervised vs. Semi-Supervised Learning

Another dimension of analysis, mainly pertaining data-driven approaches, of interest is the kind of supervision which these methods need during training. This is something very important to analyze as it is has a considerable impact on the practical applicability of the resulting approaches. Depending on the extent to which labeled instances are available, anomaly detection techniques can operate in either *supervised*, *unsupervised* or *semi-supervised* mode [33].

Supervised methods (e.g., [38]) need fully labeled data for training (i.e., the training dataset must consist of examples all of which labeled as either nominal or anomalous) and generally build predictive models for normal vs. anomaly classes. It should be noted that obtaining labeled data that is accurate as well as representative of all types of behaviors is often prohibitively expensive, in fact, labeling is often done manually by a human expert and hence substantial effort is required to obtain the labeled training dataset. Moreover, there exist domains in which obtaining a sufficient amount of anomalously labeled training example is just not possible. Think for example of safety critical systems or satellites, which are designed to ensure exceptional levels of availability and fault tolerance, for these systems, even if we were in possession of sensor logs and were willing to face the economic burden of having a human label them, it would be highly likely that such logs would not contain anomalous examples at all (as they would encompass catastrophic events), which would make the training set unsuitable for supervised anomaly detection as both nominal and anomalous samples are required. Another related and well-known issue of supervised methods for anomaly detection is that, even when labeled anomalous examples are present in the training dataset, these are likely to be far fewer compared to the normal instances. This results in an imbalanced dataset, which, in turn, makes training harder (although several methods have ben proposed to tackle the problem of imbalanced class distributions [34, 102, 139], this is a problem still not yet fully overcome). One of the advantages of supervised methods is that, provided that separate labels are available for different anomaly types, they can simultaneously perform anomaly detection and diagnosis (i.e., they can recognize that something anomalous is happening and also the nature of it). Unfortunately, this comes at the cost of

being able to only model anomalies similar to those already in the training set as these models assume to already know all possible kinds (i.e., classes) of anomalies that will ever occur with the consequence of not being able to detect previously unseen anomaly types. To overcome issues related to scarcity of anomalous examples, a number of techniques have been proposed that inject artificial anomalies into a normal dataset [38,78], however, it is not always straightforward to inject realistic faults and the incapability to recognize previously unknown anomaly types still persist. Besides these issues, supervised anomaly approaches, when applicable, deliver good results as the anomaly detection problem reduces to building standard predictive models [18], for which a great body of research is available. For all of these reasons, recent developments in robotics tend to shift towards unsupervised and especially semi-supervised learning paradigms.

Unsupervised methods (e.g., [87]) do not require a labeled training set and are thus most widely applicable. These methods make the implicit assumption that anomalous instances are far more rarely occurring than nominal ones in test data. Another limiting assumption of unsupervised approaches is that anomalous examples are sufficiently different from nominal ones. If these assumptions are not true then the detection performance of such techniques quickly deteriorates due to high false alarm rates.

Semi-supervised methods (e.g., [8, 126–128]) are those that require labels for the nominal class only (which makes them more widely applicable than supervised techniques) and are motivated by the fact that, generally, obtaining a labeled set of anomalous data instances that covers all possible type of anomalous behavior is more difficult than getting labels for normal behavior. In needing only nominal samples to be trained, semi-supervised approaches take the best from supervised and unsupervised approaches while retaining very few of their downsides. Since they learn a model for the nominal behavior only, they *(i)* do not make any assumption on how anomalies look like and are hence capable of detecting also previously unseen anomalies; *(ii)* are able to relax the assumption on the rarity of anomalies of unsupervised methods. It is worth noting that most semi-supervised techniques can be adapted to operate in unsupervised mode by using an unlabeled training set (i.e., it may contain both nominal and anomalous examples without any label). Such adaptation, however, assumes (as for natively unsupervised methods), that the test data contains very few anomalies and that the model learned during training is robust to these few anomalies.

2.3 Anomaly Detection in Multi-Robot Systems

A multi-robot system consists of multiple robots working together as a unique system by interacting among themselves in order to achieve a common goal within a physical environment. In a way, a multi-robot system can be viewed as a unique distributed robotic system that senses, thinks, and acts, just like a single-robot system, however, each one of these processes becomes distributed and significantly more complex in multi-robot systems than in a single robotic system.

In general, in a multi-robot system, each robot senses its surroundings and forms an individual local belief. Local beliefs are then (partially) transmitted to the global scope that processes them to form a global belief (global belief generation). A global representation of knowledge is what allows the multi-robot system to make intelligent decisions, such as global planning. After a global plan is selected, individual tasks are allocated (via communication) to each robot. Then, each robot applies local planning to select the steps necessary to successfully complete its assigned task and checks that those actions are not in conflict with those selected by the other robots. Once an agreement is reached, each individual robot executes its local plan. As robots perform their actions, they inevitably affect the environment, which is again sensed and the process repeats. In a multi-robot system, in addition to the same faults that could affect a single robot, each of these local and global activities might be subject to different faults, and, in turn, could disrupt the entire system.

Very few works have been proposed that tackle the detection of anomalies and faults specifically in multi-robot systems, a survey can be found in [86] . The purpose of these works is however rather distant from the one of this thesis as they propose ad hoc techniques to detect coordination and planning-related faults, while we are interested in agnostically observing a system's sensor streams and detect anomalies of whichever type.

For these reasons, in this thesis, we restrict our focus to a subfield of multi-robot systems, namely, swarm robotics, where global belief generation, global planning, task allocation and task coordination are generally not performed.

2.3.1 Anomaly Detection in Robotic Swarms

A swarm robotics system [21,58] is a system consisting of multiple intelligent robots that individually possess low intelligence/capabilities (they can process data by themselves but they can only execute simple behaviors) but that through local interconnection (exchange of messages, usually in broad-

cast, with nearby robots) and swarm intelligence [20, 84] are together able to create emergent behaviors that can be rather complex. It is important to stress that these complex emergent behaviors are achieved without any centralized control.

Although, due to the usually large numbers of robots composing a swarm, these systems are considered to be inherently fault-tolerant, it has been proved that a swarm robotics system's work can be slowed down or even blocked due to one or more faulty robots [164]. It is hence necessary to implement some mechanism to avoid these situations by early detecting faults and promptly responding to them. Some works have been presented that tackle the problem of anomaly detection specifically on swarms of robots [111].

In addition to the dimensions of analysis already identified for single-robot systems, for swarms it is possible to identify some more.

Endogenous vs. Exogenous Fault Detection

A first distinction concerns the subject of the anomaly detection w.r.t. the entity that performs it. In this regard, three alternatives can be identified: *endogenous*, *exogenous*, and *multi-layered*.

Endogenous anomalous detection (e.g., [98]) enables a robot to detect the presence of anomalies in itself by means of single-robot methods (see Section 2.2). One downside of this paradigm is that catastrophic faults, such as a malfunctioning power source or issues with the onboard computational hardware, usually cannot be detected endogenously as they render the robot completely non-operational.

In multi-robot systems, robots also have the opportunity to detect the presence of faults in one another. This is commonly referred to as exogenous anomaly detection (e.g., [39, 153–156]). Exogenous anomaly detection has the potential to detect any type of fault, including catastrophic faults.

Lastly, multi-layered anomaly detection is when robots are able to exhibit both endogenous anomaly detection and exogenous anomaly detection (e.g., [26]).

Homogeneous vs. Heterogeneous Swarms

Anomaly detection approaches in swarm robotics can also be categorized according to the type of swarm considered: *homogeneous* swarms are those in which all the robots in the system execute the same task [39, 98, 155], while, in *heterogeneous* swarms, different robots executing different tasks

(they may also be different from an hardware perspective) can be found [153, 154, 156].

Centralized vs. Distributed Anomaly Detection

Finally, anomaly detection methods can be categorized according to the architecture employed by the robots to exchange information and according to where the anomaly detection algorithm is run. In *centralized* architectures, anomaly detection is performed by only one node, often external to the swarm [61–63, 88, 89]. *Distributed* architectures are those in which the same algorithm is run on each node of the swarm [39, 98, 153–156].

2.4 Concluding Remarks

We wrap up our analysis of the related works by highlighting the main differences between the approaches proposed in this thesis and the ones described in this chapter.

For what regards the two anomaly detectors for single robots contributed by this thesis, the data-driven paradigm has been adopted for both of them. The rationale behind such choice is the desideratum to be freed from heavily reliance on domain expertise, something which is further reflected by the use of semi-supervised training techniques. We, in fact, consider semi-supervision as the best trade-off between the severe need for labeled training samples of supervised methods (and hence depending yet again on a domain expert), and the often unacceptable shortcomings of fully unsupervised ones. What differentiate our proposed approaches from other data-driven semi-supervied ones is the striving to minimize the need for nominal executions required during the training of such models. Differences between the proposed methods and similar ones from a model perspective will be discussed in the next chapters.

For what regards anomaly detection in robotic swarms, we conform to the majority of the other existing works and present an exogenous anomaly detector for homogeneous swarms by means of a distributed (data-driven) architecture. What differentiate our approach from the others is the automatic feature extraction enabled by the use of our VAE architecture.

CHAPTER 3

An HMM-based Anomaly Detector for Single-Robot Systems

The work contained in this chapter has been published in the proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS) 2020 [8].

3.1 Introduction

In this chapter, a novel method based on *Hidden Markov Models* (HMMs) for detecting anomalies in single-robot systems is presented. HMMs [136] have been successfully used for learning robot behaviors, especially in the context of Learning from Demonstration (LfD) for manipulators and humanoid robots [6, 17]. An HMM is a statistical model in which the system being modeled is assumed to be a Markov process with unobservable (hidden) states, each characterized by an emission distribution governing the probability of producing any of the observable system outputs and a transition distribution indicating which are the likely next states. Because of their robustness to spatiotemporal variations of sequential data, HMMs are also commonly used for encoding and abstracting noisy time series [27, 29, 31].

We advocate that HMMs can provide good representations also of robot behaviors in LTA contexts, where similar sequences of actions (tasks) are typically repeated multiple times. Representing robot behaviors in these domains is still wildly unexplored because of the difficulty to predict the diverse situations in which the robot may have to deal with.

This chapter proposes a new method for detecting anomalous behaviors of robotic systems involved in complex LTA scenarios, both online, while robots are operating, and offline, after robots have completed a run of their tasks. The behavior of robots is modeled using HMMs and, originally, the Hellinger distance [67] is used to compute *(i)* the dissimilarity between the probability distribution of subsequences of observations in a sliding window and the emission probability of related HMM hidden states (online approach) and *(ii)* the distance between pairs of HMMs representing nominal and observed behaviors (offline approach). The advantage of using such a distance measure instead of standard measures (such as the likelihood of observation subsequences for online approaches) is twofold: first, the Hellinger distance is bounded and thus lends itself to simpler interpretation and thresholding; second, it is less noisy and hence more informative and discriminative.

Experiments on two robot systems operating in real-world settings show that the proposed online and offline approaches outperform standard fault detection methods. The online approach allows to discover both trajectory and speed anomalies of aquatic drones performing water monitoring. In the same application, the offline approach significantly discriminates regular and anomalous behaviors observed in different runs of the same task. Anomalous execution traces are also detected in a long-term deployment of a socially assistive mobile robot supporting independence of elderly people living alone at home.

The main original contribution of this work is the novel application of two theoretical tools, HMMs and Hellinger distance, to autonomous robots and LTA. Specifically, we contribute:

- A new *online* anomaly detection algorithm based on HMMs and Hellinger distance.
- A new *offline* anomaly detection algorithm based on a bounded distance between HMMs derived from the Hellinger distance. This distance abstracts the comparison between two behaviors from the level of observations to the level of learned HMMs, providing interpretability and diagnostic capabilities.
- An extensive experimental campaign on real robots involved in two

applications requiring LTA.

3.2 Hidden Markov Models

We use HMMs [136] as a probabilistic model for the system that generated a given multivariate time series $\boldsymbol{O}$. 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 i to state j), and $\boldsymbol{B} = \{b_i(\boldsymbol{o})\}_{i=1}^{K}$ 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 i, respectively. Theory of HMMs provides algorithms to solve three important problems:

- Compute the probability (i.e., likelihood) that an observed (sub)sequence $\boldsymbol{O}$ is represented by an HMM, e.g., using the Forward algorithm [12].
- Find the parameters of an HMM, λ, to maximize the fit (likelihood) to an observed sequence $\boldsymbol{O}$, e.g., using the Baum-Welch algorithm [13].
- Compute the optimal HMM state sequence (known as Viterbi path) that best explains a given observed (sub)sequence $\boldsymbol{O}$, e.g., using the Viterbi algorithm [49].

The optimal number of hidden states and the covariance type can be found by minimizing the Bayesian Information Criterion (BIC), which finds the optimal trade-off between maximizing the likelihood of the training data w.r.t. the model learned and minimizing the number of parameters required (i.e., the number of hidden states) [18].

HMM-Based Fault Detection

HMMs have been often used for anomaly detection. Most of the works in the literature train HMMs with data recorded during non-anomalous executions (i.e., semi-supervised learning) and use one of the following two approaches for detecting anomalies: *(i)* compute the likelihood of current observations and classify them as anomalous if the likelihood is lower than a threshold, *(ii)* compute the probability of the underlying Markov chain and compare it with a fixed threshold [56, 117, 161, 168].

The works that most resemble ours are [126] and [128], 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 anomalies. One of the innovative aspects of the work presented in this chapter is that we substitute the likelihood estimation with the computation of a more informative and interpretable measure, and also provide a new offline methodology for detecting long-term shifts from the nominal behavior. Another similar work, but with a different application focus, is [143], in which anomalies represent credit card frauds and are identified by directly comparing HMMs fit at consecutive periods rather than comparing acceptance probabilities (i.e., likelihoods). Our approach is different in the fact that, instead of just comparing the emission probabilities of the states of two HMMs, we propose a new single-value metric representing the overall dissimilarity between two HMMs.

3.3 The Proposed Method

In this section, our formulation of the problem of detecting anomalies in single-robot systems is defined, the proposed online and offline anomaly detection methods are presented.

3.3.1 Problem Definition

We represent by $\boldsymbol{O} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_n\}$ a d-dimensional time series composed of n observations, where $\boldsymbol{o_t}$ is a d-dimensional vector representing the multivariate (multi-valued) observation at time t. The nominal behavior of a robot system is then represented as $\boldsymbol{O}^N = \{\boldsymbol{o}_1^N, ..., \boldsymbol{o}_{n^N}^N\}$ and the observed behavior of the same system along some time period as $\boldsymbol{O}^O = \{\boldsymbol{o}_1^O, ..., \boldsymbol{o}_{n^O}^O\}$.

If we consider $\boldsymbol{O}^O$ as a (possibly infinite) data stream, *online anomaly detection* at time t is the task of classifying the portion of the stream included in a sliding window (up to t) as anomalous or non-anomalous wrt $\boldsymbol{O}^N$.

Given a finite batch of observations $\boldsymbol{O}^O$, *offline anomaly detection* is the task of classifying the behavior displayed by the system in $\boldsymbol{O}^O$ as anomalous or non-anomalous wrt $\boldsymbol{O}^N$.

We assume the availability of $\boldsymbol{O}^N$ and, for this reason, our approach belongs to the semi-supervised family. This choice is motivated by the fact that, in robotics, the availability of nominal observations for repetitive

tasks, which are the kind of tasks on which we focus, is quite common, since it is often plausible to make ad hoc executions in nominal conditions.

3.3.2 Mathematical Background

The Hellinger distance [67], used in both our online and offline anomaly detection methods described below, is a $[0, 1]$-bounded metric that quantifies the similarity between two probability density functions $f(x)$ and $g(x)$. It is computed as follows:

$$H^2(f, g) = \frac{1}{2} \int \left(\sqrt{f(x)} - \sqrt{g(x)} \right)^2 dx. \tag{3.1}$$

For the case of two multivariate Gaussian distributions $f(x) \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $g(x) \sim \mathcal{N}(\mu_2, \Sigma_2)$, the Hellinger distance can be computed in closed form as:

$$\begin{aligned} H^2(f, g) = 1 - \frac{det(\Sigma_1)^{1/4} det(\Sigma_2)^{1/4}}{det\left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{1/2}} \cdot \\ \cdot exp\left\{ -\frac{1}{8} (\mu_1 + \mu_2)^T \left(\frac{\Sigma_1 + \Sigma_2}{2} \right)^{-1} (\mu_1 - \mu_2) \right\}. \end{aligned} \tag{3.2}$$

3.3.3 Online Anomaly Detection

The nominal behavior of the robot system is modeled as an HMM λ^N that is trained from $\boldsymbol{O}^N$ using the Baum-Welch algorithm. 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 $\boldsymbol{W}_t = \{\boldsymbol{o}^O_{t-w+1}, ..., \boldsymbol{o}^O_t\}$ of length w which includes the last w observations. For each window $\boldsymbol{W}_t$, a score is computed and, when the score exceeds a predefined threshold τ, the behavior 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 $S_t = \{s_{t-w+1}, ..., s_t\}$ of window $\boldsymbol{W}_t$ and the emission probability of the same state in λ^N.

The detailed procedure for online anomaly detection is in Algorithm 1. The algorithm starts by fitting the nominal HMM λ^N with the number of hidden states suggested by the BIC score (lines 1-2). After having specified the desired window length w (line 3) and threshold τ (line 4), the algorithm waits until w observations are collected (line 5) and then starts the online procedure (lines 6-17). The online procedure computes the Viterbi path of

Algorithm 1: Online anomaly detection

```
1  K ← number of hidden states
2  λ^N ← Baum-Welch(O^N, K)
3  w ← window size
4  τ ← threshold
5  t ← w
6  repeat
7      W_t ← {o^O_{t−w+1}, ..., o^O_t}
8      S_t ← Viterbi(λ^N, W_t)
9      ŝ_t ← most frequent state in S_t
10     X ← {o^O_j ∈ W_t : s_j = ŝ_t}
11     μ ← E[X]
12     Σ ← E[(X − μ)(X − μ)^T]
13     if H^2(b^N_{ŝ_t}, N(μ, Σ)) > τ then
14         echo warning
15     end
16     t = t + 1
17 until new data keep coming
```

the multivariate time series inside the window (line 8). For the state $\hat{s}_t$ occurring most frequently in the Viterbi path (line 9) a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is fit through maximum likelihood with the data inside the window (lines 10-12). Then the Hellinger distance is computed (using equation (3.2)) between $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and the emission probability of state $\hat{s}_t$ in λ^N (line 13). If the distance is larger than τ, then a warning is reported (line 14).

3.3.4 Offline Anomaly Detection

Offline anomaly detection is performed by learning two different HMMs, λ^N and λ^O, and computing the distance between them in order to discover if (and how) the behavior of a robot system has changed over time.

To this end, we first need to learn λ^N and λ^O from $\boldsymbol{O}^N$ and $\boldsymbol{O}^O$, respectively, with the Baum-Welch algorithm. We constrain the two models to have the same number of hidden states (i.e., the number of states of λ^N), which is reasonable since we assume the overall behavior of the robot system we model is the same.

Given the model parameters of two HMMs, defining an appropriate similarity measure between the two models is not straightforward. Most of the works in the literature employ the Kullback-Leiber (KL) divergence [94] as a distance measure between two HMMs [80]. Given two probability density functions $f(x)$ and $g(x)$, the KL divergence can be computed as:

$$D_{\mathrm{KL}}(f,g) = \int f(x)\, log\, \frac{f(x)}{g(x)} dx. \tag{3.3}$$

The KL divergence has a closed-form expression for many probability distributions, including Gaussians and, more generally, the exponential family. For more complex distributions, such as mixture models and HMMs, the integral involves the logarithm of sums of component densities, and no simple closed-form expression exists. As a consequence, the KL divergence between HMMs can only be approximated via Monte Carlo sampling [80] or through variational approximation [68]. In this work we are interested in computing the Hellinger distance between HMMs instead of the KL divergence, since, as seen before, it is a bounded measure that can provide interpretability to the anomaly detection model. Furthermore, to the best of our knowledge, no work in the literature has attempted to compute the Hellinger distance between HMMs.

We start the derivation of our Hellinger-based distance between HMMs (for offline anomaly detection) observing that although no closed-form solution exists for the KL divergence between two HMMs, some upper bounds have been proposed which can be computed in closed form. One such bound is proposed by [171] for left-to-right HMMs (a more constrained version of HMMs in which state transitions are allowed only from lower indexed states to higher indexed ones):

$$\begin{aligned} D(\lambda^1,\lambda^2) \leq \sum_{i=1}^{K} \Bigg\{ l_i^1 \Bigg[\overbrace{D_{\mathrm{KL}}\left(b_i^1, b_i^2\right)}^{\substack{\text{contribution}\\ \text{of emission}\\ \text{probabilities}}} + \overbrace{\log\left(\frac{a_{ii}^1}{a_{ii}^2}\right)}^{\substack{\text{contribution}\\ \text{of transition}\\ \text{matrices}}} \Bigg] + \\ + l_i^2 \Bigg[D_{\mathrm{KL}}\left(b_i^2, b_i^1\right) + \log\left(\frac{a_{ii}^2}{a_{ii}^1}\right) \Bigg] \Bigg\}, \end{aligned} \tag{3.4}$$

where $D_{KL}\left(b_i^1, b_i^2\right)$ is the KL divergence between the emission probabilities of state i in the two models and represents how the emission probabilities differ, $\log\left(a_{ii}^1/a_{ii}^2\right)$ is the log-likelihood ratio of the transition probabilities, representing how much the two transition matrices differ, and $l_i = 1/(1 - a_{ii})$ approximates the expected duration of state i. The second half of equation (3.4) makes the distance symmetric.

The problem with equation (3.4) is that all of its components are unbounded, resulting in an overall unbounded measure very difficult to interpret and threshold in practical applications. Moreover, the contribution of the emission probabilities and that of the transition matrices can grow with

different orders of magnitude, making it even more difficult to intuitively interpret the overall distance.

We take inspiration from equation (3.4) maintaining the idea of the two contributions and propose a new bounded (with values in $[0, 1]$) approximation of the distance between two HMMs which is based on the Hellinger distance and on the long-term probabilities of a Markov chain:

$$D(\lambda^1, \lambda^2) \approx \sum_{i=1}^{K} \left\{ l_i^1 \frac{1}{2} \left[\overbrace{H^2\left(b_i^1, b_i^2\right)}^{\substack{\text{contribution of} \\ \text{emission probabilities}}} + \underbrace{\frac{1}{\sqrt{2}} \sqrt{\sum_{j=1}^{K} \left(\sqrt{a_{ij}^1} - \sqrt{a_{ij}^2}\right)^2}}_{\text{contribution of transition matrices}} \right] \right\}, \tag{3.5}$$

where $H^2(b_i^1, b_i^2)$ is the Hellinger distance between the emission probabilities of state i in the two models (i.e., the contribution to the distance of the emission probabilities for state i) and the sum under the square root is the Hellinger distance between the rows of state i in the transition matrices of the two models (i.e., the contribution to the distance of the transition matrices for state i).

We drop the term corresponding to the second half of equation (3.4), which would make the distance symmetric, since we are only interested in how λ^O is dissimilar from λ^N and not vice-versa.

Computing the contribution of the transition matrices as in equation (3.5) instead of as in equation (3.4) has the advantage of taking into account the difference between the transition probabilities to all the states, while the log-likelihood ratio in equation (3.4) considers only the transition probabilities on the main diagonal that correspond to transitions to the same state.

In equation (3.5), l_i^1 is computed as the long term probability of remaining in state i^1 wrt the transition matrix $\boldsymbol{A}^1$. Let $\boldsymbol{A}$ be a regular transition matrix (i.e., such that some power of $\boldsymbol{A}$ has all positive entries) with states $\{1, 2, ..., K\}$, long-term probabilities $\boldsymbol{l} = \{l_1, l_2, ..., l_K\}$ are the unique solution to:

$$\begin{cases} l_j = \sum_{k=1}^{K} l_k a_{kj}, \quad j = 1, 2, ..., K \\ \sum_{i=1}^{K} l_i = 1 \end{cases}$$

Long-term probabilities have two advantages over their approximations used in equation (3.4): *(i)* they are a better proxy of the time spent in each

Algorithm 2: Offline anomaly detection

1 $K \leftarrow$ number of hidden states
2 $\lambda^N \leftarrow$ Baum-Welch($\boldsymbol{O}^N$, K)
3 $\lambda^O \leftarrow$ Baum-Welch($\boldsymbol{O}^O$, K)
4 $\tau \leftarrow$ threshold
5 $\boldsymbol{l}^N \leftarrow$ long-term probabilities of $\boldsymbol{A}^N$
6 Hungarian($\boldsymbol{B}^N$, $\boldsymbol{B}^O$)
7 **if** $D(\lambda^N, \lambda^O) > \tau$ **then**
8 | echo warning
9 **end**

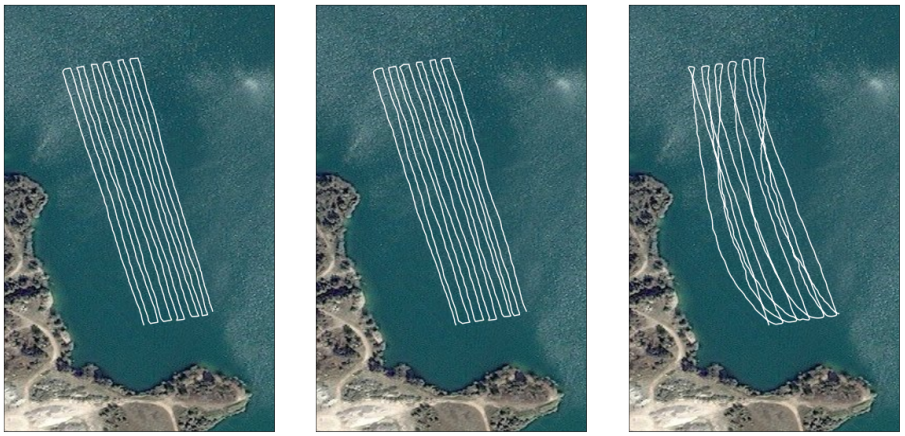
state, since their theoretical interpretation is exactly that, the long-run proportion of time spent in each state, and *(ii)* they add up to 1, making equation (3.5) a weighted average.

To compute equation (3.5), we perform a bijective matching between states of λ^N and states of λ^O using the Hungarian algorithm [93] and considering the Hellinger distance between each pair of states, namely, the distance between the emission probability distributions of those states.

In practice, for diagnostic purposes, equation (3.5) can be unrolled and its components can be inspected separately. In particular, for each state, the two contributions can be inspected and, depending on the value of l_i^1 the impact of state i on the overall distance can be identified. This could greatly help in the diagnostic process to identify the precise reason(s) why two behaviors are dissimilar and to possibly recover to a non-anomalous behavior.

Beyond interpretability, one of the main strengths of our offline approach is that it is not negatively affected by differences in the lengths of the sequences $\boldsymbol{O}^N$ and $\boldsymbol{O}^O$ (as the standard likelihood) or by possible misalignments in such sequences, since it abstracts the comparison of behaviors to the level of learned HMMs.

The detailed procedure for offline anomaly detection is reported in Algorithm 2, which starts by fitting the nominal HMM λ^N with the number of hidden states suggested by the BIC score (lines 1-2). Then the observed HMM λ^O is learned with the same number of hidden states as λ^N (line 3) and the detection threshold τ is selected (line 4). Long-term probabilities of the nominal model are computed with equation 3.3.4 (line 5). Then, after having matched the states in the two models with the Hungarian algorithm (line 6), the distance between λ^N and λ^O is computed with equation (3.5). If such distance is larger than τ, a warning is issued (lines 7-9).



(a) *Day 1* **(b)** *Day 2* **(c)** *Day 11*

Figure 3.1: *Water drone trajectories*

3.4 Experimental Results

In this section we present the results obtained by applying the proposed approach to detect anomalies of two robots operating in real-world LTA scenarios.

3.4.1 Water Monitoring Robot

The first robot operates in the context of the INTCATCH Project[1], a H2020 EU project aiming to develop a new paradigm for water monitoring in river and lakes by harmonizing a range of innovative tools into a single efficient and user-friendly model. A dataset (see Figure 3.1) has been gathered that contains 11 runs of a predefined path traveled by a Platypus drone (see Figure 3.2) on Lake Garda (Italy). The dataset consists of 76213 observations, collected at 1Hz frequency, of the following variables concerning the robot state: heading (i.e., compass direction), speed, acceleration, power signals to the left and right propellers, latitude, and longitude. A domain expert has certified the readings of the first day (see Figure 3.1(a)) as representing the nominal behavior and we use them to train an HMM. The BIC score suggests an optimal number of states $K = 3$ intuitively corresponding to:

- going *upward*, line segment A-B in Figure 3.3(a);
- going *downward*, line segment C-D in Figure 3.3(a);

[1] http://www.intcatch.eu
[2] http://senseplatypus.com/

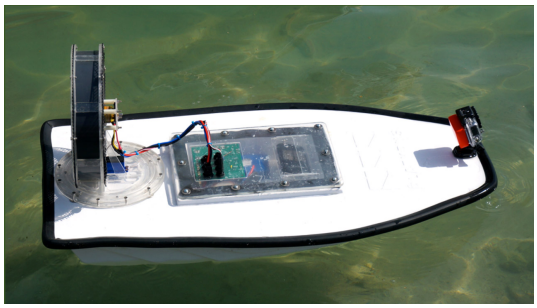


Figure 3.2: *Platypus*[2] *Lutra boat usedin the context of the INTCATCH project, about 1 m long and 0.5 m wide.*

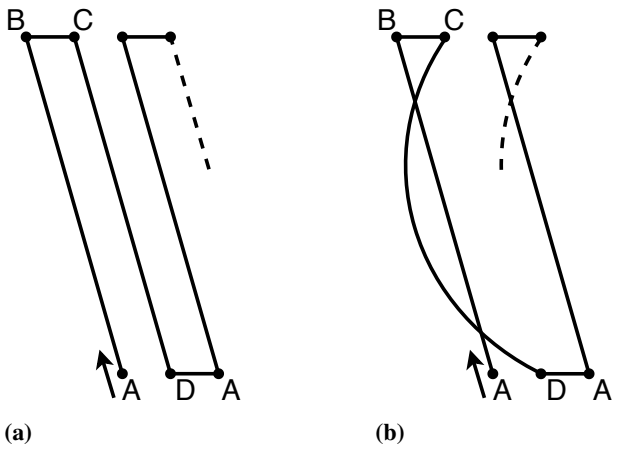


Figure 3.3: *Nominal (a) and anomalous (b) behaviors*

- going *right*, line segments B-C and D-A in Figure 3.3(a).

The domain expert also classified the runs from day 2 to day 10 as non-anomalous (Figure 3.1 just reports the trajectory of day 2) and the run of day 11 as anomalous (as it can be clearly seen from the corresponding trajectory in Figure 3.1, which we use for testing. Figure 3.3 schematically shows the difference between the regular (days 1-10) and the anomalous (day 11) behaviors.

Online anomaly detection

We compare our technique to two standard approaches (e.g., used in [126, 161, 168]): the negative log-likelihood with respect to the nominal HMM, and the negative of the logarithm of the probability of the Viterbi path. The former one (which for brevity will be referred to as likelihood) is computed

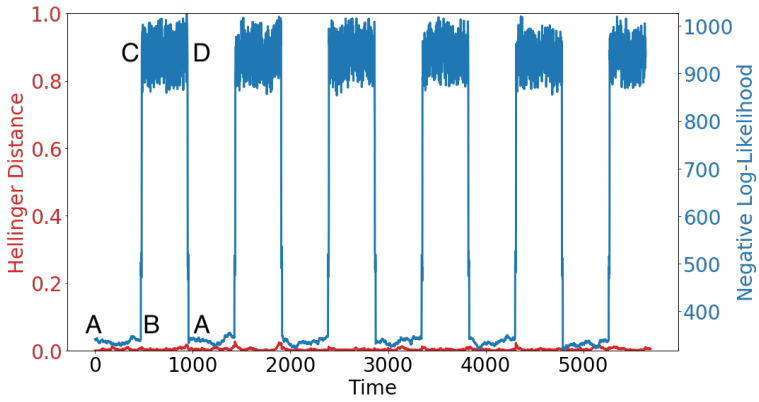


Figure 3.4: *Online anomaly detection day 2 ($w = 50$)*

with the Forward algorithm as the negative of the logarithm of $P(\boldsymbol{W}_t|\lambda^N)$. The latter is obtained by computing the Viterbi path of $\boldsymbol{W}_t$ with the Viterbi algorithm and then by taking the negative of the logarithm of the multiplication of the transition probabilities between the states in the Viterbi path.

The window size w must be set to a value between a minimum, which allows to robustly estimate a multivariate Gaussian distribution from data inside the window, and a maximum, which depends on the dynamics of the analyzed behavior (if w is too large, anomalies relative to short behaviors could be missed). After some empirical tests, we selected a window size of 50 samples.

In Figure 3.4, our anomaly measure (in red) and the likelihood (in blue) for the second day are depicted. As expected, our measure maintains always a very low value while the likelihood seems to be high during the downward segments, which visually appear to be regular. As we will see, a negative log-likelihood of 1000 is not very high (when compared to the values reached in the eleventh day), anyway, being the likelihood unbounded, it would be hard to decide a priori that such a value of likelihood does not reflect an anomaly.

Figure 3.5 reports the anomaly scores for the eleventh day, the anomalous one. The following remarks can be made:

- During the first upward segment ① an anomaly occurs, due to the fact that the speed of the robot is much higher than that observed during the nominal upward behavior. Our technique empasizes this anomaly (which is not evident from the trajectories of Figure 3.1) better than the likelihood.

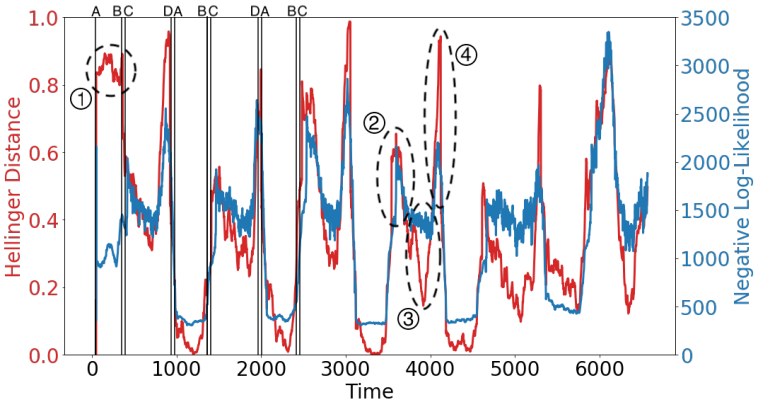


Figure 3.5: *Online anomaly detection day 11 ($w = 50$)*

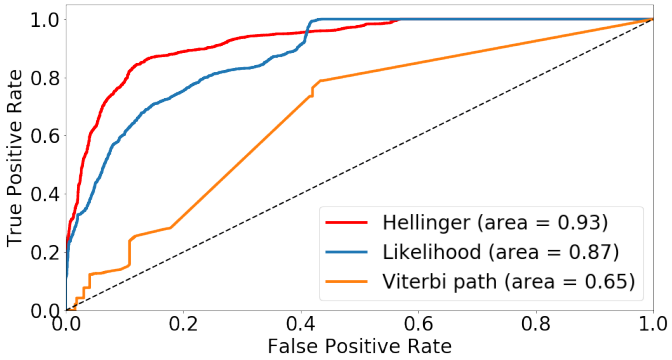


Figure 3.6: *ROC curves day 11*

- Our anomaly score better reflects the anomalies present in the downward segments. Indeed, our approach correctly assigns an higher Hellinger distance to the first half of the C-D segment ② (when the drone moves away from the optimal trajectory) and assigns a lower value to the second half ③ (when the drone gets back on track), while the negative log-likelihood reaches a plateau and does not decrease during the second half. In this sense, we can say that our approach is more expressive in capturing and representing anomalies.
- The second spike in the C-D segment ④ is correctly identified by both techniques and is due to an anomalous increase in the speed of the robot.

Figure 3.6 shows the ROC curves as τ is varied for the three methods

considered. Our method outperforms the others, improving the area under the curve (AUC) of the standard approach based on the likelihood by 6%. We omit the plots of the negative logarithm of the Viterbi path in Figures 3.4 and 3.5 since it does not detect anomalies as good as the other two approaches (as evidenced also by the lower AUC in Figure 3.6).

Besides reflecting anomalies more expressively, another advantage of the technique we propose is the ease with which it is possible to set a threshold due to its bounded nature.

If diagonal covariance matrices for the emission probabilities are used, the computational complexity of our online algorithm is linear in the number of dimensions and in the window length, while it is quadratic in the number of hidden states (i.e., the same as the two baseline methods we consider). For each window, the anomaly score can be computed in approximately 5 ms in our case study (using a commercial laptop), making our method suitable for online purposes.

Offline anomaly detection

We consider the observations of the first day as representing the nominal behavior and we use them to fit an HMM λ^1. We then fit ten more HMMs, one for each of the remaining days, called λ^2 to λ^{11}, respectively. We then compute the distance between each HMM and λ^1. Table 3.1 reports the distance between λ^1 and λ^2 (which serves as a representative for days from 2 to 10, i.e., non-anomalous days) and between λ^1 and λ^{11}. The results show a much bigger distance for the eleventh day, highlighting the presence of an anomaly, mainly caused by the downward state, which contributes 87% of $D(\lambda^1, \lambda^{11})$. The contribution of the downward state can be, in turn, further decomposed by inspecting the two contributions of equation (3.5) separately. By looking at the contribution of the transition matrices, an higher self-transition probability suggests a lower velocity. By looking at $H^2(b^1_{\text{downward}}, b^{11}_{\text{downward}})$, the contribution of the emissions probabilities, we can notice a lower mean for the velocity, confirming that the downwards segments are traversed slower than in the nominal case, and also an higher variance for the heading, which suggests that during the downward state the water drone does not manage to maintain rectilinear motion. This is an example of how the distance can be interpreted for diagnostic purposes. Note that expecting a very high distance between λ^1 and λ^{11} (i.e., close to 1) would be incorrect, since only one of the three states corresponds to anomalous behavior. In fact, the overall coverage task can be considered as partially accomplished even in presence of anomalies.

A rule of thumb to set the detection threshold τ is to choose the value of

Models	State	Distance	Total
$D(\lambda^1, \lambda^2)$	upward	0,0048	0.0112
	downward	0,0052	
	right	0,0012	
$D(\lambda^1, \lambda^{11})$	upward	0,0063	0.2424
	downward	0.2103	
	right	0.0258	

Table 3.1: *Offline anomaly detection*

the average distance between two nominal behaviors plus x times the standard deviation. For instance, $x = 3$ provides a good statistical confidence that the observed behavior is not nominal. In our experiments, the behavior of day 11 is considered anomalous since its distance from the nominal behavior of day 1 is significantly far away from the distribution of distances between day 1 and days 2 to 10. The z-score of day 11 with respect to this distribution is 69.08 (i.e., much greater than the standard threshold of 3). The p-value is < 0.0001.

3.4.2 Socially Assistive Robot

The second set of experiments is performed on data collected during the testing phase of the MoveCare project [106], a H2020 EU project developing an innovative, multi-actor platform centered around an autonomous robot for supporting the independence of elderly people living alone at home. The socially assistive autonomous mobile robot is called Giraff-X (Figure 3.7) and moves in domestic environments, which represent a typical context for LTA [107]. The goal of the robot is to provide notifications to the user. For doing so, the robot searches, identifies, and approaches the elder, and interact with him/her for stimulation by suggesting activities that aim to counteract physical and cognitive decline, as well as isolation. To localize the person, the robot starts from its charging base and visits in sequence three different rooms (living room, bedroom, and bathroom) of the test house until the elder is found. When the elder is found, the robot approaches him/her following a path suitable for Human-Robot-Interaction (HRI). After the notification is provided to the user, the robot autonomously returns back to its charging base [116]. When idle, the robot stays at its charging base.

Data are collected in a 9-day experiment simulating the same number of interventions performed in a month of use of this social assistive robot, thus

Figure 3.7: *Giraff-X socially assistive robot, developed for the MoveCare project [107]*

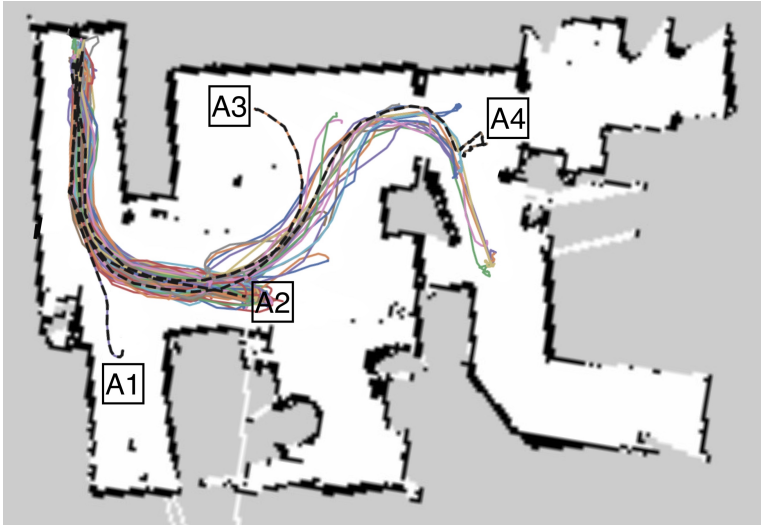


Figure 3.8: *Runs of Giraff robot*

performing multiple interventions per day for assessing LTA [107]. The dataset contains 149 runs, each one composed of a sequence of observations collected at 1 Hz and including: heading, speed, acceleration, position w.r.t. the x-axis, and position w.r.t. the y-axis (see Figure 3.8, different runs are depicted in different colors).

Out of the 149 runs, 4 are labeled as anomalous by a domain expert (denoted as A1, A2, A3, and A4 in Figure 3.8). Runs A1, A3, and A4

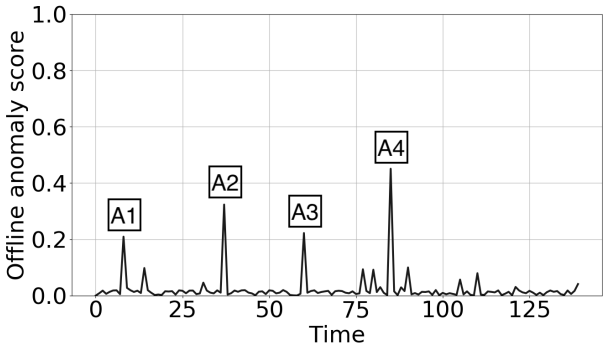


Figure 3.9: $D(\lambda^N, \lambda^{O_t})$ *for runs of the Giraff robot*

represent anomalous behaviors due to departures from the expected trajectories, while run A2 constitutes an anomaly since the robot moves at a higher speed than the nominal one. More precisely, anomalies in runs A1 and A3 are due to the fact that the robot identified the user at a different location than expected, and had to modify its path in order to find a suitable location for HRI. In run A4, after performing HRI, the robot placed itself in a position too close to furniture and got stuck there.

In this dataset a run is assessed as anomalous by considering it as a whole, thus we present only results about offline anomaly detection. A ROC curve computed online, as in Section 3.4.1, would require knowledge of which observations are actually anomalous within an overall anomalous run. Since we do not have such information, we cannot apply online anomaly detection in this case.

For each task of reaching one of the three rooms, an HMM λ^N is trained with a single run labeled as non-anomalous by the expert and considered as representing the nominal behavior. The remaining runs are tested for anomaly using our offline method. For each test run, an HMM λ^{O_t} is trained and compared with the nominal one for the task of reaching the same room. Results are shown in Figure 3.9. Our approach successfully identifies all four anomalies while reporting a low distance for all the other (correct) runs. Note that, although computing the negative log-likelihood of each whole run w.r.t. its nominal HMM could result in a plot similar to that of Figure 3.9 (yet unbounded on the y-axis), it would not be theoretically sound since each run consists of a different number of observations and, being the likelihood sensitive to trace length, the scores obtained would not actually be comparable.

3.5 Concluding Remarks

Unlike other works in the literature, we show that even a single run is enough for learning the nominal behavior, making the semi-supervised setting effectively applicable in practical real-world scenarios. Note that, in the context of LTA, a small initial supervision effort by a domain expert may be acceptable, given that the robots will operate autonomously for a long time.

In our experiments, we show that a constant detection threshold τ is enough and that the bounded nature of our anomaly scores gives a semantic meaning to such threshold. A suitable threshold τ should be chosen depending on the specific application, for example to minimize false alarms (e.g., when human verification is very costly) or to be sure to detect all anomalies while permitting some false alarms (e.g., when the robot could harm people). Although for the online approach it is easy to give a semantic interpretation to the selected threshold, for the offline approach one should choose the threshold trying to answer the question "How much am I willing to let the observed behavior be different from the nominal one and still consider it as non-anomalous?". For example, consider a case with $K = 3$ (equally important) states and two behaviors that overlap perfectly except for one state, in which they are completely different. In this case, the offline anomaly score is approximately $1/3$ and, if the application requires that an anomaly is detected when the behaviors are different in at least one state, the threshold τ should be set to a value less than $1/3$.

CHAPTER 4

Data Augmentation for HMM-based Anomaly Detection

The work contained in this chapter has been done in conjunction with colleagues from the University of Verona (Castellini, A., Masillo, F., & Farinelli, A.) and my advisor Amigoni, F. My personal contribution involved: ideation and design of the method (together with the other authors), mathematical derivation of the gradients, experiments on the two datasets involving robots (i.e., INTCATCH and ALFA). The work contained in this chapter has been submitted to the journal of Artificial Intelligence (AIJ) and is currently under review.

4.1 Introduction

In the previous chapter, an online approach has been presented for detecting anomalous behaviors of robot systems involved in complex LTA scenarios. The methodology uses HMMs to model the nominal (expected) behaviour of the robot and the Hellinger distance (H^2) [67] to evaluate the dissimilarity between the probability distribution of subsequences of observations (i.e., multivariate sensor time series) in a sliding window and the emis-

sion probability of related HMM hidden states. It has been shown that 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 it lends itself to simpler interpretation and thresholding; second, it is less noisy, hence more informative and discriminative. For simplicity, in the following we refer to this algorithm as *HMM-Hellinger-based Anomaly Detector* (*HHAD*). In this chapter, an *adversarial data augmentation and retraining technique for HHAD* (called HHAD-AUG in the following) is presented.

Data augmentation is frequently used to improve generalizability in image classifiers [148]. In our context, it is motivated by the lack of anomalously labeled examples and the noise that generally characterizes data in robotic and real systems in general. Both issues can be mitigated by the exploration and characterization of the feature space nearby (nominal) training samples (i.e., the nominal region of our anomaly detector). Following the recent and promising research field of adversarial example generation and adversarial attack generation for machine learning models [55], we ground our data augmentation method on adversarial examples, which, in our case, are perturbed time series [60,76,82,120]. More precisely, we perturb nominal examples in the training set to become adversarial using two algorithms we propose, one based on the Hellinger distance (which we call *H-ADV*), and a second one based on the HMM likelihood (called *L-ADV*). The main goal of the adversarial data augmentation is to improve the performance of HHAD when limited amounts of training data are available, however, we show that the augmentation procedure we propose achieves performance improvements also with models trained on larger datasets.

What differentiates our approach for *generating adversarial examples* from the approaches in the literature is that these approaches are mainly targeted to deep neural networks for image classification. Very recently, an approach has been proposed to perform adversarial attacks on (univariate) time series classifiers based on neural networks [82], but no method exists that is specifically targeted to HMM-based models. Our method employs the same definition of adversarial attacks for time series used in [82] but it generates adversarial examples specifically designed for HMM-based classifiers. Finally, the proposed method focuses on a specific type of classification problem, namely, anomaly detection, which we address in its semi-supervised formulation, while the classification tasks considered by the other works are generally supervised. In summary, there are three main differences between the approach for adversarial example generation presented here and those available in the literature, namely, *(i)* the procedure is

based on HMMs and Hellinger distance instead of on neural networks, *(ii)* the samples we consider are (slices of) multivariate time series instead of images, *(iii)* our target model is an anomaly detector instead of a standard classifier.

As it will be better explained in the next sections, in order to compute adversarial samples, the gradient of the anomaly score need to be computed and maximized. Since the anomaly score used by HHAD (i.e., the *Hellinger* distance between the distribution of observed samples and the distribution of related HMM emission models), is hard to express in closed form as, involving the computation of the maximally frequent state in the observation window, requires to invert the Viterbi problem, we consider also a second anomaly score: the HMM likelihood. Although the HMM likelihood is only a proxy of the anomaly score used by HHAD, its gradient is easier to express. We mathematically derive the gradients of both anomaly scores, and present a procedure to use them to augment the training dataset, which, according to our extensive experiments lead to a significant improvement in the detection performance of HHAD in terms of F1-score, especially when very few observations are available.

We evaluate the data augmentation and retraining technique on four public datasets, three of which are recognized real-world benchmarks for anomaly detection in robotic and cyber-physical systems. The first dataset is generated by the Tennessee-Eastman industrial chemical process [97,99], where the control system is tested for cyber-attacks [5]; the second dataset comes from the Secure Water Treatment (SWaT) testbed [51,110], a scaled version of an industrial water treatment plant also tested for cyber-attacks [5]; the third dataset is the Air Lab Fault and Anomaly (ALFA) dataset [83], a recent benchmark generated by real Unmanned Aerial Vehicles (UAVs); finally, the fourth dataset, used also in the previous chapter, contains multivariate sensor signals collected by aquatic drones involved in water monitoring and developed in the INTCATCH Horizon 2020 project [30]. For each dataset we first generate an anomaly detector using HHAD and then try to improve its performance using the proposed data augmentation and retraining method HHAD-AUG.

The experimental evaluation of the proposed approach confirms that *(i)* H-ADV and L-ADV can generate meaningful adversarial examples for HHAD and *(ii)* HHAD-AUG can employ these new samples to improve the performance of HHAD. Regarding the first point, we are able to generate adversarial examples for all datasets evaluated. These examples are guaranteed to have small distance from the original nominal examples, according to the definition of adversarial example for time series given

in [82]. For instance, in the dataset based on the industrial chemical process (where anomalies correspond to cyber-attacks), the approach based on the Hellinger distance generates adversarial examples (i.e., multivariate time series) with maximum $L2$ distance of 0.803 from their original examples. This corresponds to an average distance on each point of 0.002, an imperceptible amount with respect to the average range of the time series which is about 9.994, considering that all time series have been standardized. This tiny perturbation would lead the new adversarial example (which is still very probably a nominal sample) to be misclassified by the detector, however, by adding it to the training set we are able to prevent cases like this one to be wrongly classified as anomalous. The only assumption underlying our approach is that the actual nominal region in the feature space is homogeneous, which means that that, starting from a nominal point, if we move just slightly, we should reach another nominal point. This assumption is plausible in practice.

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

- We propose an algorithm able to generate adversarial examples for an anomaly detector based on HMMs and working with multivariate time series.
- We propose an algorithm for data augmentation and retraining based on adversarial examples which improves the performance of the anomaly detector.
- We evaluate, obtaining good results, both adversarial generation and data augmentation on four datasets[1] of multivariate sensory signals acquired from autonomous robots and industrial cyber-physical systems, namely, Tennessee Eastman [97], SWaT [110], ALFA [83] and INTCATCH [30].

4.2 Related Work

Three main research topics are related to the work presented in this chapter. The first one is anomaly detection for autonomous robots (which has already been covered in Chapter 2), the second one is adversarial example generation, and the third one is data augmentation. In the following, we analyze the state-of-the-art of the last two topics separately and high-

[1] We remark that the reason behind the choice of not using the MoveCare dataset for these experiments is it being composed of a rather conspicuous amount of runs already, and hence not fit for the scope.

light the differences between our approach and the most similar ones in the literature.

Adversarial Example Generation

Adversarial examples on classification models are investigated in [16] and the analysis is specialized to neural networks for image classification in [152] where the authors noticed 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 [152] 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 [55]. It produces sub-optimal adversarial examples, in terms of distance from the original sample, but being very fast it has quickly become popular and has inspired other approaches. Two examples are Deepfool [114] and the Carlini-Wagner method [25], that use iterative procedures based on local linearization of the classifier function. The approaches described so far require full knowledge of the classifier parameters and are hence said *white-box*. When such information is not available, *black-box* methods are generally used [124]. They first query the classifier and collect responses, then use this data to generate a neural network-based approximation of the classifier, and finally generate perturbations using this approximated classifier relying on adversarial example transferability [125]. A theoretical framework for analyzing the robustness of classifiers to adversarial perturbations is proposed in [48]. Adversarial training has been used as a regularization method for supervised and semi-supervised learning of neural networks in [112]. A method for generating universal adversarial perturbations is presented in [115]. Complete reviews on adversarial attack methods are proposed in [121, 166].

The differences between our method for adversarial example generation and those mentioned so far are that we focus on a one-class classification problem (i.e., semi-supervised anomaly detection), our classifier is based on HMMs and Hellinger distance, and our samples are multivariate time series. To the best of our knowledge the current literature does not provide any method for adversarial example generation in this setting. The only attack specifically targeted to HMMs which resembles adversarial attacks is that presented in [24], which is applied to speech recognition systems. However, the examples (audio voice commands) generated by the attacker in that case cannot be defined as adversarial, since they are required to

be recognizable by the speech recognition system and not recognizable by humans, hence they can be very different from original examples, while adversarial examples are defined as perturbed examples very similar (i.e., close) to the original ones. Furthermore, unlike black-box methods, our method does not need to generate neural network approximations of the detector, which would require prior knowledge about the complexity of the classification problem (to set the network architecture) and large amounts of data for training.

Methodologies for generating adversarial attacks on time series are proposed in [60, 76, 82, 120]. A strategy based on Adversarial Transformation Networks (ATNs) [10] is used in [76, 82] to generate adversarial attacks on a target classifier of time series via a student model trained using standard model distillation techniques [22, 71]. The target classifier can be a fully convolutional neural network or a 1-nearest neighbor classifier with Dynamic Time Warping. The ATN takes as input a time series x 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 x' that represents a possible adversarial sample. 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 [60] 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 [60, 76, 82] but the approach we propose is different both from the point of view of the objective and that of the method. We propose a data augmentation technique based on adversarial samples for improving HMM-based anomaly detectors that work on multivariate time series, while [82] and [60] propose new methodologies for generating adversarial attacks on time series. We derive the gradient of the specific loss function (i.e., in our case the anomaly score) of the anomaly detector, hence, our method generates adversarial examples directly on the detector, rather than on a neural network that tries to mimic it. Also the taks is different, in our case, it is an anomaly detector trained using only nominal samples, while in [82] samples 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 [3, 79], 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 on deep learning models.

Data Augmentation

Data augmentation is an established practice in image recognition with neural networks, where several methods are available for enhancing the size and quality of datasets used to train classification models. Competition-winning image classifiers such as AlexNet use data augmentation methods to achieve those high performances. In [148] data augmentation methods are partitioned according to the following categories: geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, generative adversarial networks, neural style transfer, and meta-learning. Our work would be considered as adversarial training. The main goal of data augmentation, both for images and other data types, is to prevent class imbalance and model overfitting due to data limitations, by adding synthetic samples to the available datasets [9, 35, 148, 165].

Time series data augmentation is, instead, not an established practice. The majority of state-of-the-art approaches for time series classification do not use data augmentation, and the first surveys on these techniques have been published only very recently [77, 163]. In [77] four families of time series data augmentation methods are described, among them we can find transformation-based methods, pattern mixing, generative models, and decomposition methods. We notice that methods related to adversarial training are still not considered in the literature of time series data augmentation. Most data augmentation techniques for time series [77, 100, 145, 157, 163] 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 [96], 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. This practice has been very recently applied to image classifiers [148], where adversarial data augmentation has been used to improve generalization to unseen domains. In [160] an iterative procedure is proposed to augment datasets with examples from a fictitious target domain that is hard under the current model. Adversarial examples are added to the training set at each iteration, allowing the classifier to better generalize to populations different from the training distribution in settings where data from the target distribution are inaccessible. In [149] a good performance under adversarial

input perturbations is guaranteed by considering in the learning optimization problem a Lagrangian penalty of perturbing the data distribution in a Wasserstein ball. The proposed training procedure augments model parameter updates with worst-case perturbations of training data. In [170] a novel regularization term for adversarial data augmentation in deep neural networks for image classification is proposed. This methodology encourages perturbing the underlying data source distribution to enlarge predictive uncertainty of the current model, so that the generated "hard" adversarial perturbations can improve the model robustness during training. The methodology extends [160] and reduces to it when the maximum-entropy term is discarded. These approaches show that adversarial training can be an effective data ingredient [148].

Our method applies these principles to a specific one-class classification problem (i.e., anomaly detection), for time series data instead of images. Adversarial data augmentation is applied to the specific HMM-based detector presented in [8]. We notice also that the adversarial-based strategy that we use to generate synthetic samples makes our methodology deeply different from traditional time series augmentation methods, since we do not need prior knowledge about the application to generate data transformations. Finally, we observe that our strategy allows to improve detection performance on both the original test set and the adversarial attacks generated from the same test set (see Section 4.5), hence improving both detection performance and robustness to adversarial attacks.

4.3 Background and Notation

We first informally define the problem addressed in this work, then the main strategies for adversarial example generation and data augmentation are finally presented.

4.3.1 Problem Definition

Given an online HMM-based anomaly detector HHAD [8], trained on a dataset of nominal multivariate time series, our goal is to improve the performance of the detector by augmenting its training set with adversarial examples. Moreover, we aim at improving the robustness of the detector to adversarial attacks.

4.3.2 Adversarial Example Generation

Adversarial examples have been defined in the context of image classification as misclassified examples that are only slightly different from correctly classified examples drawn from the same data distribution [55]. In other words, in problems with well-separated classes, the classifier is expected to assign the same class to an example $\boldsymbol{x}$ and a slightly perturbed example $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$, where the perturbation $\boldsymbol{\eta}$ is small enough to be not perceived as an actual change of $\boldsymbol{x}$. Common instances of adversarial examples are those of images generated by slightly perturbing an original image showing an object. These adversarial images clearly display (to human perception) the same object of the original image but they are misclassified by the classifier. Some well known examples of these images can be found, for instance, in [55]. In the context of time series data, the ground truth of a sample cannot be provided by human perception, hence it must be known in advance to evaluate the perturbed samples. Adversarial examples are then defined as slight perturbations of original samples that produce a misclassification with respect to the ground truth. Since the ground truth is not available for all samples, we use the definition of [60, 76, 82], namely, the label predicted by the classifier is assumed to be the ground truth and adversarial samples are defined as samples whose predicted class label is different from the predicted ground truth label.

A formal definition of (minimal)[2] adversarial *perturbation* for an object $\boldsymbol{x}$ is provided in [114] as the minimal perturbation $\boldsymbol{\eta}$ that is sufficient to change the label $\hat{y} = f(\boldsymbol{x})$ estimated by a classifier $f : \mathbb{R}^q \rightarrow \{1, \ldots, k\}, q \in \mathbb{N}$ for sample $\boldsymbol{x}$. The L_p distance of such a minimal perturbation is therefore:

$$\Delta(\boldsymbol{x}; f) \doteq \min_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_p \text{ subject to } f(\boldsymbol{x} + \boldsymbol{\eta}) \neq f(\boldsymbol{x}). \tag{4.1}$$

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, which measures the number of coordinates changed by the perturbation, L_2, the standard Euclidean distance between the original and the perturbed sample, and L_∞, which measures the maximum change among all coordinates [25]. Different L_p distances have a different impact on *(i)* the classification of the perturbed sample made by the classifier, *(ii)* the true class of the perturbed sample, which corresponds to human perception for images and to the ground truth class for other kinds of samples in general.

[2] In [114] this is called adversarial perturbation but we call it minimal because adversarial examples can also be affected by a non-minimal perturbation.

No distance metric is a perfect measure of similarity in the ground truth[3] and the usage of a distance metric that closely approximates the ground truth similarity is fundamental to generate good adversarial examples.

The *robustness* of a classifier f is then defined as $\rho(f) = \mathbb{E}_{\boldsymbol{x}}\Delta(\boldsymbol{x}; f)$. Namely, it is the average norm of the minimal perturbations required to change the estimated labels of all samples, given a specific classifier f and a specific set of samples. Similar definitions are also provided in [48], where a complete theoretical framework is proposed for analyzing the robustness of classifiers to adversarial perturbations. The best adversarial perturbation is defined in [48] as the point at minimal distance between $\boldsymbol{x}$ and the decision boundary of the classifier. Some bounds are defined on the robustness of classifiers. They depend on a *distinguishability* measure that captures the *difficulty* of the classification task [48].

The subtle *cause* of the existence of adversarial examples is described in [55]. Namely, the adversarial perturbation $\boldsymbol{\eta}$ is small in all its elements (Goodfellow et al. refer to the infinity norm in their work, i.e., $\|\boldsymbol{\eta}\|_\infty < \epsilon$, where ϵ is a small real number that limits the perturbation size) but it produces a strong effect on the classifier prediction if all these elements contribute concordantly to move towards the closest point of the model decision boundary. In [152] authors define adversarial examples as "low-probability (high-dimensional) pockets in the manifold represented by the deep neural classifier, which are hard to efficiently find by simply randomly sampling the input around a given example". The effect of adversarial perturbation is in fact maximized in high dimensional problems where a large number of infinitesimal (hence invisible) changes performed to a sample (e.g., an image) can add up producing a very large change on the output. The direction of these changes is however hard to find in practice and different methods have been proposed depending on the assumptions made on the type of model, the availability of model parameters and other elements of the problem.

The first technique for generating adversarial examples on deep neural networks has been proposed in [152]. It uses an optimization procedure called box-constrained *Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm* (*L-BFGS*), to solve the problem of finding the closest image to $\boldsymbol{x}$ classified with a specific label l by f, which is formally defined

[3] In [25], authors refer to human perceptual similarity but this concept can be extended to the similarity in the ground truth when compared examples are not images.

as:

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{\eta}\|_2 && (4.2a)\\ &\text{subject to} && f(\boldsymbol{x}+\boldsymbol{\eta}) = l, && (4.2b)\\ & && \boldsymbol{x}+\boldsymbol{\eta} \in [0,1]^q. && (4.2c) \end{aligned}$$

The label of the adversarial example is explicit, hence this method is said to be *targeted*. Since this is a hard problem, an approximated solution has been computed by repeatedly solving the following minimization problem:

$$\begin{aligned} &\text{minimize} && c\|\boldsymbol{\eta}\|_2 + \mathit{loss}_f(\boldsymbol{x}+\boldsymbol{\eta}, l) && (4.3a)\\ &\text{subject to} && \boldsymbol{x}+\boldsymbol{\eta} \in [0,1]^q, && (4.3b) \end{aligned}$$

with different constants $c > 0$, where $\mathit{loss}_f : \mathbb{R}^q \times \{1,\dots,k\} \to \mathbb{R}^+$ is the loss function associated to the classifier f, such as the cross-entropy. Notice that minimizing the loss function for the example $\boldsymbol{x}+\boldsymbol{\eta}$ and label l is equivalent to find a perturbation $\boldsymbol{\eta}$ where the example $\boldsymbol{x}+\boldsymbol{\eta}$ has actually label l.

To overcome the time-complexity issue of this method the *Fast Gradient Sign Method* (**FGSM**) is proposed in [55] which is *untargeted* (i.e., it does not allow to specify the target label) and optimized for the L_∞ distance metric. This method does not produce maximally close adversarial examples but it is faster than L-BFGS. Given an example $\boldsymbol{x}$, FGSM searches for an adversarial $\boldsymbol{x}'$ such that

$$\boldsymbol{x}' = \boldsymbol{x} + \epsilon \cdot \mathit{sign}(\nabla_{\boldsymbol{x}} \mathit{loss}_f(\boldsymbol{x}, y)), \tag{4.4}$$

where ∇_x is the gradient over $\boldsymbol{x}$. According to this method, given the parameters $\boldsymbol{\theta}$ of the classifier f, an example $\boldsymbol{x}$, 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 considering $\boldsymbol{x}'$ with the original label y. The method can be made targeted by using a perturbation $-\epsilon \cdot \mathit{sign}(\nabla_x \mathit{loss}_f(\boldsymbol{x}, l))$ with l being the target label. Time complexity can be improved by efficiently computing the gradient using backpropagation. Other popular methods proposed more recently are *Deepfool* [114] and *Carlini-Wagner* [25]. They usually manage to generate adversarial examples that are more similar to original examples than those of methods described so far. The reader can refer to the original papers for a detailed description.

Finally, *black-box* adversarial attacks [124] assume adversarial example transferrability [125]. They need to first generate a neural network approximation of the classifier under investigation and then they generate adversarial examples on this model using standard methods for neural networks. The main contribution of these methods is in the strategy they use to approximate the original classifier by a neural network. This process can be time consuming and it needs an initial dataset with samples for all classes in the problem, which is hardly applicable in anomaly detection since anomalies are usually not available in advance. The quality of the initial dataset can affect the quality of the approximation and therefore of the adversarial examples. A *taxonomy* of adversarial examples for deep neural network classifiers is found in [123].

4.3.3 Data Augmentation

The data augmentation problem consists in extending a dataset by adding new samples with the aim to improve the performance of a model trained on such dataset. Standard approaches for time series data augmentation have been already analyzed in Section 4.2. As, usually, new samples are generated from the original ones, the problem can be formalized as the generation from original samples $\boldsymbol{x}$ of perturbed samples $\boldsymbol{x} + \boldsymbol{\eta}$ that maximize the performance of model f. In the following we describe two baseline methods for time series data augmentation [77] that we use in Section 4.5 to compare the performance of our method. We selected these methods as baselines because they are general approaches that do not introduce specific domain knowledge, similarly to our adversarial example generation method.

Random Data Augmentation Strategy (R-AUG)

An i.i.d. random noise is added to every time point x_i^j of the time series $\boldsymbol{x}$, where i is the time instant and j the signal (i.e., sensor). The random noise is sampled from a uniform distribution which is parametrized such that a maximum perturbation ϵ is obtained for each point.

Drift Data Augmentation Strategy (D-AUG)

This method drifts the time series $\boldsymbol{x}$ gradually such that a more realistic perturbation is produced. The augmenter drifts the values of $\boldsymbol{x}$ from original ones randomly and smoothly as the time increments. The extent of drifting is controlled by the maximal drift and the number of drifted points, which we set so that a maximum perturbation of ϵ is obtained.

4.4 The Proposed Method

In this section we first present our two approaches for adversarial example generation on HHAD. The first approach is based on a loss function that uses the Hellinger distance and the second on a loss function that uses the HMM likelihood. Please notice that in this context we sometimes refer to *anomaly scores* as *loss functions* in order to be compliant with the related literature. Subsequently, we introduce the methodology for adversarial data augmentation and retraining.

4.4.1 Adversarial Example Generation

Our approach for adversarial example generation is outlined in Figure 4.1 and formalized in Algorithm 3 (for the Hellinger-based loss) and Algorithm 4 (for the Likelihood-based loss). As shown in Figure 4.1, the main idea is to take a sample $\boldsymbol{x}$ (i.e., a multivariate time series) from the complete time series $\boldsymbol{O}$, and to pass it to the adversarial example generator (Alg. 3 or Alg. 4, respectively called H-ADV and L-ADV 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 [60, 76, 82]. Algorithm 3 describes H-ADV, the procedure based on a loss function which uses the Hellinger distance. It receives five inputs, namely, *(i)* a nominal example[4], i.e., a slice of a multivariate time series $\boldsymbol{x} = \langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_w \rangle \in \mathbb{R}^{d \times w}$, where w is the length of the time series slice and $\boldsymbol{x}_t = [x_t^1, \ldots, x_t^d], t \in \{1, \ldots, w\}$ is a multivariate observation representing $\boldsymbol{x}$ 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 $\boldsymbol{x}$, *(iv)* the number of steps $c \in \mathbb{N}$ in which the interval ϵ 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., $f(\boldsymbol{x}') \neq f(\boldsymbol{x})$. HHAD is a function $f : \mathbb{R}^{d \times w} \to \{0, 1\}$, where 0 is the class for nominal behaviors and 1 that for anomalous behaviors. Notice that the goal of adversarial example generation here is not to fool the detector for generating a damage, but to augment the training set, which is assumed to contain only nominal samples. For this reason, we do not consider adversarial examples generated from anomalous points, although this kind of samples could be even more harmful if used against an anomaly detector.

[4]The algorithm assumes $\boldsymbol{x}$ to be a sample of the training set which we want to augment, hence $\boldsymbol{x}$ is nominal.

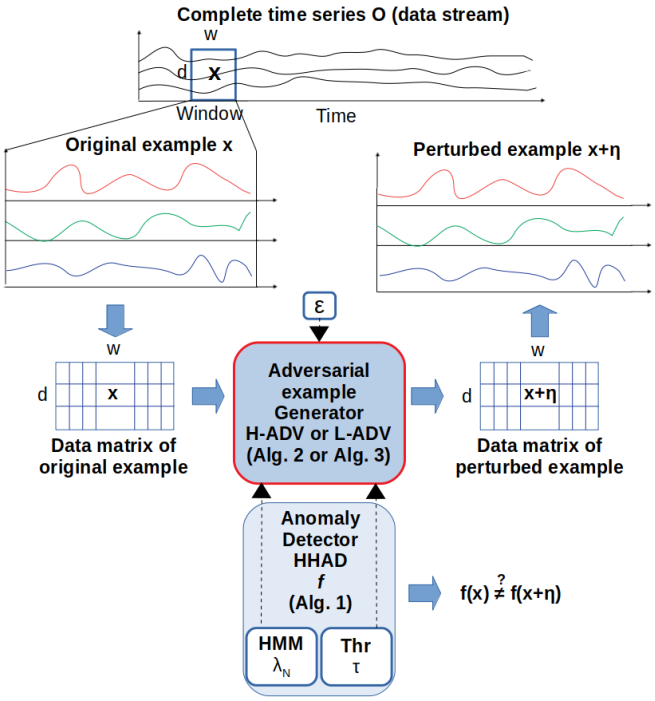


Figure 4.1: *Overview of the adversarial generation process.*

Given the example $\boldsymbol{x}$, whose class is assumed to be 0 (i.e., nominal sample), Algorithm 3 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 (see Equation 4.4). The classifier f is HHAD, hence it combines the application of the Viterbi algorithm and the threshold on the Hellinger distance between the data distribution in $\boldsymbol{x}$ and the distribution of the HMM emission model, to determine the class of the example. The loss function has a complex form in this case because it depends on the maximally frequent state in $\boldsymbol{x}$. The gradient of this function can be expressed in closed form only given the maximally frequent state $\hat{s}_t$ (see details Section 3). The main obstacle is related to the computation of the inverse-Viterbi problem, which is NP-complete [147]. 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 model $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ 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

Algorithm 3: Adversarial example generation based on Hellinger (H-ADV)

Input: $\boldsymbol{x} \in \mathbb{R}^{d \times w} \leftarrow$ original nominal example
$\lambda^N \leftarrow$ nominal HMM used by HHAD
$\epsilon \leftarrow$ maximum perturbation size
$c \in \mathbb{N} \leftarrow$ number of sampling steps per state
$\tau \in [0, 1] \leftarrow$ threshold for the Hellinger distance used by HHAD
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 $\boldsymbol{X} \leftarrow \{\boldsymbol{x}_j \in \boldsymbol{x} \ : \ s_j = \hat{s}_t\}$
4 $\boldsymbol{\mu} \leftarrow E[\boldsymbol{X}]$
5 $\boldsymbol{\Sigma} \leftarrow E[(\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{X} - \boldsymbol{\mu})^T]$
6 $\boldsymbol{g} = sign(\nabla_{\boldsymbol{x}} H^2(b^N_{\hat{s}_t}, \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}))$ // gradient of the Hellinger 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 continue=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 $\boldsymbol{X}' \leftarrow \{\boldsymbol{x}_j \in \boldsymbol{x}' \ : \ s_j = \hat{s}'_t\}$
15 $\boldsymbol{\mu}' \leftarrow E[\boldsymbol{X}']$
16 $\boldsymbol{\Sigma}' \leftarrow E[(\boldsymbol{X}' - \boldsymbol{\mu}')(\boldsymbol{X}' - \boldsymbol{\mu}')^T]$
17 $h' = H^2(b^N_{\hat{s}'_t}, \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\Sigma}'))$
18 **if** $h' > h_{max}$ **then**
19 $h_{max} = h'$ // max Hellinger update
20 continue=True
21 **end**
22 **if** $\hat{s}_t \neq \hat{s}'_t$ **then**
23 // New most frequent state
24 $\boldsymbol{g} = sign(\nabla_{\boldsymbol{x}'} H^2(b^N_{\hat{s}'_t}, \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\Sigma}'))$ // gradient update
25 $\hat{s}_t = \hat{s}'_t$
26 **end**
27 **until** $(\|\boldsymbol{x}' - \boldsymbol{x}\|_1 \geq d \cdot \epsilon) \vee$ (continue==False) $\vee \ (h_{max} > \tau)$
28 **return** $\boldsymbol{x}'$

$\boldsymbol{x}$ 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 $\boldsymbol{x}$. This 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., the perturbed example $\boldsymbol{x}'$ is classified as anomalous). Notice that not all the perturbed examples change their class and we call

adversarial only the perturbed examples that change their class[5].

Figure 4.2 provides a graphical overview of the strategy implemented by Algorithm 3. 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 sample 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 $\boldsymbol{x}^{(3)}$. This point represents the final perturbation of $\boldsymbol{x}$, which is an adversarial example since $\boldsymbol{x}'$ is classified as anomalous.

The second algorithm for adversarial example generation that we propose is applied to the standard detector HHAD but it generates adversarial examples following the gradient of the likelihood of the sample $\boldsymbol{x}$ instead of the Hellinger distance. The algorithm for adversarial example generation is called L-ADV and formalized in Algorithm 4. Its main goal is to simplify the computation of the gradient of the loss function with respect to 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 [12]. This time the algorithm for adversarial generation is not iterative, because the computation of the gradient of the likelihood is time consuming. Given an observation $\boldsymbol{x}$, first it computes the sign of the gradient of the likelihood of the sample given the nominal HMM (line 1), namely $sign(\nabla_x P(\boldsymbol{x}, \lambda^N))$ (see details Section 3). Then it moves the sample in the direction of the gradient for a step ϵ in each dimension (line 2). The algorithm returns the perturbed sample $\boldsymbol{x}'$. Notice that the perturbed example $\boldsymbol{x}'$ is an adversarial example only if the Hellinger distance between its maximally frequent observations in $\boldsymbol{x}'$ and the emission model of the maximally frequent state in $\boldsymbol{x}'$ is larger than

[5]Since original examples from the training set are nominal the change of class makes adversarial examples anomalous.

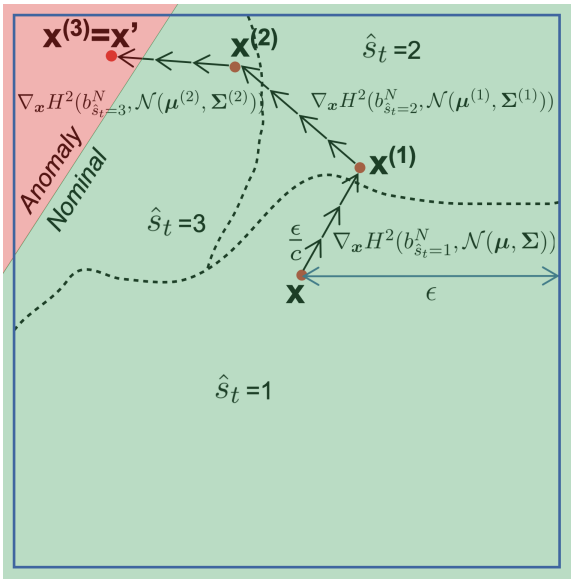


Figure 4.2: *Iterative gradient ascent strategy performed by the adversarial generation algorithm H-ADV.*

threshold τ, namely, only if HHAD classifies $\boldsymbol{x}'$ as anomalous according to its parameters λ^N and τ.

Algorithm 4: Adversarial example generation based on likelihood (L-ADV)

Input: $\boldsymbol{x} \in \mathbb{R}^{d \times w} \leftarrow$ original nominal example
$\lambda^N \leftarrow$ nominal HMM
$\epsilon \leftarrow$ maximum perturbation size
Output: $\boldsymbol{x}' \in \mathbb{R}^{d \times w}$: perturbed example

$\boldsymbol{g} = sign(\nabla_x P(\boldsymbol{x}, \lambda^N))$ // gradient of the likelihood in $\boldsymbol{x}$
$\boldsymbol{x}' = \boldsymbol{x} - \epsilon \cdot \boldsymbol{g}$ // $\boldsymbol{x}$ update
return $\boldsymbol{x}'$

The time complexity of Algorithm 3 is $O(c \cdot (wK^2 + wd))$, where $O(wK^2)$ is the computational cost of the Viterbi algorithm and $O(wd)$ is the cost of performing the gradient (cost $O(1)$) on each dimension and time step of the window. The time complexity of Algorithm 4 is $O(K^2w^2d)$. Being the gradient on the HMM likelihood not computable in closed form 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 $O(1)$. As a consequence, the computational complexity of Algorithm 4 is quadratic in the window length, which results in a considerable increase of the running time (i.e., it is ≈ 400 times slower than Algorithm 3). For-

tunately, Algorithm 4 is quite parallizable, in fact, after re-implementing it in `Cython` we managed to achieve a running time similar to that of Algorithm 3.

In the following we provide the formulas of the gradients for the two loss functions used so far, namely, the *Hellinger distance* between the distribution of the observations related to the most frequent state in $\boldsymbol{x}$ and the emission model of the most frequent state (used in lines 6 and 24 in Algorithm 3), and the *likelihood* of $\boldsymbol{x}$ given the parameters of the nominal HMM λ^N (used in line 1 of Algorithm 4).

Loss Function Based on Hellinger Distance

We consider as a loss function to be maximized by the adversarial generation algorithm H-ADV the Hellinger distance between the probability distribution of the emission model of the maximally frequent state in $\boldsymbol{x}$, called $p_1(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ in the following, and the probability distribution of the observations corresponding to the maximally frequent state in $\boldsymbol{x}$, called $p_2(\boldsymbol{x})$ in the following. Notice that example $\boldsymbol{x}$ was called $\boldsymbol{W}_t$ in Chapter 3. The Hellinger distance is computed by Equation 3.2 where $p_1(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ is the probability distribution of the emission model and $p_2(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ is the probability distribution of the example $\boldsymbol{x}$. We consider the multivariate case with diagonal covariance matrix, namely:

$$\mu_1 = \begin{bmatrix} \mu_1^1 & \cdots & \mu_1^d \end{bmatrix} \quad \mu_2 = \begin{bmatrix} \mu_2^1 & \cdots & \mu_2^d \end{bmatrix} \tag{4.5}$$

$$\Sigma_1 = \begin{bmatrix} (\sigma_1^{11})^2 & & 0 \\ & \ddots & \\ 0 & & (\sigma_1^{dd})^2 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} (\sigma_2^{11})^2 & & 0 \\ & \ddots & \\ 0 & & (\sigma_2^{dd})^2 \end{bmatrix}. \tag{4.6}$$

Substituting these four terms into Equation 3.2, the equation can be rewritten as:

$$H^2 = 1 - \frac{\left(\prod_{i=1}^{d} (\sigma_1^{ii})^2\right)^{\frac{1}{4}} \left(\prod_{i=1}^{d} \frac{\sum_{t=1}^{w} (x_t^i - \mu_2^i)^2}{w}\right)^{\frac{1}{4}}}{\left(\prod_{i=1}^{d} \frac{(\sigma_1^{ii})^2 + \frac{\sum_{t=1}^{w} (x_t^i - \mu_2^i)^2}{w}}{2}\right)^{\frac{1}{2}}} \cdot \exp\left\{ -\frac{1}{8} \cdot \sum_{i=1}^{d} \frac{2 \cdot \left(\mu_1^i - \frac{\sum_{t=1}^{w} x_t^i}{w}\right)^2}{(\sigma_1^{ii})^2 + \frac{\sum_{t=1}^{w} (x_t^i - \mu_2^i)^2}{w}} \right\}. \tag{4.7}$$

We then compute the gradient of Equation 4.7 with respect to each single element x_t^i of the example $\boldsymbol{x}$ and obtain the following formula:

$$\nabla_{x_{t^*}^{i^*}} H^2 = -\frac{\zeta^* - \iota^*}{\beta} \cdot \kappa + \nu \cdot \kappa \cdot \frac{\xi^*}{\psi^*} \tag{4.8}$$

where the seven terms $\zeta^*, \iota^*, \beta, \kappa, \nu, \xi^*$, and ψ^* are expanded in the following equations (terms with superscript * depend on $x_{t^*}^{i^*}$):

$$\beta = \prod_{i=1}^{d} \frac{(\sigma_1^{ii})^2 + \frac{\sum_{t=1}^{w}(x_t^i - \mu_2^i)^2}{w}}{2} \tag{4.9}$$

$$\gamma = \left(\prod_{i=1}^{d} (\sigma_1^{ii})^2 \right)^{\frac{1}{4}} \tag{4.10}$$

$$\delta = \prod_{i=1}^{d} \frac{\sum_{t=1}^{w}(x_t^i - \mu_2^i)^2}{w} \tag{4.11}$$

$$\zeta^* = \beta^{\frac{1}{2}} \cdot \gamma \cdot \frac{1}{2} \delta^{-\frac{3}{4}} \cdot \left[\left(\prod_{i=1, i \neq i^*}^{d} \frac{\sum_{t=1}^{w}(x_t^i - \mu_2^i)^2}{w} \right) \cdot \frac{(x_{t^*}^{i^*} - \mu_2^{i^*})}{w} \right] \tag{4.12}$$

$$\iota^* = \gamma \cdot \delta^{\frac{1}{4}} \cdot \frac{1}{2} \beta^{-\frac{1}{2}} \cdot \left[\left(\prod_{i=1, i \neq i^*}^{d} \frac{(\sigma_1^{ii})^2 + \frac{\sum_{t=1}^{w}(x_t^i - \mu_2^i)^2}{w}}{2} \right) \cdot \frac{(x_{t^*}^{i^*} - \mu_2^{i^*})}{w} \right] \tag{4.13}$$

$$\kappa = \exp\left\{ -\frac{1}{4} \cdot \sum_{i=1}^{d} \frac{\left(\mu_1^i - \frac{\sum_{t=1}^{w} x_t^i}{w} \right)^2}{(\sigma_1^{ii})^2 + \frac{\sum_{t=1}^{w}(x_t^i - \mu_2^i)^2}{w}} \right\} \tag{4.14}$$

$$\nu = -\frac{\gamma \cdot \delta^{\frac{1}{4}}}{\beta^{\frac{1}{2}}} \tag{4.15}$$

$$\begin{aligned} \xi^* = & -\frac{1}{2} \cdot \left((\sigma_1^{i^* i^*})^2 + \frac{(x_{t^*}^{i^*} - \mu_2^{i^*})^2}{w} \right) \cdot \left[\frac{-x_{t^*}^{i^*}}{w} \cdot \left(\mu_1^{i^*} - \sum_{t=1}^{w} \frac{x_t^{i^*}}{w} \right) \right] + \\ & - 4 \cdot \left(\mu_1^{i^*} - \frac{\sum_{t=1}^{w} x_t^{i^*}}{w} \right)^2 \cdot \frac{(x_{t^*}^{i^*} - \mu_2^{i^*})}{w} \end{aligned} \tag{4.16}$$

$$\psi^* = \left((\sigma_1^{i^* i^*})^2 + \frac{(x_{t^*}^{i^*} - \mu_2^{i^*})^2}{w} \right)^2 \tag{4.17}$$

The gradient of Equation 4.8 is used by H-ADV (Algorithm 3 lines 6 and 24) to iteratively compute the direction of the perturbations. As explained above, the class is always $y = 0$ (i.e., nominal sample) because the training set considers only nominal samples. Moreover, although not explicitly specified, the loss function depends on the emission model $b_{\hat{s}_t}^N$ of the most frequent state in the current example, which is represented by $f(x) \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ in the equations above.

Loss Function Based on Likelihood

Given an example $\boldsymbol{x}$, the probability to observe it given the nominal HMM λ^N can be computed by the *forward* algorithm [136] as:

$$P(\boldsymbol{x}|\lambda^N) = \sum_{i=1}^{K} \alpha_w(i), \tag{4.18}$$

where w is the dimension of the window used by HHAD and

$$\begin{aligned} \alpha_t(j) &= P(\langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_w \rangle, q_t = s_j \mid \lambda^N) \\ &= \sum_{i=1}^{K} \alpha_{t-1}(i) a_{ij} b_j(\boldsymbol{x}_t) \quad 1 \leq j \leq K, \;\; 2 \leq t \leq w, \end{aligned} \tag{4.19}$$

$$\alpha_1(j) = \pi_j b_j(\boldsymbol{x}_1) \qquad 1 \leq j \leq K, \tag{4.20}$$

in which K is the number of hidden states in λ^N, a_{ij} is the probability to switch from state s_i to state s_j given the transition matrix $\boldsymbol{A}$, $b_j(\boldsymbol{o}_t)$ is the probability to get observation $\boldsymbol{o}_t$ from state s_j at time instant t, and π_j is the probability to have initial state s_j.

We then compute the gradient of $P(\boldsymbol{x}|\lambda^N)$ in two cases, namely, when the covariance matrix $\boldsymbol{\Sigma}$ is *diagonal* (as we did for the loss function based on the Hellinger distance in Equation 4.8) and when it is *full*[6]. In the first case the observation probability becomes:

$$\begin{aligned} b_j(x_t^1, \ldots, x_t^d) &= \frac{exp\left(-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_t - \boldsymbol{\mu}_j)\right)}{\sqrt{(2\pi)^d det(\boldsymbol{\Sigma}_j)}} \\ &= \frac{exp\left(-\frac{1}{2}\sum_{i=1}^{d}\left((x_t^i - \mu_j^i)^2 \cdot \frac{1}{(\sigma_j^{ii})^2}\right)\right)}{\sqrt{(2\pi)^d \cdot \prod_{i=1}^{d}(\sigma_j^{ii})^2}} \end{aligned} \tag{4.21}$$

and, consequently, the gradient can be computed as:

$$\nabla_{x_{t^*}^{i^*}} P(\boldsymbol{x}|\lambda^N) = \sum_{i=1}^{K} \alpha_w(i), \tag{4.22}$$

where w is the dimension of the window used by HHAD and

$$\alpha_t(j) = \sum_{i=1}^{K} \alpha_{t-1}(i) a_{ij} b_j(\boldsymbol{x}_t) \quad 1 \leq j \leq K, \;\; 2 \leq t \leq w, \;\; t \neq t^* \tag{4.23}$$

[6]The case with full covariance matrix is not available for the loss function based on the Hellinger distance because in that case the gradient is computed for each window, at runtime, and the computation of the determinant present in the formula of the Hellinger distance would be computationally too expansive with full covariance matrices.

$$\alpha_t(j) = \sum_{i=1}^{K} \alpha_{t-1}(i) a_{ij} \cdot \frac{exp\left(-\frac{1}{2}\sum_{i=1}^{d}\left((x_t^i - \mu_j^i)^2 \cdot \frac{1}{(\sigma_j^{ii})^2}\right)\right)}{\sqrt{(2\pi)^d \cdot \prod_{i=1}^{d}(\sigma_j^{ii})^2}} \cdot \frac{-(x_t^{i^*} - \mu_j^{i^*})}{(\sigma_j^{i^*i^*})^2} \qquad (4.24)$$
$$1 \leq j \leq K, \;\; t = t^*$$

$$\alpha_1(j) = \pi_j b_j(\boldsymbol{x}_1) \qquad 1 \leq j \leq K, \;\; t^* \neq 1 \qquad (4.25)$$

$$\alpha_1(j) = \sum_{i=1}^{K} \pi_j \cdot \frac{exp\left(-\frac{1}{2}\sum_{i=1}^{d}\left((x_1^i - \mu_j^i)^2 \cdot \frac{1}{(\sigma_j^{ii})^2}\right)\right)}{\sqrt{(2\pi)^d \cdot \prod_{i=1}^{d}(\sigma_j^{ii})^2}} \cdot \frac{-(x_1^{i^*} - \mu_j^{i^*})}{(\sigma_j^{i^*i^*})^2}$$
$$1 \leq j \leq K, \;\; t^* = 1. \qquad (4.26)$$

In the second case (full covariance matrix) we instead define the inverse of the covariance matrix for the j-th emission model as:

$$\Sigma_j^{-1} = \begin{bmatrix} (s_j^{11})^2 & \dots & (s_j^{1d})^2 \\ \vdots & \ddots & \vdots \\ (s_j^{d1})^2 & \dots & (s_j^{dd})^2 \end{bmatrix} \qquad (4.27)$$

and the observation probability becomes:

$$b_j(x_t^1, \dots, x_t^d) = \frac{exp\left(-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu}_j)^T \Sigma_j^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}_j)\right)}{\sqrt{(2\pi)^d |\Sigma_j|}}, \qquad (4.28)$$

therefore the gradient of the likelihood $P(\boldsymbol{x}|\lambda^N)$ can be computed using Equations 4.22 and 4.23, but substituting Equation 4.24 (used when $t = t^*$) by:

$$\alpha_t(j) = \sum_{i=1}^{K} \alpha_{t-1}(i) a_{ij} \cdot \frac{exp\left(-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu}_j)^T \Sigma_j^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}_j)\right)}{\sqrt{(2\pi)^d |\Sigma_j|}} \cdot$$
$$\cdot -\frac{1}{2}\left(\sum_{i=1}^{d}(x_t^i - \mu_j^i)(s_j^{ii^*})^2 + \sum_{i=1}^{d}(x_t^i - \mu_j^i)(s_j^{i^*i})^2\right) \qquad 1 \leq j \leq K, \;\; t = t^* \qquad (4.29)$$

$$\alpha_1(j) = \sum_{i=1}^{K} \pi_j \cdot \frac{exp\left(-\frac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{\mu}_j)^T \Sigma_j^{-1}(\boldsymbol{x}_1 - \boldsymbol{\mu}_j)\right)}{\sqrt{(2\pi)^d |\Sigma_j|}} \cdot$$
$$\cdot -\frac{1}{2}\left(\sum_{i=1}^{d}(x_1^i - \mu_j^i)(s_j^{ii^*})^2 + \sum_{i=1}^{d}(x_1^i - \mu_j^i)(s_j^{i^*i})^2\right) \qquad 1 \leq j \leq K, \;\; t^* = 1. \qquad (4.30)$$

The gradient of Equation 4.22 is used by L-ADV (Algorithm 4 line 1) to compute the direction of the perturbations. In this case the gradient does not depend on the maximally frequent state $\hat{s}'$, as it happens with the Hellinger-based loss, hence the gradient is computed only once in Algorithm 4 instead of using an iterative strategy.

4.4.2 Data Augmentation and Retraining

The procedures H-ADV and L-ADV for generating adversarial examples are here integrated in a technique for data augmentation and retraining called HHAD-AUG. Algorithm 5 formalizes the proposed approach. The algorithm aims at improving the performance of HHAD and its robustness to adversarial attacks. The inputs are the nominal time series $\boldsymbol{O}$ used to train HMM λ^N, the original HMM λ^N (having K hidden states), the windows size w, the loss function $loss_f(\boldsymbol{x}, y)$ (i.e., based on Hellinger distance or likelihood), the threshold τ for the Hellinger distance, the set Y of training labels, the maximum perturbation size ϵ, the number of intervals c in which ϵ is split during adversarial generation, and the number of times m that adversarial examples are generated on the training set. The outputs are instead the augmented training set of nominal samples $\hat{\boldsymbol{W}}$, the augmented nominal HMM $\hat{\lambda}^N$ (trained on $\hat{\boldsymbol{W}}$), and the augmented threshold $\hat{\tau}$ learned from $\hat{\lambda}^N$ and $\boldsymbol{W}$.

The augmented dataset $\hat{\boldsymbol{W}}$ is first initialized to the set of examples in the training set generated by sampling the complete time series $\boldsymbol{O}$ 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 (as, since the HMM and the threshold are updated at each iteration, there is the possibility of discovering new adversarial examples at each new iteration). The steps of this loop are described in the following. For each training example $\boldsymbol{W}_t = \langle \boldsymbol{o}_{t-w+1}, \ldots, \boldsymbol{o}_t \rangle$ in the training set $\boldsymbol{W}$ (line 7) a perturbed example $\boldsymbol{W}'_t$ is generated using algorithm H-AUG or L-AUG (lines 9 and 11) depending on the loss function $loss_f(\boldsymbol{x}, y)$ chosen for adversarial generation. The perturbed example $\boldsymbol{W}'_t$ is added to the augmented set $\hat{\boldsymbol{W}}$ only if it is classified as an anomaly by HHAD (lines 13 and 15). When all examples in the training set have been perturbed the nominal HMM is retrained using the augmented dataset $\hat{\boldsymbol{W}}$ (line 18). The threshold is then updated, if its value is increased, to the value of the maximum Hellinger distance computed using the augmented HMM on examples in

Algorithm 5: Adversarial data augmentation and retraining (HHAD-AUG)

Input: $\boldsymbol{O} \leftarrow$ nominal d-dimensional time series (training set)
$\lambda^N \leftarrow$ Baum-Welch($\boldsymbol{O}$, K)
$K \leftarrow$ number of hidden states
$w \leftarrow$ window size
$loss_f(\boldsymbol{x}, y) \leftarrow$ loss function
$\tau \leftarrow$ threshold
$Y \leftarrow$ set of training set labels
$\epsilon \leftarrow$ maximum perturbation size
$c \in \mathbb{N} \leftarrow$ number of sampling steps for state
$m \in \mathbb{N} \leftarrow$ number of iterations
Output: $\hat{\boldsymbol{W}}$ augmented set of nominal samples
$\hat{\lambda}^N$ augmented nominal HMM
$\hat{\tau}$ augmented threshold

1 $\boldsymbol{W} = \{\langle \boldsymbol{o}_{t-w+1}, \ldots, \boldsymbol{o}_t \rangle \mid t = w, \ldots, n\}$ // original training set
2 $\hat{\boldsymbol{W}} = \boldsymbol{W}$ // 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 $\boldsymbol{W}_t = \langle \boldsymbol{o}_{t-w+1}, \ldots, \boldsymbol{o}_t \rangle$ // select sample from training set
8 **if** $(loss_f == H^2(\cdot,\cdot))$ **then**
9 $\boldsymbol{W}'_t =$ H-ADV$(\boldsymbol{W}_t, \hat{\lambda}^N, \epsilon, Y(t), K, H^2(\cdot,\cdot), c)$
10 **else**
11 $\boldsymbol{W}'_t =$ L-ADV$(\boldsymbol{W}_t, \hat{\lambda}^N, \epsilon, Y(t), K, ll_f(\cdot), c)$
12 **end**
13 $y =$ HHAD$(\boldsymbol{W}'_t, K, \hat{\lambda}^N, w, \hat{\tau})$
14 **if** *(y==1)* **then**
15 $\hat{\boldsymbol{W}} = \hat{\boldsymbol{W}} \cup \boldsymbol{W}'_t$ // add adversarial $\boldsymbol{W}'_t$ to augmented dataset $\hat{\boldsymbol{W}}$
16 **end**
17 **end**
18 $\hat{\lambda}^N = Baum\text{-}Welch(\hat{\boldsymbol{W}}, K)$ // retrain HMM λ^N with augmented data
19 $\tau' =$ maximum value of H^2 or ll_f for samples in $\boldsymbol{W}$ computed using $\hat{\lambda}^N$
20 **if** *($\tau' > \hat{\tau}$)* **then**
21 $\hat{\tau} = \tau'$ // update threshold τ
22 **end**
23 **end**
24 **return** $\hat{\boldsymbol{W}}, \hat{\lambda}^N, \hat{\tau}$

the original dataset $\boldsymbol{W}$ (lines 19-22). 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. This is the main idea of our approach, and it is based on the fact that the adversarial examples are very close to the original nominal examples, hence we consider them as misclassified by the original detector. Finally, we observe that in line 7 we

consider only samples $\boldsymbol{W}_t$ from the training set in all m iterations, hence adversarial examples are not not considered as original examples to generate other adversarial examples. This guarantees that the decision boundary is not moved away from the training examples indefinitely.

The process of adversarial example generation, HMM retraining, and threshold update, is iterated m times (lines 5-23) using only the examples in the original training set $\boldsymbol{O}$ as seeds for generating new adversarial examples. Our experiments, presented in Section 4.5, show that 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. Empirical tests also show that this improvement is mainly achieved in the first three iteration of the data augmentation process, hence we use $m = 3$. The computational complexity of Algorithm 5 is $O(m \cdot (|\boldsymbol{O}| - w) \cdot ADV)$, where ADV is the computational complexity of the algorithm used for generating adversarial examples, namely, H-ADV or L-ADV.

4.5 Results

We first describe the experimental setting and then analyze the results of our data augmentation technique comparing them against those obtained by baseline methods. Results are displayed for four application domains related to robotic and cyber-physical systems. We present results for each domain, focusing on the performance improvement achieved by the augmented detectors on different training set sizes. Furthermore, for some domains we also provide details about the mechanisms that generate the performance improvement. In particular, we show that the adversarial examples we generate are very similar to the original examples (to recall the image parallel, they are usually indistinguishable) and they have a specific direction of perturbation that causes performance improvement. In fact, perturbations of the same intensity but performed using one of the baseline methods (not adversarial) do not achieve any performance improvement. In the Tennessee Eastman domain, we also investigate the change of the anomaly score and threshold parameter produced by the augmented training set.

4.5.1 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), we assume to have a nominal HMM λ^N with K hidden states (chosen by the Bayesian Information Criterion BIC [64]) and trained on a training set $\boldsymbol{O}$ 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 $\boldsymbol{O}$ as described in [8]. With these three elements, namely, λ^N, w, and τ, a complete instance of the original 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 from the original anomaly detector. Here, we assume the properties of the original anomaly detector (e.g., windows size and input variables) as predefined and unchangeable, since we focus only on data augmentation for improving the performance of the detector.

The goal of our experiments is to evaluate the *performance improvement* achieved by the anomaly detector on the test set $\boldsymbol{T}$ when we update the HMM λ^N and threshold τ using the data augmentation and retraining method HHAD-AUG. In particular, we compare the F1-score of the *augmented anomaly detector* with that of the original detector on test set $\boldsymbol{T}$. Some dimensions of analysis are particularly interesting to evaluate the capabilities of HHAD-AUG in different situations. The first dimension is the *loss function* used to generate adversarial examples (i.e., Hellinger distance or likelihood). The second dimension is the *size of training set* $|\boldsymbol{O}|$, which is important because it shows the potential of the proposed approach on different amounts of training data. Table 4.1 summarizes the parameters of all experiments described in the following subsections. The parameters common to all application domains are here listed. The first one is the number of repetitions *#rep* of each test on different training sets of the same dimension, which is set to 30. In other words, given a training set size $|\boldsymbol{O}|$ we recompute the performance improvement 30 times, and each time we train the HMM λ^N and the threshold τ on different (random) training sets having size $|\boldsymbol{O}|$. The second parameter in common among all domains is the maximum perturbation size (for each dimension) ϵ, which is always set to 0.05. The last parameter in common is the number of iterations m in the data augmentation and retraining procedure HHAD-AUG, which is always set to 3.

The code is developed in Python. Baum-Welch, Viterbi, and other algorithms for training and inference on HMMs are based on `hmmlearn` library[7]. For baseline data augmentation algorithms we use the `tsaug`

[7] `https://hmmlearn.readthedocs.io/en/latest/`

Par.	Domain TE	SWaT	ALFA	INTCATCH
$\|O\|$	250,500,750, 1000,1250,1500	2500,5000, 10000,20000	250,500,750, 1000,1250,1500	250,500,750, 1000,1250,1500
#rep	30	30	30	30
$\|T\|$	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 4.1: *Summary of experimental parameters used in all application domains.*

library[8], a Python package for time series augmentation. In particular, function `AddNoise()` is used to generate random augmented samples for R-AUG. Function `Drift()` is instead used to generate drift-based augmented samples. `Cython`[9] [15] allows a considerable time improvement on algorithm L-ADV. `SciKitLearn`[10] [130] is used for data scaling and PCA. Experiments are run on a laptop with Intel Core i5 processor - 2.30 GHz x 4 cores, RAM 16 GB and operating system macOS 11.1. Performing 3 iterations of data augmentation on a training set of 1500 points and 4 variables takes on average 45.06 seconds using H-AUG and 44.11 seconds using L-AUG.

4.5.2 Performance Measures

Anomaly detection performance (algorithm HHAD) is evaluated by *F1-score* on a test set which is kept separated from the training set used to learn λ^N and τ. The formula for this score is:

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{4.31}$$

where *TP* are true positives, *TN* are true negatives, *FP* are false positives, and *FN* are false negatives [18]. We consider positive the nominal samples and negative the anomalous samples. Hence, true positives are nominal samples correctly classified by HHAD, true negatives are anomalous samples correctly detected by HHAD, and false positive (negatives) are anomalous (nominal) samples that are wrongly classified by the detector. The value of the F1-score must be maximized.

[8] `https://tsaug.readthedocs.io/en/stable/`
[9] `https://cython.org/`
[10] `https://scikit-learn.org/stable/`

Adversarial example generation (algorithms H-ADV and L-ADV) is instead evaluated by two measures. The minimum/average/maximum *distance between the original examples and the adversarial examples* is computed using norm L_2 and should be kept as small as possible to ensure a close similarity between original and adversarial examples. The *percent success rate* (*SR%*) is computed as the percentage of perturbed samples that are actually misclassified by HHAD. This measure should be kept as small as possible. Both distance and success rate depend on parameter ϵ and can be computed on the training set and on the test set. Larger values for ϵ allow more distant (i.e., dissimilar) adversarial examples and, consequently, an increased success rate. These measures do not provide absolute performance values, hence they are more suitable for comparing different methods on specific dataset/settings than for evaluating the absolute performance of the algorithm.

Data augmentation and retraining (algorithm HHAD-AUG) is evaluated by two measures. First, we compute the improvement of F1-score generated on HHAD by data augmentation and retraining. In particular, we measure the *difference between the F1-score after and before data augmentation and retraining* on the test set. To test the statistical significance of this difference we apply algorithm HHAD-AUG to several training sets containing different samples and having 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. The second performance measure aims at evaluating the improvement of robustness to adversarial attacks introduced by HHAD-AUG. We measure it as the difference in percent success rate *SR%* between original and augmented HHAD (see results in the next sections).

4.5.3 Domain D1: Tennessee-Eastman Industrial Chemical Process (TE)

Domain and dataset description. This domain is represented by a synthetic model of a real industrial chemical process realized for evaluating process control strategies [43]. The process produces two liquid products from four gaseous reactants, in addition to a byproduct and an inert, making a total of eight chemical components. This domain has recently become popular in the Industrial Control System (ICS) security community because it allows to test attack and defence approaches on a realistic (although simulated) environment. We used the dataset provided by [5] which employs the simulator proposed in [43] in the open-source Damn Vulnerable Chemical

Process (DVCP-TE) Simulink implementation[11]. The simulator is oriented towards security research and features support for performing attacks on both sensor and actuator signals. The control strategy used to keep the process stability is that presented in [97]. The dataset generated in [5] contains integrity attacks on both sensors and actuators. In particular, there are *stealth attacks* designed to slowly degrade the performance of the process and *direct damage attacks* which aim at damaging the physical equipment, driving the process to unsafe operating conditions (e.g., high temperature or pressure).

We use a dataset related to the stealth attack named SA1. The dataset has 41 variables and 4801 observations. A label is available for each sample, namely, 0 for nominal observations and 1 for anomalous observations. The training set is generated taking a slice of sequential nominal observations of length $|\boldsymbol{O}| \in \{250, 500, 750, 1000, 1250, 1500\}$ (see Table 4.1). The test set is a sequence containing $|\boldsymbol{T}| = 3201$ observations, of which 2400 are nominal and 801 anomalous. In particular, 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 then we compute average performance and related standard deviations on the performance observed 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 4.1).

Results. Figure 4.3 shows the main results. The x-axis represents the training set size $|\boldsymbol{O}|$ and the y-axis represents the F1-score. The blue solid line is the original detector HHAD with related 95%-confidence interval (shaded area). Dashed lines with other colors represent different data augmentation strategies, namely, orange is H-AUG, green is L-AUG, red is R-AUG and purple is D-AUG. The average performance improvement achieved by HHAD-AUG with Hellinger-based loss, called H-AUG, is statistically significant for all training set dimensions. We compute, 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

[11] https://github.com/satejnik/DVCP-TE

λ^N and τ). HHAD-AUG with loglikelihood-based loss, called L-AUG, provides a statistically significant improvement only for $|\boldsymbol{O}| = 750$ and $|\boldsymbol{O}| = 1500$. A motivation for this is reported in the next paragraph. Interestingly, HHAD augmented by baseline methods R-AUG and D-AUG does not achieve any significant performance improvement. Overall these results show that the proposed adversarial-based strategy for augmenting the training set is effective in this application domain.

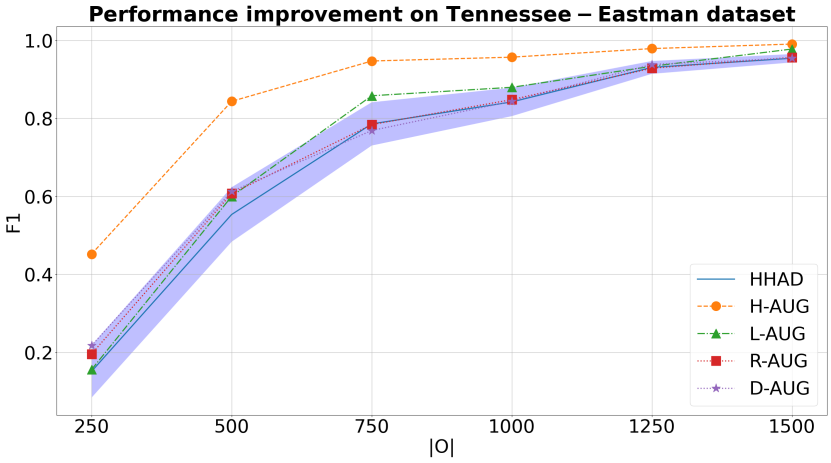


Figure 4.3: *Average F1-score for the original detector HHAD and augmented detectors H-AUG, L-AUG, R-AUG, D-AUG on different training set sizes of Tennessee-Eastman dataset. Average values are computed over 30 datasets, for each dataset size.*

Table 4.2 provides a quantitative evaluation of the performance improvement achieved by each data augmentation method with respect to the original detector HHAD. The first row shows the average F1-score μ_{F1} and related standard deviation σ_{F1} of the HHAD, where both statistics are computed over the 30 repeats performed for each training set size. Then, for each data augmentation algorithm we show in the white row the average F1-score and standard deviation, and in the gray row 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 to be statistically significant only if the p-value is less than 0.05, which corresponds to a confidence higher than 95% that the improvement is not null. These values are highlighted in bold in the table. We note that the improvement of H-AUG is larger with small training sets and it decreases as the training set size increases, hence the methodology could be used in applications with small amounts of data to improve the detection performance. For instance, the

F1-score of the augmented detector H-AUG trained with 500 samples is equivalent to the F1-score of the original detector HHAD trained with 1000 samples.

		Training set dimension $\|O\|$ (Tennessee Eastman)					
Detector	**Measure**	**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

Table 4.2: *Average F1-scores of the original anomaly detector HHAD and the augmented detectors H-AUG, L-AUG, R-AUG, and D-AUG on different training set sizes in the Tennessee-Eastman domain. Averages are computed over 30 datasets, for each dataset size. Average F1-score improvements Δ with respect to HHAD are also displayed with p-values for testing their statistical significance. Statistically significant performance improvements (having p-values < 0.05) are highlighted in bold.*

The 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 attacks generated by H-AUG and 0.15% of adversarial attacks generated by L-AUG are successful, namely, they managed to change the classification of HHAD, in each run of the algorithms. 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 sizes). The F1-score improvement of H-AUG with $|O| = 500$, for instance, is obtained using only 6.05 adversarial attacks 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.

The distribution of the $L2$ distance between the original training data and the related adversarial examples generated by H-AUG in our experiments is displayed in Figure 4.4. The minimum, median, and maximum L_2 distances from the original training data to the adversarial examples generated by H-AUG in our experiments are, respectively, 0.054, 0.476, and 0.803. Figure 4.5 shows the (four dimensional) time series of original (blue) and adversarial (red) examples for the attack having maximum distance. The difference between the two examples is clearly minimal.

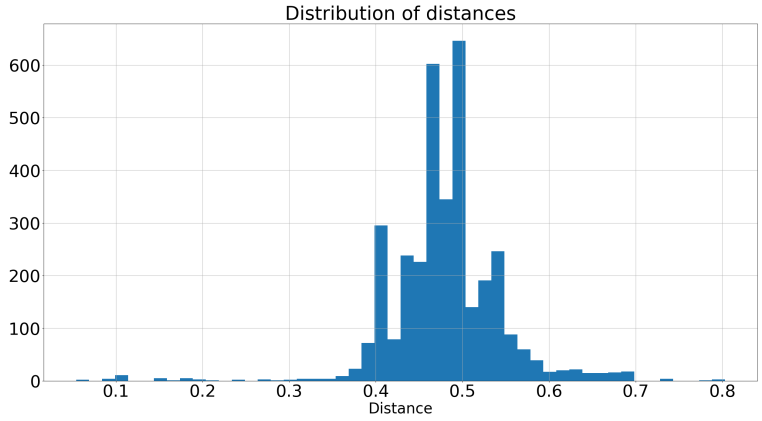


Figure 4.4: *Distribution of the $L2$ distance between the original training data and the related adversarial examples generated by H-AUG on the Tennessee Eastman dataset.*

Analysis of the mechanisms that generate the performance improvement. As a final inspection about the mechanisms that generate the performance improvement, we show in Figure 4.6 the Hellinger distance values on the test set (blue line) and the threshold τ (dashed black line) before and after data augmentation performed by H-AUG on the Tennessee Eastman dataset. These values are important because they describe the relationship between the test set samples and the decision boundary of the anomaly detector. Notice that all points in the white area on the left are nominal and all points in the red area on the right are anomalies. What we observe is that the data augmentation and retraining procedure slightly increases the threshold τ and, more importantly, it decreases and makes more stable the anomaly score (i.e., Hellinger distance) of the nominal values which are always below the threshold after the augmentation, except for a single false positive right before the start of the anomalous interval (see Figure 4.6.b) while many false positives were present before the augmentation (Figure 4.6.a). The stabilization and decrease of the anomaly score is due to the

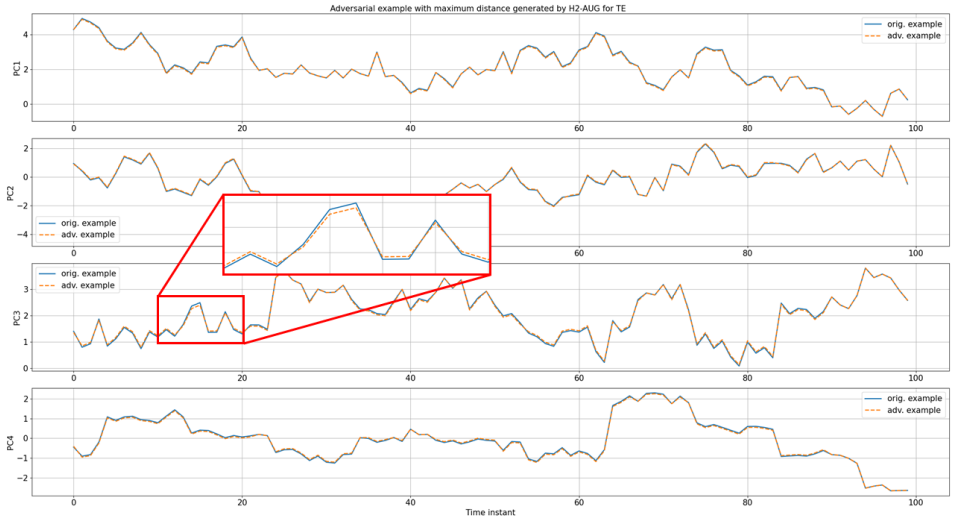


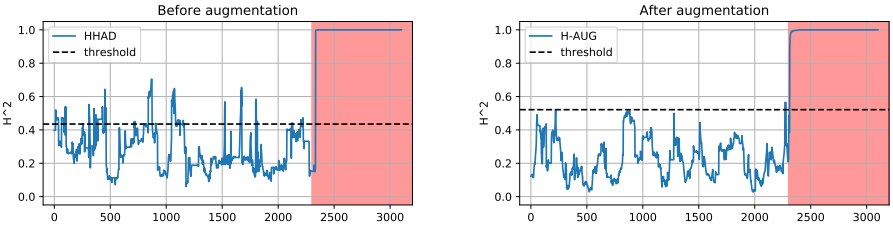
Figure 4.5: *Four-dimensional time series of the training sample (blue) and the related adversarial example (orange) for the attack having maximum distance in the experiments performed on the Tennessee-Eastman dataset.*

update of the parameters of the HMM. It clearly produces a performance improvement because all the points above the threshold in the white area are false positives (i.e., classified as anomalies even if they are nominal).

Table 4.3 shows the improvement of the robustness to adversarial attacks introduced by HHAD-AUG. The first three rows (after the heading) focus on the approach based on H-ADV and the second three rows on the approach based on L-ADV. We measure in particular the differences in the percent success rate $\Delta_{SR\%}$ of the detector before and after data augmentation and retraining, as explained in the end of Section 4.5.2. For both types of data augmentation, this difference is negative for all dataset sizes except $|\boldsymbol{O}| = 250$. The negative difference means that the percentage of successful adversarial attacks after augmentation and retraining is less than that before augmentation and retraining. The reduction is higher for H-AUG, with a maximum decrease of 8.6% than for L-AUG, which has a maximum decrease of 1.2%. Notice that these percentages are related to the dimension of the training set.

4.5.4 Domain D2: Secure Water Treatment Testbed (SWaT)

Domain and dataset description. SWaT is a scaled down version of a real-world industrial water treatment plant producing 18.92 liters per minute of



(a) *Before augmentation* **(b)** *After augmentation*

Figure 4.6: *Effect of the augmentation.* a) *Hellinger distance anomaly score before data augmentation performed by H-AUG;* b) *Hellinger distance anomaly score after data augmentation. The red areas represent parts of the test set containing anomalies.*

			Training set dimension $\|O\|$ (Tennessee Eastman)					
Detector	**Attack**	**Measure**	**250**	**500**	**750**	**1000**	**1250**	**1500**
HHAD	**H-ADV**	**SR%**	3.2%	11.2%	10.8%	13.1%	9.1%	5.5%
H-AUG	**H-ADV**	**SR%**	7.9%	7.7%	4.1%	4.5%	2.3%	1.5%
		$\Delta_{SR\%}$	4.7%	**-3.5%**	**-6.7%**	**-8.6%**	**-6.9%**	**-3.9%**
HHAD	**L-ADV**	**SR%**	0.7%	4.34%	6.25%	6.05%	5.0%	3.62%
L-AUG	**L-ADV**	**SR%**	0.8%	4.32%	5.0%	5.9%	4.6%	2.6%
		$\Delta_{SR\%}$	0.11	**-0.02**	**-1.2%**	**-0.1%**	**-0.4%**	**-1.0%**

Table 4.3: *Average success rate* SR% *of adversarial attacks H-ADV and L-ADV on the test set of the Tennessee-Eastman domain for detectors HHAD (original), H-AUG (augmented with Hellinger-based loss), and L-AUG (augmented with likelihood-based loss). Averages are computed over 30 datasets, for each dataset size. The average improvement* $\Delta_{SR\%}$ *with respect to HHAD is also displayed.*

water filtered via membrane-based ultrafiltration and reverse osmosis units [110]. This plant allowed data collection under two behavioural modes: normal and attacked. SWaT was run non-stop from its empty state to fully operational state for a total of 11 days. During this period, the first seven days the system operated normally i.e. without any attacks or faults. During the remaining days, some cyber and physical attacks were launched on SWaT while data collection continued. The dataset [51] was released to support research in the design of secure Cyber Physical Systems (CPS) in the context of modern ICS. Collected data corresponds to 51 sensor and actuator signals. During the data collection, the SWaT testbed undergoes 41 different attacks, five of them with no physical impact [5].

The dataset that we use has 51 variables and 495000 time steps acquired with sampling interval of 1 second. A label (i.e., 0 for nominal data and 1 for anomalous data) is available for each observation. The training set is generated taking slices of sequential nominal observations of length

$|\boldsymbol{O}| \in \{2500, 5000, 10000, 20000\}$ (see also Table 4.1). The test set is a time series containing $|\boldsymbol{T}| = 449919$ observations, of which 395298 are nominal and 54621 anomalous. In particular, the training sets are selected in the interval between time steps 2500 and 492500, while the test set is used in its completeness. For each training set size we generate 30 training sets (sampling the original dataset in different positions) and then we compute average performance and related standard deviations on the test set.

Experimental parameters. Given a specific training set, the dimensionality is reduced by selecting the first 2 principal components computed by standard PCA. The number of hidden states K of the nominal HMM λ^N is then selected by BIC in the interval $[2, 25]$. Diagonal covariance matrix is used. The window length is $w = 50$. The maximum perturbation size is $\epsilon = 0.05$. The number of iterations of the data augmentation and retraining procedure is $m = 3$ (see Table 4.1).

Results. The average performance improvement achieved by both H-AUG and L-AUG with respect to HHAD is statistically significant for all training set sizes, as graphically shown in Figure 4.7. Table 4.4 provides quantitative details about the improvement. Baseline augmented detectors R-AUG and D-AUG in this case manage to improve the average $F1$-score only for the smaller dataset (having 2500 samples) but the improvement (i.e., 0.032) is smaller than that achieved by H-AUG (i.e., 0.048) and L-AUG (i.e., 0.063). Moreover, for larger training set sizes the two baseline methods achieve negative or null improvement, while the proposed methods always get significant improvement. The improvement stays almost constant (between 6% and 3%) as the training set size increases, showing that the F1-score can still increase even when new data are added to an already large training set. Also in this case the gain is relevant, since the F1-score obtained by the detector augmented with H-ADV on 5000 samples (about 0.89) is larger than the F1-score obtained by the original detector HHAD using 20000 samples, inducing a saving of about 15000 samples.

In this dataset the percentage of successful attacks is on average 1.59% for H-AUG and 0.58% for L-AUG (the average is computed on training sets of different sizes). These correspond to an average of 87.06 adversarial examples added to the training set by H-AUG and 33.31 by L-AUG. We notice that these numbers are much larger than those of the Tennessee-Eastman dataset because the training set sizes are also larger.

Table 4.5 shows the improvement of the robustness to adversarial attacks

Figure 4.7: *Average F1-score for the original detector HHAD and augmented detectors H-AUG, L-AUG, R-AUG, D-AUG on different training set sizes in the SWaT dataset. Averages are computed over 30 datasets, for each dataset size.*

introduced by HHAD-AUG. For both data augmentation types H-AUG and L-AUG, the difference $\Delta_{SR\%}$ between percentage success rate before and after the augmentation is negative for all dataset sizes, meaning that the percentage of successful adversarial attacks after augmentation and retraining is less than that before augmentation and retraining. The reduction is slightly higher for H-AUG, with a maximum decrease of 6.6%, than for L-AUG, which has a maximum decrease of 5.1%.

4.5.5 Domain D3: UAV Fault and Anomaly Detection (ALFA)

Domain and dataset description. This dataset[12] presents several fault types in control surfaces of a fixed-wing Unmanned Aerial Vehicle (UAV) for use in Fault Detection and Isolation (FDI) and Anomaly Detection (AD) research [83]. The dataset 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 ground truth of the time and type of faults is provided in each scenario to enable the evaluation of new methods using the dataset. The platform used for collecting the dataset is a custom modification of the Carbon Z T-28 model plane. The plane has 2 meters of wingspan, a single electric engine in the front, ailerons, flaperons, an elevator, and a rudder.

[12] http://theairlab.org/alfa-dataset

		Training set dimension $\|O\|$ (SWaT)			
Detector	**Measure**	**2500**	**5000**	**10000**	**20000**
HHAD	μ_{F1}	0.801	0.849	0.863	0.883
	σ_{F1}	0.054	0.030	0.024	0.028
H-AUG	μ_{F1}	0.849	0.887	0.892	0.911
	σ_{F1}	0.029	0.015	0.019	0.017
	Δ_{F1}	**0.048**	**0.038**	**0.029**	**0.028**
	p-val	**7.2e-06**	**9.7e-09**	**1.8e-07**	**6.3e-07**
L-AUG	μ_{F1}	0.864	0.892	0.902	0.914
	σ_{F1}	0.027	0.031	0.017	0.015
	Δ_{F1}	**0.063**	**0.043**	**0.039**	**0.031**
	p-val	**5.1e-06**	**6.0e-09**	**9.4e-09**	**9.7e-08**
R-AUG	μ_{F1}	0.833	0.837	0.843	0.885
	σ_{F1}	0.023	0.016	0.014	0.009
	Δ_{F1}	**0.032**	-0.012	-0.020	0.002
	p-val	**0.009**	0.061	7.6e-04	0.674
D-AUG	μ_{F1}	0.830	0.836	0.846	0.883
	σ_{F1}	0.028	0.015	0.016	0.009
	Δ_{F1}	**0.029**	-0.013	-0.016	2e-4
	p-val	**0.011**	0.031	0.004	0.975

Table 4.4: *Average F1-scores of the original anomaly detector HHAD and the augmented detectors H-AUG, L-AUG, R-AUG, and D-AUG on different training set sizes in the SWaT dataset. Averages are computed over 30 datasets, for each dataset size. Average F1-score improvements Δ with respect to HHAD are also displayed with p-values for testing their statistical significance. Statistically significant performance improvements (having p-values < 0.05) are highlighted in bold.*

			Training set dimension $\|O\|$ (SWaT)			
Detector	**Attack**	**Measure**	**2500**	**5000**	**10000**	**20000**
HHAD	**H-ADV**	**SR%**	13.2%	10.1%	7.6%	6.2%
H-AUG	**H-ADV**	**SR%**	6.5%	4.8%	7.8%	2.4%
		$\Delta_{SR\%}$	**-6.6%**	**-5.3%**	**-2.9%**	**-3.8%**
HHAD	**L-ADV**	**SR%**	9.3%	8.1%	7.0%	5.2%
L-AUG	**L-ADV**	**SR%**	4.1%	3.7%	4.0%	2.7%
		$\Delta_{SR\%}$	**-5.1%**	**-4.4%**	**-3.1%**	**-2.6%**

Table 4.5: *Average success rate* SR% *of adversarial attacks H-ADV and L-ADV on the test set of the SWaT domain for detectors HHAD (original), H-AUG (augmented with Hellinger-based loss), and L-AUG (augmented with likelihood-based loss). Averages are computed over 30 datasets, for each dataset size. The average improvement $\Delta_{SR\%}$ with respect to HHAD is also displayed.*

The aircraft is equipped with a Holybro PX4 2.4.6 autopilot, a Pitot Tube, a GPS module, an Nvidia Jetson TX2 onboard computer and a radio for communication with the ground station. The onboard computer uses Robot

Operating System (ROS) Kinetic Kame on Linux Ubuntu 16.04 (Xenial) to read the flight and state information from the Pixhawk using MAVROS package (the MAVLink node for ROS).

The part of the dataset that we use is related to the run "carbonZ_2018-07-30-16-29-45_engine_failure_with_emr_traj". It has 19 variables and 2668 time steps acquired with sampling frequency of approximately 20 Hz. A label (i.e., 0 for nominal sample and 1 for anomalous sample) is available for each observation. The training set is generated taking a slice of sequential nominal observations length $|\boldsymbol{O}| \in \{250, 500, 750, 1000, 1250, 1500\}$ (see also Table 4.1). The test set is a time series containing $|\boldsymbol{T}| = 1068$ observations, of which 718 are nominal and 350 anomalous. In particular, 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 2668. For each training set size we generate 30 training set (sampling the original dataset in different positions) and then we compute average performance with related standard deviations and distributions.

Experimental parameters. The dimensionality is reduced by selecting the first 3 principal components computed by standard PCA (see Table 4.1). The number of hidden states K of the nominal HMM λ^N is then selected by BIC in the interval $[2, 20]$. Diagonal covariance matrix is used. The window length is $w = 100$. The maximum perturbation size is set to $\epsilon = 0.05$. The number of iterations of the data augmentation and retraining procedure is set to $m = 3$.

Results. The average performance improvement achieved by H-AUG with respect to HHAD on the ALFA domain is statistically significant for all training set dimensions. Figure 4.8 graphically shows these results and Table 4.6 provides all average performance values with improvement and p-values. L-AUG has on average better F1-score than HHAD and the difference is statistically significant for all training set dimensions except for $|\boldsymbol{O}| = 250$, however the magnitude of the improvement is less than that achieved by H-AUG. Baseline methods R-AUG and D-AUG do not achieve any significant performance improvement except for a small improvement obtained by D-AUG on $|\boldsymbol{O}| = 1500$. 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 $|\boldsymbol{O}| = 500$. Out of the four, this is certainly the most difficult dataset containing complex behaviours of real intelligent 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.

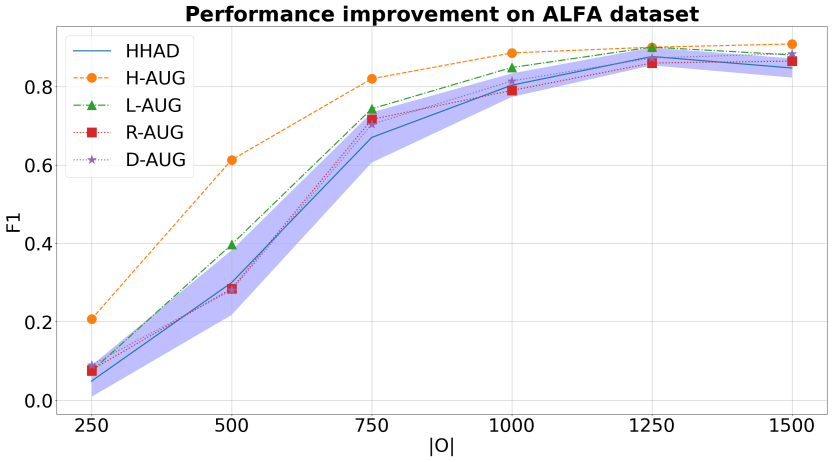


Figure 4.8: *Average F1-score for the original detector HHAD and augmented detectors H-AUG, L-AUG, R-AUG, D-AUG on different training set sizes in the ALFA dataset. Averages are computed over 30 datasets, for each dataset size.*

Table 4.7 shows the improvement of the robustness to adversarial attacks introduced by HHAD-AUG. The augmentation based on H-AUG has negative difference $\Delta_{SR\%}$ between percentage success rate before and after the augmentation for all dataset sizes, while in L-AUG this difference is negative (or null) for all dataset sizes except $|\boldsymbol{O}| = 1500$. Therefore the detector improves its robustness to adversarial attacks in almost all cases. The reduction is higher for H-AUG and larger training sets, with a maximum decrease of 7.8% with $|\boldsymbol{O}| = 1000$. The maximum reduction for L-AUG is reached also on $|\boldsymbol{O}| = 1000$ with a decrease of 2.3%.

4.5.6 Domain D4: Water Monitoring with ASV (INTCATCH)

Domain and dataset description. This dataset is the one used also in the previous chapter collected as part of INTCATCH Project[13] [28–30]. The dataset consists of observations, collected at 1 Hz frequency, of the following variables concerning the robot state: heading (i.e., compass direction), speed, acceleration, power signals to the left and right propellers, latitude, and longitude. Two runs, one completely nominal, used for training, and another partially anomalous, used for testing, are available. The runs have 9 variables, 5739 train samples and 6620 test samples acquired with sampling

[13] https://www.intcatch.eu

		Training set dimension $\|O\|$ (ALFA)					
Detector	**Measure**	**250**	**500**	**750**	**1000**	**1250**	**1500**
HHAD	μ_{F1}	0.049	0.300	0.670	0.804	0.877	0.847
	σ_{F1}	0.106	0.217	0.169	0.079	0.060	0.065
H-AUG	μ_{F1}	0.207	0.613	0.820	0.886	0.900	0.909
	σ_{F1}	0.283	0.274	0.080	0.039	0.027	0.027
	Δ_{F1}	**0.158**	**0.313**	**0.150**	**0.082**	**0.024**	**0.061**
	p-val	**0.001**	**1.9e-06**	**2.6e-04**	**6.5e-06**	**0.044**	**2.4e-05**
L-AUG	μ_{F1}	0.074	0.397	0.743	0.849	0.901	0.881
	σ_{F1}	0.146	0.278	0.156	0.069	0.052	0.059
	Δ_{F1}	0.025	**0.097**	**0.073**	**0.045**	**0.024**	**0.033**
	p-val	0.051	**0.024**	**0.038**	**8.5e-4**	**0.020**	**0.004**
R-AUG	μ_{F1}	0.077	0.285	0.716	0.790	0.860	0.865
	σ_{F1}	0.134	0.255	0.138	0.081	0.065	0.075
	Δ_{F1}	0.028	-0.016	0.046	-0.014	-0.017	0.018
	p-val	0.316	0.694	0.286	0.547	0.108	0.171
D-AUG	μ_{F1}	0.090	0.281	0.704	0.814	0.873	0.884
	σ_{F1}	0.142	0.237	0.180	0.091	0.052	0.051
	Δ_{F1}	0.041	-0.019	0.034	0.011	-0.004	**0.037**
	p-val	0.173	0.608	0.391	0.567	0.727	**0.002**

Table 4.6: *Average F1-scores of the original anomaly detector HHAD and the augmented detectors H-AUG, L-AUG, R-AUG, and D-AUG on different training set sizes in the ALFA dataset. Averages are computed over 30 datasets, for each dataset size. Average F1-score improvements Δ with respect to HHAD are also displayed with p-values for testing their statistical significance. Statistically significant performance improvements (having p-values < 0.05) are highlighted in bold.*

			Training set dimension $\|O\|$ (ALFA)					
Detector	**Attack**	**Measure**	**250**	**500**	**750**	**1000**	**1250**	**1500**
HHAD	**H-ADV**	**SR%**	0.6%	4.1%	7.9%	9.2%	5.5%	7.6%
H-AUG	**H-ADV**	**SR%**	0.1%	3.9%	4.0%	1.4%	2.4%	1.6%
		$\Delta_{SR\%}$	**-0.5%**	**-0.2%**	**-3.9%**	**-7.8%**	**-3.0%**	**-6.0%**
HHAD	**L-ADV**	**SR%**	0.6%	6.5%	8.4%	7.4%	6.7%	4.8%
L-AUG	**L-ADV**	**SR%**	0.6%	6.2%	6.5%	5.0%	5.6%	5.1%
		$\Delta_{SR\%}$	**0.0%**	**-0.3%**	**-2.00%**	**-2.3%**	**-1.1%**	0.3%

Table 4.7: *Average success rate* SR% *of adversarial attacks H-ADV and L-ADV on the test set of the ALFA domain for detectors HHAD (original), H-AUG (augmented with Hellinger-based loss), and L-AUG (augmented with likelihood-based loss). Averages are computed over 30 datasets, for each dataset size. The average improvement $\Delta_{SR\%}$ with respect to HHAD is also displayed.*

frequency of 1 Hz. A label (i.e., 0=nominal, 1=anomaly) is available for each observation. The training set is generated taking a slice of sequential nominal observations length $|\boldsymbol{O}| \in \{250, 500, 750, 1000, 1250, 1500\}$ (see

also Table 4.1). The test set is a time series containing $|\boldsymbol{T}| = 6620$ observations, of which 3783 are nominal and 2837 anomalous. For each training set size we generate 30 training set (sampling the original dataset in different positions) and then we compute average performance with related standard deviations and distributions.

Experimental parameters. The dimensionality is reduced by selecting the first 4 principal components computed by standard 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 $w = 100$. The maximum perturbation size is set to $\epsilon = 0.05$ (see Table 4.1).

Results. The average performance improvement of H-AUG with respect to HHAD is statistically significant for training set sizes of 250, 500, and 750 samples, while it is negligible for sizes of 1000, 1250, and 1500 samples, as shown in Figure 4.9. L-AUG has instead statistically significant improvement for training set sizes of 500 and 750, and negligible improvement for the other sizes. The figure shows that the performance stops growing at $|\boldsymbol{O}| = 1000$, where the F1-score reaches a plateau of about 0.820 (see also Table 4.8). A possible motivation for this is that the signals provided to the detector intrinsically contain only a part of the information needed to perform anomaly detection. In this case it would be impossible to get a performance improvement. The plateau of the performance improvement after $|\boldsymbol{O}| = 1000$ should be ascribed to the fact that the task is repetitive. In particular, the water drone performs 6 cycles (from bottom to top and back) to cover the monitoring area, see also [8]. One of these cycles lasts approximately 950 time instants, hence training sizes larger than that value gather almost no new information. The baseline methods for data augmentation and retraining, i.e., R-AUG and D-AUG, have performance similar to HHAD for all dataset sizes.

Table 4.9 shows the improvement of the robustness to adversarial attacks introduced by HHAD-AUG. Values of $\Delta_{SR\%}$ are negative for almost all combinations of augmentation method and training set size, also in this case. H-AUG reaches a maximum reduction of successful attacks of 4.7% with $|\boldsymbol{O}| = 750$, while L-AUG reaches its best improvement on the robustness to adversarial attacks on $|\boldsymbol{O}| = 1000$ with a reduction of 0.7%. The reduction is higher for H-AUG and medium size training sets.

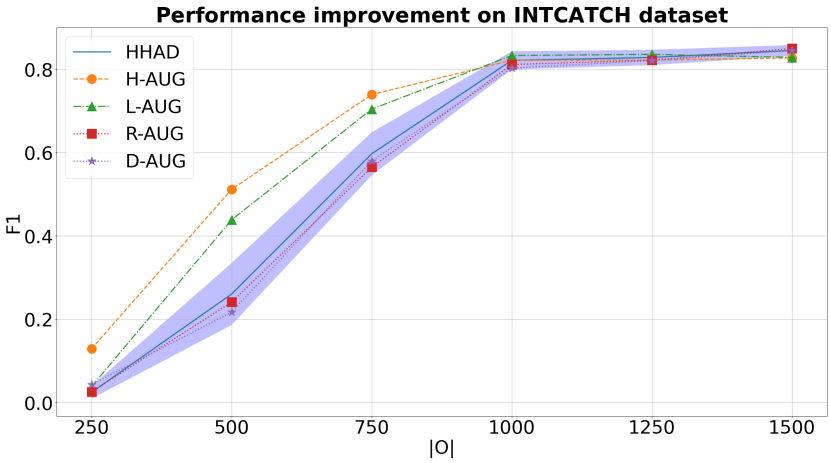


Figure 4.9: *Average F1-score for the original detector HHADand augmented detectors H-AUG, L-AUG, R-AUG, D-AUG on different training set sizes in the INTCATCH dataset. Averages are computed over 30 datasets, for each dataset size.*

4.6 Concluding Remarks

Detection of anomalous behaviors in intelligent systems, such as autonomous robotic and cyber-physical systems, requires tools able to represent their nominal behaviors and discover patterns that do not satisfy them in sensor traces. HMMs are a viable and established tool for abstracting dynamical behaviors contained in multivariate time series. The methodology of adversarial data augmentation presented in this work allows to improve the detection performance of such tools without using any prior knowledge about the form of the nominal time series, as traditional data augmentation methods require. For each domain we analyzed, our method showed a statistically significant performance improvement while standard data augmentation methods based on random perturbation showed no performance improvement, proving that our strategy to generate adversarial examples is the cause of the improvement.

Despite their simplicity, HMMs have proven to be a powerful and capable tool that can be successfully employed in many practical applications. However, when robotic tasks become very complex, there may be the need of an even more powerful formalism. As a consequence, the rest of this thesis is devoted to develop anomaly detectors based on a different underlying model: variational autoencoders.

		Training set dimension $\|O\|$ (INTCATCH)					
Detector	**Measure**	**250**	**500**	**750**	**1000**	**1250**	**1500**
HHAD	μ_{F1}	0.025	0.261	0.597	0.822	0.828	0.845
	σ_{F1}	0.040	0.196	0.136	0.057	0.048	0.034
H-AUG	μ_{F1}	0.129	0.512	0.739	0.822	0.822	0.827
	σ_{F1}	0.134	0.163	0.088	0.049	0.041	0.048
	Δ_{F1}	**0.105**	**0.251**	**0.142**	2.0e-4	-0.006	-0.018
	p-val	**5.3e-05**	**2.1e-09**	**2.9e-08**	0.983	0.547	0.080
L-AUG	μ_{F1}	0.039	0.439	0.705	0.833	0.836	0.829
	σ_{F1}	0.062	0.212	0.115	0.051	0.044	0.036
	Δ_{F1}	0.014	**0.178**	**0.107**	0.012	0.007	-0.016
	p-val	0.178	**5.5e-05**	**2.5e-05**	0.247	0.379	0.047
R-AUG	μ_{F1}	0.026	0.241	0.565	0.812	0.822	0.850
	σ_{F1}	0.059	0.180	0.154	0.060	0.066	0.044
	Δ_{F1}	0.001	-0.020	-0.032	-0.010	-0.007	0.005
	p-val	0.899	0.371	0.014	0.133	0.522	0.528
D-AUG	μ_{F1}	0.043	0.217	0.580	0.803	0.821	0.845
	σ_{F1}	0.0845	0.180	0.162	0.056	0.062	0.044
	Δ_{F1}	0.019	-0.043	-0.017	-0.019	-0.007	0.0
	p-val	0.194	0.018	0.165	0.011	0.469	1.0

Table 4.8: *Average F1-scores of the original anomaly detector HHAD and the augmented detectors H-AUG, L-AUG, R-AUG, and D-AUG on different training set sizes in the INTCATCH dataset. Averages are computed over 30 datasets, for each dataset size. Average F1-score improvements Δ with respect to HHAD are also displayed with p-values for testing their statistical significance. Statistically significant performance improvements (having p-values <0.05) are highlighted in bold.*

			Training set dimension $\|O\|$ (INTCATCH)					
Detector	**Attack**	**Measure**	**250**	**500**	**750**	**1000**	**1250**	**1500**
HHAD	**H-ADV**	**SR%**	0,8%	5,7%	7,6%	6,4%	6,0%	3,1%
H-AUG	**H-ADV**	**SR%**	1,1%	1,9%	2,9%	2,2%	2,2%	1,9%
		$\Delta_{SR\%}$	0,4%	**-3,8%**	**-4,7%**	**-4,2%**	**-3,9%**	**-1,2%**
HHAD	**L-ADV**	**SR%**	0,2%	2,0%	2,6%	2,3%	2,1%	1,5%
L-AUG	**L-ADV**	**SR%**	0,2%	1,9%	2,0%	1,6%	1,8%	1,6%
		$\Delta_{SR\%}$	**0,0%**	**-0,1%**	**-0,6%**	**-0,7%**	**-0,3%**	0,1%

Table 4.9: *Average success rate* SR% *of adversarial attacks H-ADV and L-ADV on the test set of the INTCATCH domain for detectors HHAD (original), H-AUG (augmented with Hellinger-based loss), and L-AUG (augmented with likelihood-based loss). Averages are computed over 30 datasets, for each dataset size. The average improvement $\Delta_{SR\%}$ with respect to HHAD is also displayed.*

CHAPTER 5

A VAE-based Anomaly Detector for Single-Robot Systems

Part of the work contained in this chapter has been published in IEEE Robotics and Automation Letters (RA-L) [7].

5.1 Introduction

Different data-driven methods for detecting anomalies in robot systems have been proposed, most of them being semi-supervised and requiring big batches (especially the deep learning-based ones) of observations labeled as nominal by human experts for their training (see Section 5.2.1).

In this chapter, a new deep learning-based *minimally supervised* method for detecting anomalies in autonomous robots is proposed. In particular, we propose a new VAE (Variational Auto-Encoder) [90] architecture able to model very long multivariate sensor logs of a robot performing a task. We also introduce a new incremental method for training VAEs, which induces a progress-based latent space that can be used to detect anomalies both online (at runtime) and offline. An original feature of our approach is that, differently from most approaches for anomaly detection in robotics,

it is trained with unlabeled observations, possibly including both nominal and anomalous executions. Only few (even just one) labeled nominal executions are then required to partition the learned latent space into nominal and anomalous regions. This *minimal supervision* provides a big advantage over semi-supervised approaches in practical settings, where collecting several nominal runs of a robot performing a task could be hard. This is true especially when using deep learning-based algorithms which notoriously demand big datasets to be trained effectively.

As the proposed method is trained with unlabeled data (i.e., like in unsupervised learning), but needs at least one execution labeled as nominal in order to detect anomalies, we call it *minimally supervised*.

Similarly to semi-supervised approaches, our method does not assume anomalies to be rarely occurring (like unsupervised approaches) or already known in advance (like supervised approaches). Another interesting feature of the proposed approach is that, similarly to supervised approaches, and provided that very few anomalously labeled examples are provided (at least one for each known anomaly type), it not only can detect that something wrong is happening, but can also discern between different anomaly types (i.e., diagnosis), still without assuming that every future anomaly will belong to one of the already known types (more on this later).

Experimental results on datasets collected from real robots show that our method outperforms state-of-the-art methods for anomaly detection in robots both in terms of false positive rate and alert delay.

This chapter thus presents a novel application of VAEs to anomaly detection in autonomous robots and provides the following main original contributions:

- A new VAE architecture and a new training method, which induce a progress-based latent space that is suitable to detect anomalies (Sections 5.3.3 and 5.3.4).
- A new online and a new offline anomaly detection algorithms that require as little as one execution labeled as nominal (Sections 5.3.5 and 5.3.6).
- An experimental evaluation on datasets collected from real robots involved in three applications requiring LTA (Section 5.4).

5.2 Background on Deep Autoencoders

Recently, deep learning models have been employed to re-address several spatio-temporal modeling tasks, including those relative to anomaly detec-

tion in robotics, providing significant improvements over classical state-of-the-art methods. We first introduce some background on these techniques and then survey the literature most relevant for our contribution.

Autoencoders (AEs) [70] are particular kinds of artificial neural networks which are trained to reconstruct their input, in a self-supervised manner. An AE is composed of an *encoder* network and a *decoder* network. The encoder takes as input the training data $x \in \mathbb{R}^d$, where d is the dimension of the data, and compresses these data into a *latent space* $z \in \mathbb{R}^h$, where h is the dimension of the encoding, usually $h < d$. Then, the decoder tries to map back the latent internal representation z to the original input space $\hat{x} \in \mathbb{R}^d$, through reconstruction. The encoder structure can be considered as a bottleneck, in which data pass and are compressed to extract a meaningful encoded representation. The decoder does the opposite. The two networks are characterized by f_ϕ, the encoding function, and f_θ, the decoding function, where $f_\phi : \mathbb{R}^d \rightarrow \mathbb{R}^h$ and $f_\theta : \mathbb{R}^h \rightarrow \mathbb{R}^d$. Finding weights (parameters) ϕ and θ for the two functions can be done by backpropagation, minimizing the loss function $\mathcal{L}_{AE}(x, \hat{x}) = \parallel x - \hat{x} \parallel^2$, called *reconstruction error*, given input x and model output $\hat{x}$.

Variational Autoencoders (VAEs) [90] differ from plain AEs in the fact that they assume the existence of a probabilistic model parametrized by the latent variable $z \in \mathbb{R}^h$ that generates the observed input values $x \in \mathbb{R}^d$. Being z latent (i.e., hidden), we can infer its characteristics by computing the posterior $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$; unfortunately, computing the marginalization $p(x) = \int p(x|z)p(z)dz$ at the denominator is intractable when z is high-dimensional. *Variational inference* can be used to overcome this issue by approximating $p(z|x)$ with a distribution $q(z|x)$ which is tractable and by minimizing their *KL-divergence* $D_{KL}(q(z|x) \parallel p(z|x))$ to ensure that $q(z|x)$ is similar to $p(z|x)$. By replacing $p(z|x)$ in $D_{KL}(q(z|x) \parallel p(z|x))$ with $\frac{p(x,z)}{p(x)}$, the minimization problem becomes equivalent to the maximization of $\mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x) \parallel p(z))$, where the first term represents the reconstruction likelihood (analogous to the reconstruction error in AEs), while the second term ensures that the learned distribution $q(z|x)$ is similar to the true prior distribution $p(z)$ (with the effect of regularizing the latent variable z). The distributions $q(z|x)$ and $p(z|x)$ can be parametrized by means of two artificial neural networks which can be considered as encoder (with weights ϕ), and decoder (with weights θ), respectively. Finding weights parameters ϕ and θ can be done by backpropagation, minimizing the loss function $\mathcal{L}_{VAE}(x) = D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) - \mathbb{E}_{q_\phi(z|x)}(\log p_\theta(x|z))$, which represents the variational lower bound of the data x according to the

Jensen's inequality (see [90] for full details). The posterior $q_\phi(z|x)$ is usually assumed to be normally distributed with parameters $\mathcal{N}(\mu_z, \Sigma_z)$, while a common choice for the prior distribution $p_\theta(z)$ is an isotropic normal distribution $\mathcal{N}(0, I)$.

5.2.1 AEs and VAEs for Anomaly Detection

The main idea behind the current use of AEs for anomaly detection is to train them only on nominal data so that they will not be able to accurately reconstruct anomalous behaviors (that the AEs have never seen), which will thus produce high reconstruction errors. AEs have been widely used for anomaly detection on time series, and more rarely on data coming from robots. For example, authors of [109] propose an LSTM (Long Short-Term Memory) based encoder decoder (EncDec-AD) that learns to reconstruct nominal time series and thereafter uses reconstruction error of observed samples to detect anomalies. Similarly, [169] proposes a Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) to perform anomaly detection and diagnosis in multivariate time series data. One significant example of application of AEs to detect anomalies in robots is [119], where the authors propose to convert sensor logs into images and then use a convolutional AE to detect anomalous behaviors resulting from cyber-security attacks.

After the introduction of VAEs, a lot of interest has developed around this new framework due to the continuity of its latent space and to its ability of producing probabilistic anomaly scores (see below) which are in general more powerful than AEs' reconstruction error. In [127], the authors propose an LSTM-VAE-based detector using a reconstruction-based anomaly score and a state-based threshold to detect anomalies for robot-assisted feeding, while authors of [150] apply the STORN model [14] to anomaly detection by introducing a trending prior on the latent representation. The AE/VAE-based approaches described so far use some variants of the reconstruction error as anomaly score. However, after its adoption in [4], the *reconstruction probability* $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ has become a more popular anomaly score than the reconstruction error. In this case, for each sample from the encoder, the probabilistic decoder outputs the mean and variance parameters of the reconstruction instead of the reconstructed value itself. Then, the reconstruction probability is calculated using the output parameters of the encoder, given the original input as a sample. For instance, in [36], a sliding-window convolutional variational autoencoder (SWCVAE) is proposed, which can perform real-time anomaly detection on multivariate time

series acquired from an industrial robot. Another example of the use of the reconstruction probability is in [131], which presents a variational recurrent autoencoder with attention.

All the above approaches, and many others, are trained on nominal data in a semi-supervised fashion. However, some AE/VAE-based approaches trained in an unsupervised fashion exist, such as [167], which proposes an unsupervised anomaly detection algorithm based on a VAE to detect anomalies on web servers usage time series. Also in this case, detection exploits the reconstruction probability. Another interesting approach has been proposed in [132], whose model is trained in a fully unsupervised fashion and applied to univariate healthcare time series in which anomalies are detected directly in the latent space by computing the Wasserstein distance between a test sample latent representation and other encoded samples in the test set.

Our method employs VAEs but differs from semi-supervised approaches in requiring only minimal supervision, and from unsupervised ones in modelling multivariate time series and not assuming rarity of anomalies.

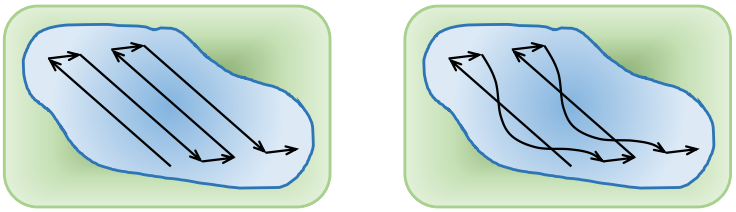
5.3 The Proposed Method

5.3.1 Problem Definition

We represent by $\boldsymbol{O} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_n\}$ a d-dimensional time series composed of n observations of a robot system performing a task, where each observation $\boldsymbol{o}_t$ is a d-dimensional vector representing the multivariate (multi-valued) observation of the robot at (discrete) time step t. Typically, $\boldsymbol{o}_t$ is extracted from sensors logs of the robot. We assume to know at least one nominal execution $\boldsymbol{O}^N = \{\boldsymbol{o}_1^N, ..., \boldsymbol{o}_{n_N}^N\}$ corresponding to the robot correctly performing its task. If the robot can display k different types of nominal behaviors when performing its task, we assume the availability of at least a nominal execution $\boldsymbol{O}^N$ for each of them. The observed behavior of the same robot along some time period is denoted by $\boldsymbol{O}^O = \{\boldsymbol{o}_1^O, ..., \boldsymbol{o}_{n_O}^O\}$. Informally, we consider an observed run $\boldsymbol{O}^O$ to be anomalous if there exists a time step $t_A \leq n_O$ from which on $\boldsymbol{O}^O$ starts to differ from one of the nominal executions $\{\boldsymbol{O}^N\}$. The details on the way in which an observation is classified as anomalous are discussed in Sections 5.3.5 and 5.3.6.

If we consider $\boldsymbol{O}^O$ as a (possibly infinite) data stream, *online anomaly detection* at time step t is the task of classifying the portion of the stream up to t as anomalous or non-anomalous w.r.t. $\{\boldsymbol{O}^N\}$.

Given a finite time series of observations $\boldsymbol{O}^O$, *offline anomaly detection*



(a) (b)

Figure 5.1: *Schematic representation of water drone nominal (a) and anomalous (b) behaviors, as they appear in the original data.*

is the task of classifying the whole behavior displayed by the robot in $\boldsymbol{O}^O$ as anomalous or non-anomalous w.r.t. $\{\boldsymbol{O}^N\}$.

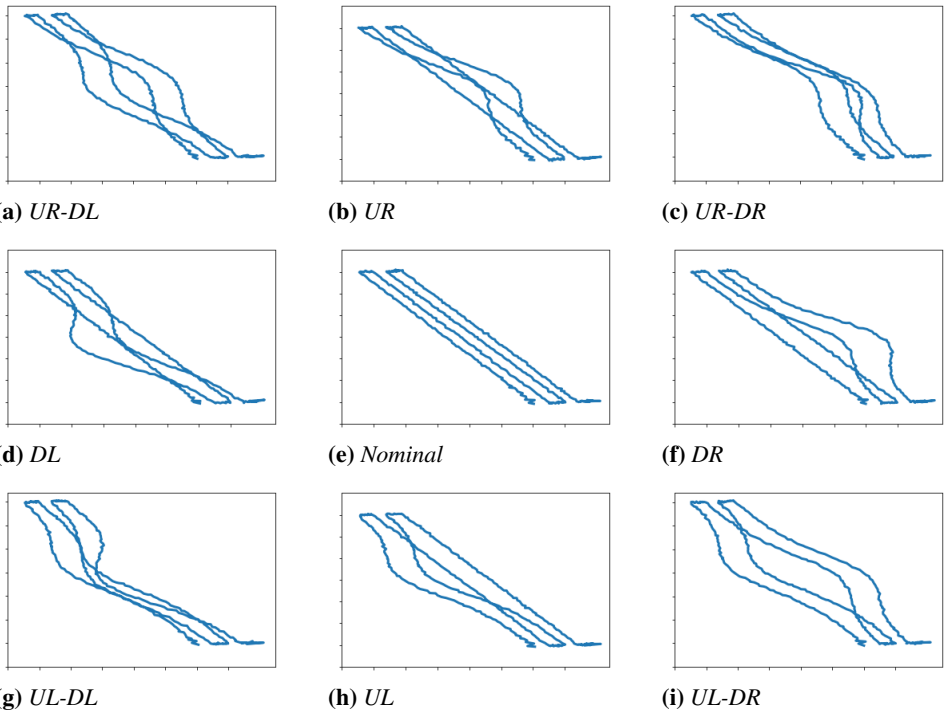
5.3.2 Running Example

As a running example we consider a synthetic dataset derived from that used in Chapter 3 and collected from a water drone (called Platypus) while performing a coverage task (Fig. 5.1a) on a lake to collect water samples [30]. Starting from some real runs, we generate 340 runs in which the following variables are recorded at each time step (1 Hz): heading, speed, acceleration, power signals to the left and right propellers, latitude, and longitude. From the anomalies observed in the real data (see 3.4.1), which present a recurring curve leaning to the left in the descending traits (Fig. 5.1b), we incorporated in our simulated dataset 8 possibles nuances of similar anomalies (Fig. 5.2).

5.3.3 Network Architecture

We present a new VAE architecture (Fig. 5.3) in which we replace the typical feed-forward layers with 1D-convolutional (Conv-1D) and Bidirectional LSTM (Bi-LSTM) layers, which are better suited to represent the temporal dependency of multivariate time series collected from robots' sensors. Moreover, the use of Conv-1D and Bi-LSTM layers allows our network to model very long runs.

Before being passed to the network for training, runs in the training set are standardized, then a Gaussian noise n_σ is added to each non-padded value of the runs in the training set X_{train} (see next section for a detailed explanation) in order to perform training using the denoising principle [159]. The noise variance σ is set to 1. The noise-corrupted input $\widetilde{X}_{train}$ is passed to stacked pairs of Conv-1D and MaxPooling layers, then a Bi-LSTM takes



(a) *UR-DL* **(b)** *UR* **(c)** *UR-DR*

(d) *DL* **(e)** *Nominal* **(f)** *DR*

(g) *UL-DL* **(h)** *UL* **(i)** *UL-DR*

Figure 5.2: *Simulated trajectories of a water drone. The nominal behavior of the robot is in the center (e): the robot starts from the bottom and goes up and down moving rigthward. (a) depicts the behavior of leaning to the right in the upward segments and to the left in the downward ones (UR-DL), (b) the behavior of leaning to the right in the upward segments (UR), (c) the behavior of leaning to the right in both the upward and the downward segments (UR-DR), (d) the behavior of leaning to the left in the downward segments (DL), (f) the behavior of leaning to the right in the downward segments (DR), (g) the behavior of leaning to the left in both the upward and the downward segments (UL-DL), (h) the behavior of leaning to the left in the upward segments (UL), (i) the behavior of leaning to the left in the upward segments and to the right in the downward ones (UL-DR).*

the output of the previous levels of convolution and returns the concatenated final states $[\overrightarrow{h_f}, \overleftarrow{h_b}]$ (i.e., the contexts in both directions). This concatenation is passed to 2 fully connected layers z_{mean} and $z_{log-var}$, which learn the parameters μ_z and σ_z^2 of the approximate posterior distribution $q_\phi(z|x)$, where ϕ is the matrix of the encoder's weights. These last two layers, together with the next one, represent the core of our VAE, in which a sample is extracted from a multivariate Gaussian distribution $\mathcal{N}(\mu_z, \sigma_z^2 I)$ by means of the reparametrization trick $z = \mu_z + \sigma_z \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$, resulting in the value of the latent variable z of the layer $z_{sampled}$, that will be used by

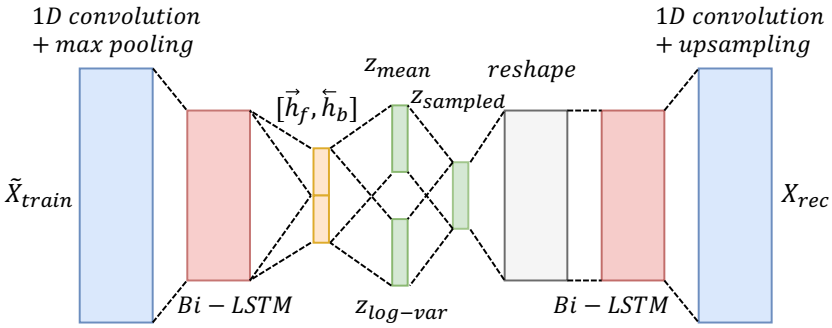


Figure 5.3: *Incr-VAE architecture.*

the decoder to reconstruct the input. The decoder network is composed of an initial fully-connected layer which reshapes $z_{sampled}$ in order to be compatible with the following layers. This is passed to the decoder Bi-LSTM that computes, for each time step of its input, a new value based on its context. Then, these sequences are passed to the stacked levels of Conv-1D and upsampling layers which expand the number of time steps of the sequences to match the original length while reconstructing their values. The final output of the last upsampling layer is used to compute the loss function $\mathcal{L}_{VAE}$ (see Section 5.2) to update the network weights by backpropagation. According to the denoising criterion, the network is trained to output the reconstruction X_{rec}, as an approximation of the original non-corrupted input X_{train}. From now on, we will refer to our VAE architecture described above as *Incr-VAE*.

5.3.4 Incremental Training

Instead of training Incr-VAE on runs corresponding to complete task executions performed by the robot or on windowed slices of such executions, as it would be the norm, we originally build a training dataset that includes also *incomplete* task executions in an incremental manner. The detailed procedure is reported in Algorithm 6. Given a batch $\mathbf{\Omega}$ of B unlabeled task executions and chosen an increment τ, we represent by X_{train} the (initially empty) training set (line 1). Each multivariate time series $\boldsymbol{O}$ in $\mathbf{\Omega}$ is inserted in X_{train} (lines 2-10) at different stages of completion by progressively including τ additional time instants (lines 3-4). Incomplete runs are zero-padded in oder to have the same length of complete ones (lines 5-7). Complete runs are also added to X_{train} (line 11). In our experiments, we use $\tau = 10$ and a maximum length of $T = 500$ (shorter complete runs are zero-padded).

Algorithm 6: Incremental Training

Input: $\mathbf{\Omega} = \{\boldsymbol{O}_1, ..., \boldsymbol{O}_B\}$, increment τ
Output: X_{train}

$X_{train} \leftarrow \{\ \}$
forall $\boldsymbol{O} = [\boldsymbol{o}_1, ..., \boldsymbol{o}_T] \in\ \mathbf{\Omega}$ **do**
  for $i = 1, ..., \lfloor T/\tau \rfloor$ **do**
    $X_{tmp} \leftarrow \{\boldsymbol{o}_1, ..., \boldsymbol{o}_{i*\tau}\}$
    for $j = (i * \tau) + 1, ..., T$ **do**
      $append(X_{tmp}, 0)$ ▷ *zero-padding*
    end
    $X_{train} \leftarrow X_{train} \cup X_{tmp}$
  end
end
$X_{train} \leftarrow X_{train} \cup \mathbf{\Omega}$

When using our incremental training approach based on the augmented training set X_{train}, the VAE induces a *progress-based* latent space, where runs at different levels of completion are encoded in different regions of the space. Fig. 5.4 shows the first 3 principal components of the latent space of our running example extracted using Principal Component Analysis (PCA) [18]. We use the same PCA projection also for the following figures. In the rightmost part of Fig. 5.4 it can be noted how incomplete runs containing just the first few observations are all represented in the same spot of the latent space as they are all indistinguishable from each other. Then, as anomalous executions start to deviate from the expected rectilinear paths (Fig. 5.2), three different bundles start to emerge, which represent the behaviors showing the same attitude in the first upward segment (e.g., UL, UL-DR, and UL-DL all lean towards left). When the water drone reaches the beginning of the first downward segment, the three bundles split and become nine as at this point all the different behaviors are distinct. Please note that in Fig. 5.4 (and in the following figures) the different behaviors have been depicted using different colors just for visualization clarity: the data used for training are unlabeled. From the figure, it appears clearly how our method leads to a structured latent space in which different behaviors are well separated.

As a consequence of the fact that our method induces a latent space which encodes both incomplete and complete executions, the same network trained only once can be used for both online and offline anomaly detection. Moreover, as a consequence of the fact that the incremental training involves some data augmentation, fewer runs are needed for training (as few as 6 runs, in our experiments) w.r.t. training VAEs in the standard way.

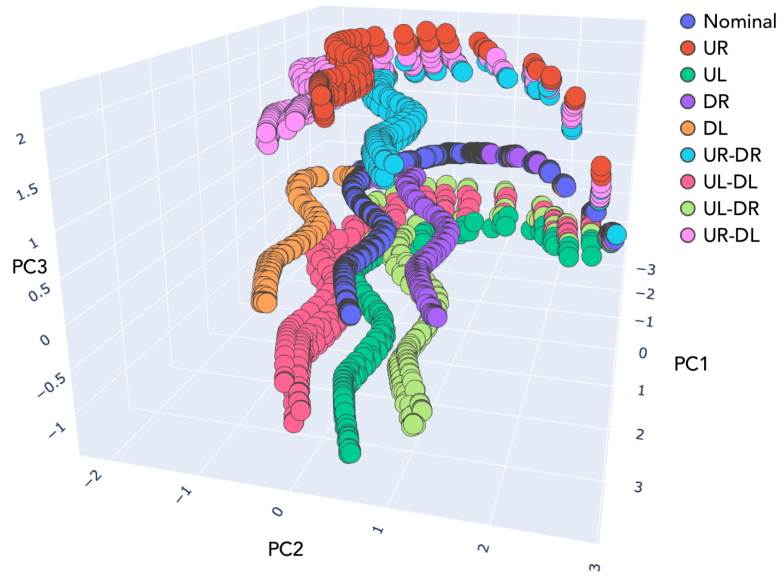
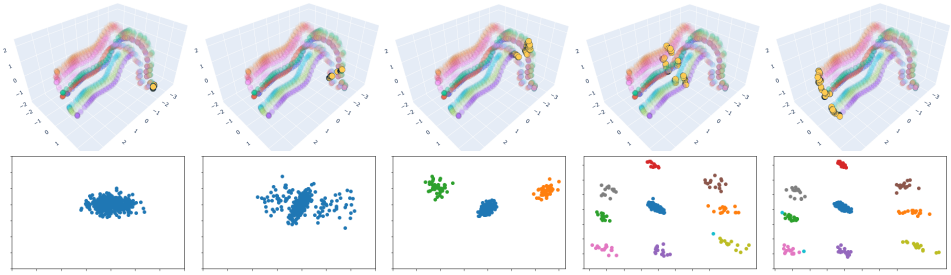


Figure 5.4: *Water monitoring robot latent space ($t = \tau$ on the right in the background, completed runs at $t = T$ in the foreground).*

5.3.5 Online Anomaly Detection

Assuming an Incr-VAE trained as illustrated above and the availability of at least one nominal execution for each of the k different types of nominal behaviors for the robot performing its task, online anomaly detection is performed by partitioning the latent space into nominal and anomalous regions according to the provided nominal executions $\{\boldsymbol{O}^N\}$ and by testing, at runtime, to which region a new incoming partial run belongs to. Note that our method detects anomalies in the latent space, differently of other methods based on AEs and VAEs (e.g., [109, 119, 127]) that detect anomalies thresholding the reconstruction error or probability.

Given X^N, obtained by applying Algorithm 6 to $\{\boldsymbol{O}^N\}$, the latent space segmentation (Algorithm 7) is performed offline (after training) using the DBSCAN algorithm [46], a density-based clustering algorithm that relies on the assumption that clusters are contiguous regions of high point density, separated from other clusters by regions of low point density. Clustering (line 8) is performed on each "slice" (line 2) of the latent space $z_{sampled}$ (i.e., on each set of points corresponding to runs with the same progress (line 5)) with the addition of the encodings of the nominal executions $\{\boldsymbol{O}^N\}$, zero-padded in order to represent the same level task completion (line 6-7). Clusters containing points belonging to nominal executions are considered



(a) $t = 10$ **(b)** $t = 30$ **(c)** $t = 80$ **(d)** $t = 200$ **(e)** $t = 500$

Figure 5.5: *Water monitoring robot latent space evolution (non-transparent encodings represent the slice to which the clustering refers to).*

Algorithm 7: Latent Space Segmentation

Input: increment τ, training set X_{train}, X^N (output of Algorithm 1 on $\{\boldsymbol{O}^N\}$), VAE encoder f_ϕ
Output: $\mathfrak{R}^N$

1 $\mathfrak{R}^N \leftarrow \{\ \}$ ▷ *nominal region*
2 **for** $i = 1, ..., \lfloor T/\tau \rfloor$ **do**
3 $\quad X^{(i)}_{train} \leftarrow x \in X_{train} \mid progress = i * \tau$
4 $\quad X^{(i)N} \leftarrow x \in X^N \mid progress = i * \tau$
5 $\quad Z^{(i)}_{train} \leftarrow f_\phi(X^{(i)}_{train})$
6 $\quad Z^{(i)N} \leftarrow f_\phi(X^{(i)N})$
7 $\quad Z^{(i)} \leftarrow Z^{(i)}_{train} \cup Z^{(i)N}$
8 $\quad C^{(i)} \leftarrow DBSCAN(Z^{(i)})$
9 $\quad$ **forall** $c \in C^{(i)}$ **do**
10 $\quad\quad$ **if** $(Z^{(i)N} \cap c) \neq \varnothing$ **then**
11 $\quad\quad\quad \mathfrak{R}^N \leftarrow \mathfrak{R}^N \cup c$
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 **end**

nominal regions (lines 10-11), while clusters not containing points from nominal executions, outliers, and the rest of the latent space are considered as *anomalous regions*. Fig. 5.5 shows how clusters evolve at different slices for the water monitoring robot running example.

At runtime (Algorithm 8), an incoming incomplete run $\boldsymbol{O}$ that needs to be tested for abnormality is firstly standardized (w.r.t. the mean and standard deviation used for the standardization of the training set) and zero-padded (line 1-2), then it is encoded into its latent representation $\hat{z}$ (line 5). The cosine similarity between $\hat{z}$ and the encodings of all the runs in the same slice is computed and if $\hat{z}$ is within a distance of ϵ (i.e., DBSCAN's threshold on the maximum distance between two samples for being consid-

Algorithm 8: Online Anomaly Detection

Input: increment τ, nominal region $\mathfrak{R}^N$, test run $\boldsymbol{O} \leftarrow [\boldsymbol{o}_1, ..., \boldsymbol{o}_t]$, threshold ϵ, VAE encoder f_ϕ
Output: $flag \in \{0, 1\}$

1 **for** $j = t + 1, ..., T$ **do**
2 $\quad append(\boldsymbol{O}, 0)$ ▷ *zero-padding*
3 **end**
4 $flag \leftarrow$ `True` ▷ *anomaly flag*
5 $\hat{z} \leftarrow f_\phi(\boldsymbol{O})$
6 **if** $t\%\tau = 0$ **then**
7 $\quad i \leftarrow t/\tau$
8 $\quad$ **forall** $z \in Z^{(i)}$ **do**
9 $\quad\quad$ **if** $Cosine(\hat{z}, z) \leq \epsilon \ \wedge \ z \in \mathfrak{R}^N$ **then**
10 $\quad\quad\quad flag \leftarrow$ `False`
11 $\quad\quad$ **end**
12 $\quad$ **end**
13 **else**
14 $\quad$ **forall** $i \in \{\lfloor t/\tau \rfloor, \lceil t/\tau \rceil\}$ **do**
15 $\quad\quad$ **forall** $z \in Z^{(i)}$ **do**
16 $\quad\quad\quad$ **if** $Cosine(\hat{z}, z) \leq \epsilon \ \wedge \ z \in \mathfrak{R}^N$ **then**
17 $\quad\quad\quad\quad flag \leftarrow$ `False`
18 $\quad\quad\quad$ **end**
19 $\quad\quad$ **end**
20 $\quad$ **end**
21 **end**

ered as neighbors of each other, $\epsilon = 0.5$ is the default value we use in our experiments) from an encoding belonging to a nominal region, the partial run is considered nominal, while an anomaly is detected otherwise (lines 9-11). For test runs whose progress is not a multiple of τ (line 6), the cosine similarity is computed w.r.t. the two slices immediately preceding and following (lines 14-20). In our experiments, we use training datasets containing few hundreds of executions, hence we perform linear search at runtime; for larger datasets it may be worth considering nearest neighbor search algorithms with sub-linear time complexity, such as space partitioning (e.g., the K-D trees) or Locality-Sensitive Hashing (LSH) [138].

5.3.6 Offline Anomaly Detection

Assuming the availability of an Incr-VAE trained as discussed in Sections 5.3.3 and 5.3.4, offline anomaly detection is obtained by performing DBSCAN on the last slice (the one containing the encodings of complete executions) of the latent space with the addition of the encodings of the nominal executions and the encoding $\hat{z}$ of the run under scrutiny. As the DBSCAN algorithm either assigns each point to a cluster or treats it as an

outlier, in case $\hat{z}$ belongs to a cluster containing the encoding of a nominal run, the behavior will be considered nominal, anomalous otherwise. In case a domain expert provides also runs labeled as anomalous and $\hat{z}$ belongs to a cluster containing one such anomalous run, it will also be possible to specify the nature of the anomaly. Outliers are considered as generic anomalies.

5.4 Experimental Results

In this section we present the results obtained by detecting anomalies in three different datasets collected from real robots.

We use two common metrics in the field of anomaly detection, namely, *alert delay* and *false positive rate*. Given an anomaly occurring at time step t_A, the alert delay d_A is computed as $d_A = t - t_A$ where $t \geq t_A$ is the time step at which the occurrence of the anomaly is detected by a method. Given the set W containing the time steps at which a method reports an anomaly, the false positive rate (FPR) is computed as the fraction of the time steps preceding the actual occurring of an anomaly which have been identified as anomalous $FPR = \frac{|t \in W,\, t < t_A|}{t_A}$. The use of just the FPR without the TPR (i.e., percentage of actual anomalies detected) is meaningful since the case in which one could obtain a $FPR = 0$ by always saying that everything is nominal is prevented from the fact that in that case the *alert delay* would result to be substantially increased.

We compare our system against a baseline, three other methods proposed in the literature for online anomaly detection in robotics, and the one proposed in the previous chapters:

- A one-class support vector machine (OSVM) trained with a sliding window size $w = 10$.
- HHAD, the HMM-based anomaly detector presented in Chapter 3. HMM parameters are chosen minimizing the BIC score, the window size is set to 10 and the 3σ-rule is used to select the detection threshold. We do not use the data-augmented version as, as shown in the previous chapter, for both robotic datasets (i.e., INTCATCH and ALFA), when the complete available dataset is used, the performance improvement provided by the data augmentation procedure is negligible.
- ENC-DEC AD [109], the first work that proposed to employ LSTM-based AEs for anomaly detection on time series. ENC-DEC AD learns to reconstruct nominal time series and then uses the reconstruction error on unseen executions to detect anomalies. We optimize the detec-

tion threshold τ by maximizing F_β (a function of precision and recall), as suggested by the authors of the method.

- Conv-AE [119], based on transforming system logs into images, which are then used to train a convolutional (2D) AE. As for ENC-DEC AD, the reconstruction error on unseen executions is used to detect anomalies. We optimize the sensitivity parameter z for each dataset according to authors' suggestions (i.e., $z \in [0, 3]$).
- LSTM-VAE [127], a state-of-the-art LSTM-based VAE with a varying state-based threshold obtained by employing a progress-based prior and a support vector regressor (SVR) for threshold prediction. We optimize the sensitivity parameter c for each dataset.

While our method is trained in a minimally supervised fashion (i.e., knowing the nominal label for k executions, where k is the number of different types of nominal behaviors), all four competitors are trained in a semi-supervised fashion (i.e., assuming that all training data are nominal) on the same datasets. Given a dataset, training our method takes some minutes on a commercial laptop, while online and offline detection of anomalies takes few milliseconds.

5.4.1 Water Monitoring Robot Dataset

This is the same dataset introduced in the running example. Our Incr-VAE network is trained using the Adam optimizer with one level of convolution in the encoder and decoder, $h = 20$ as latent dimension, 10 filters for the convolution, and 10 as convolution window. A single nominal run is used to partition the latent space.

5.4.2 Patrolling Robot Dataset

This publicly available[1] dataset has been collected within the scope of the STRANDS[2] project [66], where an autonomous robot called SCITOS-G5 performs a patrolling task in a small office every 5 minutes. A complete description of the data collection process is provided in [91]. We consider a total of 463 different executions sub-sampled at 1 Hz. We restrict the set of available sensors to those that are intuitively useful for anomaly detection, namely, robot location (x and y) and robot and camera headings. After a visual inspection of the logs, only one type of nominal behavior ($k = 1$)

[1] https://lcas.lincoln.ac.uk/nextcloud/shared/datasets/
[2] http://strands.acin.tuwien.ac.at/

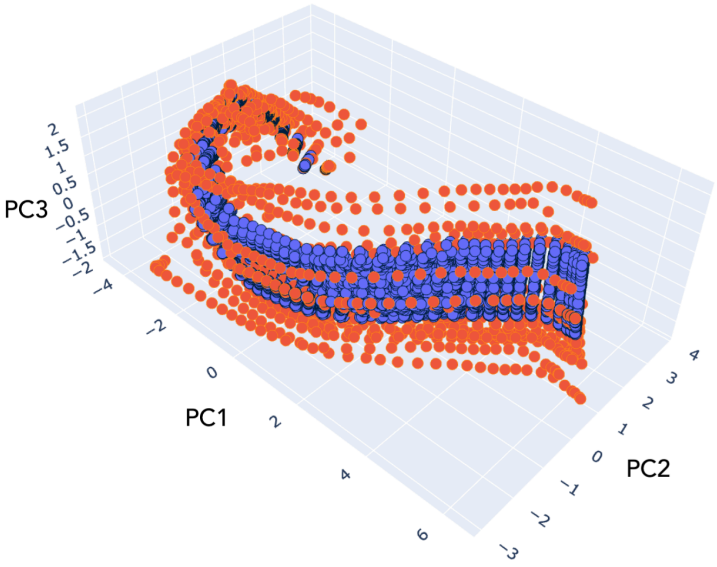


Figure 5.6: *Patrolling robot latent space ($t = \tau$ in the background, completed runs at $t = T$ in the foreground). Nominal runs in blue, anomalies in red.*

has been assumed and a subset of the runs have been manually labeled as anomalous in order to be used for testing. Examples of anomalies are deviations from the predefined path and incorrect uses of the RGB-D camera when checking for the presence of intruders. The latent space of our VAE induced by this dataset is shown in Fig. 5.6. Anomalies (in red) are clearly detached from the central nominal bundle (in blue). The network is trained using the Adam optimizer with 2 levels of convolution on both encoder and decoder, 10 as number of filters and window of convolution, $h = 20$ as latent dimension. A single nominal run is used to partition the latent space.

5.4.3 Assistive Robot Dataset

The third dataset has been collected within the same project [108] of the one in Section 3.4.2. The dataset contains 238 runs, each one composed of a sequence of observations collected at 1 Hz including: heading, speed, acceleration, position w.r.t. the x-axis, and position w.r.t. the y-axis. The dataset presents $k = 4$ different types of nominal behaviors (corresponding to reaching one of four different rooms) for each of which a domain expert provided a single nominal run. Out of the 238 runs, 12 have been classified by a domain expert as anomalies which are used for testing and correspond to the robot not being able to return back to its charging station because it

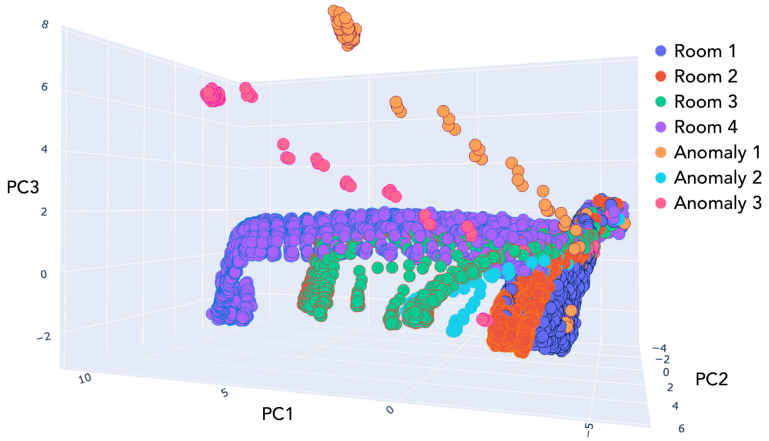


Figure 5.7: *Assistive robot latent space ($t = \tau$ on the right, completed runs at $t = T$ on the left).*

remains stuck in forniture, the robot departing from its nominal trajectory, or the robot moving too fast. We use the same network hyperparameters as in the patrolling robot dataset. The latent space induced by this dataset is in Fig. 5.7. As said, a single nominal run for each one of the $k = 4$ types of nominal behaviors is used to partition the latent space.

We do not consider the ALFA dataset in these experiments, as, being composed of flights each with a different trajectory, it would not be a good fit to the technique proposed in this chapter, which is designed to model repetitive tasks.

5.4.4 Results

Alert delays and FPRs are reported in Table 5.1 and Table 5.2, respectively. Our method provides the best performance across all datasets despite using just 1 labeled nominal execution for the first two datasets and 4 for the third one.

As highlighted also in [127], the higher alert delay and FPR of ENC-DEC AD have to be attributed to the fact that sometimes the reconstruction error is high also in nominal situations. In fact, depending on the stage of the execution, the reconstruction quality may vary. An example is when spikes (i.e., impulses to make the boat turn) appear on the currents to the motors in the time series of the first dataset, which result in high anomaly scores also in nominal runs. Thanks to its varying state-based threshold, LSTM-VAE is able to overcome the above issue after we set a tolerance

	Platypus	SCITOS-G5	Assistive robot
OSVM	1.375 (2.18)	60.5 (65.19)	5.0 (7.07)
HHAD	3.48 (2.11)	8.38 (2.18)	46.17 (58.91)
ENC-DEC AD	19.23 (11.60)	10.66 (13.42)	6.83 (6.47)
Conv-AE	6.22 (5.16)	3.40 (3.65)	6.42 (7.48)
LSTM-VAE	3.68 (2.25)	4.88 (6.33)	5.42 (6.81)
Incr-VAE	**0.5** (2.18)	**3.13** (5.56)	**3.34** (4.71)

Table 5.1: *Alert delay results.*

	Platypus	SCITOS-G5	Assistive robot
OSVM	0.123 (0.200)	0.180 (0.204)	0.130 (0.130)
HHAD	0.019 (0.053)	0.169 (0.120)	0.073 (0.103)
ENC-DEC AD	0.107 (0.151)	0.142 (0.245)	0.200 (0.200)
Conv-AE	0.049 (0.086)	0.109 (0.058)	0.170 (0.100)
LSTM-VAE	**0.0** (0.0)	0.191 (0.197)	0.084 (0.119)
Incr-VAE	**0.0** (0.0)	**0.065** (0.115)	**0.007** (0.009)

Table 5.2: *FPR results.*

value to be added to the state-based threshold of the model to avoid the incorrect detection of the spikes, even though this comes at the cost of a slight worsening of the alert delay. Comparing the two AE-based methods, Conv-AE always outperforms ENC-DEC AD, probably due to the use of convolution, that is more stable and easier to train than recurrent layers. We also note that the VAE-based methods outperform the AE-based ones, as also pointed out in [127]. The high alert delay for HHAD in the third dataset has to be ascribed to its inability to detect anomalies on the velocity of the robot due to the Markov assumption. To enable HHAD to detect also such anomalies, velocity should be explicitly modeled as an additional dimension of the multivariate time series as done in [8]. OSVM's higher FPR and bad performance in general on the second dataset result from its inability to represent the portion of time series inside the window as an actual sequence instead of as a feature vector without any time dependence. Moreover, it is difficult to adjust OSVM's threshold after training as it coincides with the SVM decision boundary.

We finally remark that our method reaches an accuracy of 100% when performing offline anomaly detection on the three datasets.

5.4.5 Latent Space Analysis

As said, instead of training Incr-VAE on runs corresponding to complete task executions performed by the robot or on windowed slices of such executions, as it usually happens, we originally build a training dataset that

includes also *incomplete* task executions in an incremental manner. Here, we investigate how the latent space would be learnt in the two usual cases just mentioned for the water monitoring robot dataset. We include in our comparison also the latent space induced by LTSM-VAE, the other method based on a variational autoencoder we consider in our experimental assessement. As a reference, the latent space of Incr-VAE when trained in the incremental way is reported in Fig. 5.4.

Complete Executions

Fig. 5.8 depicts the latent space resulting from training Incr-VAE on complete executions only. As it can be seen, our architecture manages to encode very long sequences in a meaningful way (i.e., nominal runs and each anomaly are clearly separated). Note that, as one would expect, this arrangement coincides also with the last slice of Fig. 5.5 (i.e., the one encoding complete executions). Looking at Fig. 5.8 it can be noted a very interesting feature: not only each anomaly type has its own "cluster" clearly detached from the others, but they are arranged in a meaningful and intuitive way. Take for example UL and DL, if we start from the nominal area (in the center) and proceed along a line passing between UL and DL, we reach the area of latent space in which UL-DL is located, i.e., the anomalous behavior affected by both UL and DL anomalies (note that this desired feature is true also when Incr-VAE is trained in the incremental way). One drawback of training on complete runs is that in this case the online anomaly detection problem reduces to the offline one, as complete executions are required to be provided as input to the Incr-VAE. Another drawback is that, according to our experiments, in this way Incr-VAE is harder to train (almost five times more epochs compared to incremental training).

Sliding Window

When trained using a sliding window, the latent space of Incr-VAE is not structured as in Fig. 5.4, where there is a clear concept of beginning and end of a run. Our method, when trained incrementally, can tell if a partial run is nominal or anomalous up to a given point; when using sliding windows, it would tell only if a specific window is anomalous. For example, Fig. 5.9 depicts the latent space when Incr-VAE is trained using a sliding window of 10 time-steps. Some structure is still present (as reflected by the groupings of the colors), but most of the interpretability is lost. The temporal progression is also lost and consecutive instances of the same sub-task (e.g., the first and the second ascending traits in Fig. 5.1a) are now encoded in the

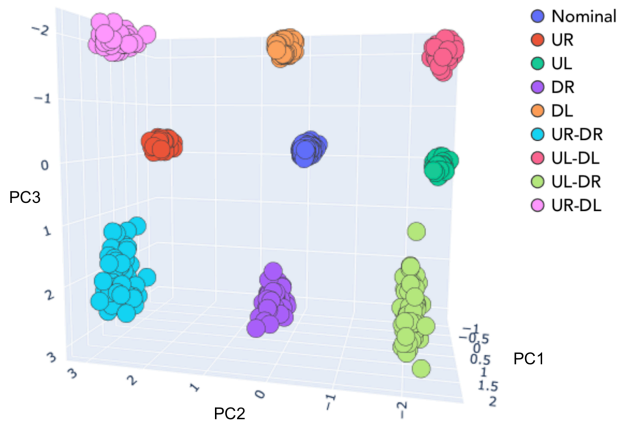


Figure 5.8: *Incr-VAE latent space when trained on complete runs.*

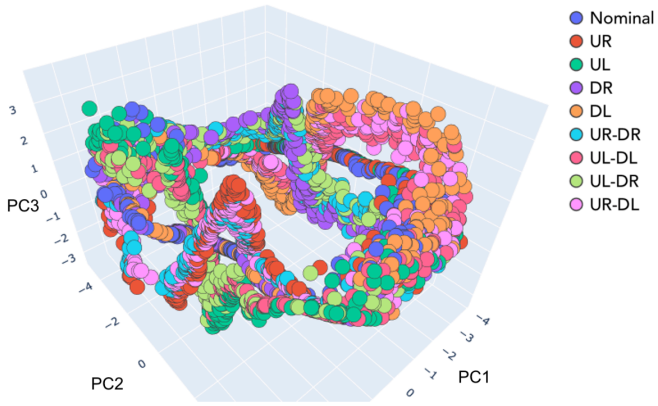
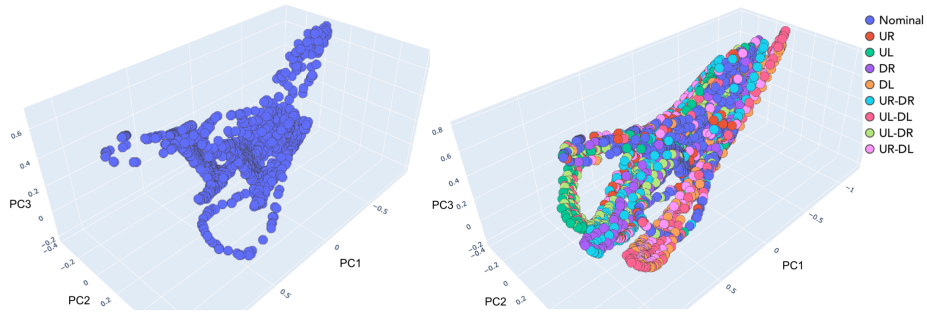


Figure 5.9: *Incr-VAE latent space when trained using a sliding window.*

same spot. Moreover, our incrementally-trained method can do both online and offline anomaly detection with a single network trained only once. When training using sliding windows, offline anomaly detection could not be performed.

LSTM-VAE

Fig. 5.10 depicts the latent space of LSTM-VAE when trained only with nominal executions of the water drone. Fig. 5.10a represents the encodings of the training set (i.e., only nominal runs). In order to make LSTM-VAE's latent space comparable to the other ones, Fig. 5.10b shows the encodings of the training set and of some anomalous runs, in different colors (the model is trained on nominal runs only, then, after training, some anoma-



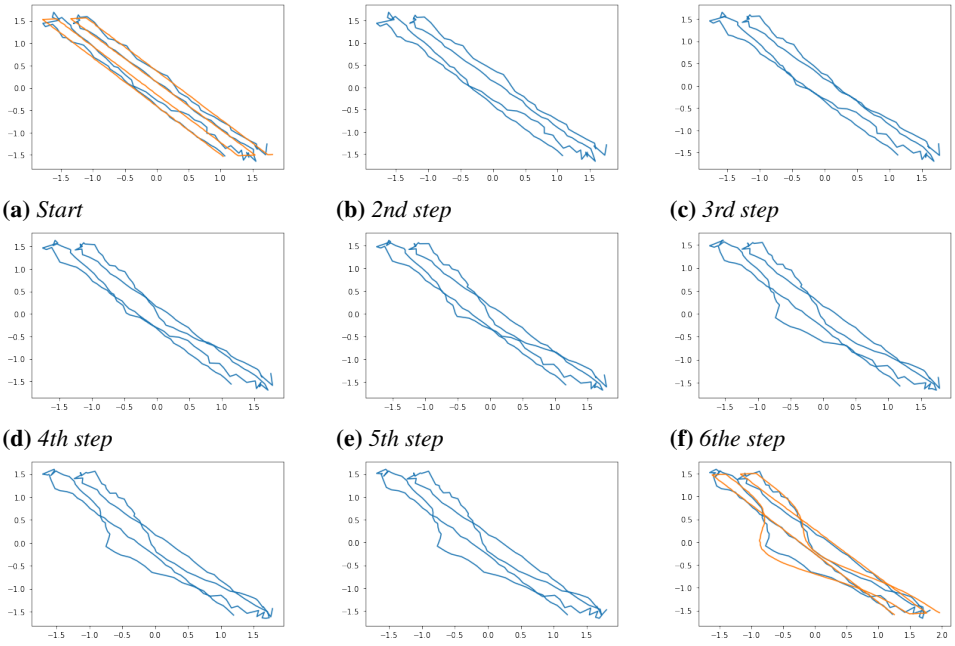
(a) **(b)**

Figure 5.10: *LSTM-VAE latent space.*

lous runs have been passed through the encoder). As it can be seen, some structure is present, which is enough for the model to learn good reconstructions and, as a consequence, detect correctly most of the anomalies (as our experimental results show). However, the latent space of LSTM-VAE is less separable, not very interpretable, and a progress-based structure is not present.

Latent Space Interpolation

One of the most important features of VAEs is the smoothness and continuity of their latent spaces, which means that, for example, by interpolating points between two encoded values that represent two different runs $\boldsymbol{O}_1$ and $\boldsymbol{O}_2$, and generating new runs using the decoder, the in-between generated runs will change smoothly from $\boldsymbol{O}_1$ to $\boldsymbol{O}_2$. In Fig. 5.11 it is shown the case in which starting from the embedding (i.e., latent representation) of a nominal run for the water monitoring robot dataset, and interpolating towards the embedding of an anomalous one (DL), the reconstructed runs becomes incrementally more anomalous. This feature could be used to generate possible anomalies that could affect a robot system, in order to develop contingency strategies before the anomalies actually occur. Moreover, latent space interpolation could be useful to explain observed anomalies. Both directions are left as future work.



(a) *Start* **(b)** *2nd step* **(c)** *3rd step*

(d) *4th step* **(e)** *5th step* **(f)** *6the step*

(g) *7th step* **(h)** *8th step* **(i)** *end*

Figure 5.11: *Interpolation (reconstructions in blue, input runs in orange).*

CHAPTER 6

Detecting Anomalies in Robot Swarms

6.1 Introduction

In this chapter, an adaptation of the method presented in the previous one to detect anomalies in robot swarms is presented.

Although some centralized approaches to detect anomalies in swarms of robots have been proposed, when these systems are operating in real-world scenarios, and especially outdoor, it is usually not possible to depend on external observation infrastructures to identify faults in a centralized manner. As a consequence, the only viable solution is to have the robots of the swarm execute anomaly-detection software onboard and look for anomalies either on themselves (i.e., endogenous anomaly detection), on one another (i.e., exogenous anomaly detection), or both (i.e., multi-layered anomaly detection). In order for our method to be general and suitable to be employed in a broad range of situation, the work presented in this chapter commits to the decentralized architecture. Moreover, since the capabilities of our VAE-based method of being suitable to be used by robots for detecting anomalies in themselves have already been demonstrated in the previous chapter, here just exogenous anomaly detection will be considered. This choice is also motivated by the fact that most of the approaches

(e.g., [39, 153–156]) proposed in the literature of swarm robotics' anomaly detection belong to the decentralized and exogenous paradigms.

We perform our experiments using the ARGoS simulator [134] and the e-puck swarm robot platform [113]. Four homogeneous swarm behaviors are considered: dispersion, aggregation, homing and flocking.

We compare our method against an immune system-inspired approach [154] that needs significant manual tuning and show how we are able to achieve a similar detection accuracy while adopting a completely automatic and end-to-end approach. The work in [154], in fact, relies on the extraction of several heavily hand-engineered features and the careful setting of a considerable amount of parameters in order to achieve optimal functioning. Please note that, although working in simulation would allow also for completely supervised learning paradigms to be adopted (as anomalies can be simulated with ease), we prefer a semi-supervised one with the aim of making our method more general and, especially, more easily applicabile in real-world settings.

6.2 The Proposed Method

An online semi-supervised method based on a sliding window approach is proposed, which allows robots to detect, at runtime, anomalies in the behavior of other nearby robots which lie within their field of view.

Problem Formulation

Given a swarm R of robots r_i, $i = 1, .., |R|$, we frame the problem of online anomaly detection at time t in terms of each robot r_i of the swarm collecting information on nearby robots $r_j \in R$, observable from r_i (i.e., such that each robot r_j is located within sensing range of r_i), and then expressing a vote on each observed robot nominality by means of an online anomaly detector (it is the same for all robots) previously trained on nominal executions.

As for the other methods presented in this thesis, we assume the availability of nominal training data and, for this reason, our approach belongs to the semi-supervised family. This choice is motivated by the fact that we use simulated data for our experiments, however, as already motivated in previous chapters, it is usually plausible to assume the availability of some nominal executions also when real robots are considered.

Sensor Data Processing

Given a swarm R of robots r_i, $i = 1, .., |R|$, at each control cycle t, a robot r_i receives range d_{ij} and bearing ϕ_{ij} observations from each robot r_j in line of sight and within a 1 m radius from r_i. Such information about range and bearing are then processed to build multivariate time series windows to be used for detecting anomalies in each observed robot.

The window employed by robot r_i for performing anomaly detection on robot r_j at time t contains, for each one of the w time instant of the window, the following information:

- Observed range d_{ij}: the range of observed robot r_j from robot r_i at time t, $d_{ij} \in [0, 1]$ m.
- d_{ij}'s first derivative: as all control cycles have the same length, this corresponds to the difference of d_{ij}-s at consecutive time instants.
- Observed bearing ϕ_{ij}: the bearing of observed robot r_j from robot r_i at time t, $\phi_{ij} \in [0, 360]$ degrees.
- ϕ_{ij}'s first derivative: as for the range, as all control cycles have the same length, this corresponds to the difference of ϕ_{ij}-s at consecutive time instants.
- Distance traveled by r_j in the last w control cycles (computed using using a forward kinematics model of a two-wheeled nonholonomic differential-drive mobile robot).

Time series widows of $w = 100$ (i.e., 10 s) are considered.

A remark needs to be made regarding the choice of using a range-and-bearing sensor board to perform the exogenous observation. Although it may be perceived as a limiting factor on the effective practical applicability of the proposed method, it must be noted that range and bearing extension boards represent baseline sensors and are available for many swarm robot platforms, including Khepera III [135], eye-bot [141], foot-bot [142] and e-puck [57].

Online Anomaly Detection

A nominal model for each one of the homogeneous behaviors is learned by training the VAE architecture proposed in the previous chapter (see Section 5.3.3) on windows of width w extracted from nominal executions. We use f_ϕ and f_θ to refer to the encoder and decoder networks, respectively.

Online anomaly detection at time t is performed by computing the reconstruction error on the current window $\boldsymbol{W}_t = \{\boldsymbol{o}^O_{t-w+1}, ..., \boldsymbol{o}^O_t\}$. When the reconstruction error exceeds a predefined threshold τ, i.e.,

$$\left|\boldsymbol{W}_t - f_\theta\big(f_\phi(\boldsymbol{W}_t)\big)\right| > \tau,$$

the behavior of the observed robot r_j is considered anomalous. The threshold τ is set using the 3-σ rule.

Please note that, in this scenario, the incremental training procedure proposed in the previous chapter cannot be adopted as tasks are less structured and repetitive, but, most importantly, because given two robots r_i and r_j, there is no guarantee that they will remain in line of sight for the whole duration of the experiment (in fact, they most likely will not be). Thus, a sliding window approach and the reconstruction error as anomaly score need to be employed.

6.3 Experimental Results

6.3.1 Experimental Setting

A similar experimental setting to the one proposed in [154] is adopted.

Simulator and Robots

We use ARGoS[1] [134], a physics-based, multi-engine multi-robot simulator designed to realistically reproduce complex experiments involving large swarms of (possibly heterogeneous) robots. Thanks to its modular and open-source approach, we have been able to easily modify its source code in order to implement the necessary simulations. We employ a robot swarm composed of 20 e-puck robots[2] [113] (see Fig. 6.1) that move inside a square arena of 3 by 3 m. The e-puck robot has a diameter of 7.5 cm, an inter-wheel distance of 5.3 cm, a maximum speed of 10 cm/s, and a control cycle of length 0.1 s. Each e-puck robot hosts onboard eight infrared proximity sensors (used for obstacle avoidance) and two actuators which control the robots speed and direction. Each robot is also equipped with a range and bearing extension board [57], which enables a robot to estimate the relative location and orientation of neighboring robots.

[1] https://www.argos-sim.info/
[2] http://www.e-puck.org/

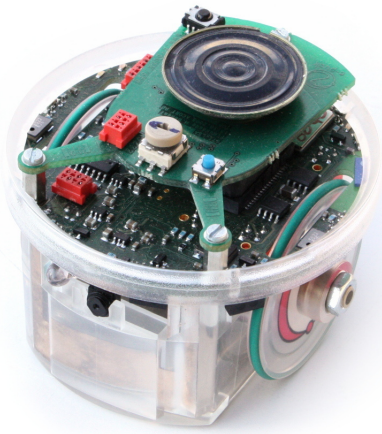


Figure 6.1: *e-puck robot (image credits: http://www.e-puck.org/).*

Swarm Behaviors

The following homogeneous swarm behaviors (i.e., all the robots in the swarm execute the same task) are considered:

- *Dispersion*: robots move in the opposite direction of the center of mass of their neighbors. In real-world settings, this could translate to the task of maximizing the sensor coverage of an area while preserving the connectivity within the swarm. Dispersion of robotic swarms appears to be applicable and useful in missions such as planetary exploration, hurricane surveillance, or nuclear decontamination, where the robots with maximal coverage collect samples from the unknown surface, detect the victims, or collect nuclear waste, respectively.

- *Aggregation*: robots move towards the center of mass of surrounding neighbors, but disperse away if too close to their neighbors to avoid collisions. This allows individuals of the swarm to get spatially close to each other for further interaction.

- *Homing*: robots move towards a single pre-specified stationary beacon that serves as a landmark, and move away if too close to the landmark or to other robots. The position of the beacon for homing is selected at random in the arena at the start of the experiment. The homing behavior is an essential part of more complex swarm behaviors such as foraging.

- *Flocking*: cohesive and ordered motion of a group of individuals in a common direction. Robots continually adjust their velocity to that of neighboring robots, where the velocity of the neighboring robots is estimated over a time-window. The flocking robots aggregate towards and disperse from neighbors, if they are too far away or close by, respectively. One of the most common applications of flocking behaviors in real-world scenarios is the control of UAVs.

Faults

We use ARGoS to inject faults directly in the sensors and actuators of the e-puck robots. As a consequence, the resulting anomalous behavior of the robots correspond to the actions performed by their controllers, which are either provided with corrupted inputs from faulty sensors, or whose commands to actuators are not executed correctly by the underlying hardware. The following anomalies affecting wheels actuators and proximity sensors are considered:

- *BACT*: both wheels are prevented from rotating. This fault can refer to one of two real-world scenarios: robot malfunction, or robot's wheel rim worn out. In ARGoS, it is simulated by setting the wheels speed to to 0 cm/s when the fault occurs.
- *RACT*: the right wheel is prevented from rotating. In ARGoS, it is simulated by setting the right wheel speed to to 0 cm/s when the fault occurs.
- *LACT*: the left wheel is prevented from rotating. In ARGoS, it is simulated by setting the left wheel speed to to 0 cm/s when the fault occurs.
- *PMIN*: proximity sensors return the smallest possible value. This fault may mirror a disconnected proximity sensor in real scenarios.
- *PMAX*: proximity sensors return the largest possible value. Real situations of this kind are, for example, obstructions or dirty proximity sensors.
- *PRND*: a random value for the proximity sensors is uniformly sampled between the minimum and the maximum. A partially dislodged proximity sensor could produce faults of this kind.

Please note that, although the faults just introduced represent component-level anomalies, we do not make use of any sensory information directly observing such components, nor are we interested in detecting them at the

component level. We are instead interested in detecting the consequences that such faults have on the general behavior of individual robots, e.g., a *BACT* fault resulting in a motionless robot, or a *LACT* fault resulting in the robot spinning on itself.

Experiments Execution

Experiments are performed with a swarm of 20 robots. For collecting the training dataset, 10 nominal executions (i.e., all 20 robots behave nominally) have been collected for each homogeneous behavior (i.e., either aggregation, dispersion, flocking, or homing). For testing, 19 of the 20 robots perform one of the nominal behaviors, while, the remaining one, performs one of the faulty behaviors, PMIN, PMAX, PRND, LACT, RACT, and BACT (the faulty robot is anomalous for the whole duration of the run). 20 different executions are performed for each of the 24 combinations of 4 normal and 6 faulty behaviors. Each experiment lasts 6,000 control cycles (corresponding to 600 s).

6.3.2 Results

We compare our approach against the method proposed in [154], where the authors propose an immune system-inspired anomaly detector based on the crossregulation model (CRM). The CRM [103] describes the dynamics of effector T-cell populations and regulatory T-cell populations during interactions with antigen presenting cells. Their method (which, for brevity, will be referred to as *CRM-based* from now on), is composed of the following three phases:

1. Robots observe and characterize the behavior of their neighbors over a period of time, and, for each robot in their field of view they estimate the corresponding features. A total of six features are computed: the first two encode the observed robot's immediate environment, two other features capture the observed robot's actions, and, lastly, two features are used to describe the observed robot's response to events. Once collected, features are then shared with nearby robots to consolidate them (i.e., majority voting).

2. Every robot classifies the observed and consolidated features as normal/abnormal according to the CRM (which is run onboard).

3. The robots form voting coalitions to consolidate (i.e., majority voting) their individual-level decisions on the detected anomalies.

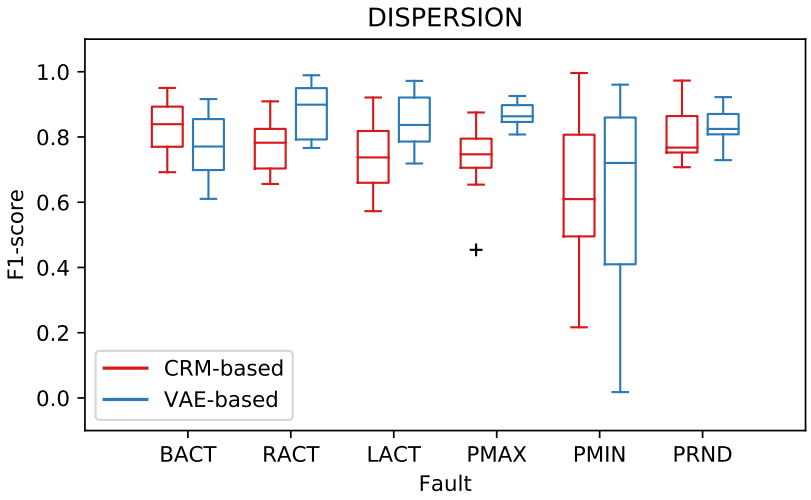


Figure 6.2: *Comparison of detection performance for the dispersion behavior.*

Results of the experiments are shown in Fig. 6.2, Fig. 6.3, Fig. 6.4 and Fig. 6.5, the F1-score is used for quantifying the goodness of the two approaches on the different combinations of behavior/fault. It can be noted how our method achieves a detection performance comparable to the one proposed in [154]. The poorer detection performance for the PMIN fault should be ascribed to the fact that normal behaving robots compensate for the faulty one. Although a comparable detection performance is achieved, it must be noted that our method does so while requiring much less injection of domain knowledge both for what regards feature engineering and model parametrization. In particular, our methods learns to automatically extract meaningful features from multivariate time series windows, while the features employed in [154] are heavily hand-engineered and have a specific meaning w.r.t. the domain under study. Moreover, the CRM-based model in [154] requires a total number of 17 parameters to be set, while our method requires to set only the number of convolutional layers, the number of filters and window of convolution and the latent dimension.

6.4 Concluding Remarks

In this chapter, we have presented an exogenous fault detection method with the aim of better investigating the exogenous capabilities of our approach (as the endogenous ones for single-robot systems had already been tested in the previous chapter). An interesting direction of future research

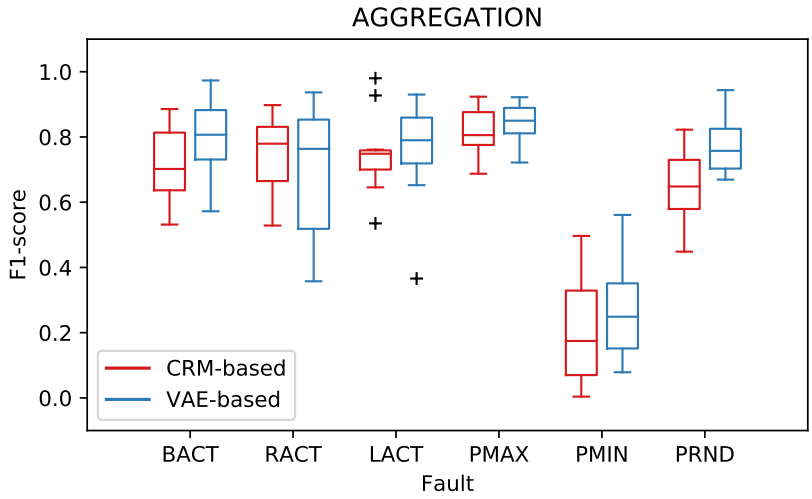


Figure 6.3: *Comparison of detection performance for the aggregation behavior.*

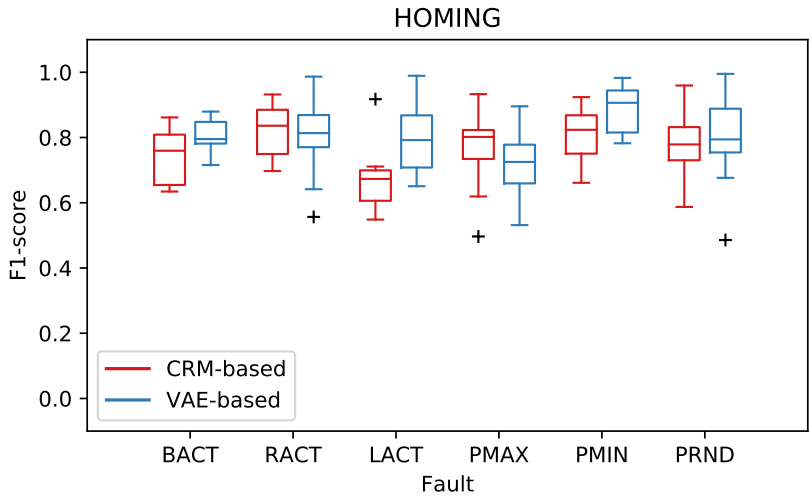


Figure 6.4: *Comparison of detection performance for the homing behavior.*

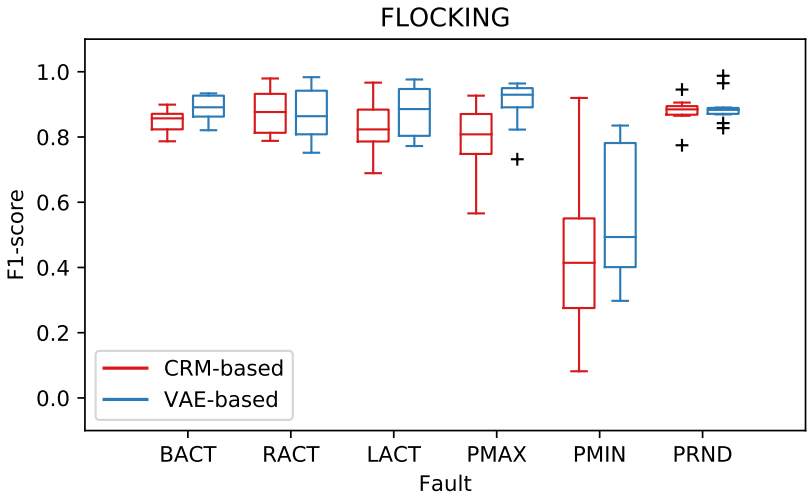


Figure 6.5: *Comparison of detection performance for the flocking behavior.*

is to extend the proposed solution to account also for endogenous anomaly detection and achieve a multi-layered perspective.

CHAPTER 7

Conclusion

As robots are less and less relegated in factories and are becoming a relevant part of humans' everyday life, anomaly detection techniques become increasingly important. Since these sophisticated machines are susceptible to different types of faults, detecting anomalies is needed not only to allow these systems to reliably operate for long periods of time, but also to make sure that otherwise undetected faults would not result in the robot damaging itself, or harming surrounding people.

To address the above issues, we have presented some new approaches for detecting various kinds of anomalies both in single- and multi-robot systems operating in different real scenarios requiring LTA.

We have presented a novel approach based on HMMs and Hellinger distance for online and offline anomaly detection, and showed how it improves over traditional methods both in detection performance and in interpretability of the results. Unlike other works in the literature, we show that even a single run is enough for learning the nominal behavior, making the semi-supervised setting effectively applicable in practical real-world scenarios. We have also proved how the methodology of adversarial data augmentation presented allows to improve the detection performance of HMM-based anomaly detectors without using any prior knowledge about the appearance

of the nominal time series, as traditional data augmentation methods require. The adversarial examples we generate are multivariate time series very close to the original ones but we empirically prove that they produce a performance improvement when used to augment the dataset with which the detector is trained. Furthermore, the same examples improve also the robustness of the detector to adversarial attacks in different domains.

We have then presented a new approach based on VAEs for detecting, both online and offline, anomalies in the behavior of autonomous robots. Our method outperforms other methods that have been recently proposed for anomaly detection in robotics and does so by requiring significantly less labeled data. We have shown how just even a single labeled nominal execution is sufficient for our method to partition a latent space (previously learned in an unsupervised fashion) in a meaningful way for detecting anomalies. We have also shown how a variant of the proposed VAE architecture can be used by individual robots in a swarm to detect anomalies in one another. Experiments on four different swarm behaviors showed that our method achieves a detection performance comparable to a state-of-the-art approach for anomaly detection in robotic swarms.

Although widely applicable in many different scenarios, as shown by our extensive experiments, several factors should be taken into consideration when choosing which one of the methods proposed in this thesis to adopt in a particular situation.

Data availability certainly plays a big role; in fact, even though all the methods proposed require very few (even just one) nominal executions to be trained, the VAE-based approach requires also a certain amount of unlabeled executions in order to be able to structure its latent space.

Another important aspect to consider is the effort for retraining; in fact, if retraining directly onboard is envisaged, depending on the available computing power of the robot considered, the non-data-augmented HMM-based anomaly detector may be more suited than the others as it requires significantly less computation for training.

Another thing to take into consideration is the different nominality semantics adopted by the various models; in fact, nominality at time t for the HMM-based approach means that the observations inside that specific window are overall nominal, while the VAE-based approach considers nominal a specific time instant t only if all observation from $t = 0$ up to time t are nominal.

If very complex robotic tasks are considered, the VAE-based method should be favoured.

7.1 Future Work

One of the biggest limitations of this thesis is certainly not having had the opportunity of implementing the proposed anomaly detectors on real robots. In fact, despite it being in the plans, it has not been possible to do so due to the travel restrictions imposed by the COVID-19 pandemic. A promising way of incorporating our proposed methods into the decision-making mechanism of a real robot could be to rely on methods belonging to the very recent research field of competence-aware systems [11], a new paradigm that allows autonomous robots to operate at varying levels of autonomy.

What follows is a list of additional directions of future work specifically for each one of the methods proposed.

Future work for the HMM-based approach includes employing more advanced versions of HMMs able to represent more complex behaviors as well as developing new distances between such models. Specifically, HMMs with Gaussian Mixture emission probabilities (GM-HMM) and a variation of the Maximum Mean Discrepancy (MMD) would allow to overcome the assumption on the same number of hidden states for the HMMs representing nominal and observed behavior. Regarding the data augmentation procedure, future work will concentrate on three main extension to the proposed methodology. The first one concerns the introduction of specific time series distance measures in the strategy for adversarial example generation. The second one is to investigate whether adversarial examples obtained with our method provide an increase in performance also when used to augment the training set of other anomaly detectors, such as AEs and one-class support vector machines. The third direction of future work focuses on the application of adversarial data augmentation to active anomaly detection, i.e., when a system is actively looking for possible anomalies in order to detect them more precisely and in advance.

Future work for the VAE-based approach includes employing a Gaussian mixture prior on z to better represent different types of nominal behaviors. Another interesting future direction is employing β-VAEs [69], as their ability of inducing a disentangled z could lead to an even more intuitive and interpretable latent space. When considering multi-robot systems, future work will focus on investigating strategies for sharing observed features among nearby robots. One promising way of achieving such goal is represented by Graph Neural Networks (GNNs) [104].

The contributions of this thesis constitute a step towards the longer-term goal of granting robots LTA capabilities. In the future, we will keep developing better and better anomaly detection solutions that will help achieving the harmonious coexistence of humans and autonomous robots.

Bibliography

[1] Naveed Akhtar and Anastassia Kuestenmacher. Using naive physics for unknown external faults in robotics. In *Proc. DX@PHM*, pages 23–29, 2011.

[2] Angelo Alessandri, Massimo Caccia, and Gianmarco Veruggio. Fault detection of actuator faults in unmanned underwater vehicles. *Control Eng Pract*, 7(3):357–368, 1999.

[3] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proc. EMNLP*, pages 2890–2896, 2018.

[4] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[5] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In *Proc. CCS*, pages 817–831, 2018.

[6] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot Auton Syst*, 57(5):469–483, 2009.

[7] Davide Azzalini, Luca Bonali, and Francesco Amigoni. A minimally supervised approach based on variational autoencoders for anomaly detection in autonomous robots. *IEEE RA-L*, 6(2):2985–2992, 2021.

[8] Davide Azzalini, Alberto Castellini, Matteo Luperto, Alessandro Farinelli, and Francesco Amigoni. HMMs for anomaly detection in autonomous robots. In *Proc. AAMAS*, pages 105–113, 2020.

[9] Kasra Babaei, ZhiYuan Chen, and Tomas Maul. Data augmentation by autoencoders for unsupervised anomaly detection. *arXiv:1912.13384*, pages 1–8, 2019.

[10] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Proc. AAAI*, pages 2687–2695, 2018.

[11] Connor Basich, Justin Svegliato, Kyle Hollins Wray, Stefan Witwicki, Joydeep Biswas, and Shlomo Zilberstein. Learning to optimize autonomy in competence-aware systems. In *Proc. AAMAS*, pages 123–131, 2020.

[12] Leonard Baum and John Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull Amer Math Soc*, 73(3):360–363, 1967.

[13] Leonard Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann Math Stat*, 41(1):164–171, 1970.

[14] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv:1411.7610*, pages 1–9, 2014.

[15] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Comput Sci Eng*, 13(2):31 –39, 2011.

[16] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Proc. ECML-PKDD*, pages 387–402, 2013.

[17] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. *Springer Handbook of Robotics*, pages 1371–1394, 2008.

[18] Christopher Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[19] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *ACM Comput Surv*, 54(3):1–33, 2021.

[20] Eric Bonabeau, Guy Theraulaz, and Marco Dorigo. *Swarm intelligence*. Springer, 1999.

[21] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[22] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proc. KDD*, pages 535–541, 2006.

[23] Alfonso Capozzoli, Marco Savino Piscitelli, Silvio Brandi, Daniele Grassi, and Gianfranco Chicco. Automated load pattern learning and anomaly detection for enhancing energy management in smart buildings. *Energy*, 157:336–352, 2018.

[24] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *Proc. USENIX*, pages 513–530, 2016.

[25] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. IEEE Symp. Secur. Priv.*, pages 39–57, 2017.

[26] Rodrigo A Carrasco, Felipe Núñez, and Aldo Cipriano. Fault detection and isolation in cooperative mobile robots using multilayer architecture and dynamic observers. *Robotica*, 29(4):555–562, 2011.

[27] Alberto Castellini, Giovanni Alberto Beltrame, Manuele Bicego, Domenico Bloisi, Jason J Blum, Matteo Denitto, and Alessandro Farinelli. Activity recognition for autonomous water drones based on unsupervised learning methods. In *Proc. AIRO@AI*IA*, pages 1–6, 2018.

[28] Alberto Castellini, Manuele Bicego, Domenico Bloisi, Jason Blum, Francesco Masillo, Sergio Peignier, and Alessandro Farinelli. Subspace clustering for situation assessment in aquatic drones: A sensitivity analysis for state-model improvement. *Cybern Syst*, 50(8):658–671, 2019.

[29] Alberto Castellini, Manuele Bicego, Francesco Masillo, Maddalena Zuccotto, and Alessandro Farinelli. Time series segmentation for state-model generation of autonomous aquatic drones: A systematic framework. *Eng. Appl. Artif. Intell.*, 90:1–13, 2020.

[30] Alberto Castellini, Domenico Bloisi, Jason Blum, Francesco Masillo, and Alessandro Farinelli. Multivariate sensor signals collected by aquatic drones involved in water monitoring: A complete dataset. *Data Brief*, 30:1–8, 2020.

[31] Alberto Castellini, Francesco Masillo, Manuele Bicego, Domenico Bloisi, Jason Blum, Alessandro Farinelli, and Sergio Peigner. Subspace clustering for situation assessment in aquatic drones. In *Proc. SAC*, pages 930–937, 2019.

[32] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv:1901.03407*, 2019.

[33] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput Surv*, 41(3):1–58, 2009.

[34] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, pages 875–886, 2009.

[35] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *J Artif Intell Res*, 16(1):321–357, 2002.

[36] Tingting Chen, Xueping Liu, Bizhong Xia, Wei Wang, and Yongzhi Lai. Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder. *IEEE Access*, 8:47072–47081, 2020.

[37] Leo H Chiang, Evan L Russell, and Richard D Braatz. *Fault detection and diagnosis in industrial systems*. Springer, 2000.

[38] Anders Christensen, Rehan OGrady, Mauro Birattari, and Marco Dorigo. Fault detection in autonomous robots based on fault injection and learning. *Auton Robot*, 24(1):49–67, 2008.

[39] Anders Lyhne Christensen, Rehan OGrady, and Marco Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Trans. Evol. Comput.*, 13(4):754–766, 2009.

[40] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, pages 1–9, 2014.

[41] Ankush Desai, Tommaso Dreossi, and Sanjit A Seshia. Combining model checking and runtime verification for safe robotics. In *Proc. RV*, pages 172–189, 2017.

[42] BS Dhillon. Robot reliability. In *Robot Reliability and Safety*, pages 119–149. Springer, 1991.

[43] James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Comput. Chem. Eng.*, 17(3):245 – 255, 1993.

[44] Zhuohua Duan, Zixing Cai, and Jinxia Yu. Adaptive particle filter for unknown fault detection of wheeled mobile robots. In *Proc. IROS*, pages 1312–1315, 2006.

[45] Francis Ysidro Edgeworth. XLI. on discordant observations. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 23(143):364–375, 1887.

[46] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[47] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[48] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Mach. Learn.*, 107(3):481–508, 2018.

[49] David Forney. The Viterbi algorithm. *P IEEE*, 61(3):268–278, 1973.

[50] Mohammed Foughali, Saddek Bensalem, Jacques Combaz, and Félix Ingrand. Runtime verification of timed properties in autonomous robots. In *Proc. MEMOCODE*, pages 1–12, 2020.

[51] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *Proc. CRITIS*, pages 88–99, 2016.

[52] Jonathan Goh, Sridhar Adepu, Marcus Tan, and Zi Shan Lee. Anomaly detection in cyber physical systems using recurrent neural networks. In *Proc. HASE*, pages 140–145, 2017.

[53] Raphael Golombek, Sebastian Wrede, Marc Hanheide, and Martin Heckmann. Learning a probabilistic self-awareness model for robotic systems. In *Proc. IROS*, pages 2745–2750, 2010.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[55] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, pages 1–15, 2015.

[56] Nico Görnitz, Mikio Braun, and Marius Kloft. Hidden Markov anomaly detection. In *Proc. ICML*, pages 1833–1842, 2015.

[57] Álvaro Gutiérrez, Alexandre Campo, Marco Dorigo, Jesus Donate, Félix Monasterio-Huelin, and Luis Magdalena. Open e-puck range & bearing miniaturized board for local communication in swarm robotics. In *Proc. ICRA*, pages 3111–3116, 2009.

[58] Heiko Hamann. *Swarm robotics: A formal approach*. Springer, 2018.

[59] James Douglas Hamilton. *Time series analysis*. Princeton university press, 1994.

[60] Samuel Harford, Fazle Karim, and Houshang Darabi. Generating adversarial samples on multivariate time series using variational autoencoders. *IEEE/CAA J. Automatica Sinica*, 8(9):1523–1538, 2021.

[61] Fouzi Harrou, Belkacem Khaldi, Ying Sun, and Foudil Cherif. Monitoring robotic swarm systems under noisy conditions using an effective fault detection strategy. *IEEE Sensors Journal*, 19(3):1141–1152, 2018.

[62] Fouzi Harrou, Belkacem Khaldi, Ying Sun, and Foudil Cherif. Statistical detection of faults in swarm robots under noisy conditions. In *Proc. CEIT*, pages 1–6, 2018.

[63] Fouzi Harrou, Belkacem Khaldi, Ying Sun, and Foudil Cherif. An efficient statistical strategy to monitor a robot swarm. *IEEE Sens. J.*, 20(4):2214–2223, 2019.

[64] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning*. Springer, 2001.

[65] Kai Häussermann, Oliver Zweigle, and Paul Levi. A novel framework for anomaly detection of robot behaviors. *J Intell Robot Syst*, 77(2):361–375, 2015.

[66] Nick Hawes, Christopher Burbridge, et al. The STRANDS project: Long-term autonomy in everyday environments. *IEEE RAM*, 24(3):146–156, 2017.

[67] Ernst Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271, 1909.

[68] John Hershey, Peder Olsen, and Steven Rennie. Variational Kullback-Leibler divergence for hidden Markov models. In *Proc. ASRU*, pages 323–328, 2007.

[69] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proc. ICLR*, pages 1–22, 2017.

[70] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[71] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, pages 1–9, 2015.

[72] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[73] Tesheng Hsiao and Mao-Chiao Weng. A hierarchical multiple-model approach for detection and isolation of robotic actuator faults. *Robot Auton Syst*, 60(2):154–166, 2012.

[74] Jeff Huang, Cansu Erdogan, Yi Zhang, Brandon Moore, Qingzhou Luo, Aravind Sundaresan, and Grigore Rosu. Rosrv: Runtime verification for robots. In *Proc. RV*, pages 247–254, 2014.

[75] Rolf Isermann. Model-based fault-detection and diagnosis – status and applications. *Annu Rev Control*, 29(1):71–85, 2005.

[76] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Adversarial attacks on deep neural networks for time series classification. In *Proc. IJCNN*, pages 1–8, 2019.

[77] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):1–32, 2021.

[78] Saurabh Jha, Subho Banerjee, Timothy Tsai, Siva KS Hari, Michael B Sullivan, Zbigniew T Kalbarczyk, Stephen W Keckler, and Ravishankar K Iyer. Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection. In *Proc. DSN*, pages 112–124, 2019.

[79] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proc. EMNLP*, pages 2021–2031, 2017.

[80] Biing-Hwang Juang and Lawrence Rabiner. A probabilistic distance measure for hidden Markov models. *AT&T Tech J*, 64(2):391–408, 1985.

[81] Mateusz Kalisch. Fault detection method using context-based approach. In *Advanced and Intelligent Computations in Diagnosis and Control*, pages 383–395. 2016.

[82] Fazle Karim, Somshubra Majumdar, and Houshang Darabi. Adversarial attacks on time series. *IEEE PAMI*, pages 3309–3320, 2020.

[83] Azarakhsh Keipour, Mohammadreza Mousaei, and Sebastian Scherer. ALFA: A Dataset for UAV Fault and Anomaly Detection. *Int J Rob Res*, 0(0):1–6, 2020.

[84] James Kennedy. Swarm intelligence. In *Handbook of nature-inspired and innovative computing*, pages 187–219. Springer, 2006.

[85] Eliahu Khalastchi and Meir Kalech. On fault detection and diagnosis in robotic systems. *ACM Comput Surv*, 51(1):1–24, 2018.

[86] Eliahu Khalastchi and Meir Kalech. Fault detection and diagnosis in multi-robot systems: a survey. *Sensors*, 19(18):4019, 2019.

[87] Eliahu Khalastchi, Meir Kalech, Gal Kaminka, and Raz Lin. Online data-driven anomaly detection in autonomous robots. *Knowl Inf Syst*, 43(3):657–688, 2015.

[88] Belkacem Khaldi, Fouzi Harrou, Foudil Cherif, and Ying Sun. Monitoring a robot swarm using a data-driven fault detection approach. *Robot Auton Syst*, 97:193–203, 2017.

[89] Belkacem Khaldi, Fouzi Harrou, Ying Sun, and Foudil Cherif. A measurement-based fault detection approach applied to monitor robots swarm. In *Proc. ICSC*, pages 21–26, 2017.

[90] Diederik Kingma and Max Welling. Auto-encoding variational Bayes. In *Proc. ICLR*, pages 1–14, 2014.

[91] Tomáš Krajník, Jaime Fentanes, Grzegorz Cielniak, Christian Dondrup, and Tom Duckett. Spectral analysis for long-term robotic mapping. In *Proc. ICRA*, pages 3706–3711, 2014.

[92] Tomáš Krajník, Jaime P Fentanes, Joao M Santos, and Tom Duckett. Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments. *IEEE T-RO*, 33(4):964–977, 2017.

[93] Harold Kuhn. The Hungarian method for the assignment problem. *Nav Res Logist Q*, 2(1-2):83–97, 1955.

[94] Solomon Kullback and Richard Leibler. On information and sufficiency. *Ann Math Stat*, 22(1):79–86, 1951.

[95] Lars Kunze, Nick Hawes, Tom Duckett, Marc Hanheide, and Tomás Krajník. Artificial intelligence for long-term robot autonomy: A survey. *IEEE RA-L*, 3(4):4023–4030, 2018.

[96] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv:1611.01236*, pages 1–17, 2016.

[97] Truls Larsson, Kristin Hestetun, Espen Hovland, and Sigurd Skogestad. Self-optimizing control of a large-scale plant: The tennessee eastman process. *Ind. Eng. Chem. Res.*, 40(22):4889–4901, 2001.

[98] HuiKeng Lau, Iain Bate, Paul Cairns, and Jon Timmis. Adaptive data-driven error detection in swarm robotics with statistical classifiers. *Robot Auton Syst*, 59(12):1021–1035, 2011.

[99] N. Lawrence Ricker. Decentralized control of the tennessee eastman challenge process. *J. Process Control*, 6(4):205 – 221, 1996.

[100] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data Augmentation for Time Series Classification using Convolutional Neural Networks. In *Proc. AALTD@ECML/PKDD*, pages 1–8, 2016.

[101] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[102] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *JMLR*, 18(1):559–563, 2017.

[103] Kalet Leon, Rolando Perez, Agustin Lage, and Jorge Carneiro. Modelling t-cell-mediated suppression dependent on interactions in multicellular conjugates. *J. Theor. Biol.*, 207(2):231–254, 2000.

[104] Zhiyuan Liu and Jie Zhou. Introduction to graph neural networks. *Synth. Lect. Artif. Intell. Mach. Learn.*, 14(2):1–127, 2020.

[105] Alex Lotz, Andreas Steck, and Christian Schlegel. Runtime monitoring of robotics software components: Increasing robustness of service robotic systems. In *Proc. ICAR*, pages 285–290, 2011.

[106] Francesca Lunardini, Matteo Luperto, Marta Romeo, Jennifer Renoux, Nicola Basilico, Andrej Krpič, Simona Ferrante, and N.Alberto Borghese. The MOVECARE project: Home-based monitoring of frailty. In *Proc. BHI*, pages 1–4, 2019.

[107] Matteo Luperto, Danilo Fusi, N Alberto Borghese, and Francesco Amigoni. Robot exploration using knowledge of inaccurate floor plans robot exploration using knowledge of inaccurate floor plans. In *Proc. ECMR*, pages 1–7, 2019.

[108] Matteo Luperto, Javier Monroy, J Ruiz-Sarmiento, Francisco-Angel Moreno, Nicola Basilico, Javier Gonzalez-Jimenez, and N Alberto Borghese. Towards long-term deployment of a mobile robot for at-home ambient assisted living of the elderly. In *Proc. ECMR*, pages 1–6, 2019.

[109] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. LSTM-based encoder-decoder for multi-sensor anomaly detection. In *Proc. AD@ICML*, pages 1–5, 2016.

[110] Aditya P Mathur and Nils Ole Tippenhauer. Swat: a water treatment testbed for research and training on ics security. In *Proc. CySWater@CPS*, pages 31–36, 2016.

[111] Olivier Graham Miller and Vaibhav Gandhi. A survey of modern exogenous fault detection and diagnosis methods for swarm robotics. *Journal of King Saud University-Engineering Sciences*, 33(1):43–53, 2021.

[112] Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE PAMI*, 41(8):1979–1993, 2019.

[113] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proc. ICARSC*, pages 59–65, 2009.

[114] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv:1511.04599*, pages 1–9, 2015.

[115] Konda Reddy Mopuri, Aditya Ganeshan, and R. Venkatesh Babu. Generalizable data-free objective for crafting universal adversarial perturbations. *IEEE PAMI*, 41(10):2452–2465, 2019.

[116] Francisco-Angel Moreno, Javier Monroy, Jose-Raul Ruiz-Sarmiento, Cipriano Galindo, and Javier Gonzalez-Jimenez. Automatic waypoint generation to improve robot navigation through narrow spaces. *Sensors*, 20(1):240, 2020.

[117] Brendan Tran Morris and Mohan Manubhai Trivedi. Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach. *IEEE PAMI*, 33(11):2287–2301, 2011.

[118] Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, and Xinghuo Yu. Spatiotemporal anomaly detection using deep learning for real-time video surveillance. *IEEE Trans Industr Inform*, 16(1):393–402, 2019.

[119] Matteo Olivato, Omar Cotugno, Lorenzo Brigato, Domenico Bloisi, Alessandro Farinelli, and Luca Iocchi. A comparative analysis on the use of autoencoders for robot security anomaly detection. In *Proc. IROS*, pages 984–989, 2019.

[120] Izaskun Oregi, Javier Del Ser, Aritz Pérez, and José Antonio Lozano. Adversarial sample crafting for time series classification with elastic similarity measures. In *Proc. IDC*, pages 26–39, 2018.

[121] Guillermo Ortiz-Jiménez, Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Optimism in the face of adversity: Understanding and improving deep learning through adversarial robustness. *P IEEE*, 109(5):635–659, 2021.

[122] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Comput Surv*, 54(2):1–38, 2021.

[123] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Proc. IEEE EuroSP*, pages 372–387, 2016.

[124] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proc. ASIA-CCS*, pages 506–519, 2017.

[125] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv:1605.07277*, pages 1–8, 2016.

[126] Daehyung Park, Zackory Erickson, Tapomayukh Bhattacharjee, and Charles Kemp. Multimodal execution monitoring for anomaly detection during robot manipulation. In *Proc. ICRA*, pages 407–414, 2016.

[127] Daehyung Park, Yuuna Hoshi, and Charles Kemp. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE RA-L*, (3):1544–1551, 2018.

[128] Daehyung Park, Hokeun Kim, and Charles Kemp. Multimodal anomaly detection for assistive robots. *Auton Robot*, 43(3):611–629, 2019.

[129] RJ Patton, FJ Uppal, and CJ Lopez-Toribio. Soft computing approaches to fault diagnosis for dynamic systems: a survey. *IFAC*, 33(11):303–315, 2000.

[130] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

[131] Joao Pereira and Margarida Silveira. Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In *Proc. ICMLA*, pages 1275–1282, 2018.

[132] Joao Pereira and Margarida Silveira. Learning representations from healthcare time series data for unsupervised anomaly detection. In *Proc. BigComp*, pages 1–7, 2019.

[133] Ola Pettersson. Execution monitoring in robotics: A survey. *Robot Auton Syst*, 53(2):73–88, 2005.

[134] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6(4):271–295, 2012.

[135] Jim Pugh, Xavier Raemy, Cedric Favre, Riccardo Falconi, and Alcherio Martinoli. A fast onboard relative positioning module for multirobot systems. *IEEE/ASME Trans. Mechatron.*, 14(2):151–162, 2009.

[136] Lawrence Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *P IEEE*, 77(2):257–286, 1989.

[137] S Benson Edwin Raj and A Annie Portia. Analysis on credit card fraud detection methods. In *Proc. ICCCET*, pages 152–156, 2011.

[138] Anand Rajaraman and Jeffrey Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[139] D Ramyachitra and P Manikandan. Imbalanced dataset classification and solutions: a review. *IJCBR*, 5(4):1–29, 2014.

[140] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[141] James F Roberts, Timothy Stirling, Jean-Christophe Zufferey, and Dario Floreano. 3-d relative positioning sensor for indoor flying robots. *Auton Robot*, 33(1):5–20, 2012.

[142] James F Roberts, Timothy S Stirling, Jean-Christophe Zufferey, and Dario Floreano. 2.5 d infrared range and bearing system for collective robotics. In *Proc. IROS*, pages 3659–3664, 2009.

[143] William Robinson and Andrea Aria. Sequential fraud detection for prepaid cards using hidden Markov model divergence. *Expert Syst Appl*, 91:235–251, 2018.

[144] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *P IEEE*, 2021.

[145] Addisson Salazar, Luis Vergara, and Gonzalo Safont. Generative adversarial networks and markov random fields for oversampling very small training sets. *Expert Syst. Appl.*, 163:1–12, 2021.

[146] César Sánchez, Gerardo Schneider, Wolfgang Ahrendt, Ezio Bartocci, Domenico Bianculli, Christian Colombo, Yliès Falcone, Adrian Francalanza, Srđan Krstić, Joao M Lourenço, et al. A survey of challenges for runtime verification from advanced application domains (beyond software). *Form Methods Syst Des*, 54(3):279–335, 2019.

[147] Michael Schnall-Levin, Leonid Chindelevitch, and Bonnie Berger. Inverting the viterbi algorithm: An abstract framework for structure design. In *Proc. ICML*, pages 904–911, 2008.

[148] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *J Big Data*, 6(60):1 – 48, 2019.

[149] Aman Sinha, Hongseok Namkoong, and John C. Duchi. Certifying some distributional robustness with principled adversarial training. In *Proc. ICLR*, pages 1–10, 2018.

[150] Maximilian Sölch, Justin Bayer, Marvin Ludersdorfer, and Patrick van der Smagt. Variational inference for on-line anomaly detection in high-dimensional time series. In *Proc. AD@ICML*, 2016.

[151] Gerald Steinbauer. A survey about faults of robots used in robocup. In *Proc. RobCup*, pages 344–355, 2012.

[152] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, pages 1–10, 2014.

[153] Danesh Tarapore, Anders Lyhne Christensen, and Jon Timmis. Abnormality detection in robots exhibiting composite swarm behaviours. In *Proc. ECAL*, pages 406–413, 2015.

[154] Danesh Tarapore, Anders Lyhne Christensen, and Jon Timmis. Generic, scalable and decentralized fault detection for robot swarms. *PloS One*, 12(8):1–29, 2017.

[155] Danesh Tarapore, Pedro U Lima, Jorge Carneiro, and Anders Lyhne Christensen. To err is robotic, to tolerate immunological: fault detection in multirobot systems. *Bioinspir. Biomim.*, 10(1):1–19, 2015.

[156] Danesh Tarapore, Jon Timmis, and Anders Lyhne Christensen. Fault detection in a swarm of physical robots based on behavioral outlier detection. *IEEE T-RO.*, 35(6):1516–1522, 2019.

[157] Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proc. ICMI*, pages 216–220, 2017.

[158] Vandi Verma, Geoff Gordon, Reid Simmons, and Sebastian Thrun. Real-time fault diagnosis [robot fault diagnosis]. *IEEE RAM*, 11(2):56–66, 2004.

[159] Pascal Vincent, Hugo Larochelle, Y. Bengio, and Pierre Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. ICML*, pages 1096–1103, 2008.

[160] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Proc. NIPS*, pages 5334–5344, 2018.

[161] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proc. SSP*, pages 133–145, 1999.

[162] Matt Webster, Clare Dixon, Michael Fisher, Maha Salem, Joe Saunders, Kheng Lee Koay, Kerstin Dautenhahn, and Joan Saez-Pons. Toward reliable autonomous robotic assistants through formal verification: A case study. *IEEE Trans. Hum.-Mach. Syst.*, 46(2):186–196, 2015.

[163] Qingsong Wen, Liang Sun, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *arXiv:2002.12478*, pages 1–10, 2020.

[164] Alan FT Winfield, Jin Sa, Mari-Carmen Fernández-Gago, Clare Dixon, and Michael Fisher. On formal specification of emergent behaviours in swarm robotic systems. *Int. J. Adv. Robot. Syst.*, 2(4):39, 2005.

[165] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. Understanding data augmentation for classification: when to warp? In *Proc. DICTA*, pages 1–6, 2016.

[166] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *arXiv:1909.08072*, pages 1–12, 2019.

[167] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao. Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. In *Proc. WWW*, pages 187–196, 2018.

[168] Nong Ye. A Markov chain model of temporal behavior for anomaly detection. In *Proc. IA@IEEE SMC*, pages 171–174, 2000.

[169] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proc. AAAI*, pages 1409–1416, 2019.

[170] Long Zhao, Ting Liu, Xi Peng, and Dimitris Metaxas. Maximum-entropy adversarial data augmentation for improved generalization and robustness. *arXiv:2010.08001*, pages 1–10, 2020.

[171] Yong Zhao, Chengsuo Zhang, Frank Soong, Min Chu, and Xi Xiao. Measuring attribute dissimilarity with HMM KL-divergence for speech synthesis. In *Proc. SSW*, pages 206–210, 2007.

[172] Duan Zhuo-Hua, CAI Zi-xing, and YU Jin-xia. Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: A survey. In *Proc. ICRA*, pages 3428–3433, 2005.