Machine learning for anomaly detection: From surface to deep

David Novoa Paradela

Doctoral Thesis

February 2024

PhD Advisors:
Óscar Fontenla Romero · Bertha Guijarro Berdiñas

PhD Program in Computer Science

UNIVERSIDADE DA CORUÑA

Óscar Fontenla Romero
Catedrático de Universidad
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

Bertha Guijarro Berdiñas
Titular de Universidad
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

CERTIFICAN

Que la memoria titulada "*Machine learning for anomaly detection: From surface to deep*" ha sido realizada por D. David Novoa Paradela bajo nuestra dirección en el Departamento de Ciencias de la Computación y Tecnologías de la Información de la Universidade da Coruña, y concluye la Tesis Doctoral que presenta para optar al grado de Doctor en Ingeniería Informática.

En A Coruña, a 28 de febrero de 2023

Fdo.: Óscar Fontenla Romero
Director de la Tesis Doctoral

Fdo.: Bertha Guijarro Berdiñas
Directora de la Tesis Doctoral

Fdo.: David Novoa Paradela
Autor de la Tesis Doctoral

Acknowledgments

Comenzar un doctorado en plena cuarentena mundial podría parecer —como mínimo— una tarea complicada. Y muchos coincidirán en que realmente lo será siempre, independientemente del momento y del lugar. En mi caso particular, ha sido un proceso por momentos amargo, pero haciendo un balance global, muy pero que muy beneficioso. Esta etapa me ha enseñado multitud de lecciones que serán de gran ayuda, no solo a lo largo de mi carrera, sino en todos los aspectos de mi vida.

Quiero reconocer en primer lugar a Óscar y Bertha por su esfuerzo, dedicación y comprensión. Solo tengo buenas palabras para vosotros. Habéis sido una especie de segundos padres con los que he medrado académica y profesionalmente. Si tuviese que volver a empezar, me gustaría hacerlo otra vez con vosotros a mí lado.

También me gustaría darle las gracias a los miembros del grupo LIDIA, compañeros y excompañeros de laboratorio, de café, y a todos aquellos que me habéis acogido a lo largo de estos años. Desde el momento en el que os conocí me he sentido como en casa.

Por último, pero no menos importante, quiero agradecerle a mi familia y amigos el haber estado siempre ahí, independientemente de encontrarnos cerca o lejos. Sin vosotros no habría llegado hasta aquí.

A veces, lo que comienza pareciendo una *anomalía*, puede acabar convirtiéndose en lo más *normal* del mundo.

David Novoa Paradela

On the shoulders of giants.

Resumo

A detección de anomalías é a rama da aprendizaxe automática encargada de construír modelos capaces de diferenciar entre datos normais e anómalos. *A priori*, isto converte a detección de anomalías nun problema de clasificación en dúas clases. Con todo, dado que as anomalías adoitan ocorrer de forma esporádica, os datos normais son os que prevalecen nestes escenarios, polo que é habitual que se requiran modelos específicos cuxo adestramento sexa levado a cabo empregando unicamente datos da clase normal. Estes sistemas xogan un papel vital nunha ampla gama de aplicacións reais, como a medicina, a detección de fraudes bancarias, intrusións na rede, ou o mantemento preditivo de sistemas industriais.

Esta tese centrouse no desenvolvemento de novos algoritmos de detección de anomalías para tres escenarios diferentes, desde modelos baseados na xeometría dos datos, ata modelos máis complexos baseados en aprendizaxe profunda. No primeiro escenario propoñemos un método baseado en peches non convexos subdivisibles para escenarios tradicionais, nos que algúns dos principais problemas son a dimensionalidade dos datos de adestramento e a forma da nube de puntos no espacio n-dimensional. No segundo escenario preséntase unha rede *autoencoder* profunda válida para escenarios de computación no borde e aprendizaxe federado debido ao seu adestramento non iterativo, así como unha arquitectura federada para a súa implementación semi-centralizada. Por último, en liña coas novas tendencias da intelixencia artificial aplicable, estudouse a aplicabilidade da detección de anomalías sobre textos, propoñendo un *pipeline* para a detección de recensións anómalas explicable en plataformas de comercio electrónico.

O traballo desenvolvido foi compartido coa comunidade investigadora a través de publicacións en revistas e conferencias científicas, ademais de repositorios de código aberto, contribuíndo así ao avance do campo da detección de anomalías.

Resumen

La detección de anomalías es la rama del aprendizaje automático encargada de construir modelos capaces de diferenciar entre datos normales y anómalos. *A priori*, esto convierte la detección de anomalías en un problema de clasificación en dos únicas clases. Sin embargo, dado que las anomalías suelen ocurrir de forma esporádica, los datos normales son los que prevalecen en estos escenarios, por lo que es habitual que se requieran modelos específicos cuyo entrenamiento se lleve a cabo empleando únicamente datos de la clase normal. Estos sistemas juegan un papel vital en una amplia gama de aplicaciones reales, como la medicina, la detección de fraudes bancarios, intrusiones en la red, o el mantenimiento predictivo de sistemas industriales.

Esta tesis se ha centrado en el desarrollo de nuevos algoritmos de detección de anomalías para tres escenarios diferentes, desde modelos basados en la geometría de los datos, hasta modelos más complejos basados en aprendizaje profundo. En el primer escenario proponemos un método basado en cierres no convexos subdivisibles para entornos tradicionales, en los que algunos de los principales problemas son la dimensionalidad de los datos de entrenamiento y la forma de la nube de puntos en el espacio n-dimensional. En el segundo escenario se presenta una red *autoencoder* profunda válida para entornos de computación en el borde y aprendizaje federado debido a su entrenamiento no iterativo, así como una arquitectura federada para su implementación semi-centralizada. Por último, en línea con las nuevas tendencias de inteligencia artificial aplicable, se ha estudiado la aplicabilidad de la detección de anomalías sobre textos, proponiendo un *pipeline* para la detección de reseñas anómalas explicable en plataformas de comercio electrónico.

El trabajo desarrollado ha sido compartido con la comunidad investigadora a través de publicaciones en revistas y congresos científicos, así como repositorios de

código abierto, contribuyendo de este modo al avance del campo de la detección de anomalías.

Abstract

Anomaly detection is the branch of machine learning in charge of building models capable of differentiating between normal and anomalous data. *A priori*, this makes anomaly detection a two-class classification problem. However, since anomalies tend to occur sporadically, normal data are the ones that prevail in these scenarios, so it is common to require specific models whose training is carried out using only data from the normal class. These systems play a vital role in a wide range of real-world applications, such as medicine, bank fraud detection, network intrusions, or predictive maintenance of industrial systems.

This thesis has focused on the development of new anomaly detection algorithms for three different scenarios, from models based on data geometry to more complex models based on deep learning. In the first scenario, we propose a method based on subdivisible non-convex hulls for traditional environments, where some of the main problems are the dimensionality of the training data and the shape of the point cloud in the n-dimensional space. In the second scenario, a deep autoencoder network valid for edge computing and federated learning environments is presented due to its non-iterative training, as well as a federated architecture for its semi-centralized implementation. Finally, in accordance with the new trends in applied artificial intelligence, the applicability of anomaly detection on text has been studied, proposing a pipeline for the detection of explainable anomalous reviews in e-commerce platforms.

The work developed has been shared with the research community through publications in scientific journals and conferences, as well as open source repositories, thus contributing to the advancement of the field of anomaly detection.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, society has witnessed an unprecedented explosion of data in practically every aspect of our lives. This vast volume of information, known as Big Data, has transformed how we live, work, and socialize. The appearance of this trend is due to three main reasons:

1. Improved computational capacity: technological advances have led to the emergence of devices with high computing power and storage capacity.

2. Increase in the number of devices: the cheapening of these devices has allowed their widespread use, giving rise to millions of data sources of all kinds.

3. Global connectivity: the expansion of the Internet has generated a global network of data exchange that connects all these devices.

This data-rich ecosystem is ideal for areas of computer science such as Machine Learning (ML), whose algorithms draw directly from these heterogeneous data sources. This is one of the main reasons why the field of Artificial Intelligence (AI) in general, and machine learning in particular, have undergone a major boost in the last decade. A tendency that seems likely to continue, at least for the next few years. Among the innumerable problems that artificial intelligence can solve, one of them is Anomaly Detection.

1.1. Anomaly detection

Anomaly detection (AD) [29, 153, 175], also referred to as outlier detection and sometimes as novelty detection, is the branch of machine learning that builds models capable of differentiating between normal and anomalous data. *A priori*, this turns anomaly detection into a classification problem with only two classes. However, since anomalies tend to occur sporadically, normal data are the ones that prevail in these scenarios so, commonly, models are required that can be trained with only data of the normal class. Assuming all training instances have only one class label, in anomaly detection is common to use One-Class Classification techniques. One-Class Classification (OCC) [94] is a special case of binary classification, where data observed during training is from a single class. The objective is to represent the boundary of the normal class with high precision to be able to classify new data by comparing them with these limits.

By understanding the patterns and characteristics of the normal class, the anomaly detection model can identify instances that deviate significantly from this learned distribution, thereby detecting anomalies more accurately. This approach allows for the detection of abnormal or unusual behaviors that falls outside the expected patterns observed in the normal class, enabling effective anomaly detection in various domains and applications. Among the most common application areas of anomaly detection are fraud detection [162, 3, 101, 241, 88, 45], cyber-intrusion [96, 115, 3, 16, 187, 107], medicine [52, 220, 82, 152, 56, 70], or industry [194, 126, 89, 104, 91, 27]. In many real-world scenarios, the response time to the detection of an anomaly can be critical, as in the case of failures in industrial systems or network intrusions. In other settings, such as medical or banking, the privacy of the data being exchanged is one of the main requirements. In scenarios such as those using Internet Of Things (IoT) devices, the main requirement could be low computational and bandwidth cost.

Despite being a successful discipline with a wide range of uses today, anomaly detection, like other areas of machine learning and artificial intelligence, is an ever-expanding line of work that faces numerous challenges:

- **Non-convex and disjointed shapes.** In anomaly detection, the distribution presented by the datasets, understood as how data is spread in the

n-dimensional space, can be decisive for the application of certain types of techniques. Anomaly detection systems play a vital role in a wide range of real-world applications where the shape of the normal class will not always be convex. A convex representation can never provide good characterization of a non-convex distribution, so it is necessary to develop specific methods capable of operating on non-convex datasets. In addition, the distribution of the data can be composed of several separate regions in space that are difficult to represent using a single hull (see Figure 1.1).

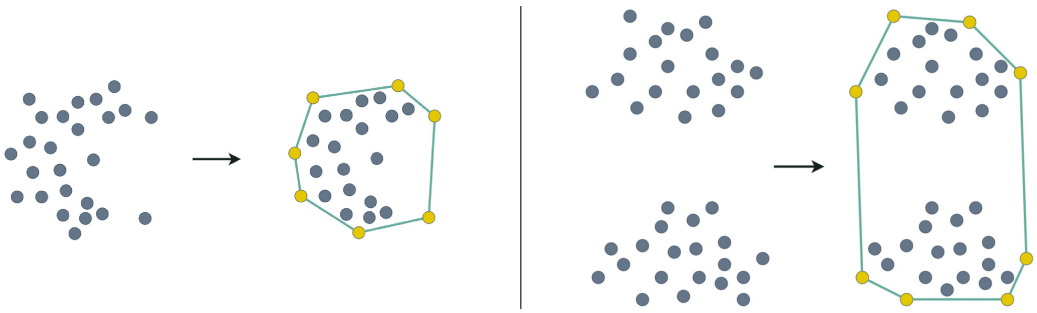


Figure 1.1: When the point cloud that forms the normal class has non-convex regions (left) or regions separated in space (right), a convex hull may not provide sufficient precision for its representation.

Boundary-based methods are those that generate a limit based on the structure of the training data to separate classes in space. These classifiers become insensitive to the size and density of each class since the classification of the new data is determined only by its location concerning the boundary, and not by its density [37, 167]. Computing the boundary of the normal class in the original space is a strategy followed by some methods [112, 154]. However, working with high-dimensional datasets makes it difficult to calculate these structures due to their high computational cost. One solution is to reduce the dimensionality of the dataset to make these operations manageable. The most common methods to simplify the representation of the problem by reducing the dimensionality of data are Principal Component Analysis (PCA) [75], Linear Discriminant Analysis (LDA) [81], or Autoencoder Networks (AE) [11].

Whether in the original space or reduced dimensional spaces, anomaly detection models must be able to represent non-convex and disjoint regions, which requires specific techniques.

- **Edge Computing.** Technological development in recent years has led to a substantial increase in the variety of computing devices such as smartphones, wearables, or IoT devices, both for personal and industrial use. Despite their small size, these have enough computational power to perform tasks that a few years ago were considered unaffordable, such as training small machine learning models, real-time inference or exchanging large amounts of data at high speed.

  Due to the abundance of these devices and the inefficiencies of traditional cloud computing to perform processes requiring low latencies, a new computing paradigm called Edge Computing (EC) [26, 95, 209] has emerged. Edge computing moves computing from data centers to the edge of the network, bringing cloud computing services and utilities closer to the end user and their devices. This enables faster information processing and response time, as well as freeing up network bandwidth. It plays an important application role in Content Delivery Network (CDN) [232], industrial IoT (IIoT) [165], smart home [229], smart transportation [116], and other fields.

  Edge computing presents challenges such as resource constraints, device and network heterogeneity, data security and privacy, reliability and fault tolerance, complexity management, and scalability. These challenges require efficient resource utilization, interoperability, robust security measures, fault-tolerant mechanisms, simplified management, and scalable architectures to ensure the successful implementation and widespread adoption of edge computing. Because of this, adapting classical anomaly detection algorithms or designing new ones to be used in these scenarios is a necessary task.

- **Federated learning.** From a machine learning point of view, this current device-plagued technological scenario is well suited for the use of federated learning. Federated Learning (FL) [236, 110, 20] is a collaborative machine learning scheme that allows heterogeneous devices with different private datasets to work together to train a global model. It plays an important role in a wide range of areas such as language model development [65], recommendation systems [227], intelligent medical diagnostic [173], or wireless communications [143].

  Challenges in federated learning encompass privacy protection, communication

cost, heterogeneity of devices involved, and unreliable model uploads. Privacy concerns arise due to the potential leakage of sensitive information during model communication. Communication efficiency becomes crucial when dealing with large-scale federated networks. The heterogeneity of systems, including hardware and network variations, presents challenges in fault tolerance. Additionally, preventing unreliable model uploads is essential to maintain the integrity of federated learning. Addressing these challenges requires the development of methods that prioritize privacy preservation, optimize communication efficiency, adapt to system heterogeneity, and mitigate the impact of unreliable model uploads. Adapting classical anomaly detection algorithms or designing new ones to be used in these scenarios is also a necessary task.

- **Natural Language Processing.** Although there are models capable of dealing directly with images or text, classical machine learning methods are usually designed to work with structured data. In the area of natural language processing (NLP), there are multiple techniques to represent texts using vectors of real numbers, known as embeddings. These techniques make it possible to generate vector spaces that represent the semantic relations and similarities of the language, so that, for example, two synonymous words will be found at a shorter distance in space than two unrelated words.

  Technological development in recent years has enabled the construction of very powerful models for the NLP field [34]. However, unlike other tasks such as sentiment analysis [197] or question answering [97], the application of anomaly detection in texts is not common, probably due to the difficulty in defining the concept of anomaly in this type of scenario. Bringing anomaly detection to the NLP area is an exercise that can be beneficial for many applications such as log analysis in cybersecurity [225], online social network analysis [184], and customer feedback analysis [178].

- **Explainability.** As black-box machine learning models are increasingly used to make important predictions in critical contexts, the demand for transparency is growing. The danger lies in using decisions that are not justifiable, legitimate or that do not allow for detailed explanations of their behavior. Explanations that support the output of a model are crucial, especially in scenarios that can directly affect us as humans, such as medicine, autonomous

driving, or the safety of industrial systems.

When machine learning models do not meet the criteria imposed to be considered explainable, a model-independent method must be applied to explain their decisions. This is the purpose of post-hoc explainability techniques [14], whose objective is to generate understandable information on how an already developed model produces its predictions for any given input. Within these techniques, the so-called model-agnostic techniques are those designed to be used on any model to extract information from its internal reasoning, such as LIME [172] or SHAP [121].

The growing demand for transparency, interpretability, and confidence in the systems also includes anomaly detection models, so it is necessary to incorporate this feature in the development of new algorithms and to study how to evaluate the quality of such explanations.

In this thesis, we decided to develop anomaly detection methods addressing the aspects mentioned above. First, we have developed and evaluated a boundary-based anomaly detection method capable of dealing with non-convex and disjointed shapes, as well as being parallelizable. Then, we developed and evaluated a new implementation of non-iterative and deep autoencoder networks valid for edge computing and federated learning scenarios. Furthermore, we have proposed a semi-decentralized architecture for these scenarios. Finally, we have brought anomaly detection closer to the NLP field by proposing a pipeline to detect anomalous reviews in e-commerce platforms, using explainability models to justify the decisions made by the system.

1.2. Structure of this thesis

This thesis is divided into four parts, each focused on addressing the issues discussed above and providing valuable insights and practical approaches to meet the challenges outlined. These parts are:

1. **The field of anomaly detection.** The field of anomaly detection has certain characteristics that make it unique compared to other machine learning tasks.

Because of this, the methods used for anomaly detection, as well as their applicability and limitations, need to be studied in depth.

In Chapter 2 we present a theoretical introduction to the field covering the aspects mentioned above, which will serve as a context for the research carried out during this thesis presented in the following chapters.

2. **Anomaly detection in traditional scenarios.** One of the major difficulties faced by boundary-based anomaly detection algorithms occurs when the normal class distribution follows a non-convex shape with disjointed regions. This is even more complex if the operations performed by these algorithms do not scale correctly with the dimensionality of the data, and may even become unaffordable in reasonable times.

   In Chapter 3 we propose a one-class classification method valid for non-convex and disjointed datasets employing random projections and expandable non-convex hulls. The method offers a robust behavior in the different tests carried out, and its performance is remarkable, positioning itself as an alternative for both convex and non-convex problems. In addition, it is easily configurable through its hyperparameters, the models generated with this algorithm are light on memory requirements, and its training and execution can be parallelized.

3. **Anomaly detection in edge computing and federated learning scenarios.** Autoencoder networks are one of the most powerful and widely used methods in anomaly detection, however, their training can be too expensive if complex architectures with a high number of layers and neurons are used. This makes difficult to use this type of model in edge computing and federated learning scenarios, where the devices are usually not computationally powerful.

   In Chapter 4 we present a novel, fast, and privacy preserving implementation of deep autoencoders. Unlike traditional neural networks, the method trains a deep autoencoder network in a non-iterative way, which drastically reduces training time. Training can be performed incrementally, in parallel, and distributed and, thanks to its mathematical formulation, the information to be exchanged does not endanger the privacy of the training data. The method has

been evaluated and compared with other state-of-the-art autoencoders, showing interesting results in terms of accuracy, speed, and utilization of available resources, which makes it a valid method for edge computing and federated learning, in addition to classical machine learning scenarios.

In this chapter we also present a semi-decentralized architecture for the use of the proposed network in a federated manner, establishing the communication protocol and the roles of the nodes, and demonstrating its effectiveness by simulating different scenarios.

4. **Anomaly detection in natural language scenarios.** Anomaly detection in text scenarios is not one of the most common tasks, probably due to the difficulty in defining the meaning of an anomaly in these situations. Likewise, both providing explainability in these scenarios and evaluating such explanations can become a challenging task.

   In Chapter 5 we propose a pipeline to detect anomalous reviews in online platforms, allowing the detection of reviews that do not generate value for users due to either accidental or malicious composition. The classifications are accompanied by a normality score and an explanation that justifies the decision made by the architecture. The pipeline capabilities have been evaluated using different datasets created from a large Amazon database.

   Additionally, a study comparing three explainability techniques involving 241 participants was conducted to evaluate this element of the pipeline. The study aimed to measure the impact of explanations on the respondents' ability to reproduce the classification model and their perceived usefulness, as well as to reflect on whether it is possible to explain tasks as humanly subjective as this one.

Finally, the main conclusions and contributions of this thesis are summarized in Chapter 6. In addition, the appendices include the values of the hyperparameters chosen during the experiments carried out (Appendix A), the scientific publications that support this thesis (Appendix B), and an extended summary (Appendix C).

Chapter 2

The field of anomaly detection

The search for outliers is as old as data analysis itself. The inclusion of outliers during model training can make it difficult, if not impossible, to correctly represent the structure of the data to be characterized. In the early days, outliers were considered as anomalous, possibly erroneous observations that had to be identified and removed from the analysis, or at least given less weight when building the model. However, it was soon realized that the outliers found representing anomalous observations could be reliable data and symbolize events of interest, such as the occurrence of unexpected faults in a system. The focus on these anomalous patterns and how to identify them has led to the emergence of a whole discipline within artificial intelligence called anomaly detection.

In this chapter, we introduce the field of anomaly detection by presenting the characteristics and foundations of the area that will be used throughout this thesis. First, we define the task of anomaly detection, then we present the types of techniques used to address this problem, subsequently that we highlight the most common application areas in anomaly detection, and finally we discuss the main challenges of the field.

2.1. The anomaly detection task

Anomaly Detection (AD) [29, 153, 175], also referred to as outlier detection, and sometimes as novelty detection, is the branch of machine learning that builds models capable of differentiating between normal and anomalous data. *A priori*, this turns anomaly detection into a classification problem with only two classes, also known as binary classification [102]. However, anomalies tend to occur infrequently, and when they do, they are often not labeled, or are insufficient to fully model the anomalous distribution, or directly do not follow any distribution at all. As a result, normal data are the prevailing data in these scenarios. Therefore, in anomaly detection, the most common scenario is to use models that are trained in an unsupervised way assuming that all the training data — or at least most of them — belong to the considered as the normal class.

The objective is to represent as accurately as possible the boundaries of the so-called normal distribution to be able to classify new data by comparing them with these boundaries. By understanding the patterns and characteristics of the normal class, the model will be able to perform the classification task. Given new data, if it fits the distribution learned by the model it will be classified as normal, while if it deviates significantly it will be classified as anomalous.

This approach allows for the detection of abnormal or unusual behaviors that fall outside the expected patterns observed in the normal class, enabling effective anomaly detection in various domains and applications. Among the most common application areas of anomaly detection are fraud detection [162, 3, 101, 241, 88, 45], cyber-intrusion [96, 115, 3, 16, 187, 107], medicine [52, 220, 82, 152, 56, 70] or industry [194, 126, 89, 104, 91, 27]. In many real-world scenarios, the *response time* to the detection of an anomaly can be critical, as in the case of failures in industrial systems or network intrusions. In other settings, such as medical or banking, the *privacy of the data* being exchanged is one of the main requirements. In scenarios such as those using Internet Of Things (IoT) devices, the main requirement is *low computational and bandwidth cost*. As with other machine learning disciplines, each scenario in which anomaly detection is adopted presents particular challenges and requirements.

2.1.1. What are anomalies?

In anomaly detection, defining the concept of anomaly is a complex task since anomalies can manifest themselves in a variety of ways and there is no single definition that fits all scenarios. Furthermore, what may be considered an anomaly in one context may be perfectly normal in another. The interpretation of what constitutes an anomaly will depend on the application domain and the specific characteristics of the data. Therefore, a thorough analysis and consideration of different perspectives is necessary to establish a precise and relevant definition of anomaly in each particular case.

Nevertheless, we can state that in a generalized way, and following the definition of Chandola *et al.* [29] in one of the most important papers in the area of anomaly detection, we can define an anomaly as:

> *"Anomalies are patterns in data that do not conform to a well defined notion of normal behavior."*

And based on the definition made by Lukas *et al.* [174], current referents in the field, we can complement the above with:

> *"Also known as outlier or novelty, such an observation may be termed unusual, irregular, atypical, inconsistent, unexpected, rare, erroneous, faulty, fraudulent, malicious, unnatural, or simply strange—depending on the situation."*

In short, and as we will see below, the definition of anomaly is partially dependent on the context in which it occurs, making anomaly detection a multidisciplinary area and very rich in terms of the number and nature of existing techniques, this being motivated precisely by the particular difficulties of each type of scenario.

2.1.2. Types of anomalies

Anomaly detection techniques depend heavily on the nature of the anomalies they are designed to detect. Originally, anomalies have been classified into three

main categories [29]: point anomalies, contextual anomalies, and collective anomalies.

Point anomalies are individual data instances that may be considered anomalous concerning the rest of the data, such as the transmission of a malicious network packet through a public network. This is the simplest and most commonly studied type of anomaly in the area of anomaly detection [238, 33].

Contextual anomalies, also known as *conditional anomalies*, are data instances that are anomalous in a specific context such as time or space but may not be so in other contexts. For example, that a heating system works to raise the temperature of a house to a high value is a behavior that we can consider normal if we are in a cold season. However, if this occurs during the hottest time of the year when the system should be working at minimum power, we are dealing with a contextual anomaly. Contextual anomalies are very frequent in time-series [189, 19, 171] and spatio-temporal [223, 69, 15] problems.

A *group* or *collective anomaly* is a set of related or dependent points that is anomalous concerning the entire dataset. The individual instances may not be anomalous by themselves, but their appearance together reflects an anomalous behavior. While point anomalies can occur in any dataset, collective anomalies can only occur in datasets where data instances are interrelated to each other, such as a denial of service (DoS) attack performed by a cluster of instances [1, 2].

However, with the rise of deep learning and the consequent proliferation of anomaly detection in new domains, this classification of existing anomaly types has been extended from new perspectives. Nowadays, it is also important to distinguish between low-level sensory anomalies and high-level semantic anomalies [174]. This approach focuses on the hierarchy of patterns present in domains such as images or texts, usually addressed by neural network-based techniques.

A *low-level sensory anomaly* will be one located in the features associated with the lowest levels of these pattern hierarchies, such as pixels, edges or textures in the case of images, or characters and words in the case of text. An example of this kind of anomaly is the detection of product defects using images [198].

A *high-level sensory anomaly* will be one located in the features associated with the higher levels of these pattern hierarchies, such as the semantic meaning of an

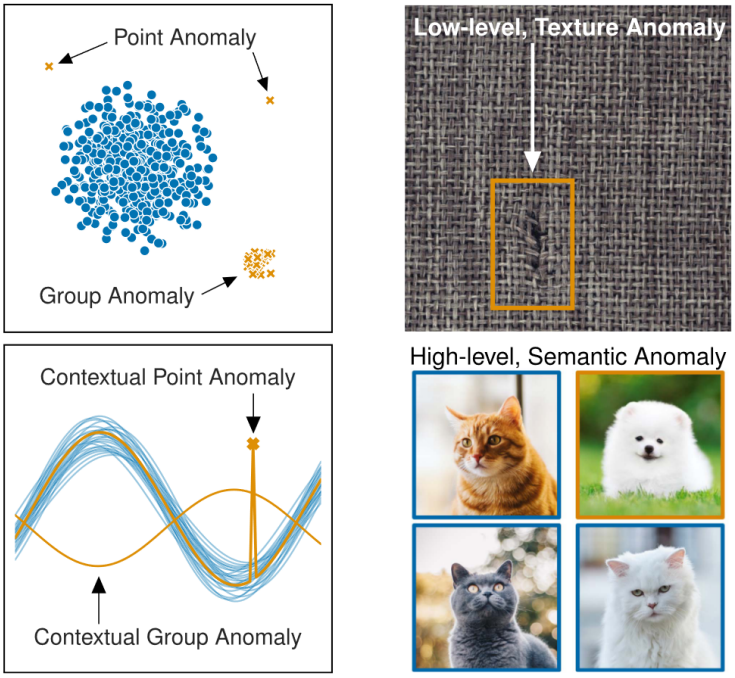


Figure 2.1: Illustration of the different types of anomalies (image taken from *Unifying review of deep and shallow anomaly detection* [174]).

image or text. Given a model trained to determine whether an image corresponds to a cat or not, such an anomaly could be an image of a dog that, although at a low level (pixel level) may share many characteristics with certain cat breeds, at a high level is not the expected type of animal.

It is important to recognize that anomalies can present in a variety of forms (see Figure 2.1). Recognizing and analyzing these different classes of anomalies enhances our ability to develop more effective and adapted approaches to anomaly detection.

2.1.3. Anomaly, outlier, or novelty

Anomalies, outliers, and novelties are all instances of data found in low probability regions. These terms appear numerous times in the literature interchangeably, although there can be found some key differences between the three [174].

An *anomaly* is an instance that comes from a different distribution than the underlying distribution of the data. For example, if we have a dataset of cats, a dog

would be an anomaly.

An *outlier* is an instance that has a low probability of being present in the underlying distribution of the data. For example, a sphinx cat (furless) could be considered an outlier within the cat dataset.

A *novelty* is an instance that is new and has not been seen before, which may indicate a shift in the distribution considered to be normal, implying an update to the models. For example, a new breed of cat would be a novelty.

The methods for detecting anomalies, outliers, and novelties are mostly similar. However, the objective of detecting each type of instance may be different. For example, the detection of an anomaly is usually associated with the occurrence of a problem, while outliers can be detected to remove noise from the data. In this thesis, we will use the terms interchangeably.

2.1.4. Data format

In machine learning, and therefore in anomaly detection, data can be classified into two main types: structured and unstructured data.

Structured data refers to data that is organized in a predefined format and can be easily represented in tables or matrices. Each row in the table usually represents an individual sample, while each column represents a specific feature or attribute of that sample. Structured data is typically found in relational databases, spreadsheets, or CSV files. Examples of structured data include numerical values, categorical data, timestamps, and more. This type of data is well-suited for traditional machine learning algorithms.

Unstructured data, on the other hand, lacks a predefined structure and does not fit neatly into tables or matrices. It is more complex and can include text, images, video, social media posts, and more. Unstructured data is often represented as a collection of raw text, images, or audio files, without a clear organization. Analyzing and making sense of unstructured data is more challenging because it requires advanced techniques such as Natural Language Processing (NLP) or Computer Vision (CV) methods.

Anomaly detection on unstructured data sometimes involves converting the raw data into a structured format so that traditional machine learning algorithms can be applied. For example, in NLP, text data can be tokenized and transformed into numerical representations using techniques like word embeddings [103] or bag-of-words [235].

In recent years, advancements in deep learning have revolutionized the field of unstructured data analysis. Deep learning models, such as Convolutional Neural Networks (CNN) [193] for images, or Recurrent Neural Networks (RNN) [195] and Transformers [117] for texts, have shown remarkable success in processing and understanding unstructured data.

In summary, structured data is well-organized and fits into tables, making it suitable for traditional machine learning algorithms, while unstructured data is more complex and requires specialized techniques like deep learning to extract meaningful insights and patterns. The field of anomaly detection on unstructured data is much less explored. Both structured and unstructured data are essential for building comprehensive and powerful machine learning applications in various domains.

2.1.5. Data labels

Classifying a data instance as normal or anomalous can become challenging. Labeling can be performed manually by a human expert but requires considerable effort. Identifying a set of anomalous instances that covers all possible anomalous behavior is often an impossible task. Anomalous behavior is often dynamic in nature, which means that new types of anomalies may emerge for which no labeled training data exist. Depending on the availability of labels, anomaly detection techniques can be trained in three ways: unsupervised, supervised, and semi-supervised.

Unsupervised anomaly detection [60, 64] is characterized by the availability of only unlabeled data to train a model. This is the most common approach [139] in anomaly detection due to the inherent difficulties in obtaining anomalous data. In these scenarios, it is common to assume that all training data are normal, although in practice it is usual for the dataset to contain not only normal examples but also noise or contamination from anomalies not detected during data collection.

This presents a challenge for anomaly detection techniques that must deal with the presence of noise or contaminated data in the training dataset. For example, certain models that attempt to determine the boundaries of the normal distribution assume that all training data are normal to perform such a characterization [28]. Therefore, they are more sensitive to the presence of noise or contamination, which may even invalidate their training, than other approaches such as autoencoder networks [12].

Supervised anomaly detection [62, 149] is the second most common case [139], and is characterized by the availability of labeled training data from both the normal and anomalous classes. One of the main challenges faced by this approach is the frequent scarcity of anomalous instances, which generates an unbalance between classes. Furthermore, unlike the normal class, anomalies do not necessarily always follow a well-defined distribution, which makes the training more difficult. In those cases where the anomalies follow a well-defined distribution, the problem would be equivalent to a binary classification problem [102].

Semi-supervised anomaly detection [213] is the most infrequent training type among the three, and is characterized by the use of both labeled and unlabeled data. Typically, due to the high cost associated with manual labeling of the data, in these scenarios, most of the training data is unlabeled, with a small fraction of the data labeled. Despite this, it has been found that the incorporation of these labeled data, especially if they are anomalies, can significantly improve the performance of the models [177]. In some cases, the labeled data corresponding to anomalies can even be extracted from auxiliary datasets close to the target domain, such as other image or text datasets, thus favoring the generalization of the models. This technique is known as outlier exposure [72, 176, 120].

2.1.6. Output of anomaly detection

An important factor to consider in any anomaly detection technique is the reporting mechanism for anomalies. Typically, anomaly detection techniques produce outputs of two types [29]: scores and labels.

Scores. Scoring techniques assign an anomaly score to each instance in the test

data based on its deviation from normal behavior. These techniques generate a ranked list of anomalies, allowing analysts to focus on the top-ranked anomalies or set a cutoff threshold for the score to select anomalies for further analysis.

Labels. Techniques in this category assign a label (normal or anomalous) to each test instance. While scoring-based techniques offer flexibility in selecting anomalies using domain-specific thresholds, techniques providing binary labels do not offer direct control over anomaly selection. However, analysts can indirectly influence the selection through hyperparameter choices involved in training.

Choosing the appropriate reporting mechanism depends on the specific requirements of the application and the analyst's preference in dealing with anomalies. Both scoring and labeling approaches offer valuable insights for anomaly detection, enabling effective analysis and decision-making based on the identified anomalies.

2.1.7. Measuring performance in anomaly detection

Anomaly detection models can make two types of errors that have distinct implications: false alarms (type I errors or false positives) and missed anomalies (type II errors or false negatives).

A *false alarm* or *false positive (FP)* occurs when a model incorrectly flags a truly normal data instance as anomalous. This can lead to unnecessary disruptions, investigations, or interventions when there is no actual anomaly present. On the other hand, a *missed anomaly* or *false negative (FN)* happens when a model fails to identify a true anomaly, allowing it to go undetected. This can have serious consequences, especially in critical applications such as medical diagnostics or fraud detection, where the missed anomaly could represent a significant risk or threat. Similarly, when a model successfully detects an anomalous instance as such, we will be dealing with what is known in anomaly detection as a *true positive (TP)*, while when the model detects a normal instance as such, we will be dealing with a *true negative (TN)*. Table 2.1 shows the four possible cases.

Finding the right balance between false alarms and missed anomalies is crucial, however, the costs associated with each error type can vary depending on the specific application domain and its requirements. Determining the optimal trade-off

between false alarms and missed anomalies requires a careful analysis of the specific application context and the potential consequences of each error type. This decision should align with the priorities and objectives of the application, striking a balance that minimizes the overall impact of errors while maximizing the effectiveness of anomaly detection.

When we have to evaluate the performance of an anomaly detection model we can make use of different evaluation metrics that are also common in other classification problems. Two of the most important metrics are the F1-score and the Area Under the Curve.

The *F1-score* [49] is based on the confusion matrix from the labels of the classified test data (see Table 2.1). It combines precision and recall into a single measure by calculating their harmonic mean, penalizing cases where precision and recall differ significantly, thus providing a more comprehensive view of the performance of the anomaly detection model. Precision refers to the proportion of instances classified as anomalies that are truly anomalies, indicating the model's ability to avoid false positives. On the other hand, recall (also known as sensitivity or true positive rate) refers to the proportion of anomalies detected out of the total anomalies present in the data, measuring the model's ability to find all true anomalies and avoid false negatives. The F1-score is therefore calculated as:

$$F_1 = 2\frac{precision \cdot recall}{precision + recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{2.1}$$

A high F1-score indicates a balance between high precision and high recall. In anomaly detection problems, where the presence of anomalies is generally much lower compared to normal data, the F1-score is a particularly useful metric. By considering both false positives and false negatives, the F1-score provides a more balanced and reliable assessment of a model's performance in anomaly detection.

Table 2.1: Confusion matrix for anomaly detection

	Normal data	Anomalous data
Classified as normal	True negative, T^-	False negative, F^-
Classified as anomalous	False positive, F^+	True positive, T^+

The *Area Under the Receiver Operating Characteristic curve (AUROC)*, commonly known as the *Area Under the Curve (AUC)* [21], is a widely used evaluation measure in anomaly detection. It considers the full range of decision thresholds on a given test set, for example, if we are dealing with a model that outputs a score instead of a label, providing a comprehensive assessment of its performance. The AUC is computed by plotting the Receiver Operating Characteristic (ROC) curve, which shows the relationship between the false alarm rate and recall at various thresholds. The AUC value ranges from 0.5 (random guessing) to 1 (perfect classification). The AUC is favored due to its interpretability and comparability across different application scenarios.

In cases of highly imbalanced test sets, the AUC can yield optimistic scores. In such scenarios, the Area Under the Precision-Recall Curve (AUPRC) is recommended. The Precision-Recall (PR) curve plots precision against recall, and the AUPRC provides a more informative measure when precision is crucial. However, one downside of the AUPRC is its interpretation variability due to the random guessing baseline, which depends on the fraction of anomalies in the test set. When the test set is not highly imbalanced, the AUROC and AUPRC exhibit similar trends.

2.2. Taxonomy of anomaly detection techniques

Based on the mechanisms they employ and the assumptions they make, we can distinguish four main types of anomaly detection methods: probabilistic, distance-based, reconstruction-based, and domain-based methods [161, 29, 139, 182].

2.2.1. Probabilistic methods

The main objective of probabilistic methods is the estimation of the density function that generates the normal data. These methods assume that the training data, usually belonging to the normal class, are generated by a probability distribution, which can be estimated using those same data. The normal data will be in regions with high model probabilities and the anomalies in the lower probability regions. Thus, the classification of new data by these methods is based on determining the

probability that such data may have been generated by the estimated distribution, being classified as normal if this probability exceeds an established threshold or as an anomaly otherwise. Parametric and non-parametric techniques are usually used to model these density functions.

Parametric techniques are based on the assumption that normal data follow a specific probability distribution. These parametric methods assume an underlying probabilistic model and use estimated parameters to characterize the distribution of normal data. Depending on the type of distribution assumed, we can speak of normal, multivariate, or mixture distribution models.

In normal distribution models [230], data belonging to the normal class are assumed to follow a normal (Gaussian) distribution. The model parameters, such as the mean and standard deviation, are estimated from the normal data. The normal probability density function is then used to calculate the probability of an instance and classify it as normal or anomalous based on a defined threshold. On the other hand, multivariate distribution models [125, 157] are used when normal data has multiple features or dimensions. It is assumed that the data follows a multivariate distribution, such as a multivariate normal distribution. The model parameters, like as the mean vector and covariance matrix, are estimated from the normal data. The multivariate probability density function is used to calculate the probability of an instance and classify it as normal or anomalous. Finally, mixture models [111, 106] assume that normal data is a combination of several probability distributions. A mixture model is estimated that includes multiple components, where each component represents a different distribution. The model parameters, such as the weights of each component and the parameters of the individual distributions, are estimated from the normal data. The probability of belonging to each component is used to classify instances as normal or anomalous.

Non-parametric techniques do not make strong assumptions about the underlying probability distribution of the data. Instead, they focus on estimating the data density directly from the available observations. These non-parametric methods offer flexibility and can capture complex patterns in the data. Some common non-parametric probabilistic approaches in anomaly detection are based on histograms and kernels.

The simplest technique is to use histograms [98, 224, 59] to generate a profile of normal data. These are also known as frequency-based or count-based techniques. In this approach, the feature space is divided into bins, and the number of data points falling into each bin is counted. The height of each bin represents the estimated density in that region of the feature space. To detect anomalies, we seek to identify bins with a low frequency of data points, indicating unrepresentative or unusual regions. On the other hand, kernel-based techniques [105, 202, 219] estimate the data density by placing a kernel function on each data point and summing up their contributions. The density at a particular point is determined by the number of neighboring points within a specified distance. Anomalies are identified as instances with low density values, indicating that they deviate significantly from the majority of data points.

Probabilistic methods offer a strong mathematical foundation and can effectively detect novel data by accurately estimating the probability distribution function. Additionally, once the model is constructed, it requires minimal information to represent it, reducing the storage requirements compared to the entire training dataset. However, the performance of these methods can be limited when the training dataset is small, particularly in moderately high-dimensional spaces. In many real-life scenarios, there is no prior knowledge of the data distributions, making parametric approaches challenging if the data does not adhere to the assumed distribution. Consequently, non-parametric techniques are appealing as they make fewer assumptions about the distribution characteristics. For instance, kernel functions exhibit reasonable scalability for multivariate data and are computationally efficient.

2.2.2. Distance-based methods

Distance-based methods, including nearest-neighbors and clustering methods, are another type of technique that can be used to perform anomaly detection. These methods focus on measuring the similarity or distance between data instances to identify anomalies. They rely on the assumption that anomalies are often far from their neighbors or exhibit different patterns compared to normal instances. The key idea is to examine the local relationships and structure of the data to detect unusual data points.

Nearest-neighbour methods assume that normal data occur in the form of dense neighborhoods, while anomalies are found far from such neighborhoods. The approach known as k-nearest neighbor (k-NN) [199, 113, 204] represents the most widely used group of techniques. It bases its operation on the fact that the score of a data is given by the distance to its k nearest neighbors, so that new data can be classified based on its score and a threshold, or the type of its k neighbors, for example, by a majority voting scheme. Data points in regions with low density are more likely to be classified as anomalies. Local Outlier Factor (LOF) [23] is a popular extension of k-NN, designed to capture the local density deviation of data points instead of distances. In both approaches, the techniques estimate the density of neighborhoods in such a way that a datum in a neighborhood with low density will be anomalous, while a datum in a high-density neighborhood will be considered normal.

Clustering methods aim to group data points into clusters based on their similarity or distance in the feature space. Anomalies are then detected as data points that do not belong to any cluster or fall into clusters with low membership. Based on the assumptions they make about the data, we can distinguish three categories. In the first category, it is assumed that normal instances belong to a cluster in the data, while anomalies do not belong to any cluster. Based on this assumption, they apply traditional clustering methods such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [92] to the dataset and declare any instance that does not belong to any cluster as anomalous. In the second category, the techniques assume that normal data are located near the centroid of their nearest clusters, while anomalies will be located far away. After calculating the clusters, the score of the data will therefore be its distance to the centroid of the nearest cluster. Establishing a distance threshold will allow the classification of new data. The initial clustering process can be performed using algorithms such as k-means [135]. Finally, in the third category are techniques that assume that normal data belong to large and dense clusters, while anomalies belong to small or sparse clusters. In this case, the classification will be based on a threshold of size or density rather than distances. One of the most commonly used techniques is Isolation Forest (IF) [119].

Distance-based techniques in anomaly detection offer several advantages, including their ability to operate in unsupervised mode and their adaptability to complex

datasets through modification of the clustering algorithm. These methods provide fast predictions on simple datasets since they only compare new data to a small constant set of clusters. However, they have significant drawbacks, as their performance heavily relies on the effectiveness of the clustering algorithm employed. Certain clustering methods may assign all data to a cluster, leading to anomalies being grouped, conflicting with the assumption that anomalies do not belong to any cluster. Consequently, some techniques are only effective when anomalies do not form significant clusters among themselves. Some of these techniques such as k-NN may suffer from the curse of dimensionality [100], which reduces their performance.

2.2.3. Reconstruction-based methods

Reconstruction-based methods are often used in regression and classification problems. In anomaly detection scenarios, these methods can autonomously model the training dataset, and classify new data by calculating its reconstruction error. This reconstruction error is defined as the distance between the value provided by the model and the target value, thus serving as a score. Under the assumption that normal data will have low reconstruction errors, reconstruction-based methods can be further classified into two groups, those based on neural networks and those based on subspaces.

Neural networks approaches have been proposed for anomaly detection, the most used being Autoencoder networks (AE) [10]. Autoencoders are a type of self-associative neural network whose output layer seeks to reproduce the data presented to the input layer after having gone through a dimensional compression phase. In this way, they manage to obtain a representation of the input data in a space with a dimension smaller than the original, learning a compact representation of the data, retaining the important information, and compressing the redundant one. For this reason, both autoencoder networks and their extensions are widely used for the elaboration of noise-robust models, an important quality in anomaly detection and regression problems [8, 242, 32, 237, 239].

Subspace-based techniques use a combination of attributes to describe the variability in the training data. These methods project or embed the data into a lower dimensional subspace, where normal and abnormal data can be distinguished. Prin-

cipal Components Analysis (PCA) [24] is a technique used to transform the data into a lower-dimensional subspace, reducing the number of features needed for data representation. This approach constructs a model of the training data distribution in the transformed space using the principal components that account for most of the data variance. The last principal component helps identify points deviating significantly from the data's correlation structure, with normal points having low values and anomalies having large values for such projections. This enables effective anomaly detection by finding features not apparent in the original variables.

Reconstruction-based techniques offer several advantages, including their ability to train in both supervised and unsupervised modes and their independence from prior data knowledge. However, neural network-based methods can be time-consuming during the learning phase, and subspace-based techniques may suffer from computational complexity, limiting their applicability in cases where normal and abnormal datasets are not easily separable in the projected space.

2.2.4. Domain-based methods

Domain-based methods, also known as One-Class Classification (OCC) methods, require the creation of a boundary based on the structure of the training dataset. Class membership of the unknown data is then determined by its location concerning the final boundary. The most common methods are One-Class Support Vector Machine and Support Vector Data Description.

Support Vector Machine (SVM) [144] is a domain-based model widely used in binary or multiclass classification problems. Different extensions are available for their use in anomaly detection scenarios such as the *One-Class Support Vector Machine (OC-SVM)* [31, 109, 217] algorithm. In these approaches, the aim is to determine the hyperplanes that best represent the target class. The data that are closest to these hyperplanes and that allow their formation are known as the support vectors. The rest of the data in the training dataset that do not form support vectors are not taken into account. In this way, the distribution of the data is ignored, which simplifies the problem.

Another important method is *Support Vector Data Description (SVDD)* [200,

156]. SVDD seeks to construct a sphere (or hypersphere in higher dimensional spaces) that encompasses most of the normal data. The center of the sphere and its radius define a region where the normal data is expected to be found. Points that fall outside this sphere are considered anomalies. This algorithm has been extended leading to more advanced methods such as Deep-SVDD [175].

These methods are usually insensitive to the specific sampling and density of the normal class, because they describe its boundary or domain, and not the density of the data. They are particularly useful when anomalous data are found in infrequent and sparse regions within the feature space. However, they can have difficulty detecting anomalies that lie near the boundaries separating the two classes. Due to this high sensitivity to space partitioning, it is necessary to compensate for the existence of data that may be misclassified in the class splitting regions. If a noisy or anomalous point is used in training the system may be overfitted, which will worsen its future performance. Therefore, these boundaries between classes are often softened. A common way to do this is to set a percentage of normal data that may fall outside the normal boundary in exchange for less forced class separation.

2.3. Applications

Anomaly detection has many practical real-life applications in different fields. Although it is very challenging to group these domains into a limited number of categories, in this section we have attempted to present the main application areas of anomaly detection through five main categories [161, 29, 139, 182].

2.3.1. Internet security systems

Anomaly detection has become a crucial area of research for internet security systems, covering a wide range of applications, including network intrusion detection and fraud detection. Network intrusion detection [96, 115, 3, 16, 51, 187, 107] aims to identify and thwart malicious activities targeting computers and other electronic devices. Since frequent attacks on computer systems represent serious threats, early intrusion detection is vital to prevent system shutdown or total system collapse.

Intrusion detection not only helps to discover and neutralize malicious programs in the operating system but also plays a key role in identifying unauthorized access to computer networks. Intrusions often exhibit different behaviors than normal users, which makes anomaly detection a powerful tool for recognizing potential threats.

On the other hand, fraud detection [162, 101, 76, 160, 99, 205] plays a critical role in protecting various domains, such as insurance claims, credit card purchases, cell phone usage, and financial transactions. For example, the purchasing behavior of credit card thieves differs significantly from that of legitimate cardholders. By identifying changes in credit card usage patterns, anomaly detection models can proactively prevent subsequent periods of fraudulent activity, protecting individuals and organizations from financial loss and potential security breaches.

Deploying robust and accurate anomaly detection models not only strengthens the integrity and security of electronic systems but also instills confidence in users and organizations while thwarting potential cyber threats and fraudulent activities.

2.3.2. Healthcare

Healthcare is a crucial domain where anomaly detection approaches find significant applications [52, 220, 82, 152, 56, 70]. These methods allow the analysis of patient records that exhibit unusual symptoms or test results, which may indicate potential health issues requiring attention. One of the primary objectives in healthcare anomaly detection is to distinguish between instrumentation or recording errors and clinically relevant changes in a patient's condition. By achieving this discrimination, timely medical intervention can be initiated in cases where it is essential.

Patient records in the medical context encompass a wide array of features, such as age, weight, vital signs (e.g., heart rate), physiological signals (e.g., electrocardiogram readings), blood test results, and medical image data. The fusion of these diverse data types enables a comprehensive assessment of a patient's health status and empowers healthcare professionals to detect anomalies and early signs of potential health issues accurately. For instance, anomalies in a patient's vital signs might indicate a deteriorating condition, prompting immediate medical attention.

Moreover, medical imaging [208, 234] plays a key role in detecting anomalies,

as it allows the visualization of internal structures and functions, aiding in the identification of abnormal patterns or growths. Medical imaging anomalies, such as X-rays, computed tomography scans, or magnetic resonance imaging, can signify the presence of tumors or other medical conditions that require further investigation and treatment.

Overall, applying anomaly detection in healthcare leads to enhanced patient care, improved diagnostic accuracy, and the timely detection of critical health conditions, thereby contributing to better medical outcomes and ultimately saving lives.

2.3.3. Industrial systems

In the industrial context, anomaly detection plays a crucial role in monitoring the health and performance of industrial assets [194, 126, 89, 104, 91, 27]. Industrial machinery and equipment are subject to wear and tear over time due to continuous usage and environmental factors. Detecting anomalies in the operation of these assets is vital to identify early signs of deterioration and potential faults. Early detection allows for timely maintenance and repairs, preventing further escalation of issues and minimizing downtime, which ultimately results in cost savings.

To achieve effective anomaly detection in industrial settings, high-value machinery is often equipped with specialized sensors dedicated to monitoring various aspects of their operation. These sensors collect diverse data, such as temperature, pressure, and vibration amplitude. By continuously monitoring this data, deviations from the expected patterns can be detected, signaling potential anomalies or abnormal behaviors in the machinery's operation.

The significance of anomaly detection in the industrial domain extends beyond merely identifying faults. It also enables predictive maintenance strategies [245, 166, 130], where data-driven insights can be used to forecast equipment failures and determine the optimal time for maintenance interventions. By adopting proactive maintenance approaches, companies can reduce the likelihood of unexpected breakdowns, improve machine performance, and enhance overall operational efficiency.

Moreover, anomaly detection in industrial processes is also instrumental in en-

suring the safety of personnel and equipment. Identifying anomalies in real-time can help prevent hazardous situations, as certain deviations may indicate imminent malfunctions or safety risks. By promptly addressing these anomalies, companies can maintain a safe working environment and protect their assets from potential damages.

In conclusion, anomaly detection in the industrial domain empowers businesses with valuable insights into the health, performance, and safety of their machinery and equipment. By harnessing the data collected from specialized sensors, companies can detect early signs of deterioration, optimize machine performance, reduce maintenance costs, and enhance operational efficiency. Embracing anomaly detection technologies as part of industrial asset management strategies is a proactive approach that enables companies to stay ahead in a competitive landscape while ensuring the reliability and safety of their operations.

2.3.4. IoT devices

With the rapid proliferation of Internet of Things (IoT) devices in various domains, the importance of anomaly detection in preserving data privacy and ensuring network security has become increasingly evident [66, 36, 30, 5, 50, 61, 122]. IoT devices are interconnected and gather vast amounts of sensitive data, making them susceptible to data breaches, frauds, and malicious attacks. However, unlike traditional computing environments, these devices often have limited computational resources, requiring the development of lightweight anomaly detection models that can efficiently operate in such constrained settings.

Privacy preservation is also a critical concern in IoT environments, where data collected by sensor nodes may include personal information, location data, and sensitive industrial or healthcare records. Effective anomaly detection methods can help identify and prevent data leaks, unauthorized access, and potential privacy violations, safeguarding the confidentiality of sensitive information and complying with privacy regulations.

Furthermore, IoT networks are exposed to various security threats, including unauthorized intrusion, network attacks, and data manipulation. The deployment

of anomaly detection models in these scenarios can proactively detect abnormal activities and patterns, enabling swift responses to potential threats and fortifying the overall network security.

In conclusion, the integration of anomaly detection in IoT sensor networks is essential for maintaining data integrity, enhancing network reliability, and fortifying security measures. As IoT technologies continue to advance and find applications in various industries, the role of anomaly detection becomes increasingly critical for ensuring the seamless and secure operation of IoT ecosystems.

2.3.5. Unstructured data

In recent years, anomaly detection has witnessed a remarkable expansion into new fields, encompassing diverse data types such as images, videos, and text, alongside traditional structured data. This surge in diversification can be attributed to the proliferation of advanced technologies, including deep learning models and natural language processing algorithms, which have revolutionized the capabilities of anomaly detection in handling complex data.

Anomaly detection has found extensive applications in recognizing novel objects in images and video streams [40, 131, 35, 68, 196, 180, 140, 243]. This area of research has gained significant attention due to the abundance of video data available and the increasing need for automated methods to extract crucial information from such media. The goal is to automatically identify novel objects or events that deviate from the norm. This could include detecting unusual activities in surveillance footage, recognizing new objects in images, or identifying unexpected events in video streams.

Moreover, in the context of text data, anomaly detection aims to automatically identify novel topics, spam, fake news, new interesting events, or emerging stories within a given collection of documents or news articles [179, 124, 63]. Text data pose unique challenges as they are typically high-dimensional and sparse, represented as bag-of-word features or more sophisticated linguistic representations. Detecting novel patterns and events in such data can provide valuable insights for various applications, including news monitoring, social media analysis, and trend detection.

The application of novelty detection in text, images, and video showcases its

versatility and wide-ranging impact across different domains. By automatically identifying novel and significant patterns, these approaches contribute to better understanding, decision-making, and situational awareness in various real-world scenarios. As the amount of available data continues to grow, the need for efficient and effective novelty detection techniques becomes ever more critical in harnessing the vast potential of these information-rich sources.

2.4. Challenges

Despite being a successful discipline with a wide range of uses today, anomaly detection, like other areas of machine learning and artificial intelligence, is an ever-expanding area that faces numerous challenges. In this section, we present the five main challenges that motivate this doctoral thesis.

2.4.1. Non-convex and disjointed shapes

In anomaly detection, the distribution presented by the target class can be decisive for the application of certain types of techniques. Anomaly detection systems play a vital role in a wide range of real-world applications where the shape of the normal class will not always be convex. A convex representation can never provide good characterization of a non-convex distribution, so it is necessary to develop specific methods capable of operating on non-convex datasets. In addition, the distribution of the data can be composed of several separate regions in space that are difficult to represent using a single hull (see Figure 2.2).

Computing the boundary of the normal class in the original space is a strategy followed by some methods [112, 154]. However, working with high-dimensional data makes it difficult to calculate these structures due to their high computational cost. One solution is to reduce the dimensionality of the dataset to make these operations manageable. The most common methods to simplify the representation of the problem by reducing the dimensionality of data are Principal Component Analysis (PCA) [75], Linear Discriminant Analysis (LDA) [81], or Autoencoder Networks (AE) [11].

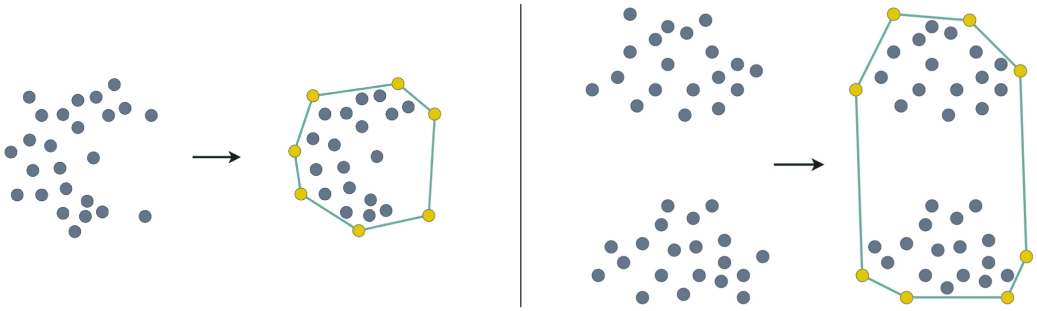


Figure 2.2: When the point cloud that forms the normal class has non-convex regions (left) or regions separated in space (right), a convex hull may not provide sufficient precision for its representation.

Whether in the original space or reduced dimensional spaces, anomaly detection models must be able to represent non-convex and disjoint regions, which requires specific techniques.

2.4.2. Edge computing

Technological development in recent years has led to a substantial increase in the number of computing devices such as smartphones or IoT devices, both for personal and industrial use. Despite their small size, these have enough computational power to perform tasks that a few years ago were considered unaffordable, such as training small machine learning models, real-time inference, or exchanging large amounts of data at high speed.

Due to the abundance of these devices and the inefficiencies of traditional cloud computing to perform processes requiring low latencies, a new computing paradigm called Edge Computing (EC) [26, 95, 209] has emerged. Edge computing moves computing from data centers to the edge of the network, bringing cloud computing services and utilities closer to the end user and their devices. This enables faster information processing and response time, as well as freeing up network bandwidth. It plays an important application role in Content Delivery Network (CDN) [232], industrial IoT (IIoT) [165], smart home [229], smart transportation [116] and other fields.

Edge computing presents challenges such as resource constraints, device and

network heterogeneity, data security and privacy, reliability and fault tolerance, complexity management, and scalability. These challenges require efficient resource utilization, interoperability, robust security measures, fault-tolerant mechanisms, simplified management, and scalable architectures to ensure the successful implementation and widespread adoption of edge computing. Because of this, adapting classical anomaly detection algorithms or designing new ones to be used in these scenarios is a necessary task.

2.4.3. Federated learning

From a machine learning point of view, this current device-plagued technological scenario is well suited for the use of federated learning. Federated Learning (FL) [236, 110, 20] is a collaborative machine learning scheme that allows heterogeneous devices with different private datasets to work together to train a global model. It plays an important role in a wide range of areas such as language model development [65], recommendation systems [227], intelligent medical diagnostic systems [173], wireless communications [143] and other fields.

Challenges in federated learning encompass privacy protection, communication cost, heterogeneity of devices involved, and unreliable model uploads. Privacy concerns arise due to the potential leakage of sensitive information during model communication. Communication efficiency becomes crucial when dealing with large-scale federated networks. The heterogeneity of systems, including hardware and network variations, presents challenges in fault tolerance. Additionally, preventing unreliable model uploads is essential to maintain the integrity of federated learning. Addressing these challenges requires the development of methods that prioritize privacy preservation, optimize communication efficiency, adapt to system heterogeneity, and mitigate the impact of unreliable model uploads. Adapting classical anomaly detection algorithms or designing new ones to be used in these scenarios is also a necessary task.

2.4.4. Natural language processing

Although there are models capable of dealing directly with images or text, classical machine learning methods are usually designed to work with structured data. In the area of natural language processing (NLP), there are multiple techniques for representing text using vectors of real numbers, known as embeddings. These techniques make it possible to generate vector spaces that represent the semantic relations and similarities of the language, so that, for example, two synonymous words will be found at a shorter distance in space than two unrelated words.

Technological development in recent years has enabled the construction of very powerful models for the NLP field [34]. However, unlike other tasks such as sentiment analysis [197] or question answering [97], the application of anomaly detection in texts is not common, probably due to the difficulty in defining the concept of anomaly in this type of scenario. Bringing anomaly detection to the NLP area is an exercise that can be beneficial for many applications such as log analysis in cybersecurity [225], online social network analysis [184], and customer feedback analysis [178].

2.4.5. Explainability

As black-box machine learning models are increasingly used to make important predictions in critical contexts, the demand for transparency is growing [71]. The danger lies in using decisions that are not justifiable, or legitimate or that do not allow for detailed explanations of their behavior. Explanations that support the output of a model are crucial, especially in scenarios that can directly affect us as humans, such as medicine, autonomous driving, or the safety of industrial systems.

When machine learning models do not meet the criteria imposed to be considered explainable, a model-independent method must be applied to explain their decisions. This is the purpose of post-hoc explainability techniques [14], whose objective is to generate understandable information on how an already developed model produces its predictions for any given input. Within these techniques, the so-called model-agnostic techniques are those designed to be used on any model to extract information from its internal reasoning, such as LIME [172] or SHAP [121].

The growing demand for transparency, interpretability and confidence in the systems also includes anomaly detection models, so it is necessary to incorporate this feature in the development of new algorithms. In addition, it is still necessary to study about how to evaluate the quality of these explainability techniques in certain areas.

Chapter 3

A proposal for anomaly detection in traditional scenarios

In anomaly detection, the distribution presented by the datasets, understood as how data is spread in the n-dimensional space, can be decisive for the application of certain types of techniques. Anomaly detection systems play a vital role in a wide range of real-world applications where the shape of the normal class will not always be convex. A convex representation can never provide good characterization of a non-convex distribution, so it is necessary to develop specific methods capable of operating on non-convex datasets. In addition, the distribution of the data can be composed of several separate regions in space that are difficult to represent using a single hull.

This chapter presents OCENCH (One-class Classification via Expanded Non-Convex Hulls) [146], a one-class classification and anomaly detection method based on the use of random projections [211] of the original data space to reduce their complexity, followed by a process based on Delaunay triangulation [47] to geometrically represent the normal class in these low-dimensional spaces through subdivisible and expandable non-convex hulls. The limits of the normal class are iteratively adapted during the training phase. This process is carried out based on a normalized hyper-parameter that controls the adjustment level and can be easily tuned by the user for each scenario. Furthermore, if in a low-dimensional space the normal class cannot be accurately represented by a single non-convex hull, it will be subdivided as many

times as necessary to fit the shape of the data. The developed OCENCH algorithm allows working with non-convex datasets in a novel way, offering a robust behavior and remarkable performance, positioning itself as an alternative for both convex and non-convex problems.

3.1. Related work

Domain-based anomaly detection methods, also known as One-Class Classification (OCC) methods, require the creation of a boundary based on the structure of the training dataset. Class membership of the unknown data is then determined by its location with respect to the final boundary. These methods are usually insensitive to the specific sampling and density of the normal class, because they describe its boundary, or domain, and not the density of the data. They are particularly useful when anomalous data are found in infrequent and sparse regions within the feature space. However, they can have difficulty detecting anomalies that lie near the boundaries separating the two classes. Due to this high sensitivity to space partitioning, it is necessary to compensate for the existence of data that may be misclassified in the class splitting regions. If a noisy or anomalous point is used in training the system may be overfitted, which will worsen its future performance. Therefore, these boundaries between classes are often softened.

This work is focused on boundary-based anomaly detection methods. An important group of boundary-based techniques that have successfully solved OCC problems are those based on convex hulls [28, 87]. A convex hull (CH) is the smallest convex polyhedron that contains a set of data points. The usefulness of this geometric shape in One-Class Classification and Anomaly Detection is to serve as a limit that contains the normal dataset and allows the classification of new data based on whether they are inside the convex hull (normal data) or outside (anomaly). However, these systems play a vital role in a wide range of real-world applications where the shape of the normal class will not always be convex. As a convex representation can never provide good characterization of a non-convex distribution, a whole line of research has emerged to work with non-convex shapes using non-convex hulls (NCH). The objective of these methods is the same, to build a non-convex hull around the normal data as a decision boundary. We can distinguish between two

types of approximations depending on whether they try to calculate the non-convex hull in the original space or a dimensionally reduced space.

Computing the boundary in the original space. Calculating the limits of the normal class in the original data space despite the high cost, is a strategy followed by some methods. A common way to solve the problem is by calculating the convex hull and digging to obtain the non-convex version. For example, DINA [112] algorithm produces a non-convex hull by removing the hollow spaces from the n-dimensional convex hull. To do this, it uses the normal vectors of the hyperplanes that form the facets of the structure, giving rise to what is called oriented non-convex hulls. The normal vector of a facet in a non-convex hull is the normal vector of the corresponding hyperplane, where this facet is located, in the direction from this hyperplane towards the space outside the non-convex hull. Given the facets of an oriented non-convex hull, the algorithm can determine if a new point is inside or outside this non-convex hull. Another approach is the one developed by Park *et al.* [154], a method that follows the same philosophy but carries out the digging stage using the closest data from inside the hull as support. An edge will be pruned if its size in relation to the distance to its support point exceeds a certain threshold defined by the user. This threshold influences the smoothness of the non-convex hulls. Although they can be appropriate for low dimensional problems, the calculation of non-convex hulls in high dimensional spaces is computationally very expensive and challenging, so there are not too many methods of this type.

Computing the boundary in a reduced space. Working with high-dimensional datasets makes it difficult to calculate these structures due to their high computational cost. One solution is to reduce the dimensionality of the dataset to make these operations manageable. The most common methods to simplify the representation of the problem by reducing the dimensionality of data are Principal Component Analysis [75], Linear Discriminant Analysis [81] or Autoencoder Networks [11]. In the best of cases, a two-dimensional representation will be obtained in which the calculation of both convex and non-convex hulls is a simpler operation. However, this remarkable reduction is not always possible, in most cases it is necessary to use more than two dimensions in order not to lose information. To achieve a good two-dimensional representation, some methods opt for the use of random projections. This random projection technique is based on the idea that high-dimensional

data spaces can be projected into a lower dimensional space without significantly losing the data structure if multiple projections are used [86]. An example of this is NCBoP [87] which, based on the Approximate Polytope Ensemble (APE) algorithm [28], takes advantage of these reduced spaces to build non-convex hulls simply.

3.2. Background

This section summarizes the main ideas of the two methods taken as the basis for the development of OCENCH: the random projections used in the APE algorithm developed by Casale *et al.* to reduce the complexity of the datasets [28], and the process for the generation of simple polygons developed by Duckham *et al.* based on Delaunay triangulation [47].

3.2.1. Approximate Polytope Ensemble

A convex hull (CH) is the smallest convex polyhedron that contains a set of data points. Figure 3.1 shows the CH of a two-dimensional dataset, in this case, a polygon. The usefulness of this geometric structure in anomaly detection problems is to represent the normal class, so that classifying new data consists of checking whether it is inside the CH (normal data) or outside (anomaly).

To reduce the computational load of calculating the CH in high-dimensional spaces, APE [28] chooses to project the data into two-dimensional spaces where it

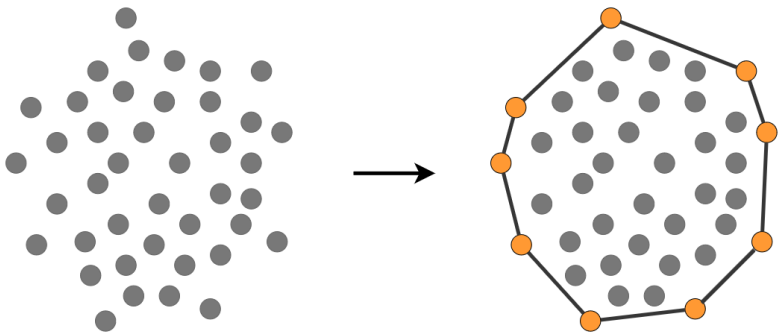


Figure 3.1: Construction of a convex hull from a cloud of points.

will calculate the limits in a more computationally affordable way.

In the training phase, the APE algorithm projects the normal dataset using random projections and calculates the convex hulls in each of these two-dimensional spaces. To classify a new data point, first, it is projected in all two-dimensional spaces. Afterwards, for each of the two-dimensional spaces, given the set of vertices of its CH, it is possible to check if the point is within the CH (normal class) or not. Finally, a point will be classified as normal only if it is within the CH for every projection. If in any of the projections the point is outside, it will be considered anomalous. Figure 3.2 shows this idea.

Building a CH in two-dimensional spaces and checking if a point falls inside are common tasks in geometric computing for which there are very efficient solutions [13]. However, the convex nature of this kind of method makes it unsuitable for non-convex datasets. Furthermore, the use of a single CH per projection space makes it difficult to represent disjointed regions.

3.2.2. Generation of simple polygons to characterize the shape of a set of points

The procedure developed by Duckham *et al.* [47] makes it possible to efficiently and flexibly construct the non-convex simple polygon that characterizes the shape of a set of input points in a plane. This technique is based on Delaunay triangulation, which allows to model a dataset using a polygonal surface. An example of Delaunay triangulation for a half-moon dataset can be seen in Figure 3.3. The resulting surface presents the following properties:

- All the points in the dataset are directly or indirectly connected to each other and form as many triangles as possible without their edges crossing.
- Triangles are defined so that the closest points are connected to each other by an edge.
- The triangles formed are as regular as possible, that is, their minor angles are maximized and the length of their edges is minimized.

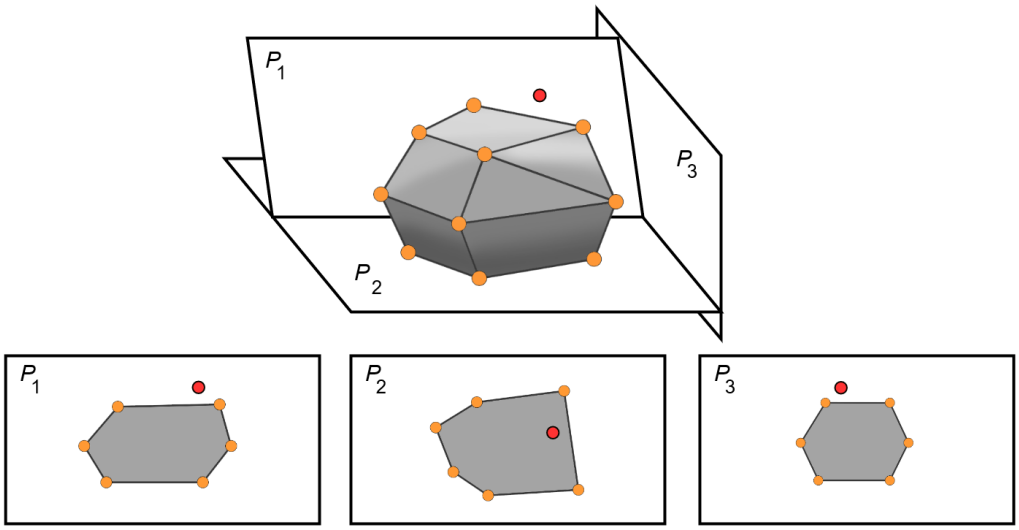


Figure 3.2: Three 2-D projections of a three-dimensional boundary and a new point (red) that will be classified as an anomaly since in at least one of the three projections it lies outside the limits of the normal class (grey).

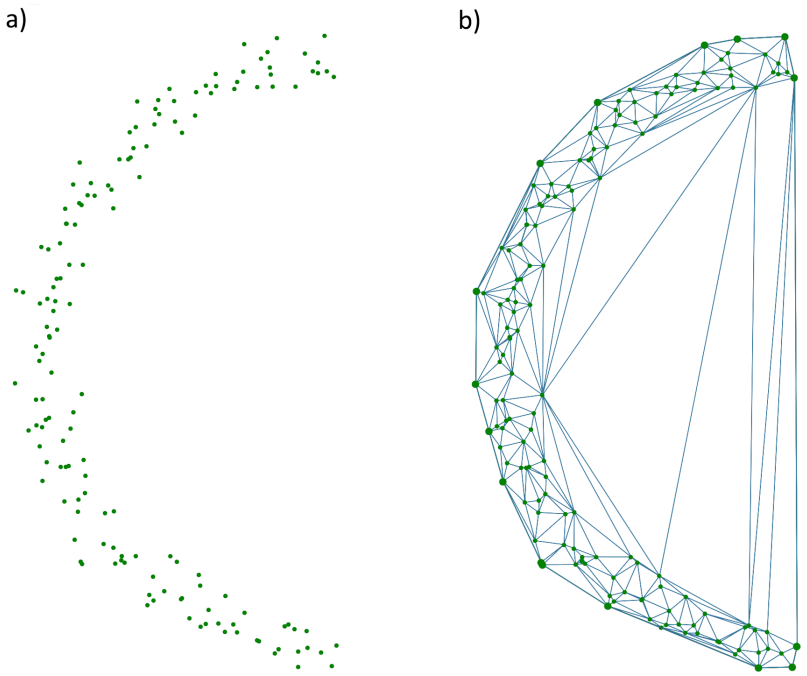


Figure 3.3: Delaunay triangulation (b) of a cloud of points (a).

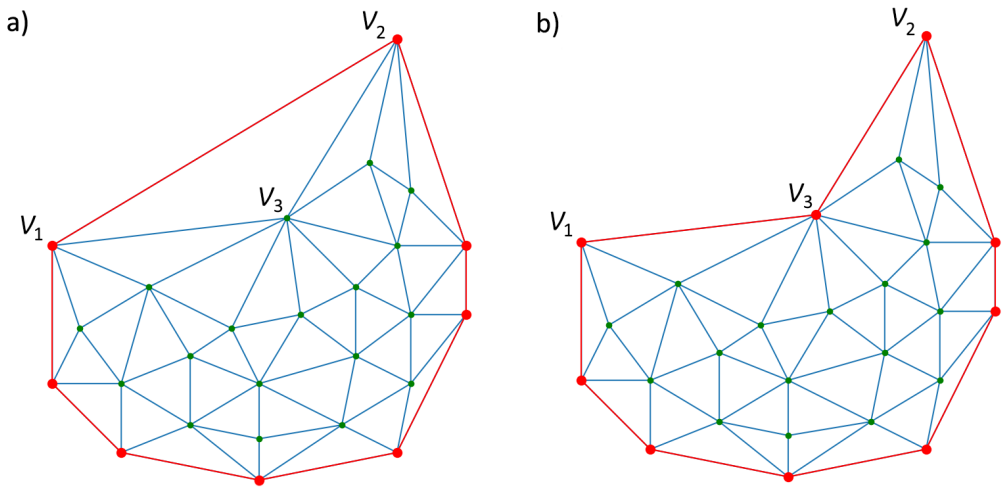


Figure 3.4: Pruning process of an edge: (a) The edge formed by vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ is the longest of the closure (red lines) and exceeds the threshold established by l, therefore it must be pruned; (b) The two edges that were part of the triangle in which the pruned edge was located (edge between $\mathbf{v}_1$ and $\mathbf{v}_3$ and between $\mathbf{v}_2$ and $\mathbf{v}_3$) become part of the new provisional closure.

The properties of triangulation allow the application of a pruning process to adapt the boundaries to non-convex shapes. This pruning process iteratively removes those exterior edges of the closure that exceed a size defined through a hyperparameter l. The exterior edges are sorted from largest to smallest, after which it is checked if they exceed the threshold. If an edge is greater than l, then it is eliminated, and the two edges with which it formed a triangle will become part of the provisional closure. Figure 3.4 shows this procedure. The pruning process will be repeated until no edge of the closure exceeds the size defined by the l hyperparameter. Starting from the triangulation in Figure 3.3, Figure 3.5 shows an example of how the hyperparameter l can influence the adjustment level of a boundary. As can be seen, the dataset is non-convex, so the edges of the interior of the half-moon (the non-convex area) will be the first to be pruned because they are the largest. A too large l value results in insufficiently pruned closures (underfitting), while too small a value can generate very steep boundaries (overfitting).

The method achieves excellent results, however, it cannot be precisely adapted in cases where the data is separated into different areas of the space (see Figure 3.6a). These situations, in which it is necessary to represent more than one isolated region

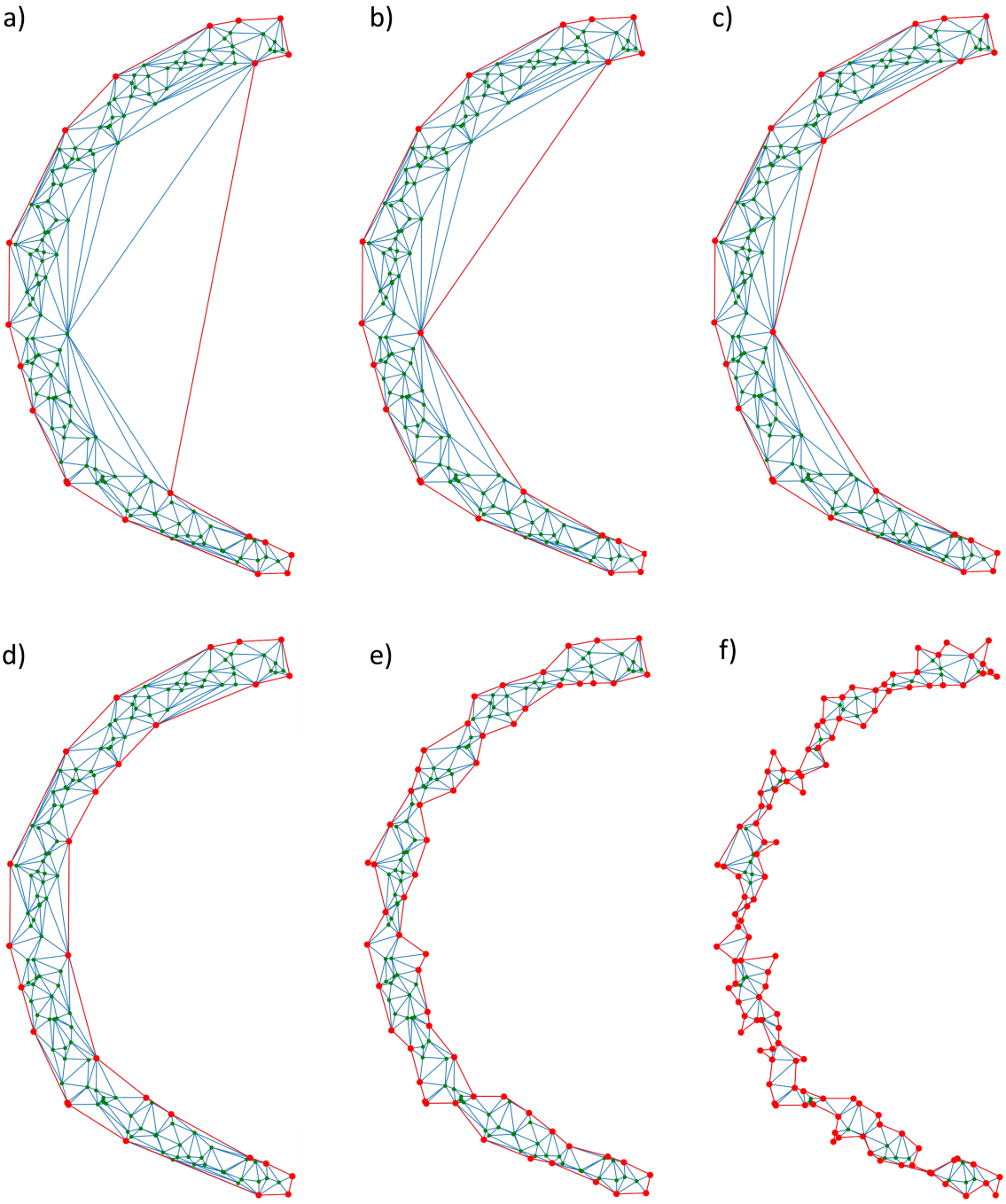


Figure 3.5: Final non-convex hulls (red) result of applying the pruning process with different values of l over a non-convex dataset (green) previously triangulated (blue): (a) $l = 1.8$; (b) $l = 1.6$; (c) $l = 1.2$; (d) $l = 0.6$; (e) $l = 0.2$; (f) $l = 0.1$.

in space, are common in real problems of AD or OCC. In these scenarios, it would be convenient to have more than one non-convex hull to be able to represent each region independently. As seen in Figure 3.6c, the results obtained by the method of Duckham *et al.* are not the most accurate in these circumstances, as a single non-convex hull is used to represent the whole set of points.

For all this, in this work, we also will propose an improvement of the Duckam *et al.* algorithm for its application in disjointed regions which, together with the idea of random projections, has allowed us to design OCENCH, a new one-class classification algorithm that allows working not only with both convex and non-convex datasets, but also with datasets that present connected and unconnected regions.

3.3. Proposed method

The main objective of OCENCH is to fit robustly to the non-convex shape of the data of the normal class during the training phase to achieve a good performance when the system is classifying new data. This section presents the theoretical foundations of the steps followed by the method:

- Dimensionality reduction: reduce the complexity of the normal dataset by projecting the normal class data into two-dimensional spaces.
- Calculation of the non-convex hulls: construction of non-convex hulls in these two-dimensional spaces to represent the limits of the normal class.
- Subdivision of non-convex hulls: subdivision of non-convex hulls to more accurately represent the shape of the data.
- Calculation of the expanded non-convex hulls: expansion of the final non-convex hulls to smooth the classification of new instances.

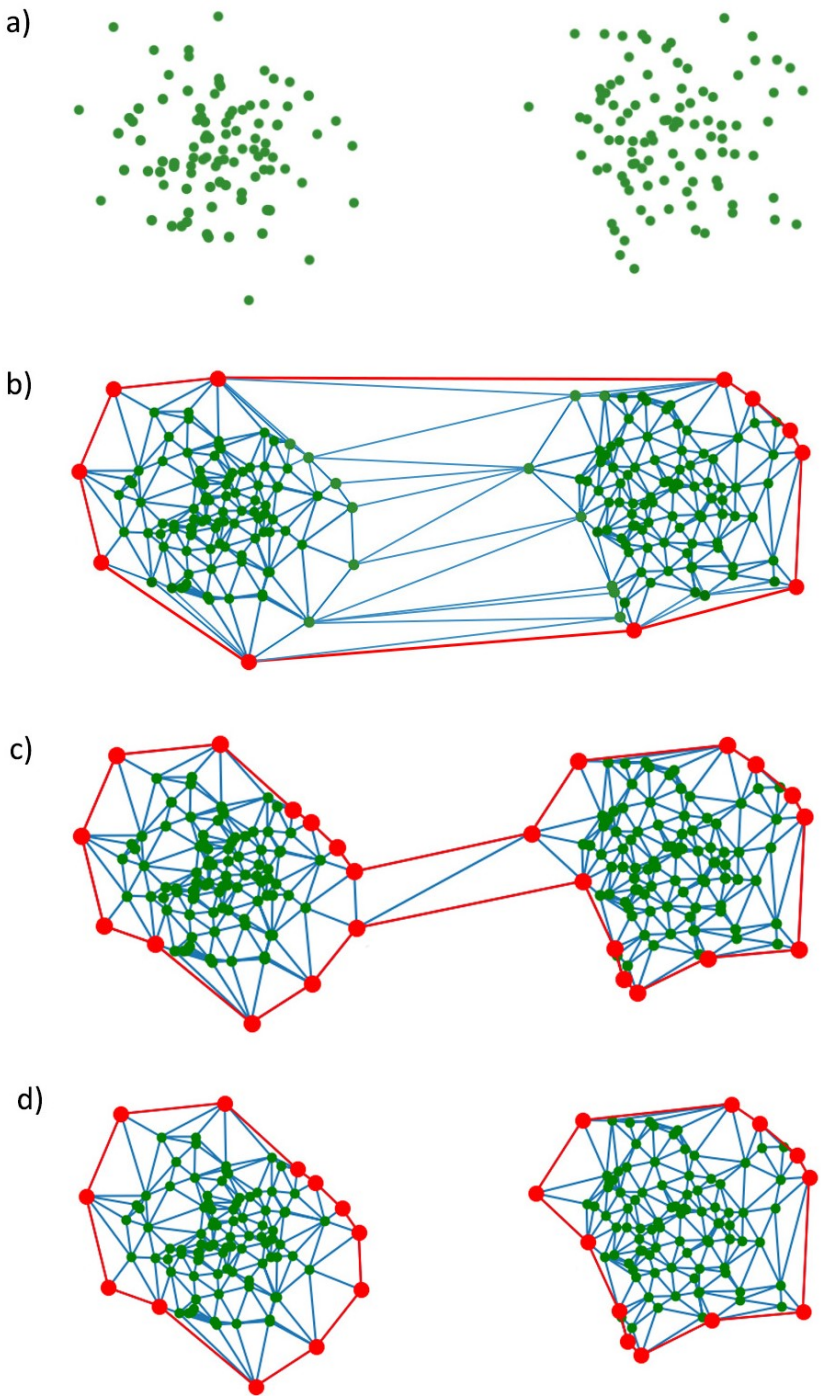


Figure 3.6: Characterization of two separate point regions using OCENCH: (a) Projected dataset; (b) Initial convex hull; (c) Pruned non-convex hull; (d) Subdivided non-convex hulls.

3.3.1. Dimensionality reduction

Given a training dataset $\mathbf{X} \in \mathbb{R}^{d \times n}$ (d variables $\times$ n samples) of normal data, the first step of the algorithm is to reduce its dimensionality carrying out projections into 2-D spaces. To project the data, a certain number of $[2 \times d]$ projection matrices are randomly generated. The number of projections τ is the first hyperparameter that must be defined by the user, taking into account that more projections imply a representation of the class with greater precision in exchange for higher computational cost. To determine the appropriate number of projections for a scenario, it is advisable to experiment with different values to find a balance between precision and time. As a guide, there are experimental studies [39, 38, 17] based on the Johnson–Lindenstrauss lemma that allow estimating the appropriate number of projections based on characteristics such as the size of the dataset. Once the projection matrices have been generated, the training set will be projected using each of these projection matrices. The steps described below are applied independently to the convex hull at each of the projections.

3.3.2. Calculation of the non-convex hulls

Once the training data are projected in two-dimensional spaces (see Figure 3.6a), the convex hull that surrounds the data is calculated for each projection using the algorithm proposed by Duckam *et al.* [47] (see Figure 3.6b). Also following the Duckam *et al.* [47] method, an edge pruning process is carried out to adjust the limits to non-convex shapes (see Figure 3.6c). As commented in section 3.2.2, this pruning process iteratively removes those edges of the closure that exceed a size defined by the user through a hyperparameter l. Since projecting the data causes that the size of the hulls is not similar in all the projections, to facilitate the choice of the hyperparameter l we propose to normalize the data of each projection independently using a standard scaler with zero mean and unit variance. That process allows to choose a value of l that consistently fits the data across all projections.

3.3.3. Subdivision of non-convex hulls

As shown in Figure 3.6c, the problem of the method presented by Duckam *et al.* is its inability to accurately represent separate regions in space. However, the use of a single closure is not enough in general and, therefore, we proposed a subdivision process [145] that results in the coexistence of multiple non-convex hulls.

To design the process of subdivision of closures, the detailed behavior of the Duckham *et al.* algorithm when applied to datasets with separate regions in space was analyzed. As can be seen in Figure 3.7, if there is more than one region, the method produces a single closure that wraps these separate regions in space, keeping them connected by a union region. This union region is characterized by the lack of data points in its interior, being formed by only two triangles whose edges were not pruned in the stage described in section 3.3.2, since their elimination would alter the properties of the triangulation. We can define these unions as a region of the triangulation composed of two triangles that make up a quadrilateral, which has two opposite external edges and no data points inside.

In Figure 3.7, the union region is formed by triangles $\mathbf{v}_1$-$\mathbf{v}_2$-$\mathbf{v}_3$ and $\mathbf{v}_2$-$\mathbf{v}_3$-$\mathbf{v}_4$. An intuitive way to subdivide this non-convex hull in two would be to eliminate the two external edges $\mathbf{v}_1$-$\mathbf{v}_2$ and $\mathbf{v}_3$-$\mathbf{v}_4$ and the common edge of both triangles $\mathbf{v}_2$-$\mathbf{v}_3$.

Since the existence of these union regions is very frequent after the application of the pruning process, it was decided to implement a process of subdivision of closures that tries to locate these structures to treat them in a specific way. To

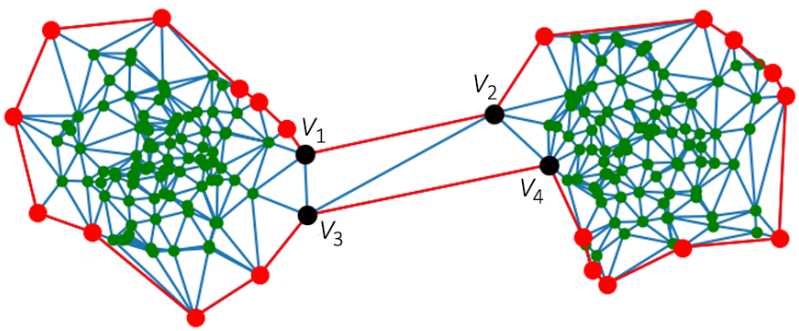


Figure 3.7: Typical morphology of a Duckham *et al.* non-convex hull that envelops two separate regions in space generating a region of union ($\mathbf{v}_1$-$\mathbf{v}_2$-$\mathbf{v}_3$-$\mathbf{v}_4$).

do this, the edges of the closure will be iteratively traversed to verify the existence of this type of union regions. If the existence of a union like the one described in Figure 3.7 is determined, it will be eliminated, giving rise to two independent non-convex hulls. These two new non-convex hulls will be readjusted, if needed, through the pruning process described in section 3.3.2. The subdivision and pruning processes are repeated iteratively until there are no union regions and the non-convex hulls are fully adjusted. In Figure 3.6d it can be seen the result of applying this subdivision process. The identification of several closures will allow a more precise characterization of the regions.

Although this subdivision process is recommended, a *subdivision* hyperparameter was included in the final algorithm to enable or disable this behavior. Given the existence of scenarios in which it is known with certainty that there are no disjointed regions, omitting this subdivision stage could be useful to reduce the algorithm training time.

3.3.4. Calculation of the expanded non-convex hulls

In some cases, there may be normal data that is wrongly classified as anomalies because they are very close to the training data but outside the limits of the non-convex hulls (NCH). To avoid the effect of over-adjustment of the training data, the method uses a third hyperparameter, the λ expansion factor. This hyperparameter allows to generate expandable non-convex hulls (ENCH), whose purpose is to soften the classification decision of new data. Figure 3.8 shows an example of an ENCH. Each vertex of the closing boundary will be expanded based on the incenter of the triangle that it forms with the other two boundary vertices with which it is connected as will be explained below.

Equations used to calculate the incenters and the ENCH are the following. Given a list of vertex $V \subseteq \mathbb{R}^2$ of a NCH and one vertex $\mathbf{v}_i \in V$ connected with vertices $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$, its expanded version $\mathbf{v}_i^\lambda$ is defined with respect to their incenter point $\mathbf{c}_i$ as:

$$\mathbf{v}_i^\lambda = \mathbf{v}_i + s\lambda \frac{(\mathbf{v}_i - \mathbf{c}_i)}{\|\mathbf{v}_i - \mathbf{c}_i\|}, \forall\ \mathbf{v}_i \in V \tag{3.1}$$

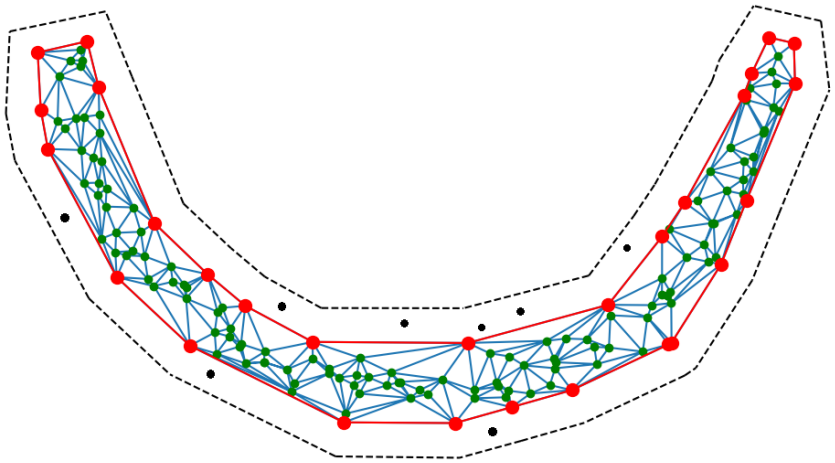


Figure 3.8: Expanded non-convex hull (black lines) obtained from an initial non-convex hull (red lines). The normal data (black dots) will be correctly classified thanks to the use of the expanded margin.

where $||\mathbf{v}_i - \mathbf{c}_i||$ is the euclidean distance between the vertex $\mathbf{v}_i$ and $\mathbf{c}_i$, s is the sign of the expansion and $\lambda \in [0, +\infty)$ is the expansion hyperparameter. The incenter $\mathbf{c}_i$ is defined as:

$$\mathbf{c}_i = \frac{\mathbf{v}_{i-1}||\mathbf{v}_i - \mathbf{v}_{i+1}|| + \mathbf{v}_i||\mathbf{v}_{i-1} - \mathbf{v}_{i+1}|| + \mathbf{v}_{i+1}||\mathbf{v}_{i-1} - \mathbf{v}_i||}{||\mathbf{v}_i - \mathbf{v}_{i+1}|| + ||\mathbf{v}_{i-1} - \mathbf{v}_{i+1}|| + ||\mathbf{v}_{i-1} - \mathbf{v}_i||}, \forall\ \mathbf{v}_i \in V \tag{3.2}$$

The sign of the expansion s of a vertex $\mathbf{v}_i$ is defined based on the internal angle α that it forms with the vertices $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$ as:

$$s = \begin{cases} 1, & \text{if } \alpha \leq 180 \\ -1, & \text{if } \alpha > 180 \end{cases} \tag{3.3}$$

Depending on the interior angle α that the vertex $\mathbf{v}_i$ forms with the other two ($\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$), we can determine if the area where $\mathbf{v}_i$ is located is convex ($\alpha < 180$)

or non-convex ($\alpha > 180$). This will influence the expansion of the vertex since the incenter of a non-convex zone is always located outside the NCH, which will invert the sign of the expansion operation (Equation 3.1). Figure 3.9 exemplifies this behavior.

The hyperparameter λ specifies a constant contraction ($0 < \lambda < 1$) or extension ($\lambda > 1$) of the NCH with respect to $\mathbf{c}_i$. We will always use $\lambda > 1$ since we are not interested in reducing the NCH. Using a too high λ value to expand a non-convex vertex could cause the ENCH to intersect itself, resulting in a complex polygon. To avoid this behavior, after each expansion, the algorithm checks whether the generated ENCH is a simple or complex polygon. If the defined value of λ causes a complex hull in some projection, the NCH will not be expanded.

3.3.5. Pseudocode

Algorithm 1 contains the pseudocode for the OCENCH training phase where the training data will be projected, the NCHs of each projection will be calculated and, finally, expanded to be able, later on, to classify new data. In line 4, a matrix is randomly generated to, in line 5, project the training data into a two-dimensional space. In line 6, a standardization model is used to normalize the projected data, and in line 7 the non-convex hulls (NCH) that surround these data are calculated. In line 8, the expanded version of the non-convex hulls (ENCH) are calculated from the vertices of the NCH and their incenters. In line 9, the system checks if the chosen λ value is valid by verifying if the expanded polygons are simple or complex.

The method to compute the NCH is shown in Algorithm 2, based on the one developed by Duckham *et al.* to geometrically represent the normal class in low-dimensional spaces through non-convex hulls but improved in this work to be able to perform subdivisions. In line 3, a Delaunay triangulation is made to get the initial convex hull. In lines 4 to 6, this initial convex hull is pruned (Algorithm 3) to fit the shape of the data. If necessary, in line 7 the closure will be subdivided (Algorithm 4) if union regions are detected. If this happens, in lines 8 to 13 the subdivision and pruning processes will alternate until all the non-convex hulls are fully adjusted.

Algorithm 3, used by Algorithm 2, describes the pruning process of a NCH based

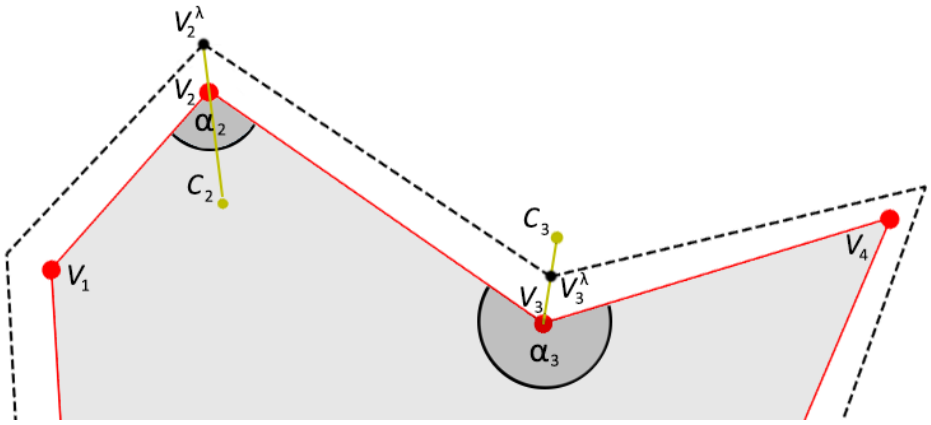


Figure 3.9: Expansion process of two vertices ($\mathbf{v}_2$ and $\mathbf{v}_3$) of a NCH (red lines). The interior angle formed by vertex $\mathbf{v}_2$ with vertices $\mathbf{v}_1$ and $\mathbf{v}_3$ is convex ($\alpha_2 < 180$), so the incenter of the triangle is inside the NCH ($\mathbf{c}_2$). However, the interior angle formed by vertex $\mathbf{v}_3$ with vertices $\mathbf{v}_2$ and $\mathbf{v}_4$ is non-convex ($\alpha_3 > 180$), so the incenter of this triangle is outside the NCH ($\mathbf{c}_3$). The incenters allow calculating the direction (yellow lines) in which the vertices should be expanded ($\mathbf{v}_2^\lambda$ and $\mathbf{v}_3^\lambda$), thus obtaining the ENCH (black lines).

on the l hyperparameter. This algorithm modifies that developed by Duckham *et al.* by adding the detection of union regions. This is done while iterating over the edges to prune them, marking those that could form union regions. On line 16, those edges whose removal would alter the properties of the triangulation will be added to a list of candidate edges. This operation does not increase the complexity of the pruning algorithm, and it greatly benefits the subdivision algorithm since this list of candidate edges considerably reduces the number of necessary checks.

Algorithm 4, also used by Algorithm 2, describes the subdivision process of a NCH based on the l hyperparameter. The goal is to use the list of candidate edges generated by Algorithm 3 to determine if union regions exist. If so, in lines 12 to 19, the method takes care of subdividing the NCH into two NCHs, as presented in section 3.3.3.

Algorithm 5 describes the expansion process of the NCHs based on the λ hyperparameter used in Algorithm 1. For each vertex of the NCH, in lines 5 to 8, the incenter and the interior angle is calculated, whilst in lines 9 to 15 the NCH is expanded.

Finally, Algorithm 6 describes the classification phase of the OCENCH to determine if a new point is normal or is an anomaly. In lines 7-8, if the new point is outside all ENCHs for some projection, it will be classified as an anomaly and it will be not necessary to continue checking the remaining projections.

Because the main operations of the algorithm, such as the construction of the NCH, are executed independently for each projection, the implementation of the method (training and classification phases) has been parallelized through the use of multi-threaded libraries. The complete implementation of the algorithm is available in GitHub [1].

Algorithm 1 : OCENCH training phase

Inputs: $\mathbf{X} \in \mathbb{R}^{d \times n}$, training dataset ($d$ variables $\times$ n samples); $\tau \in \mathbb{N}^{+}$, number of random 2D-projections; $l \in \mathbb{R}^{+}$, maximum edge length allowed in the NCH; $\lambda \in \mathbb{R}^{+}$, expansion hyperparameter of the NCH.

Output: M, model composed of τ *ENCHs*, projection matrices and normalizers.

1: **function** TRAIN
2: $M = \varnothing$
3: **for** $t = 1 \dots \tau$ **do**
4: $\mathbf{P}_t \sim N(0, 1)$ ▷ Random projection matrix $[2 \times d]$
5: $\bar{\mathbf{X}}_t = \mathbf{P}_t \mathbf{X}$ ▷ Project the data
6: $\mathbf{N}_t, normalizer_t = normalize(\mathbf{X}_t)$ ▷ Data normalization
7: $NCH_t = computeNCH(\mathbf{N}_t, l)$ ▷ NCH calculation
8: $ENCH_t = expandNCH(NCH_t, \lambda)$ ▷ ENCH calculation
9: **if** isSimple($ENCH_t$) **then** ▷ Check if the ENCHs are simple
10: $M = M \cup (ENCH_t, \mathbf{P}_t, normalizer_t)$
11: **else**
12: **return** Error ▷ Lambda value is not feasible
13: **end if**
14: **end for**
15: **return** M ▷ Returns the final model
16: **end function**

[1] https://github.com/DavidNovoaP/OCENCH

Algorithm 2 : Function used during training to construct, prune and subdivide a 2-D non-convex hull from a set of points based on a hyperparameter l

Inputs: $\mathbf{N} \in \mathbb{R}^{2\times n}$, projected data (2 variables $\times$ n samples); $l \in \mathbb{R}^{+}$, maximum edge length allowed in the NCH.

Output: *NCH*, 2-D non-convex hull.

```
function computeNCH
    Δ, E, V, S = ∅                                   ▷ Auxiliary lists
    Δ[1] = Delaunay triangulation of N
    E[1] = Boundary edges of Δ
    V[1] = Boundary vertices of Δ
    E[1], V[1], Δ[1], S[1] = Pruning(E[1], V[1], Δ[1], l)
    E, Δ, continue = Subdivision(E, Δ, S, l)
    while continue == True do
        for i = 1..len(E) do                         ▷ For each NCH
            E[i], V[i], Δ[i], S[i] = Pruning(E[i], V[i], Δ[i], l)
        end for
        E, Δ, continue = Subdivision(E, Δ, S, l)
    end while
    return E
end function
```

Algorithm 3 : Function used during training to prune the edges of a 2-D non-convex hull.

Inputs: E_i, non-convex hull boundary edges; V_i, boundary vertices; Δ_i, triangulation; $l \in \mathbb{R}^+$, maximum edge length allowed in the NCH.

Output: FB, new boundary edges; V_i, new boundary vertices; Δ_i, new triangulation; C_i, candidate triangles that could form union regions.

function PRUNING
  $FB = \varnothing$ ▷ List to store the final boundaries edges
  $C_i = \varnothing$ ▷ List to store potential union regions
  Sort the list E_i in descending order by length
  Sort the list V_i of vertices to form a chain of connected vertices
  while E_i is not empty **do**
    $\mathbf{e} \leftarrow$ head(E_i)
    Remove **e** from E_i
    if $\|\mathbf{e}\| > l$ **then**
      Find the triangle t of Δ_i that contains the two vertices of **e** and find its third vertex **v**
      if not (**v** in V_i) **then**
        Remove **e** from the triangulation Δ_i
        Insert the other two edges of t into E_i in order
        Insert **v** into V_i in order
      else
        Insert t into C_i
      end if
    else
      Insert **e** into FB
    end if
  end while
  return FB, V_i, Δ_i, C_i
end function

Algorithm 4 : Function used during training to detect and subdivide 2-D NCHs.

Inputs: E, NCH boundary edges, one list per NCH; Δ, list of triangulations, one per NCH; C, candidate edges that could form union regions, one list per NCH; $l \in \mathbb{R}^+$, maximum edge length allowed in the NCH.

Output: E, updated list of NCH boundary edges; Δ, updated list of triangulations; *continue*, $True$ if there has been any subdivision or $False$ if not.

```
1: function SUBDIVISION
2:     continue = False
3:     for i = 1 ... card(E) do                          ▷ For each NCH
4:         for e_1 in C_i do                  ▷ For each candidate edge of the NCH
5:             t_1 = triangle of Δ_i that contains the vertices of e_1
6:             v_1 = third vertex of t_1
7:             e_list = list of edges of C_i that contain vertex v_1
8:             for e_2 in e_list do
9:                 t_2 = triangle of Δ_i that contains the vertices of e_2
10:                v_2 = third vertex of t_2
11:                if v_2 ∈ e_1 then
12:                    Remove the edge e_1 from E_i and Δ_i
13:                    Remove the edge e_2 from E_i and Δ_i
14:                    Remove the edge v_1-v_2 from Δ_i
15:                    Add new boundary edges to E
16:                    Divide E_i structure in two and update E
17:                    Divide Δ_i structure in two and update Δ
18:                    continue = True
19:                    Break                          ▷ Finish subdivision
20:                end if
21:            end for
22:        end for
23:    end for
24:    return E, Δ, continue
25: end function
```

Algorithm 5 : Function used during training to expand a NCH

Inputs: *NCH*, list of NCHs, output from *computeNCH*; $\lambda \in \mathbb{R}^+$, expansion hyperparameter.

Output: *ENCH*, expanded NCHs.

function EXPANDNCH
  $ENCH = \varnothing$ ▷ List to store the final ENCHs
  for **c** in *NCH* **do** ▷ For each NCH
    for $i = 1 \dots card(\mathbf{c}.V)$ **do** ▷ For each vertex of the NCH
      $ENCH\_aux = \varnothing$ ▷ Auxiliary list to store one ENCH
      Get vertices $\mathbf{v}_i$, $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$ of $\mathbf{c}.V$
      Calculate the incenter $\mathbf{c}_i$ for the vertex $\mathbf{v}_i$ using equation (3.2)
      Calculate the interior angle (α) for the vertex $\mathbf{v}_i$
      if $\alpha > 180°$ **then**
        sign = -1 ▷ Non-convex angle
      else
        sign = 1 ▷ Convex angle
      end if
      $\mathbf{v}_{expanded} = \mathbf{v}_i + sign * \lambda * (\mathbf{v}_i - \mathbf{c}_i)/||\mathbf{v}_i - \mathbf{c}_i||$
      $ENCH\_aux = ENCH\_aux \cup (\mathbf{v}_{expanded})$
    end for
    Add $ENCH\_aux$ to the *ENCH* list
  end for
  return *ENCH*
end function

Algorithm 6 : OCENCH classification phase

Input: $\mathbf{x} \in \mathbb{R}^d$, datum to be classified; M, trained model.
Output: $Result \in \{NORMAL, ANOMALY\}$.

function CLASSIFY
  $Result = NORMAL$
  for $t = 1 \dots length(M.P)$ **do** ▷ For each projection
    $\bar{\mathbf{x}}_t = \mathbf{P}_t\mathbf{x}$ ▷ Project the new point
    $\mathbf{n}_t = normalize(\bar{\mathbf{x}}_t, normalizer_t)$ ▷ Normalize the data
    if not $isIn(\mathbf{n}_t, ENCH_t).any$ **then**
      $Result = ANOMALY$ ▷ The point is an anomaly
      $Break$
    end if
  end for
  return $Result$
end function

3.4. Evaluation

In this section, several experiments are presented to show the behavior of the proposed algorithm in real scenarios. In this study, we tested the proposed method against the Non-Convex Boundary over Projections (NCBoP) method [87], a similar method based on the use of random projections, as well as against other known machine learning methods for anomaly detection: Autoencoder (AE) [11], Isolation Forest (IF) [118], Local Outlier Factor (LOF) [22], One-Class Support Vector Machine (OCSVM) [167], Robust Covariance (RC) [159] and Support Vector Data Description (SVDD) [201].

These algorithms have been evaluated over 10 real datasets available in the UCI Machine Learning Repository [9] and in the ODDS benchmark [168]. The characteristics of these datasets are summarized in Table 3.1. The data have been normalized using standard scalers with zero mean and unit variance. To assess the performance of each algorithm, the data has been divided into training and test sets using a 10-fold cross validation. All algorithms have been trained using only normal data,

while the test phase included data from both classes. The choice of the hyperparameters of the different methods has been carried out using grid search, selecting those configurations that produce the best results. The combinations of hyperparameters that reported the best results are available in Appendix A.1.

The metric used to measure the performance of the algorithms was the F1-score which relates precision and recall. In order to obtain more reliable results the experiments were repeated twice.

Table 3.2 summarizes the mean test results. The chosen statistical test was Nemenyi, a non-parametric test that makes a pairwise comparison between models [41, 57]. Using a significance level of 5% ($\alpha = 0.05$) and the F1-scores obtained for each dataset independently, the best values in Table 3.2 have been highlighted in bold. As can be seen, the OCENCH algorithm presents a very robust behavior, achieving good performance for most datasets. In five of the ten datasets, the subdivision of the NCH has been beneficial, increasing the performance obtained by the method (Annthyroid, Shuttle, and Optdigit) or achieving similar results in a significantly shorter time (Telescope and Miniboone).

A statistical test was also carried out to compare the global performance of the algorithms. The chosen test was again Nemenyi. Using a significance level of 5% and the F1-scores of the algorithms for the different datasets, OCENCH ranks within the group of best methods, represented graphically by Figure 3.10. As can be seen, the null hypothesis of algorithms having similar performance is accepted for OCENCH, LOF, OCSVM, RC, IF, NCBoP, and AE, so we can affirm that in these

Table 3.1: Characteristics of the datasets used.

Dataset	# Samples	Anomalies	Input dimension
Annthyroid	7200	534 (7.4%)	6
Thyroid	3772	93 (2.5%)	6
Shuttle	49097	3511 (7.2%)	9
Telescope	19020	6573 (34.5%)	10
Pendigits	6870	156 (2.3%)	16
Cardio	1831	176 (9.6%)	21
Ionosphere	351	126 (35.9%)	33
Miniboone	130065	36499 (28.1%)	50
Optdigit	5216	64 (2.9%)	64
MNIST	7603	700 (9.2%)	100

Table 3.2: Average test F1-score ± standard deviation for the different datasets. Best values according to the statistical test have been highlighted in bold.

Dataset	OCENCH	NCBoP	AE	IF	LOF	OCSVM	RC	SVDD
Annthyroid	**77.5±0.9**	51.0±1.7	**78.9±2.6**	**84.2±1.7**	72.4±0.7	66.8±0.4	**85.9±0.5**	63.5±4.4
Thyroid	**93.4±1.1**	89.9±1.0	87.5±1.7	**94.9±1.4**	**92.7±1.4**	**92.4±1.4**	**95.0±1.1**	83.9±3.3
Shuttle	**99.2±0.2**	97.5±0.1	86.3±0.4	**98.1±0.2**	95.2±0.6	96.9±0.4	96.3±0.3	85.0±10.7
Telescope	**70.1±0.9**	**69.7±0.7**	66.7±1.1	63.5±2.7	**71.0±0.4**	66.6±0.4	**70.7±0.4**	66.8±0.1
Pendigits	**93.73±1.1**	**92.9±1.1**	82.4±1.3	**93.6±1.4**	**95.2±0.9**	**94.2±0.7**	84.1±1.3	**96.4±0.6**
Cardio	**94.3±1.4**	**92.0±1.9**	86.7±1.5	**90.7±1.1**	89.7±1.6	**92.3±0.7**	79.5±1.7	80.5±3.1
Ionosphere	**85.9±2.6**	**84.3±4.3**	**85.9±3.5**	82.8±3.5	**85.2±3.6**	**87.9±2.4**	80.1±2.7	**89.8±2.3**
Miniboone	**76.6±0.5**	49.9±2.3	**71.9±0.3**	64.0±1.9	**72.2±0.6**	67.9±0.2	68.5±0.3	69.6±1.8
Optdigit	**84.3±1.4**	74.7±1.9	71.5±0.6	72.2±6.4	**95.8±0.8**	**85.2±0.6**	75.7±1.6	66.7±0.2
MNIST	80.2±0.9	73.7±1.6	72.5±0.3	78.0±1.1	**88.9±0.4**	**93.7±0.6**	**82.7±2.2**	73.7±5.4

tests our method was found among the best models. In addition, we can affirm that OCENCH performance was significantly better than SVDD.

Table 3.3 shows the ranking obtained by each algorithm for each dataset according to the average test F1-score. To calculate this ranking, when two algorithms obtained the same average F1-score, they were assigned the same ranking. For example, in Ionosphere, OCENCH and AE rank 3rd and 4th based on their F1-score. Since their F1-score has been the same (85.9), they were both assigned the same position in the table (3.5). The final average ranking is shown in the last row. As can be observed, OCENCH was the algorithm that obtained the best average ranking (2.8) with a considerable difference between the average ranking of the first three methods (OCENCH, LOF, and OCSVM) and the rest.

Table 3.4 shows the mean training time of each algorithm (lower values than 0.05 have been represented as 0.0). As can be seen, the training time of the methods that use random projections such as OCENCH or NCBoP is higher than that of most algorithms, however they are still reasonable times for their use. Test times have not been included in this work because they are very low for all the algorithms.

3.5. Ablation study

In order to understand the impact of the different hyperparameters of the method on its performance, an ablation study has been carried out. As has been seen, the OCENCH method has four hyperparameters to adapt the algorithm to the scenario

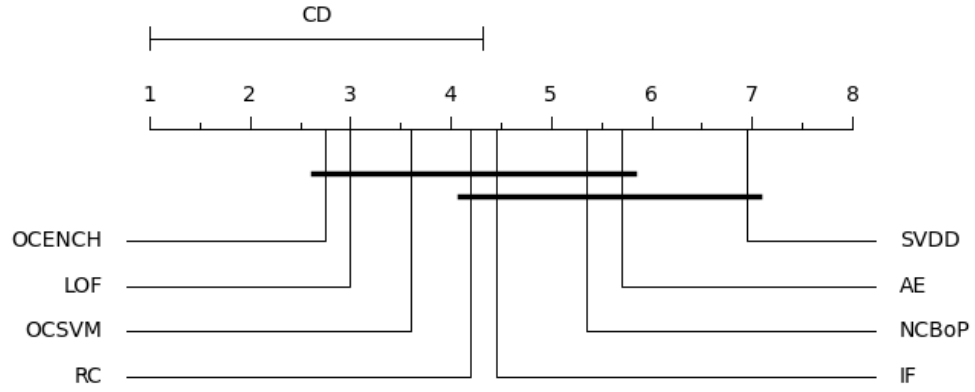


Figure 3.10: Graphical representation of Nemenyi test with $\alpha = 0.05$. The critical distance (CD) obtained was 3.32.

Table 3.3: Ranking of the algorithms for the different datasets and global average ranking.

Dataset	OCENCH	NCBoP	AE	IF	LOF	OCSVM	RC	SVDD
Annthyroid	4	8	3	2	5	6	1	7
Thyroid	3	6	7	2	4	5	1	8
Shuttle	1	3	7	2	6	4	5	8
Telescope	3	4	6	8	1	7	2	5
Pendigits	4	6	8	5	2	3	7	1
Cardio	1	3	6	4	5	2	8	7
Ionosphere	3.5	6	3.5	6	5	2	8	1
Miniboone	1	8	3	7	2	6	5	4
Optdigit	3	5	7.5	6	1	2	4	7.5
MNIST	4	6.5	8	5	2	1	3	6.5
Avg. Rank	**2.8**	5.6	5.9	4.7	3.3	3.8	4.4	5.5

Table 3.4: Average training time ± standard deviation for the different datasets.

Dataset	OCENCH	NCBoP	AE	IF	LOF	OCSVM	RC	SVDD
Annthyroid	329.5±9.1	622.9±14.7	2.6±0.7	0.2±0.0	0.1±0.0	0.5±0.0	1.1±0.2	50.7±1.3
Thyroid	1.7±0.1	16.8±2.7	1.1±0.1	0.4±0.0	0.0±0.0	0.1±0.0	0.8±0.1	8.9±1.6
Shuttle	7.6±1.4	0.7±0.0	1.5±0.8	0.6±0.0	0.7±0.0	0.2±0.0	3.5±0.7	125.8±9.6
Telescope	75.8±3.2	193.2±8.7	1.51±0.2	0.0±0.0	0.2±0.0	0.5±0.0	3.2±0.9	264.8±43.9
Pendigits	0.9±0.0	0.5±0.0	2.6±0.6	0.5±0.0	0.3±0.0	0.2±0.0	1.8±0.0	53.5±3.2
Cardio	0.5±0.0	0.8±0.0	1.1±0.1	0.3±0.0	0.1±0.0	0.0±0.0	0.6±0.1	1.2±0.1
Ionosphere	0.1±0.0	0.0±0.0	5.8±1.9	0.0±0.0	0.0±0.0	0.0±0.0	0.1±0.0	0.0±0.0
Miniboone	139.4±4.2	494.2±19.1	3.4±0.5	0.7±0.0	1.4±0.0	55.1±4.2	17.2±2.6	76.1±3.6
Optdigit	11.6±2.7	252.7±9.8	2.1±0.4	0.2±0.0	0.4±0.0	0.4±0.0	2.7±0.0	13.6±0.9
MNIST	1.7±0.0	133.3±1.2	10.5±2.3	1.4±0.0	0.7±0.0	0.2±0.0	9.1±0.2	30.0±1.4
Avg. Rank	5.2	5.3	5.4	1.6	1.2	1.7	4.4	6.5

in which it is going to be used:

- Number of projections (τ): number of random projections used to represent the target dataset. A very low number of projections may imply a too simple representation, while a very high number carries a higher computational cost that in some cases may not be worth it. The goal is to use a sufficient number of projections to represent the class accurately while maintaining a feasible computational cost.

- Subdivision (boolean value): by default it is recommended to allow the subdivision of closures to represent the dataset. If *a priori* it is known that the dataset does not present disjointed regions, this subdivision process could be avoided to reduce the computational cost of the algorithm. In any case, if subdivision is allowed, it will be carried out by OCENCH automatically only when necessary.

- Maximum edge size (l): value that regulates the edge pruning process and, if it is enabled, the subdivision process. Since the projected data is normalized in each 2-D subspace independently, it is only necessary to define a single value for this hyperparameter that will be used universally in all these subspaces. A too small value will cause a very extreme fit to the shape of the data, which can cause an excessive number of subdivisions with the consequent computational cost, while a too large value could cause a poorly adjusted representation, making it difficult to subdivide the closures. The objective is to use a maximum

edge size that allows the class to be represented accurately while maintaining a feasible computational cost.

- Expansion hyperparameter (λ): to avoid the effect of over-adjustment of the training data for the anomaly detection problem, it is possible to expand the final closures in a controlled way. This process is computationally inexpensive and in certain scenarios it can be beneficial, such as in cases where there are not a large number of instances of the class to be modeled.

To show the influence of the hyperparameter selection on the performance of the OCENCH algorithm (F1-score), several experiments have been carried out using the datasets tested in this work and the best hyperparameter combinations for each of them (Appendix A.1). These experiments are aimed to study the influence of each of the hyperparameters on the performance of the algorithm when the other three hyperparameters take a fixed value. The hyperparameters that have been studied are the number of projections, the maximum edge size l, and the λ expansion factor. The boolean hyperparameter *Subdivision* is always on, it is not modified.

Since the data are normalized, to study the behavior of the hyperparameter l, it has taken values between 0.1 and 1.7, with increments of 0.1. As can be seen in Figure 3.11, the range of values with which the best results are obtained for these 10 scenarios is between 0.1 and 1.1. However, there is a particular subrange of values of l for each dataset in which the best performance of the algorithm is obtained, for example, $l \in [0.3, 1.1]$ in the case of Thyroid. Using a grid search with the proposed generic values is an easy way to estimate the best operating range for this hyperparameter that controls the level of adjustment of the closures and their subdivision.

To study the influence of the number of projections on the performance of the algorithm, between 5 and 2000 projections have been used. Figure 3.12, whose horizontal axis is on a logarithmic scale, shows the results obtained. As can be seen, the number of projections required to characterize the dataset varies considerably depending on the characteristics of each scenario (see Table 3.1). In any case, as with the l hyperparameter, there is a subrange of values for each dataset in which the best performance of the algorithm is obtained. Using an initial grid search with several projections in the interval [5, 2000] is an easy way to estimate the best

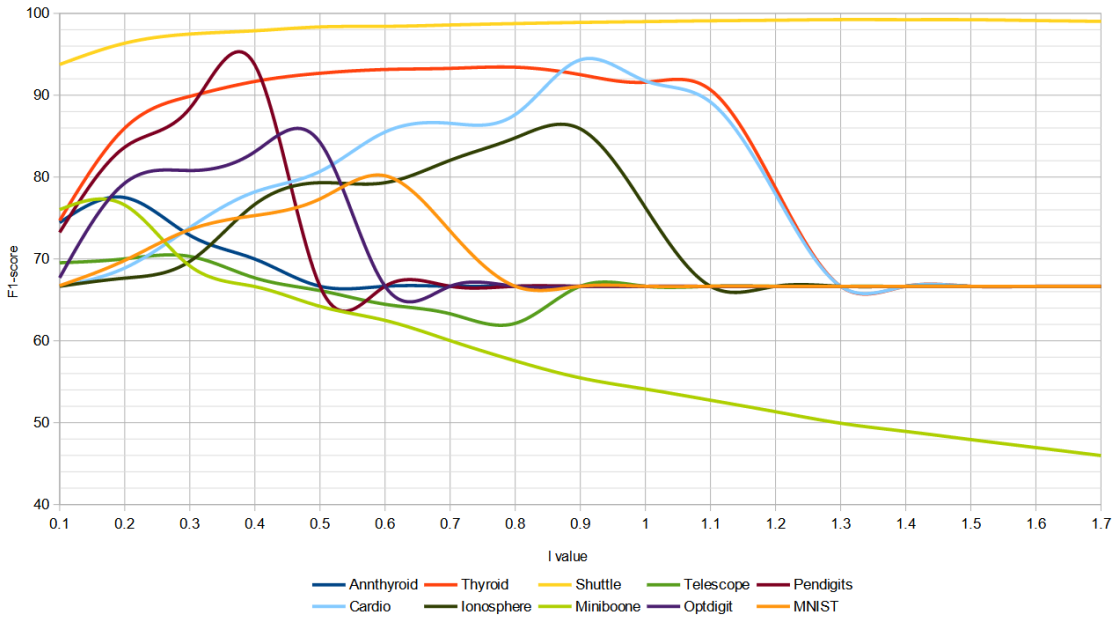


Figure 3.11: Influence of l on the F1-score when the number of projections and the λ expansion factor are fixed.

operating range for this hyperparameter.

To study the influence of the expansion factor on the performance of the algorithm, values between 0.0001 (0.01%) and 2 (200%) have been used. Figure 3.13, whose horizontal axis is on a logarithmic scale, shows the results obtained. In all but three of the datasets (Pendigits, Miniboone, and Annthyroid), the expansion has improved the performance of the algorithm compared to using the original NCHs. Using an initial grid search with values in the interval [0, 2] is an easy way to estimate if expansion is beneficial to algorithm performance and if so, determine the best value for λ.

3.6. Conclusions

The developed OCENCH algorithm allows working with non-convex and disjointed datasets in a novel way, offering a robust behavior in the different tests carried out. Its performance is remarkable, positioning itself as an alternative for both convex and non-convex problems. In addition, OCENCH is easily configurable

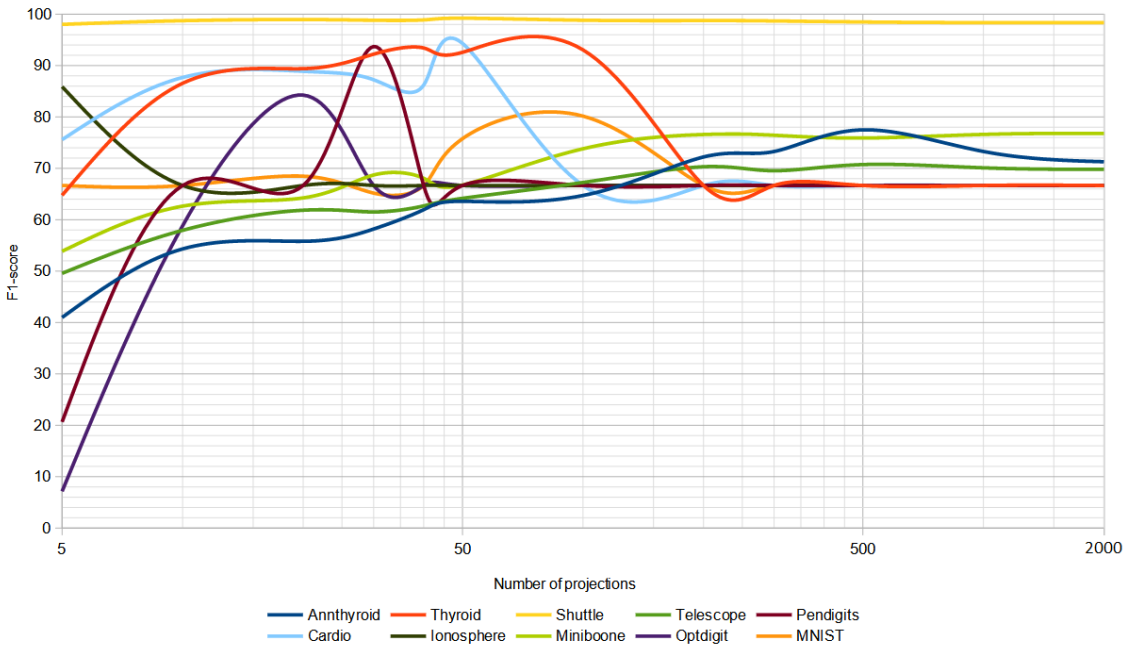


Figure 3.12: Influence of the number of projections on the F1-score when l and the λ expansion factor are fixed.

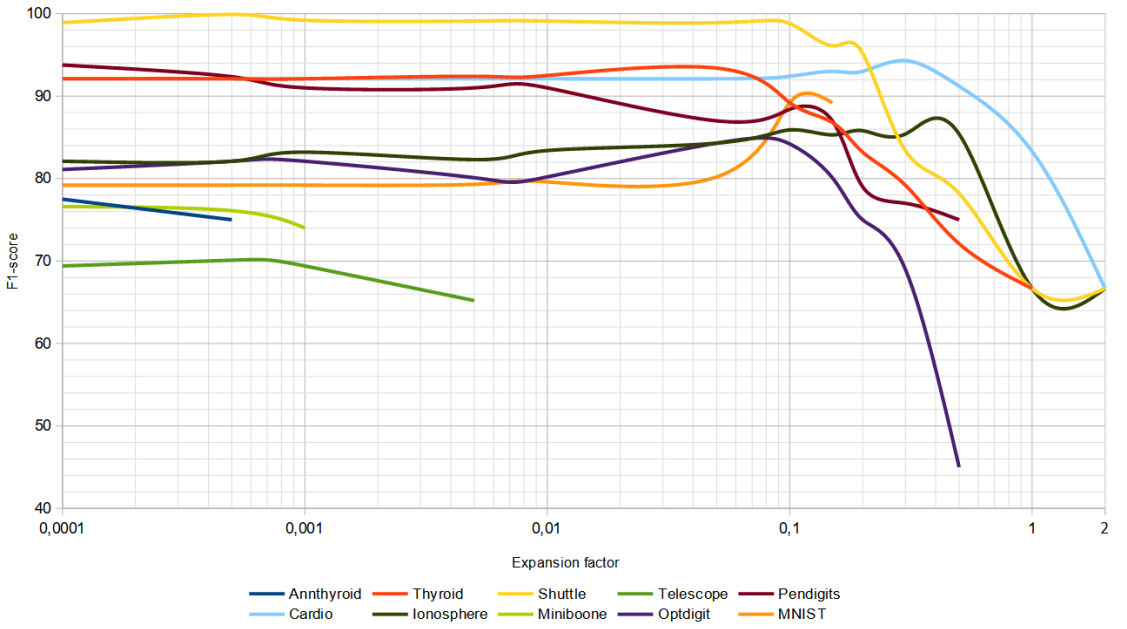


Figure 3.13: Influence of the expansion factor on the F1-score when the number of projections and l are fixed. The lines are drawn up to the maximum possible value without generating complex polygons.

through its hyperparameters, it only needs normal data to be trained and is not necessary to know *a priori* the percentage of anomalies in the dataset. The models generated with this algorithm are light on memory requirements since it is not necessary to store all the data used for the creation of the non-convex hulls. The model only needs the information of the vertices of each non-convex hull to determine if new data is inside or outside the normal class. Finally, the algorithm has been implemented using specialized libraries so that it can be executed in parallel. The calculations of each projection are carried out independently, so its parallelization allows for drastically reducing the execution time of the training and classification phases.

As future work, it would be interesting to develop and test a version of the algorithm that could detect the existence of empty regions inside dense regions to be able to fit more precisely in these scenarios, such as rings or concentric rings. Another possible line of work would be the implementation of a forgetting mechanism to extend the method to online learning environments. In addition, it could also be interesting to adapt and test the algorithm in edge computing and federated learning environments, for example using different projections on each device.

Chapter 4

A proposal for anomaly detection in edge computing and federated learning scenarios

As happened at the time with the massive adoption of personal computers, the technological development of recent years has caused a substantial increase in the number of small computing machines such as smartphones or Internet of Things (IoT) devices, for both industrial and personal use. Despite their size, they have enough computing power to perform tasks that a few years ago were considered unapproachable, such as the training of small machine learning models, real-time inference, or the exchange of large amounts of information at high speeds. Due to the abundance of these devices and the inefficiencies of traditional cloud computing for applications that demand low latencies, a new computing paradigm called edge computing has emerged [26]. Edge computing (EC) moves computing away from data centers to the edge of the network, bringing cloud computing services and utilities closer to the end user and their devices. This allows faster information processing and response time, as well as freeing up the network bandwidth.

From a machine learning (ML) point of view, this new technological scenario is very suitable for the use of federated learning [236]. Federated learning (FL) is a collaborative machine learning scheme that allows heterogeneous devices with different private datasets to work together to train a global model. In addition, this

work scheme emphasizes the preservation of the privacy of local data collected on each device by implementing mechanisms that prevent possible direct and indirect leaks of their data.

If we focus on anomaly detection, in many real systems, the response time to the recognition of an anomaly can be critical, as is the case of failures in industrial systems [44] or network intrusions detection [4]. In addition, in certain scenarios such as the medical or banking field, the privacy of the data that is exchanged is essential. For this reason, the development of anomaly detection techniques based on edge computing and federated learning may be the solution to reduce these response times and infrastructure limitations while preserving data privacy.

In this chapter we introduce DAEF (Deep AutoEncoder for Federated learning) [147], a fast and privacy-preserving deep autoencoder very suitable for edge computing and federated learning scenarios, in addition to classic machine learning environments. Unlike traditional deep neural networks, its learning method is non-iterative, which drastically reduces its training time. Its training can be carried out incrementally (aggregation of models), in a distributed way (training shared among multiple nodes), and in parallel (at the node level if it has several cores), and due to its mathematical formulation, the information that is exchanged does not endanger the privacy of the training data. All of this makes DAEF a useful method for edge computing and federated training, capable of performing tasks such as anomaly detection on large datasets while maintaining the performance of traditional (iterative) autoencoders.

4.1. Related work

Anomaly detection is a field that has a large number of algorithms that solve the problem of distinguishing between normal and anomalous instances in a wide variety of ways [26, 29]. Depending on the assumptions and processes they employ, in traditional anomaly detection, we can distinguish between five main types of methods: probabilistic, distance-based, information theory-based, boundary-based, and reconstruction-based methods. In general, these algorithms are characterized by their high performance when classifying new data, however, they do not focus on

other aspects which from a centralized perspective may seem less important, such as data privacy and incremental learning. This makes it difficult to apply many of these classical methods in decentralized environments. For this reason, the strong expansion of edge computing has brought with it a new line of research in the field of anomaly detection in charge of designing new algorithms capable of learning in a distributed and, in some cases, incremental way, while preserving data privacy.

Due to their good performance, it is common for these anomaly detection methods to be based on reconstruction (neural networks). In this section we will distinguish between reconstruction-based methods that use autoencoders [12] and those that do not. Among those that do not use autoencoders is DÏOT [142], a self-learning distributed system for security monitoring of IoT devices that utilizes a novel anomaly detection approach based on representing network packets as symbols, allowing to use a language analysis technique to detect anomalies. B. Hussain *et al.* [80] presented a deep learning framework to monitor user activities of multiple cells and thus detect anomalies using feedforward deep neural networks. R. Abdel *et al.* [183] introduced a federated stacked long short-time memory model to solve multi-task problems using IoT sensors in smart buildings. Y. Zhao *et al.* [240] propose a multi-task deep neural network in federated learning to perform simultaneously network anomaly detection, VPN traffic recognition, and traffic classification. Other authors like D. Preuveneers *et al.* [163] propose the use of blockchain technology to carry out a decentralized registry of federated model updates. This guarantees the integrity of incrementally-learned machine learning models by cryptographically chaining one machine learning model to the next. These solutions obtain good results, however, they do not emphasize privacy preservation and their iterative learning can lead to long training times.

On the other hand, if we focus on autoencoders [12], it is also possible to find works oriented towards edge computing and/or federated learning scenarios. Autoencoders (AE) are a type of self-associative neural network whose output layer seeks to reproduce the data presented to the input layer after having gone through a dimensional compression phase. In this way, they manage to obtain a representation of the input data in a space with a dimension smaller than the original, learning a compact representation of the data, retaining the important information, and compressing the redundant one. For this reason, they are widely used for the

elaboration of models that are robust to noise, an important quality in anomaly detection and regression problems. Figure 1 represents the traditional architecture of an autoencoder.

T. Luo *et al.* [123] propose to use autoencoders for anomaly detection in wireless sensor networks, however, each edge device does not train a local model with its data. These devices send their local data to a central cloud node from which the training of the global model is carried out. In the approach presented by M. Ngo *et al.* [141], an adaptive hierarchical edge computing system composed of three autoencoder models of increasing complexity is used for IoT anomaly detection.

In the two previous works, as well as in the majority that use this type of networks, the autoencoders are trained during several iterations to adjust their parameters (weights, bias) using techniques such as gradient descent and backpropagation. This greatly increases training time, especially when dealing with large datasets or complex network architectures, which in edge computing scenarios can be critical. However, there is a line of work that allows training autoencoders in a non-iterative way. This is based on Extreme Learning Machines (ELM) [77], an alternative learning algorithm originally formulated for single-hidden layer feedforward neural networks (SLFNs). This algorithm tends to provide good generalization performance and an extremely fast learning speed. Over time, more advanced versions such as MLELM [90], a multilayer version of ELM, or DELM [43], a deep version of ELM, have been developed. For anomaly detection in edge computing and federated learning scenarios, R. Ito *et al.* [83] propose to combine OS-ELM (Online Sequential Extreme Learning Machine) [114] with autoencoders. This allows each edge device to train its own local model and incrementally update it with the results obtained by the other devices. Nevertheless, a possible limitation of this solution is its autoencoder architecture with only one hidden layer, which in some cases may not be sufficient.

In this work we present DAEF, a deep autoencoder with the following characteristics:

- The architecture is deep (more than one hidden layer) and asymmetrical.
- The training process is non-iterative and therefore faster than a traditional autoencoder.

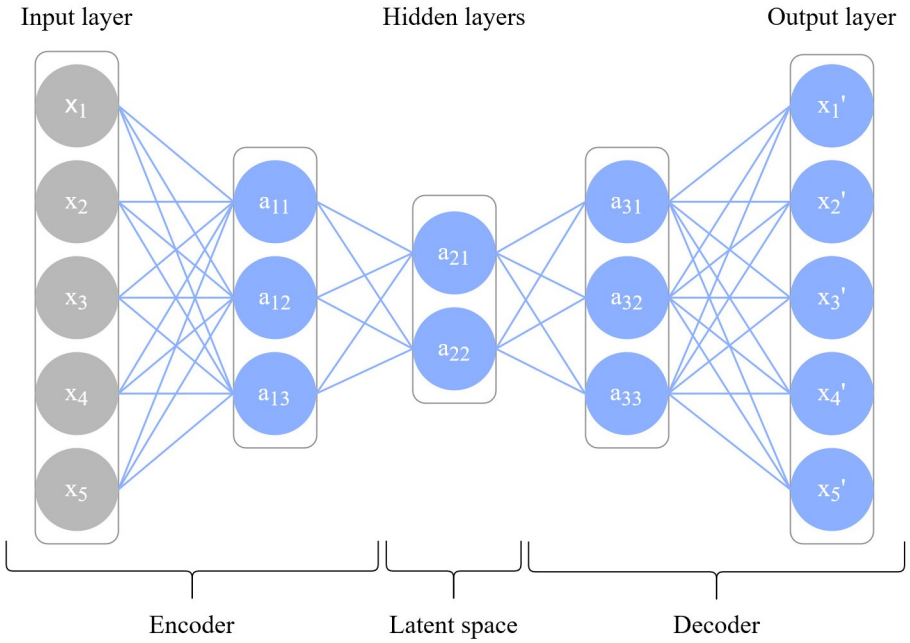


Figure 4.1: Example of autoencoder neural network architecture.

- It can be trained in a distributed and incremental way, which makes it suitable for edge computing and federated learning environments.
- It is a privacy-preserving method.

4.2. Background

This section introduces the theoretical foundations and as well as ideas taken as the basis for the development of DAEF.

4.2.1. Federated learning architectures

Federated Learning systems can be classified into four types based on the level of centralization of their architectures [233]: centralized, hierarchical, regional, and decentralized. In centralized FL systems, all edge nodes are connected to a central aggregation node to update local weights and distribute models. Hierarchical architectures achieve less level of centralization by keeping a central node but including

several regional aggregation nodes whose goal is to reduce data exchange and manage local devices. Regional architectures completely eliminate the central node to eliminate the risk of a single point of failure. Decentralized approaches move all tasks to the edge, so that edge nodes will handle both local training and model aggregation.

The most common is to use centralized or semi-centralized architectures, in which the aggregator node is a server with high computational capabilities. In this work, we want to design a valid method to be used with more decentralized architectures on FL and EC scenarios, in which the training and aggregations are carried out in the edge devices without requiring a large computational power, and exchange of data through the network is secure and preserves the privacy of local datasets.

4.2.2. Distributed Singular Value Decomposition Autoencoder

DSVD-autoencoder (Distributed Singular Value Decomposition Autoencoder) [55] is a hidden single-layer autoencoder network for anomaly detection, consisting of an encoder followed by a decoder. The idea of its encoder will serve as the basis for our work and, therefore, is the one that will be explained in detail below.

Regarding its encoder part, the aim is to learn a vector space embedding of the input data $\mathbf{X} \in \mathbb{R}^{m_0 \times n}$, where m_0 is the number of input variables and n the number of data samples, extracting a meaningful but lower dimensional representation, known as the latent space. This can be accomplished by a low-rank matrix approximation of $\mathbf{X}$, which is a minimization problem that tries to approximate a given matrix by another one subject to the constraint that the approximating matrix has reduced rank [48]. The size of this rank is determined by the size of the hidden layer. Thus, being m_1 the number of neurons of this layer, the Singular Value Decomposition (SVD) of rank-m_1 of matrix $\mathbf{X}$ is used to obtain the weights $\mathbf{W}_1$ of this first layer, as follows. Let's consider the full SVD of $\mathbf{X}$, a factorization of the form:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{4.1}$$

where $\mathbf{S} \in \mathbb{R}^{m_0 \times n}$ is a diagonal matrix with descending ordered non-negative values on the diagonal that are the singular values of $\mathbf{X}$, while $\mathbf{U} \in \mathbb{R}^{m_0 \times m_0}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$

are orthogonal matrices containing the left and right singular vectors of $\mathbf{X}$. In a low-rank approximation, the optimal rank-m_1 approximation of $\mathbf{X}$ can be computed by taking the first m_1 columns of $\mathbf{U}$ and rows of $\mathbf{V}^T$ and truncating $\mathbf{S}$ to the first m_1 diagonal elements. The new truncated matrices $\mathbf{U}_{m_1} \in \mathbb{R}^{m_0 \times m_1}$ and $\mathbf{V}_{m_1}^T \in \mathbb{R}^{m_1 \times n}$ are, respectively, m_1-dimensional representations of rows (features) and columns (samples) of the input data $\mathbf{X}$. Therefore, $\mathbf{U}_{m_1}$ is used as the weights $\mathbf{W}_1$ for the first layer as it contains the m_1-dimensional transformation of the input space ($\mathbb{R}^{m_0} \rightarrow \mathbb{R}^{m_1}$).

Moreover, in a distributed scenario, the SVD of an entire data matrix $\mathbf{X}$ distributed into P several blocks, that is $\mathbf{X} = \left[\mathbf{X}^1|\mathbf{X}^2|\cdots|\mathbf{X}^P\right]$, can be also computed distributively by calculating at each site $p = 1, \ldots, P$ the local matrices $\mathbf{U}^p$ and $\mathbf{S}^p$, corresponding to the rank-m_1 SVD of $\mathbf{X}^p$, and then arbitrarily computing the following operation:

$$[\mathbf{U}_{m_1}, \mathbf{S}_{m_1}, \sim] = SVD([\mathbf{U}^1\mathbf{S}^1|\ldots|\mathbf{U}^P\mathbf{S}^P]). \tag{4.2}$$

Notice that, as only the local SVD decompositions are exchanged between nodes, from which the original training data could not be deduced, data privacy is preserved in this process. Algorithm 7 shows this process.

Algorithm 7 Distributed singular value decomposition (DSVD)

Input: $\mathbf{X} = \left[\mathbf{X}^1|\mathbf{X}^2|\cdots|\mathbf{X}^P\right] \in \mathbb{R}^{m_0 \times n}$, training data split into P partitions (m_0 variables $\times$ n samples); m_1, rank of the SVD.

Output: $\mathbf{U}_{m_1}$, $\mathbf{S}_{m_1}$ corresponding to the rank-m_1 decomposition of $\mathbf{X}$.

1: **function** DSVD
2: **parallelFor** $p = 1...P$ ▷ Each partition in parallel
3: $(\mathbf{U}^p, \mathbf{S}^p, \sim) = SVD(\mathbf{X}^p)$
4: **end parallelFor**
5: $[\mathbf{U}, \mathbf{S}, \sim] = SVD\left([(\mathbf{U}^1\mathbf{S}^1 \mid \cdots \mid \mathbf{U}^p\mathbf{S}^p)]\right)$ ▷ Recalculate
6: $\mathbf{U}_{m_1} = \mathbf{U}\left[:, 1 : m_1\right]$ ▷ Obtain the rank-m_1
7: $\mathbf{S}_{m_1} = \mathbf{S}\left[:, 1 : m_1\right]$ ▷ Obtain the rank-m_1
8: **end function**

4.2.3. Regularized One-Layer Neural Network

ROLANN (Regularized One-Layer Neural Networks) [53] is a L2 norm-regularized training method that allows training single-layer neural networks (without hidden layers) in a non-iterative manner by minimizing the mean squared error (MSE) measured before the activation function of the output neurons, as described in [54]. In addition to not being iterative, the complexity of the method depends on the smaller of the dimensions that define the size of the training set (number of samples and number of variables), which together make the method computationally very efficient. Moreover, the algorithm can be used in an incremental and distributed way while preserving privacy, two possibilities that can be combined as needed to obtain, for instance, a distributed learning environment that also learns incrementally at each location. These characteristics make the method a perfect fit for federated learning environments.

Its bases are as follows. Let's consider the training input data $\mathbf{X} \in \mathbb{R}^{m \times n}$, where m is the number of input variables and n the number of data samples. After propagating this data through the network we can obtain, for each data point, the values of the derivative $\mathbf{f}'$ and the inverse $\bar{\mathbf{d}}$ of the neural function $\mathbf{f}$. With this information the weights $\mathbf{W}$ of the network can be analytically calculated as:

$$[\mathbf{U}, \mathbf{S}, \sim] = SVD\left(\mathbf{X}\mathbf{F}\right), \tag{4.3}$$

$$\mathbf{M} = \mathbf{X} * \left(\mathbf{f}'. * \mathbf{f}'. * \bar{\mathbf{d}}\right), \tag{4.4}$$

$$\mathbf{W} = \mathbf{U} * inv\left(\mathbf{S} * \mathbf{S} + \lambda \mathbf{I}\right) * \left(\mathbf{U}^T * \mathbf{M}\right) \tag{4.5}$$

where $\mathbf{F}$ is the diagonal matrix of $\mathbf{f}$, λ is a weight-decay based regularization parameter and $\mathbf{I}$ is the identity matrix.

As mentioned, this method allows incremental and distributed learning. Suppose $\mathbf{M}^p$, $\mathbf{U}^p$ and $\mathbf{S}^p$ correspond to the knowledge obtained in the current data partition p. If there were a previous k data partition that ROLANN was already trained on,

this method can learn from both partitions by calculating:

$$\left[\mathbf{U}^{k|p}, \mathbf{S}^{k|p}, \sim\right] = SVD\left(\mathbf{U}^{k}\mathbf{S}^{k} \mid \mathbf{U}^{p}\mathbf{S}^{p}\right), \tag{4.6}$$

$$\mathbf{M}^{k|p} = \mathbf{M}^{k} + \mathbf{M}^{p}, \tag{4.7}$$

and finally obtaining the weights as:

$$\mathbf{W} = \mathbf{U}^{k|p} * inv\left(\mathbf{S}^{k|p} * \mathbf{S}^{k|p} + \lambda\mathbf{I}\right) * \left(\mathbf{U}^{k|p^{T}} * \mathbf{M}^{k|p}\right), \tag{4.8}$$

Algorithm 8 shows this process in detail.

Algorithm 8 ROLANN for incremental/distributed regularized learning

Input: $\mathbf{X}_p \in \mathbb{R}^{m\times n}$, new training data ($m$ variables $\times$ n samples); $\mathbf{d}_p \in \mathbb{R}^{n\times 1}$, desired outputs; f, nonlinear activation function (invertible); λ, regularization hyperparameter; $\mathbf{M}_k, \mathbf{U}_k, \mathbf{S}_k$, matrices calculated using previous data (optional).

Output: $\mathbf{w} \in \mathbb{R}^{m\times 1}$, optimal weights; $\mathbf{M}_{k|p}, \mathbf{U}_{k|p}, \mathbf{S}_{k|p}$, updated $\mathbf{M}, \mathbf{U}, \mathbf{S}$ matrices.

function ROLANN
  $\mathbf{X}_p = [ones\,(1, n)\,; \mathbf{X}_p]$ ▷ Bias is added
  $\overline{\mathbf{d}}_p = f^{-1}\,(\mathbf{d}_p)$ ▷ Inverse of the neural function
  $\mathbf{f}^{'} = f^{'}\left(\overline{\mathbf{d}}_p\right)$ ▷ Derivate of the neural function
  $\mathbf{F}_p = diag(\mathbf{f}^{'})$ ▷ Diagonal matrix
  if ($\mathbf{M}_k$ & $\mathbf{U}_k$ & $\mathbf{S}_k$ are empty matrices) **then**
    $\left[\mathbf{U}_{k|p}, \mathbf{S}_{k|p}, \sim\right] = SVD(\mathbf{X}_p * \mathbf{F}_p)$ ▷ Economy size
    $\mathbf{M}_{k|p} = \mathbf{X}_p * (\mathbf{f}^{'} .* \mathbf{f}^{'} .* \overline{\mathbf{d}}_p)$
  else ▷ Combine previous knowledge
    $\mathbf{M}_{k|p} = \mathbf{M}_k + \mathbf{X}_p * (\mathbf{f}^{'} .* \mathbf{f}^{'} .* \overline{\mathbf{d}}_p)$
    $[\mathbf{U}_p, \mathbf{S}_p, \sim] = SVD(\mathbf{X}_p * \mathbf{F}_p)$
    $\left[\mathbf{U}_{k|p}, \mathbf{S}_{k|p}, \sim\right] = SVD([\mathbf{U}_k\mathbf{S}_k|\mathbf{U}_p\mathbf{S}_p])$
  end if
  $\mathbf{w} = \mathbf{U}_{k|p} * inv(\mathbf{S}_{k|p} * \mathbf{S}_{k|p} + \lambda\mathbf{I}) * (\mathbf{U}_{k|p}^{T} * \mathbf{M}_{k|p})$
end function

4.2.4. Multilayer Extreme Learning Machine

MLELM (Multilayer Extreme Learning Machine) [90] is a multilayer neural network that, in each layer, makes use of unsupervised learning to train its parameters, eliminating the need to fine-tune the network. As a result, the authors obtain a method to train multilayer networks in a non-iterative, fast, and mathematically simple way. Specifically, it obtains the weights at each layer by using an ELM-AE (Extreme Learning Machine-Autoencoder) [90], which is an unsupervised single hidden layer autoencoder that, like any other, tries to reproduce the input signal at the output. This mechanism has served as an inspiration for the work presented here.

4.3. Proposed method

The main objective DAEF (Deep Autoencoder for Federated learning) is to learn a compressed representation of the normal data and, from this reduced space, to reconstruct the inputs at the output of the autoencoder. These tasks should be carried out in a distributed way, and incrementally where possible, to apply the algorithm in edge computing and federated learning environments. To achieve this, DAEF employs an asymmetric autoencoder architecture as shown in Figure 2. A first single-layer encoder reduces the dimensionality of the input data and it is adjusted using a distributed SVD process. It is followed by a multi-layer decoder to reconstruct the input signal at the output which is trained in a layer-by-layer basis through a non-iterative process. This section presents in detail the steps followed by the method and its theoretical foundations.

4.3.1. The encoder

Given an input dataset $\mathbf{X} \in \mathbb{R}^{m_0 \times n}$, where m_0 is the number of input variables and n the number of data samples, the goal of the encoder is to transform $\mathbf{X}$ into a vector space embedding of lower-dimension. This will be done using the Distributed Singular Value Decomposition (Algorithm 7). Therefore, the weights of the first layer $\mathbf{W}_1 = \mathbf{U}_1$ are obtained collaboratively across all node locations into which data is partitioned. Finally, the outputs of the first hidden layer of the network can

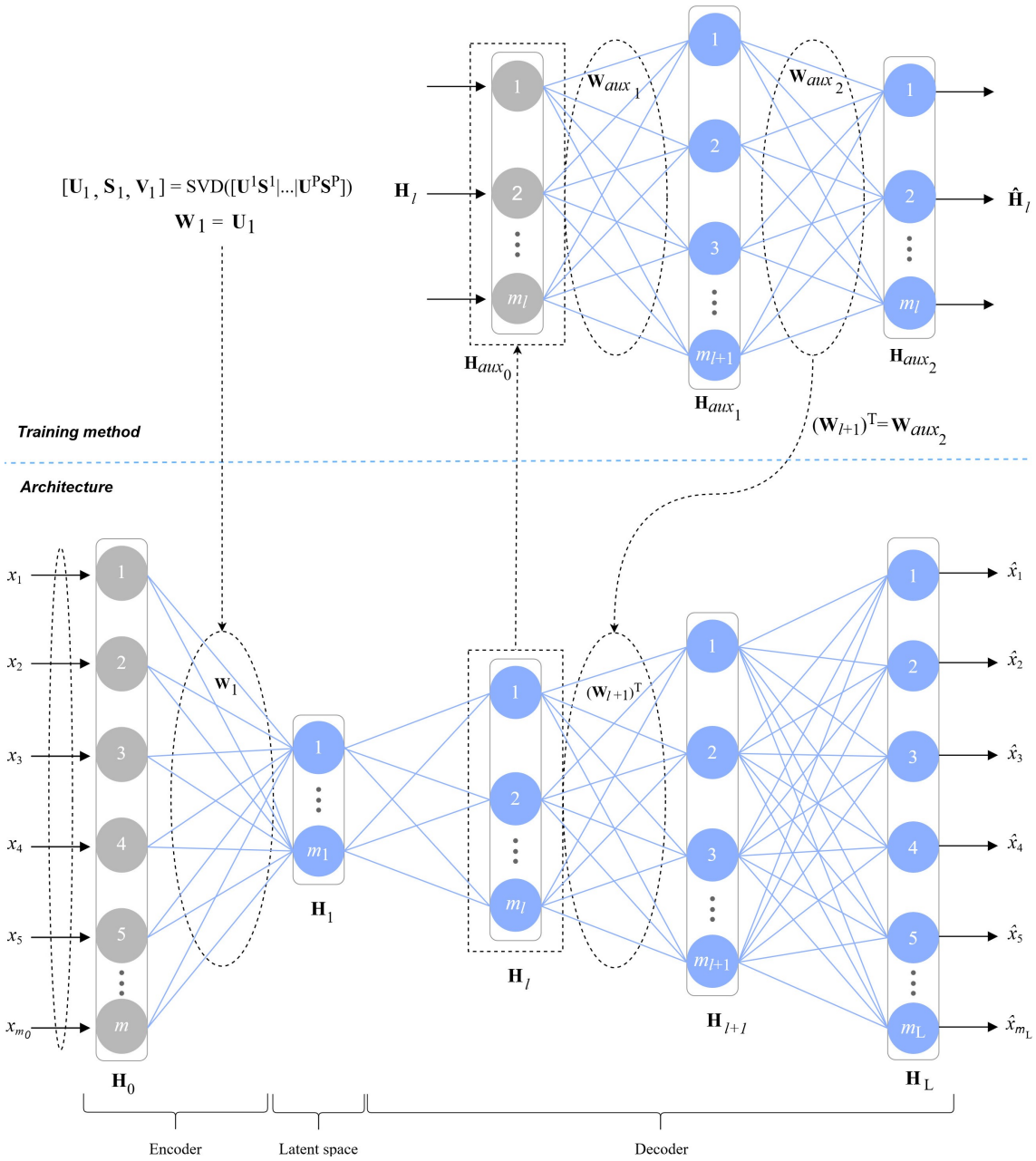


Figure 4.2: The asymmetric deep autoencoder DAEF.

be calculated, at each location p, as:

$$\mathbf{H}_1^p = f_1\left(\mathbf{W}_1^T\mathbf{X}^p\right); \forall p = 1, ..., P, \tag{4.9}$$

where f_1 is the activation function of the first hidden layer.

It has been decided to use a single-layer encoder, i.e., a single dimensionality reduction using SVD, as chaining several SVD processes sequentially and progressively (one per layer) did not show better performance. Since the encoder consists of a single layer and the decoder of several layers, the architecture of this autoencoder is asymmetrical, as can be seen in Figure 2.

4.3.2. The decoder

In the decoder, the goal is to reconstruct the input from the low-dimensional representation provided by the output of the first hidden layer $\mathbf{H}_1$ (see Equation 4.9). In order to be able to work with large datasets in a fast and efficient way, we propose to apply a non-iterative learning method to obtain the decoder parameters.

Similar to ELM-AE [90], DAEF employs an auxiliary network to determine the parameters of each layer of the decoder in an unsupervised way, layer by layer. In the DAEF decoder, the weights and bias of a given $(l+1)$th hidden layer will be calculated with an auxiliary network, as will be explained. Finally, the output matrix of $(l+1)$th layer ($\mathbf{H}_{l+1}$) can be obtained as:

$$\mathbf{H}_{l+1} = f_{l+1}\left(\mathbf{W}_{l+1}^T\mathbf{H}_l + \mathbf{b}_{l+1}\mathbf{1}^T\right) \tag{4.10}$$

being f_{l+1} the activation function, $\mathbf{H}_l$ the output matrix of the lth layer with m_l neurons, $\mathbf{W}_{l+1} \in \mathbb{R}^{m_l \times m_{l+1}}$ and $\mathbf{b}_{l+1} \in \mathbb{R}^{m_{l+1} \times 1}$ the estimated weight matrix and bias vector of the layer, respectively, and $\mathbf{1}$ a column vector of n ones.

The use of the already mentioned auxiliary network is shown in the top right of Figure 2, where $\mathbf{W}_{l+1}$ are obtained as the output weights of this network. As can be seen, the auxiliary network is a single-hidden layer sparse autoencoder. Regarding its structure, to calculate the parameters between the l and the $(l+1)$ hidden layers

of the principal network, the number of neurons in the input and the output layers of the auxiliary network will be identical to m_l, the number of neurons of its hidden layer will be m_{l+1}, and f_{l+1} will be used as activation function.

The training of this auxiliary network can be divided into two stages: the training of the first half of the network, in which the input of the network $\mathbf{H}_{aux_0}$ (which is the output $\mathbf{H}_l$ of the DAEF's principal network lth layer) is transformed at the hidden layer; and a second stage, in which, using the data $\mathbf{H}_{aux_1}$ coming from the hidden layer, the original input is reconstructed at the output $\mathbf{H}_{aux_2}$ of the auxiliary network.

The weights $\mathbf{W}_{aux_1}$ and the bias $\mathbf{b}_{aux_1}$ of the first stage are initialized and fixed (the different initialization schemes will be studied in section 4.6.2). Subsequently, the $\mathbf{H}_{aux_1}$ output of the hidden layer can be calculated as:

$$\mathbf{H}_{aux_1} = f\left(\mathbf{W}_{aux_1}^T \mathbf{H}_{aux_0} + \mathbf{b}_{aux_1}\mathbf{1}^T\right), \tag{4.11}$$

In the second stage, the weights $\mathbf{W}_{aux_2}$ are obtained in a supervised way using ROLANN (Algorithm 8). Considering that each output of the neural network depends solely on a set of independent weights, this second stage can be computed in parallel. Once this is calculated, the weights between the principal network's lth and $(l+1)$th layers can be obtained as $\mathbf{W}_{l+1} = \mathbf{W}_{aux_2}^T$, and the output $\mathbf{H}_{l+1}$ can be calculated using Equation 4.10.

This process will be repeated for each of the hidden layers of the decoder, using the outputs of one layer to calculate the weights of the next one, until reaching the last layer. Finally, as the output target values for the DAEF's last layer are known (the same as in the DAEF's input layer), the weights of the last layer can be calculated directly in a supervised and distributed way using ROLANN (Algorithm 8). We can summarize the DAEF training as follows:

1. Dimensionality reduction in the first layer using distributed SVD (encoder).
2. Unsupervised/supervised training, layer by layer, using an auxiliary network in which ROLANN is used as a regularization method (decoder).
3. Supervised training of the last layer using ROLANN (decoder).

4.3.3. Incremental, distributed and parallel learning

DAEF performs various operations that can be computed in a parallel way if the node (device) on which it is executed has several cores. These operations are the SVD computation of the encoder (the dataset can be divided and the partial SVDs concatenated and recalculated) and the ROLANN regularization processes in the decoder (the weights concerning the output layer can be calculated in parallel).

In addition to this, the trained DAEF models can be updated when new data arrives thanks to their incremental learning capacity. A node can add knowledge to its model without having to retrain from scratch, incorporating the new knowledge quickly and inexpensively. A DAEF network trained with a data partition can incorporate the knowledge obtained by a second DAEF network trained with a different partition if the latter shares the $\mathbf{U}_{m_1}$ matrices of its encoder [55], and the $\mathbf{M}_k$, $\mathbf{U}_k$, and $\mathbf{S}_k$ matrices of each layer of its decoder [53]. By adding this information, the first DAEF network can recalculate its weights and will have learned incrementally.

If we are faced with a multi-node environment, such as an IoT scenario, where each node has a partition of the global dataset, we can take advantage of the incremental capacity of the DAEF network to carry out a distributed training. Each node (device) would train a DAEF autoencoder network with its local data, and using a protocol such as MQTT [129], these nodes can publish their local model information through a broker to share their particular knowledge with the rest of the devices. The broker will be in charge of sending this information to the nodes that are subscribed to the updates, which will be able to aggregate the information received to their model.

We consider the local dataset of each node as a partition of a global dataset, so all the nodes must use a DAEF autoencoder network with a similar architecture. For the model information shared between nodes to be compatible with each other, the nodes must also use the same weights generated by the Xavier Glorot initialization scheme and the same bias. At the start of the training, one of the nodes must define the architecture, generate the weights and bias, and publish them through the broker. Figure 3 shows this scenario using the MQTT protocol.

The private data of each node will be protected since the information that is sent

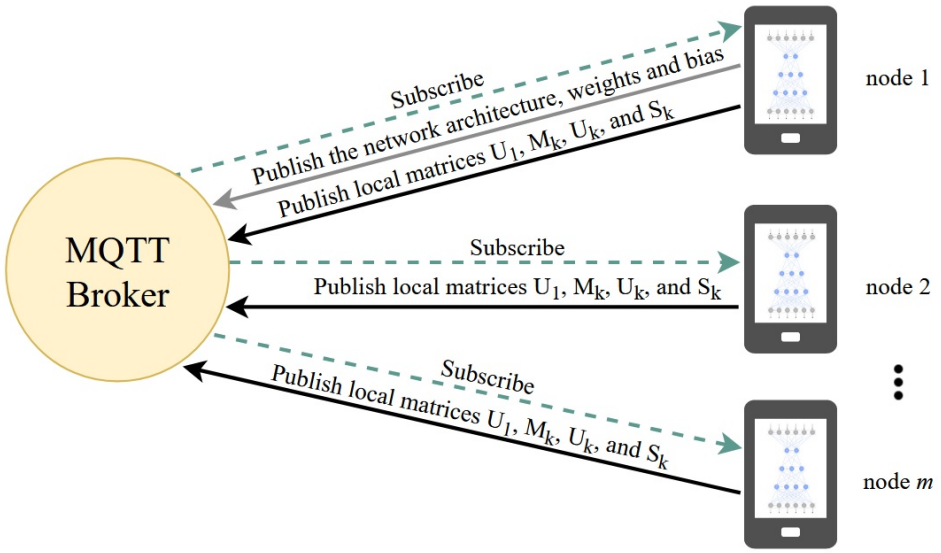


Figure 4.3: DAEF networks collaborating through an MQTT protocol.

through the broker is another. The data shared by each DAEF model (node) will be the $\mathbf{U}_{m_1}$ matrices of the encoder, and the $\mathbf{M}_k$, $\mathbf{U}_k$, and $\mathbf{S}_k$ matrices of each layer of the decoder, from which the original data are not recoverable [55, 53]. The DAEF network matrices mentioned above are the only information needed to perform the federated learning, so if desired, the original dataset of each node can be removed to save space. Storing these matrices is not a problem since their size is independent of the number of instances of the original dataset.

Note that DAEF could also be used in a centralized scenario in which the information from the local models would be sent to a central node, which would be in charge of aggregating the information, obtaining the global model and sharing it with the network nodes. A semi-centralized architecture for the use of DAEF in Edge Computing and Federated Learning scenarios will be studied in depth in Section 4.5.

4.3.4. Pseudocode

Algorithm 9 contains the pseudocode for the DAEF training phase for one node with t available cores. The processes carried out in the encoder are described between

lines 5 and 12. In line 7 the dimensionality of the data is reduced using SVD, in a parallel way splitting the local dataset into t partitions, obtaining the encoder weights and, in line 9, the encoder output. Between lines 13 and 20, the hidden layers of the decoder are trained one by one. For this, Algorithm 10 is used in line 15. In lines 21 and 22, the last layer of the decoder is trained directly using ROLANN also using parallelization.

Algorithm 10 contains the pseudocode of the auxiliary function used in algorithm 9 to train the different hidden layers of the decoder using an auxiliary autoencoder. In lines 2 and 3, the weights and bias are generated respectively, while in line 4 the output of the hidden layer is computed. Between lines 5 and 7, the decoder weights and the output are calculated using ROLANN. Since the weights concerning each neuron of the output layer are calculated independently, the t available cores will be used to perform these calculations in parallel.

Algorithm 11 contains the pseudocode for the DAEF prediction phase where the trained network will reconstruct a test sample. Comparing the input value and its reconstruction after passing through the network (reconstruction error), it could be classified as normal or anomalous. The complete implementation of the algorithm is available in GitHub [1].

4.4. Privacy treatment

In distributed environments (EC and FL), preserving the privacy of data at each node is a critical aspect, even more so if they contain sensitive information, such as personal data. Due to this, in this section, we will analyze the privacy preservation capacity of the DAEF method. To do so we will consider two main threat scenarios [133].

4.4.1. Preventing direct leakage

In classic environments, it is common for the original data from the nodes to be sent to other nodes or to a central server, for example, to be analyzed, pre-processed

[1] https://github.com/DavidNovoaP/DAEF

Algorithm 9 DAEF training phase

Input: $\mathbf{X} \in \mathbb{R}^{m_0 \times n}$, training dataset ($m_0$ variables $\times$ n samples); L, number of layers; a, list of number of neurons per layer; λ_{hid} and λ_{last}, regularization hyper-parameters of the hidden and last layers; f_{hid} and f_{last}, activation functions of the hidden and last layers; t, available cores.

Output: $Model$, trained model composed of the weights W_{list} and bias b_{list}, the training output $\mathbf{H}_L$, and the $matrix_{list}$ needed for incremental learning, i.e., $\mathbf{U}_1$ and $\mathbf{S}_1$ matrices of the encoder and the $\mathbf{M}_l$, $\mathbf{U}_l$, and $\mathbf{S}_l$ matrices of each layer of the decoder.

function DAEF_TRAIN
  $W_{list} = \varnothing$ ▷ Layer weight list
  $H_{list} = \varnothing$ ▷ Layer output list
  $matrices_{list} = \varnothing$ ▷ matrix list for incremental learning
  ▷ Learning the encoder
  $\mathbf{X}_{partitioned}$ = Split $\mathbf{X}$ in t partitions
  $\mathbf{U}_1, \mathbf{S}_1 = DSVD(\mathbf{X}_{partitioned}, a[1])$ ▷ a[1], latent space dimension
  $\mathbf{W}_1 = \mathbf{U}_1$ ▷ Weights of the *encoder*
  $\mathbf{H}_1 = f_{hid}((\mathbf{W}_1)^T\mathbf{X})$
  Append $\mathbf{W}_1$ to W_{list}
  Append $\mathbf{H}_1$ to H_{list}
  Append $[\mathbf{U}_1, \mathbf{S}_1]$ to $matrix_{list}$
  ▷ Learning the decoder
  for $l = 2..L-1$ **do** ▷ $L-1$, decoder hidden layers
    $\mathbf{W}_l, \mathbf{b}_l, \mathbf{H}_l, \mathbf{M}_l, \mathbf{U}_l, \mathbf{S}_l = TLD(H_{list}[l-1], a[l], \lambda_{hid}, f_{hid}, t)$ ▷ Call to Algorithm 2
    Append $\mathbf{W}_l$ to W_{list}
    Append $\mathbf{b}_l$ to b_{list}
    Append $\mathbf{H}_l$ to H_{list}
    Append $[\mathbf{M}_l, \mathbf{U}_l, \mathbf{S}_l]$ to $matrix_{list}$
  end for
  $pool$ =Pool(t) ▷ Pool of t processes
  $\mathbf{W}_L, \mathbf{b}_L, \mathbf{M}_L, \mathbf{U}_L, \mathbf{S}_L = pool.map(ROLANN, (H_{list}[L-1], \mathbf{X}, \lambda_{last}, []))$ ▷ Parallel training of the last layer with ROLANN
  $\mathbf{H}_L = f_{last}((\mathbf{W}_L)^T H_{list}[l-1])$
  Append $\mathbf{W}_L$ to W_{list}
  Append $\mathbf{b}_L$ to b_{list}
  Append $[\mathbf{M}_L, \mathbf{U}_L, \mathbf{S}_L]$ to $matrix_{list}$
  $Model = W_{list}, b_{list}, \mathbf{H}_L, matrix_{list}$
end function

Algorithm 10 Train one layer of the decoder (TLD)

Input: $\mathbf{H}_{l-1} \in \mathbb{R}^{m_l \times n}$ output of principal network's layer $l-1$ used as training data for this algorithm; m_l, number of neurons of the layer l; λ_{hid}, regularization hyperparameter of the hidden layer; f_{hid}, activation function of the layer; t, available cores.

Output: $\mathbf{W}_l$, weights of the layer l; $\mathbf{b}_l$ bias of the layer; $\mathbf{H}_l$, output of the layer l; and a list with the matrices needed for incremental learning, [**M**, **U**, **S**].

1: **function** TLD
2: $\mathbf{W}_{aux_1} = Xavier(m_{l-1}, m_l)$ ▷ Initial weights
3: $\mathbf{b}_{aux_1} = Random(m_l, 1)$ ▷ Initial bias
4: $\mathbf{H}_{aux_1} = f_{hid}(\mathbf{W}_{aux_1}^T \mathbf{H}_{l-1} + \mathbf{b}_{aux_1}\mathbf{1}^T)$
5: $pool$ =Pool(t) ▷ Pool of t processes
6: $\mathbf{W}_l, \mathbf{b}_l, \mathbf{M}_l, \mathbf{U}_l, \mathbf{S}_l = pool.map(ROLANN, (\mathbf{H}_{aux_1}, \mathbf{H}_{l-1}, \lambda_{hid}, []))$ ▷ Parallel training of the layer
7: $\mathbf{H}_l = f_{hid}(\mathbf{W}_l^T \mathbf{H}_{l-1} + \mathbf{b}_l\mathbf{1}^T)$
8: **end function**

Algorithm 11 DAEF prediction phase

Input: $\mathbf{X} \in \mathbb{R}^{m_0 \times n}$, test dataset ($m_0$ variables $\times$ n samples); W_{list}, weights of the trained network; b_{list}, bias of the trained network; f_{hid} and f_{last}, activation functions of the hidden and last layers; a, list of number of neurons per layer.

Output: *prediction*, reconstruction of the input **X** after passing through the network.

1: **function** DAEF_PREDICT
2: $\mathbf{H}_1 = f_{hid}((W_{list}[1])^T \mathbf{X})$
3: **for** $l = 2..length(W_{list}) - 1$ **do**
4: $\mathbf{H}_l = f_{hid}((W_{list}[l])^T \mathbf{H}_{l-1} + b_{list}[l-1]\mathbf{1}^T)$
5: **end for**
6: $prediction = f_{last}((W_{list}[l])^T \mathbf{H}_l + b_{list}[l]\mathbf{1}^T)$
7: **end function**

or to build a global machine learning model. This puts the privacy of the data at risk, which can be used maliciously and not to carry out the original tasks.

In the case of the DAEF method, the data shared for training the global model is not the original data ($\mathbf{X}$). Regarding the encoder block, each node p, using its local data ($\mathbf{X}^p$), computes an $SVD = \mathbf{USV}^p$ and the only information shared to carry out the federated learning is the product $\mathbf{U}^p\mathbf{S}^p$. Since the matrix $\mathbf{V}^p$ is neither calculated nor sent, the original data $\mathbf{X}^p$ cannot be retrieved through the factorization expression described in Equation 1. Regarding the decoder block, the federated learning is carried out by interchanging the $\mathbf{M}^p$, $\mathbf{U}^p$ and $\mathbf{S}^p$ matrices obtained through ROLANN regularization, so the original data is also kept safe. Once the global model is obtained, it is distributed to each of the local nodes p to be used privately, so there is also no direct data leakage in the operation phase.

4.4.2. Preventing indirect leakage

Another possible scenario is one in which a malicious node impersonates a real participant of the distributed learning protocol to try to obtain the private data of other nodes. Due to the nature, when we train models in a distributed way, it is common for nodes to share their calculations and parameters. Using this information, several authors have proposed specific methods that in certain cases can obtain the original training data, putting the privacy of the nodes at risk. The model inversion attack [216], once the network has been trained, follows the gradient used to adjust the weights of the network and obtains a reverse-engineered example for all represented classes in the model. Another way to address the problem is to train a Generative Adversarial Network (GAN) [74] in parallel to the attacked network using its gradients, so that it manages to extract information about classes of data that it does not know.

In the case of DAEF, the method is not iterative, so this type of attack is not a problem. The model parameters are calculated in a single step, so it is not possible to train GAN networks in this way. In addition, as we have seen previously, stochastic gradients are not shared (they are not used), so model inversion attacks are also not possible.

4.5. Proposed architecture implementation

In this work, we want to design an architecture for the use of the DAEF model in FL and EC scenarios, in which the training and aggregations are carried out in the edge devices without requiring large computational power, and exchange of data through the network is secure and preserves the privacy of local datasets. Model aggregation at the edge will also be managed by an edge node that assumes the role of coordinator and can be replaced at any time by another node, so although it cannot be considered a fully decentralized architecture, the level of dependence of the system on the central node is considerably reduced.

4.5.1. The MQTT communication protocol

Various lightweight protocols have been introduced for efficient communication between IoT devices [136]. One of the most used is Message Queuing Telemetry Transfer (MQTT) [203]. The MQTT protocol is a set of rules that defines how the devices can publish and subscribe to data over the Internet. The protocol is event-driven and connects devices using the publish-subscribe pattern. The sender (Publisher) and the receiver (Subscriber) communicate via Topics and are decoupled from each other. The connection between them is handled by the MQTT broker, which filters all incoming messages and correctly distributes them to the Subscribers. Due to its simplicity and lightness, in this work, MQTT will be used as the communication protocol between edge devices.

4.5.2. Definition of terms, roles, and assumptions

We define our scenario as a set of nodes (devices) of variable number, heterogeneous, with different computational power, and located at the edge of a network. Each node of the network gathers a private local dataset, with several instances that might not be the same. We can therefore consider each of these local datasets as a partition of a global one. In addition, over time, nodes could expand their local datasets with new data. The ultimate goal of the architecture is to achieve the training of a global DAEF model through the cooperation of the different nodes

(*workers*) that are part of the network. To do this, a node called *initializer* will send to each *worker* the DAEF model that will be trained by the *workers* with their local data. *Workers* who complete their local training may submit a request to the *coordinator* node to add their local knowledge to the global model. The *coordinator* will manage the requests of the *workers*, assigning turns so that they add their local model to the global one. The *coordinator* will manage the turns using a queue so that at any moment only one *worker* can be adding their local model to the global one. The *coordinator* will also be in charge of periodically sending updated versions of the global model to the *workers*. When all devices on the network have completed their training and all local models have been aggregated, the global training process is completed until the arrival of new devices or new data. Unlike other FL methods, the DAEF method does not require several rounds of learning on the edge devices, since it is a non-iterative method, and it is only necessary to add the local weights once. This implies greater speed, less network traffic, and lower energy consumption. Therefore, the roles that the nodes will take during the training are:

- *Initializer*: There will only be one initializer. Its task is to send to all the nodes of the network the configuration of the DAEF model that will be used during training. It is at the same time a worker with its local dataset.
- *Worker*: The worker's task is to train its own DAEF model using its local data and the initializer configuration and, after requesting a turn from the coordinator, add the information from its local model to the global one.
- *Coordinator*: There will be only one coordinator node. The coordinator's task is to manage the queue of FL requests from the worker nodes, store the state of the global model, and periodically forward it to the workers. The coordinator is at the same time a worker with its own local dataset.

As explained above, the network nodes use MQTT as the communication protocol. The publish-subscribe pattern decouples the node that sends a message (publisher) from the node or nodes that receive the messages (subscribers). MQTT uses the topic (subject) of the message to determine which message goes to which node (subscriber). Therefore a topic is a hierarchically structured string that can be used to filter and route messages. We define five main topics:

- *configuration*: Used by the initializer (publisher) to send to all the other nodes of the network (subscribers) the DAEF configuration that will be used during the training. The initializer will publish the number of layers, neurons per layer, initial weights, and bias of the model through the MQTT broker.
- *ask_turn*: Used by the coordinator (subscriber) to receive from the workers (publishers) their requests to add their local models to the global one.
- *order_to_train*: Used by the coordinator (publisher) to communicate to the workers (subscribers) which one has been chosen to perform the aggregation. The current global model is also sent.
- *local_update*: Used by workers (publishers) to send the updated global model to the coordinator (subscriber) after adding their local model.
- *send_global_model*: Used by the coordinator (publisher) to send the updated global model to all the workers (subscribers).

4.5.3. Architecture operation

The training process begins with the assignment of roles to the nodes or Edge Devices (ED) of the network. First, all nodes will assume the *worker* role. After this, a randomly selected node will also be designated as the *initializer* and a second node as the *coordinator*. These nodes will perform tasks in addition to those assigned to the *workers*. If at any time they are disconnected from the network, their role will be assigned to another randomly selected node of the network.

The *initializer* will create the *configuration* topic, to which all other nodes on the network will subscribe to receive the DAEF configuration to be used during training (see Figure 4.4). If in the future a new *worker* joins the network, it will also receive the DAEF configuration through the topic.

Once all the network nodes know the configuration, they start training their DAEF networks using their local data. When a node has completed its training, it can request permission from the *coordinator* to add its local knowledge to the global model. To do this, the *ask_turn* topic is used, a topic to which only the *coordinator* is subscribed, and through which the other nodes (publishers) ask for a

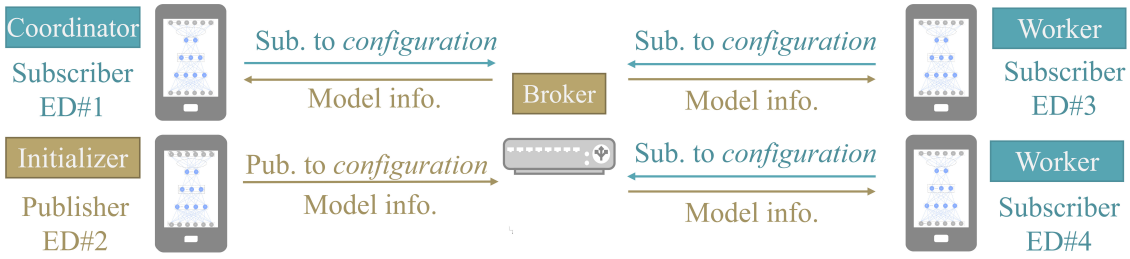


Figure 4.4: Initialization of the DAEF models using the *configuration* topic.

turn to perform the aggregations. Figure 4.5 shows this behavior. The *coordinator* manages requests in order of arrival using a queue.

The nodes that have requested to carry out the federated learning will subscribe to the *order_to_train* topic, through which the *coordinator* will publish which node of the queue has been designated to carry out the next aggregation. In addition to sending the identifier of the chosen node, the *coordinator* will send the necessary information about the current global model for the selected worker to aggregate its local model. Figure 4.6 shows this procedure.

The node designated to perform the FL will aggregate the information from its local model with the global one in an inexpensive way [147]. This new version of the global model will be sent back from this node (publisher) to the *coordinator* (subscriber) through the *local_update* topic (see Figure 4.7).

When the *coordinator* considers it necessary, for example, after a fixed number of aggregations, the current global model can be made available to the other nodes of the network through the topic *global_model* (see Figure 4.8). Although network devices have their local model trained on their local data, as the FL progresses,

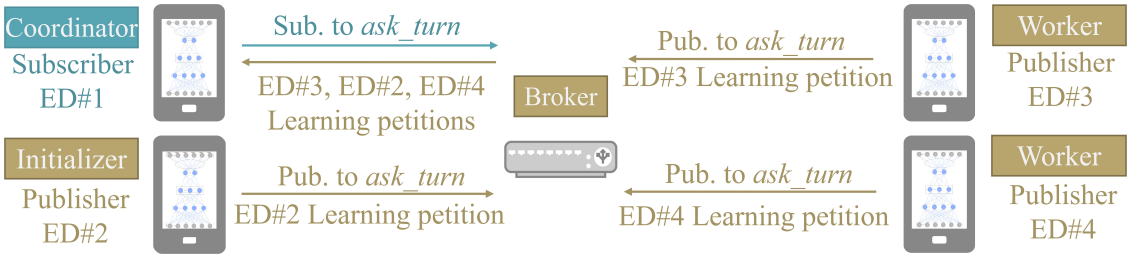


Figure 4.5: Workers (ED#3, ED#2, and ED#4) asking the coordinator for their turn to add their local model to the global one using the *ask_turn* topic.

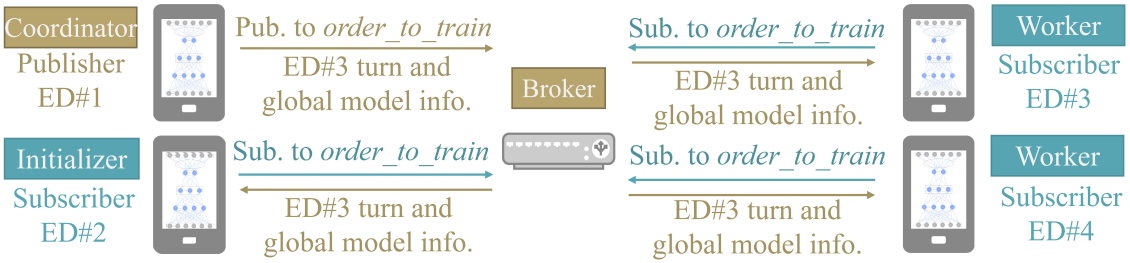


Figure 4.6: Coordinator giving the turn to worker ED#3 (first in the queue) and sending the current global model through *order_to_train* to perform the FL.

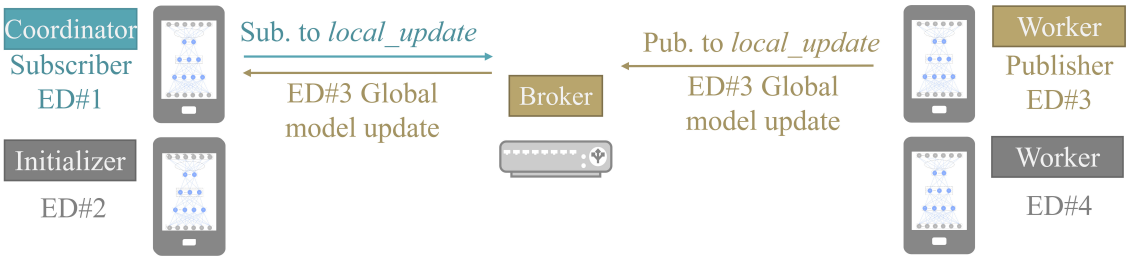


Figure 4.7: Sending the updated global model of the chosen worker (ED#3) after the federated learning to the coordinator through the *local_update* topic. Devices ED#2 and ED#4 could be publishers in the future after getting their turn.

devices will infer using the current global model for better results. Initial local models can be useful in the early stages where sufficient data is not yet available. We recommended to share the updated global model with the network after each device's aggregation to avoid problems when the coordinator node crashes, as will be seen in the next section.

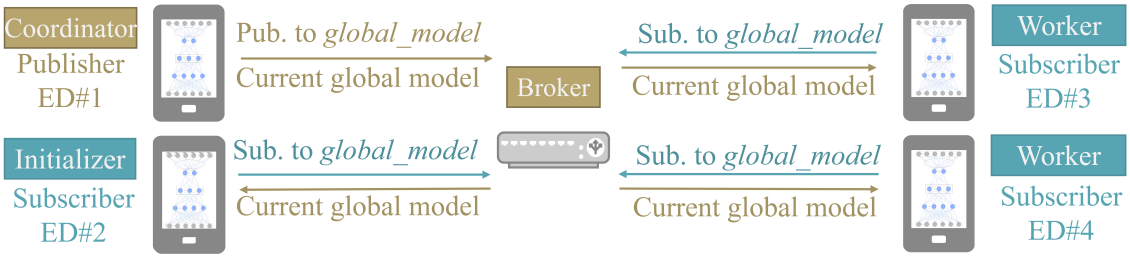


Figure 4.8: Coordinator sharing the current version of the global model with the nodes subscribed to *global_model*.

4.5.4. Fail-safe mechanisms

In EC and FL scenarios is reasonable to assume that some of the nodes will occasionally disconnect unexpectedly. This can occur due to loss of connection, empty batteries, or many other reasons. The MQTT protocol allows to differentiate between expected and unexpected disconnections, and therefore respond appropriately to each type. In an unexpected disconnection in MQTT, the Last Will and Testament (LWT) feature is used to notify other nodes about the disconnection. Each node can specify its LWT message when it connects to a broker. This is a normal MQTT message with a topic that is retained. The broker stores the message until it detects that the node has disconnected unexpectedly. In response to the unexpected disconnection, the broker sends the LWT message to all subscribed nodes of the last-will message topic. If the node performs an expected disconnection, the broker discards the stored LWT message.

There are several possible problems derived from the unexpected disconnection of the different devices that need to be taken into account:

- *Initializer*: The initializing function takes place every time a new node is to be added to the network, so all the nodes that are already part of the network have the DAEF model configuration. Therefore, reassigning this function to another randomly selected node does not create problems.

- *Worker*: The unexpected disconnection of a worker that is not in the process of adding its local knowledge to the global model does not cause problems, beyond the loss of the raw data that it captures locally. However, if the worker has requested to add its model or if is already adding it, an unexpected disconnection can be dangerous. On the one hand, if the worker has requested to add its model and its request is in the queue, it needs to be removed. For that, we propose the use of the worker's LWT message to inform the coordinator of the disconnection. In addition, if the worker was already adding its model when it disconnected, the coordinator could wait indefinitely for the results. To solve this problem, we propose two actions: (1) the definition of a maximum time to carry out the training, combined with (2) the use of the aforementioned worker's LWT message to inform the coordinator of the

disconnection. Defining a maximum timeout also prevents a malicious node from intentionally stopping training.

- *Coordinator*: The coordinator node stores both the training request queue and the current state of the global model. If the node goes down, a new node will be randomly designated as the coordinator. Since it is not convenient to share the queue of requests over the network, after the fall of the coordinator and the loss of the queue, the workers will have to request a turn again to add their local model to the global one, since each of them is aware of whether or not they were in the queue. For this, the new coordinator will notify through the topic *order_to_train* to which the workers can respond with requests through the topic *ask_turn*. With the failure of the coordinator, the current version of the global model is also lost. That is why in the previous section we recommended publishing the global model after each aggregation through the topic *send_global_model*, or at least sending it periodically. This will allow both its use by the nodes to infer, and continue to operate normally if the coordinator goes down (all nodes will have a copy of the global model).

4.6. Evaluation

Although autoencoder networks have several uses, the main task for which DAEF has been designed is anomaly detection. In this section, several experiments are presented to show its behavior in this kind of scenario.

4.6.1. Experimental setup

DAEF emerges as a fast alternative to perform anomaly detection in environments where time can be a critical factor, such as edge computing and federated learning scenarios. Iterative approaches achieve high accuracy in anomaly detection, but in certain cases, their long training times may make them unsuitable for these environments. This study aims to analyze the performance achieved by DAEF compared to the adaptation of Online Sequential Extreme Learning Machine (OS-ELM) for federated learning [83], another non-iterative approach that uses a single hidden

layer, as well as iterative single hidden layer autoencoders (SHL-AE) and iterative multiple hidden layer autoencoders (MHL-AE), models that follow a classical approach based on iterative learning. For this, a machine equipped with an Intel Core i7-11700k processor and 64GB of RAM has been used.

The algorithms were evaluated over seven datasets available in the UCI Machine Learning Repository [46], Kaggle [222] and ODDS [169]. Their characteristics are summarized in Table 1. All datasets were normalized using standard scalers with zero mean and unit variance. To assess the performance of each algorithm, the data were split using a 10-fold cross validation. The algorithms were trained using only normal data, while the test phase included data from both classes (50% normal and 50% anomalies).

Given an autoencoder network trained with normal data, the classification of new instances can be carried out by comparing their value at the network input with the values obtained at the output. The difference is known as the reconstruction error, and since anomalies are very rare in these scenarios, instances corresponding to the normal class will have a low reconstruction error, while anomalies will emit a much higher error. The reconstruction errors are calculated using the MSE (Mean Squared Error) and, after training the network, it is necessary to establish an error threshold above which input samples are classified as anomalies. There are many alternatives, a popular one being to define this threshold based on the interquartile range (IQR) of the reconstruction errors of the training examples, i.e.:

$$IQR = Q_3 - Q_1 \tag{4.12}$$

Table 4.1: Characteristics of the datasets used. All represent anomaly detection data, except Covertype whose class 4 was taken as anomalous.

Dataset	# Samples	Anomalies	Input dimension
Shuttle	49097	3511 (7.2%)	9
Covertype	581012	2747 (1.0%)	10
Pendigits	6870	156 (2.3%)	16
Cardio	1831	176 (9.6%)	21
Credit card	284807	492 (0.2%)	29
Ionosphere	351	126 (35.9%)	33
Optdigit	5216	64 (2.9%)	64

where Q_1 and Q_3 represent the first and the third quartiles. In this work, we define two error thresholds, one for outliers ($outlierIQR$) and another for extreme outliers ($extremeIQR$):

$$outlierIQR = Q_3 + 1.5 \times IQR \tag{4.13}$$

$$extremeIQR = Q_3 + 3 \times IQR \tag{4.14}$$

In addition, we also considered other thresholds using fixed percentiles (Q_{95}, Q_{90}, Q_{80}, Q_{70}, Q_{60} and Q_{50}).

Considering the anomalous class as the positive one, and based on the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), to measure the performance of the algorithms the F1-score metric was used (Equation 2.1).

Finally, the combinations of hyperparameters chosen for each algorithm, as well as the error thresholds, were selected using a grid search and are available in Appendix A.2.

4.6.2. Choosing the best initialization for DAEF

Firstly, the effects of using different types of initialization for the weights and biases of the DAEF network were studied. Table 4.2 shows the results obtained by the DAEF model using three types of initializations: orthogonal, random, and Xavier Glorot. In all cases, we have used sigmoid activation functions in the hidden layers and linear functions in the outputs since we want to reconstruct in the output any real input data to the network.

As can be seen, the performance of DAEF is very similar for the three types of initializations, with Xavier Glorot being slightly higher in some cases, such as the Covertype dataset. To validate this statement, statistical tests were carried out to compare the global performance of the three approaches. The chosen procedure has been Kruskal–Wallis and Tukey's HSD as a post hoc test [151, 137]. Using a

Table 4.2: Average test F1-score ± standard deviation for the different datasets.

Dataset	DAEF Ortho.	DAEF Random	DAEF Xavier
Shuttle	94.1±4.9	93.9±5.1	96.0±3.5
Covertype	81.4±6.3	80.1±9.7	86.0±4.4
Pendigits	70.2±9.3	70.6±12.1	76.3±5.5
Cardio	86.4±2.6	85.2±2.2	85.5±3.6
Credit card	89.1±1.8	87.6±2.3	90.0±0.6
Ionosphere	93.7±3.1	93.4±4.1	94.1±4.1
Optdigit	76.0±2.3	75.2±6.1	77.0±4.0

Table 4.3: Average training time (seconds) ± standard deviation for the different datasets.

Dataset	DAEF Ortho.	DAEF Random	DAEF Xavier
Shuttle	1.8±0.1	1.8±0.1	1.8±0.1
Covertype	4.2±0.0	4.3±0.0	4.2±0.0
Pendigits	1.8±0.0	1.8±0.0	1.8±0.0
Cardio	1.9±0.0	1.9±0.0	1.9±0.0
Credit card	36.3±0.4	36.4±0.3	36.3±0.3
Ionosphere	1.9±0.0	1.8±0.0	1.9±0.0
Optdigit	3.4±0.0	3.4±0.0	3.6±0.1

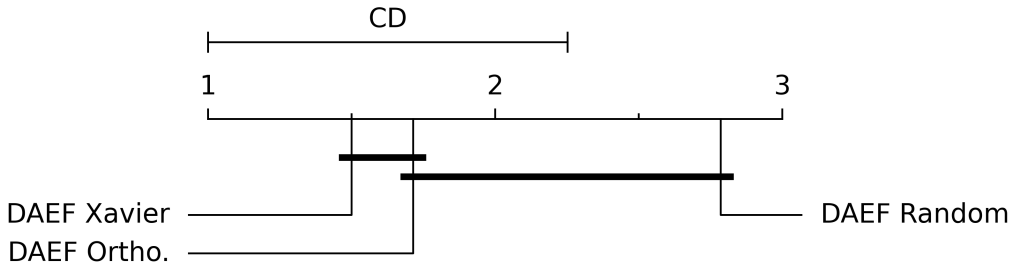


Figure 4.9: Graphical representation of Nemenyi test with $\alpha = 0.05$. The critical distance (CD) obtained was 1.25.

significance level of 5% in both cases and the F1-scores of the algorithms for the different datasets, the Xavier Glorot initialization, and the orthogonal initialization rank first, while random initialization is in a lower group, represented graphically by Figure 4.9. This allows us to affirm that the performance of the DAEF model is equivalent using Xavier Glorot and orthogonal initializations, these being slightly better than the totally random one. This could be related to the difficulty of sigmoid activation functions to deal with small random weights [58]

Table 4.3 shows the mean training time of each algorithm (lower values than 0.05 have been represented as 0.0). Test times have not been included in this work because they are very low for all the algorithms. As can be seen, the training times are very similar regardless of the initialization used. Due to this, and his slightly better performance, Xavier Glorot will be considered the default initialization for DAEF and will be used in the remaining tests.

4.6.3. DAEF vs. other anomaly detection algorithms

In this section, we compare the performance obtained by DAEF (Xavier Glorot initialization) and the algorithms mentioned in section 4.6.1 (OS-ELM, SHL-AE, and MHL-AE). Since the goal is to measure only their ability to detect anomalies, DAEF and OS-ELM were executed using a single node. Notice that running the OS-ELM adaptation for FL [83] using a single node is equivalent to using the original OS-ELM algorithm [114].

Table 4.4 summarizes the mean test results. Again, Kruskal–Wallis and Tukey's HSD tests (significance level of 5%) have been used for each dataset to highlight in bold the models that rank first. As can be seen, the DAEF algorithm achieves a good performance for most datasets, matching the performance of the best models in four of the seven datasets (Covertype, Cardio, Credit card, and Ionosphere), and getting a very close performance on datasets such as Shuttle. A Nemenyi statistical test [41, 57] was carried out to compare the global performance of the algorithms. Using a significance level of 5% and the F1-scores of the algorithms for the different datasets, the four methods rank in the same position, represented graphically by Figure 4.10.

Table 4.5 shows the mean training time of each algorithm. Because DAEF training is non-iterative, its training times are much shorter than those required by traditional iterative autoencoders (SHL-AE and MHL-AE). However, it fails to outperform OS-ELM which is extremely fast. This difference in training time is due, in part, to the fact that OS-ELM uses a single hidden layer, while DAEF uses several.

Finally, Tables 4.6 and 4.7 show an estimation of carbon dioxide emissions (CO_2-eq) and energy consumption (kWh) for the machine on which the tests were run [185]. As can be seen, in most cases the consumption and emissions of DAEF are much lower than those of the iterative autoencoders (SHL-AE and MHL-AE) although, as expected, higher than those of OS-ELM.

4.6.4. DAEF in federated learning scenarios

To test the behavior of DAEF in federated environments, several experiments were carried out. The objective was to study its performance, training time, and energy consumption as a function of the number of nodes. The results were compared with the reference method OS-ELM [83]. Given a simulated FL environment with n nodes with similar computational resources, and a dataset, it is divided evenly into n partitions of equal size. Each of these nodes trains a DAEF network with its local dataset partition, and then adds its local model to the global one. To ensure that the partitions are made up of a sufficiently large number of instances, the datasets chosen were the three datasets with the largest number of instances: Shuttle, Covertype, and Credit card. Training time was measured as the sum of the time needed by the

Table 4.4: Average test F1-score ± standard deviation for the different datasets. Best values according to the statistical test have been highlighted in bold.

Dataset	DAEF	OS-ELM	SHL-AE	MHL-AE
Shuttle	96.0±3.5	**97.9±0.3**	**98.0±1.1**	**98.4±1.0**
Covertype	**86.0±4.4**	**85.8±0.2**	72.0±4.9	**81.5±13.3**
Pendigits	76.3±5.5	**88.2±2.0**	**84.2±4.4**	**86.3±6.1**
Cardio	**85.5±3.6**	**88.1±1.3**	**87.8±1.7**	**87.2±2.7**
Credit card	**90.0±0.6**	**90.7±0.5**	**91.0±0.6**	89.0±1.6
Ionosphere	**94.1±4.1**	**96.7±3.2**	91.9±2.4	**94.0±2.3**
Optdigit	77.0±4.0	81.7±0.8	83.1±7.7	**88.9±3.7**

Table 4.5: Average training time (seconds) ± standard deviation for the different datasets. Best values according to the statistical test have been highlighted in bold.

Dataset	DAEF	OS-ELM	SHL-AE	MHL-AE
Shuttle	1.8±0.1	0.0±0.0	53.9±1.0	157.8 ±1.7
Covertype	4.2±0.0	0.2±0.0	157.1±1.3	239.6±2.3
Pendigits	1.8±0.0	0.0±0.0	19.6 ± 2.2	58.6±4.0
Cardio	1.9±0.0	0.0±0.0	11.9±0.1	2.4±0.1
Credit card	36.3±0.3	0.9±0.0	387.2±6.2	194.7±3.6
Ionosphere	1.9±0.0	0.0±0.0	4.0±0.1	2.59±0.2
Optdigit	3.6±0.1	0.0±0.0	3.9 ±0.4	13.04±0.4

Table 4.6: Estimated emissions (CO_2-eq) for the different datasets.

Dataset	DAEF	OS-ELM	SHL-AE	MHL-AE
Shuttle	1.4×10^{-6}	2.8×10^{-7}	7.9×10^{-4}	2.5×10^{-3}
Covertype	3.3×10^{-5}	3.0×10^{-6}	2.2×10^{-3}	3.7×10^{-3}
Pendigits	5.0×10^{-6}	2.3×10^{-7}	2.9×10^{-4}	8.5×10^{-4}
Cardio	5.3×10^{-6}	7.9×10^{-8}	1.7×10^{-4}	3.1×10^{-5}
Credit card	4.2×10^{-4}	1.2×10^{-5}	5.3×10^{-3}	2.9×10^{-3}
Ionosphere	4.4×10^{-6}	3.5×10^{-8}	5.1×10^{-5}	3.7×10^{-5}
Optdigit	6.9×10^{-5}	2.7×10^{-7}	5.6×10^{-5}	1.9×10^{-4}
Mean	7.7×10^{-5}	2.3×10^{-6}	1.3×10^{-3}	1.5×10^{-3}

Table 4.7: Estimated energy consumed (kWh) for the different datasets.

Dataset	DAEF	OS-ELM	SHL-AE	MHL-AE
Shuttle	1.7×10^{-6}	4.7×10^{-7}	1.3×10^{-3}	4.1×10^{-3}
Covertype	5.3×10^{-5}	5.0×10^{-6}	3.7×10^{-3}	6.2×10^{-3}
Pendigits	8.5×10^{-6}	3.9×10^{-7}	4.8×10^{-4}	1.4×10^{-3}
Cardio	8.9×10^{-6}	1.3×10^{-7}	2.8×10^{-4}	5.3×10^{-5}
Credit card	7.0×10^{-4}	2.0×10^{-5}	8.9×10^{-3}	4.9×10^{-3}
Ionosphere	7.4×10^{-6}	5.9×10^{-8}	8.6×10^{-5}	6.2×10^{-5}
Optdigit	1.1×10^{-4}	4.6×10^{-7}	9.4×10^{-5}	3.3×10^{-4}
Mean	1.3×10^{-4}	3.8×10^{-6}	2.1×10^{-3}	2.4×10^{-3}

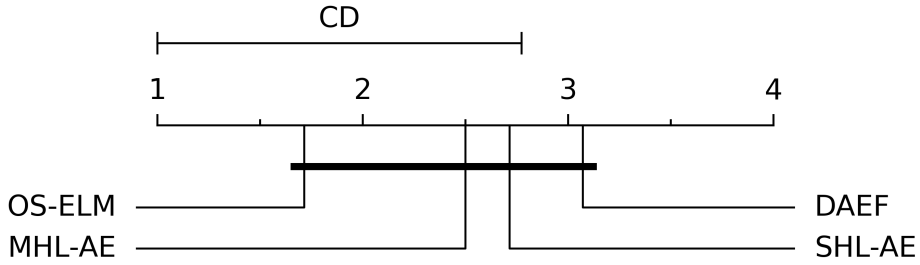


Figure 4.10: Graphical representation of Nemenyi test with $\alpha = 0.05$. The critical distance (CD) obtained was 1.77.

slowest node during local training and the time of all aggregations performed during incremental learning.

Figure 4.11 represents the average F1-score obtained using a different number of nodes (between 1 and 50). As can be seen, the performance of both models remains similar.

However, a common scenario in FL is one in which, either due to the existence of a very high number of nodes, or due to the scarcity of instances in the dataset, the number of instances per node is low. To test the DAEF and OS-ELM methods in this type of scenario, we used all the datasets mentioned in this work (see Table 4.1). For each of these datasets, the F1-score of both methods was measured based on the number of nodes used, in a similar way to the previous experiments, but using a sufficiently high number of nodes to cause the number of instances assigned to each node to be low ($\sim$50 instances per node). Taking this into account, we have observed that the performance of the DAEF model remains stable regardless of the number of data per node, while the OS-ELM method tends to generate bad results when the number of data goes below a certain value (depending on the dataset). Table 4.8 shows, for each dataset, the conditions from which the performance of OS-ELM starts to degrade significantly. Except the Covertype dataset, where none of the models degrades its performance, we can state that, in general, it is remarkable how OS-ELM presents difficulties to learn in a federated way when the number of instances per node is extremely low (between 14 and 60), while DAEF maintains its original performance.

Regarding the training times, Figure 4.12 contains the sum of the local training time and the aggregation of local models when using between 1 and 50 nodes. In

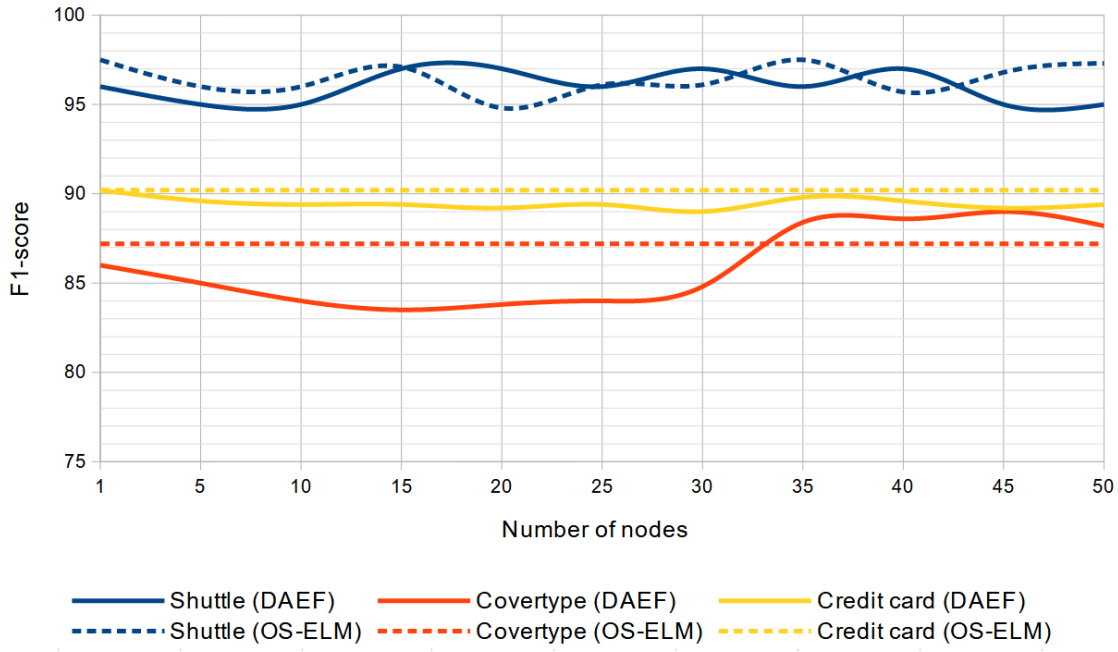


Figure 4.11: Average F1-score of the final global model as a function of the number of nodes used in a federated scenario with DAEF and OS-ELM.

Table 4.8: DAEF vs. OS-ELM (F1-score) when the amount of samples per node is extremely low. The Samples per node column shows the critical point at which the OS-ELM model exhibits a significant drop in performance. The DAEF and OS-ELM columns contain the F1-score reached at that point.

Dataset	Nodes	Samples per node	DAEF	OS-ELM
Shuttle	1000	∼50	96.1	86.2
Covertype	–	–	–	–
Pendigits	100	∼60	76.2	60.5
Cardio	100	∼14	86.1	74.1
Credit Card	8000	∼31	89.5	71.2
Ionosphere	3	∼67	85.7	53.5
Optdigit	100	∼46	75.5	77.9

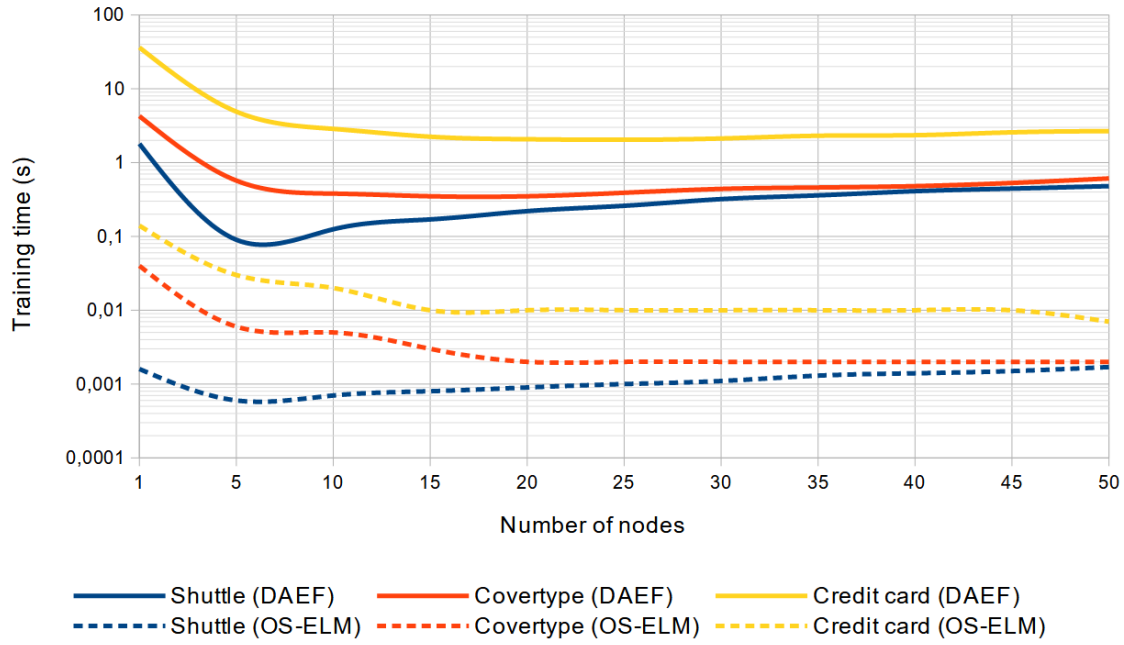


Figure 4.12: Average training time (seconds) required by DAEF and OS-ELM in a federated learning scenario based on the number of nodes used.

both methods, the time consumed using multiple nodes is smaller than the time required for a single node centrally trained on the entire dataset. This is because the partitioning of the dataset greatly accelerates local training, making most of the time required to complete training associated with the aggregation process. Even though the times of the federated version are better, we can observe a slight increase in training time as the number of nodes grows. This is due to the aggregation process performed by the coordinator that, although being a not very complex operation, scales linearly with the number of nodes used. Compared to the OS-ELM method, the latter is much faster, largely due to the use of a single hidden layer. Despite this, DAEF is fast enough for FL scenarios.

Finally, Figure 4.13 shows the estimated energy consumption (kWh) during the training taking into account all the nodes involved (again between 1 and 50 nodes). Regarding DAEF's behavior, in the case of Shuttle, since it is the smaller dataset, the aggregation of models is the process with the greatest weight in total consumption, which justifies its marked increasing trend with respect to the number of nodes. In the case of Covertype and Credit card, consumption shows slower growth in the

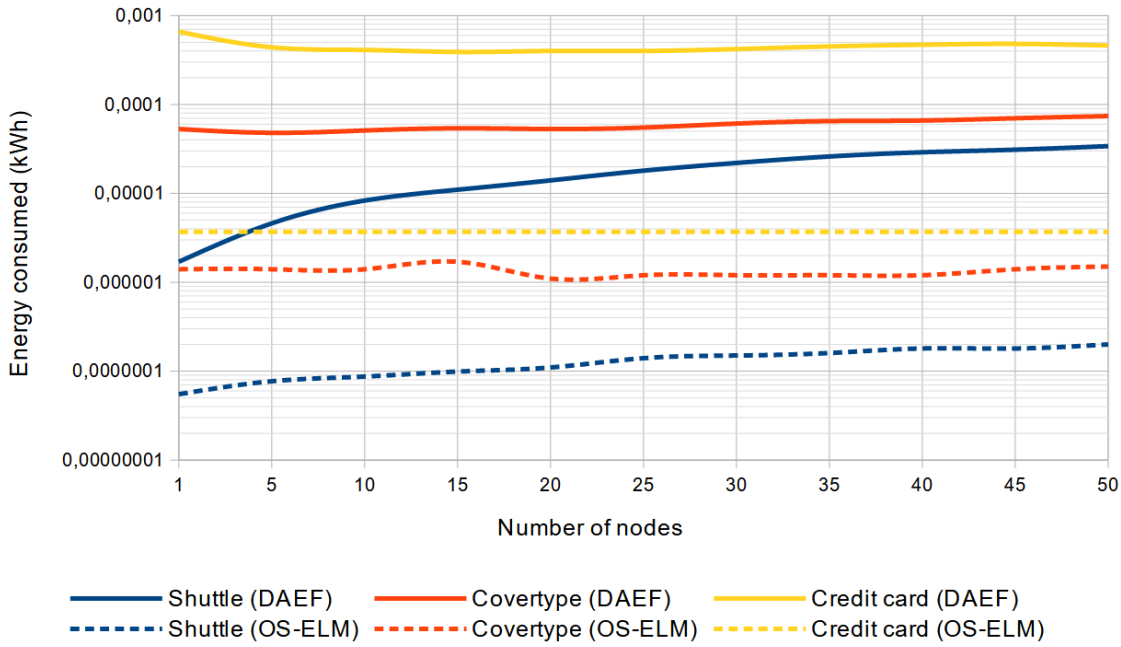


Figure 4.13: Estimated energy consumption (kWh) during the training of DAEF and OS-ELM in a federated scenario based as a function of the number of nodes.

initial stages because, for the same reasons, the computational cost of local training has a greater weight. Compared to the OS-ELM method, and in accordance with the training times already observed in Figure 4.12, it is natural that the energy consumption of DAEF is higher. Again, this does not prevent its use in FL scenarios as it is still a very competitive consumption.

4.6.5. DAEF vs. non-IID data

In horizontal federated learning scenarios, when the training data is not independent and identically distributed (Non-IID), the models usually perform worse than centralized alternatives [244]. To demonstrate that DAEF allows obtaining a global model equivalent to training with all the data in a centralized way also in Non-IID scenarios, a test has been carried out.

For this, the Covertype dataset has been used, which has seven different classes. Considering one of them as anomalous and the rest as normal, a non-IID scenario with no overlap between classes has been simulated. Simulating six nodes, each one

has been trained with the instances corresponding to one of the six classes considered normal. After this, the local models have been added, giving rise to a global model. The performance of this global model has been compared with the performance of a second model trained with all the normal data at once in a centralized way (IID). This test has been repeated ten times using a 10-fold. As can be seen in Table 4.9, the performance of both types of training was similar.

4.7. Conclusions

An alternative method to traditional deep autoencoder networks has been presented. Its distributed, parallel, and incremental learning capability, its low computational cost and its privacy preservation make it a valid solution for edge computing and federated learning environments. For these scenarios, this work has proposed a federated architecture to implement DAEF using MQTT as the communication protocol. It can be classified as semi-decentralized since, even though local and incremental learning is carried out in the edge nodes, it uses a coordinator node to manage aspects of the system such as the order of training or the sending of information through the broker. Fail-safe mechanisms are also proposed to combat the disconnection of the different devices, including the coordinator, and it has been empirically shown that the federated model is equivalent to training with all the datasets in a centralized way.

Moreover, although the strengths of the method lie in its usefulness in edge

Table 4.9: Average test F1-score ± standard deviation for the Covertype dataset considering each class as the anomalous and two types of training: IID centralized and Non-IID decentralized.

Anomalous class	IID Centralized	Non-IDD Decentralized
Class 1	75.0±3.1	75.2±2.6
Class 2	69.1±5.8	67.9±4.0
Class 3	91.5±3.2	90.3±3.6
Class 4	86.0±4.4	85.2±5.1
Class 5	82.4±3.9	81.8±4.6
Class 6	87.2±4.3	87.3±3.3
Class 7	79.06±5.1	80.2±2.3

computing and federated learning scenarios, it is also an interesting solution for centralized classical machine learning scenarios. To prove this claim, DAEF has been compared with traditional approaches as well as with OS-ELM, a very efficient federated autoencoder. In all cases, DAEF shows a fairly competitive performance, with training times and energy consumption much lower than traditional iterative approaches. Regarding OS-ELM, its efficiency is hardly surpassable, however, DAEF has shown better performance when the number of instances per node is low and also allows the use of deep architectures, which positions it as a good alternative for edge computing and federated learning scenarios that require more complex models, a context for which there are currently not many solutions.

As future work, it would be interesting to test the algorithm in real edge computing or federated learning environments using real physical devices that act as independent nodes, instead of simulating them. Another possible line of work would be to adapt the method to scenarios in which the distribution of the normal class changes over time (distribution shift), implementing, for example, a forgetting mechanism. It would be also interesting to try other communication protocols such as Secure MQTT [190]. Another possible line of work would be to redesign the architecture to be completely decentralized, thus dispensing with the coordinator and turning it into peer-to-peer, as well as the application of blockchain technology to preserve the integrity of the global model collaboratively [163].

Chapter 5

A proposal for anomaly detection in natural language scenarios

Nowadays more than ever in history, user opinions about products and services have a great impact on the future of the company that offers them. In such a globalized and highly competitive world, online review platforms, such as electronic commerce (e-commerce), play a crucial role in the credibility of products and services. These reviews usually come in the form of text reviews or numerical ratings made by users, accompanied in some cases by images or videos, and provide other users with information about the product or service they are considering purchasing, which directly influences the number of sales. Most people make purchase decisions based on ratings and reviews from other users [215].

In the case of many companies, such as Amazon, each product in the store has a list of text reviews published by customers of the platform. Users can access this list of reviews (opinions) to obtain extra information about the product, being able to mark the reviews as useful, which will position those reviews with the most votes at the top of the list. In addition to this, users can report to Amazon if they feel a review is inappropriate, for example, if its content is incorrect. This procedure to rank reviews based on their usefulness and report inappropriate reviews is carried out manually by platform users. As a result, Amazon reported more than 200 million suspected fake reviews in 2020 alone [7]. This problem does not only occur on Amazon but affects all platforms that allow their users to post reviews. For example,

Tripadvisor uses an automatic system capable of distinguishing between normal, suspicious, and inappropriate reviews [206]. Inappropriate ones are automatically removed (3.1% of review submissions in 2020), while those classified as suspicious are reviewed again by a human moderator (5.1% of review submissions in 2020).

In this chapter we propose a flexible and robust pipeline [148] that addresses review filtering as an anomaly detection problem in e-commerce platforms, allowing to classify reviews associated with a product as normal or anomalous. This allows to locate those that do not describe characteristics of the product to which they are associated, and therefore have no value for the users of the platform. In addition to this, each review classified by the system is associated with a normality score and an explanation that justifies its classification. The ability of the pipeline to solve the anomaly detection task was evaluated using different datasets derived from a large Amazon database [6]. In addition, to evaluate the explainability module, a study was conducted to compare three explainability techniques involving a total of 241 participants. The aim of this study is both to measure the impact of the explanations on the reproducibility of the classification model by the respondents, and the usefulness of these explanations in scenarios where subjectivity plays such an important role as the one posed in this work.

5.1. Related work

Anomaly detection has been a consolidated research field for years. Its great utility has allowed its techniques to be applied in numerous areas: medicine [186], industrial systems [207], electronic fraud [73], cybersecurity [79], etc. However, when we talk about texts and NLP, there is no massive application of these anomaly detection techniques as in the previous cases. This may be due to the difficulty in defining the idea of an anomaly in texts. Contrary to other scenarios, such as monitoring an industrial system through its sensors, in which the anomalous class will correspond to faults in the system, when the data is text, defining the concept of an anomaly is not trivial.

One of the most important lines of research related to the detection of anomalies in texts is fake reviews detection, also known as spam review detection, fake opinion

detection, and spam opinion detection [132]. The main problem associated with fake review detection is classifying the review as either fake or genuine. There are generally three types of fake reviews [85]:

- **Type 1 (untruthful opinions)**: Fake reviews describing users who post negative reviews to damage a product's reputation or post positive reviews to promote it. These reviews are called fake or deceptive reviews, and they are difficult to detect simply by reading, as real and fake reviews are similar to each other.
- **Type 2 (reviews on brands only)**: Those that do not comment on the products themselves, but talk about the brands, manufacturers, or sellers of the products. Although they can be useful, they are sometimes considered spam because they are not targeted at specific products.
- **Type 3 (non-reviews)**: Non-reviews that are irrelevant and offer no genuine opinion.

There are a wide variety of ways to distinguish between intentionally written reviews and fake ones in e-commerce scenarios. In the work carried out by J. Salminen *et al.* [181], the authors try to distinguish genuine reviews from fake reviews on Amazon. To have a labeled dataset of fake reviews, they use GPT-2 to artificially generate them. After this, they solve the task of distinguishing between genuine and fake reviews by fine-tuning a pretrained RoBERTa model. Once trained, it is shown that this model is also capable of detecting fake reviews manually written by humans. Ş. Öztürk Birim *et al.* [246] proposed, instead of directly handling the encoded text of the reviews, to use relevant information as the review length, purchase verification, sentiment score, or topic distribution as features to represent costumer reviews. Based on these features, well-known machine learning classifiers like random forests (RF) are applied for fake detection. In another approach, D. U. Vidanagama *et al.* [212] incorporate review-related features such as linguistic features, Part-of-Speech (POS) features, and sentiment analysis features using a domain ontology to detect fake reviews with a rule-based classifier.

If instead of focusing only on reviews we focus on detecting anomalies in texts in a more general way, we could find specific methods to solve this task such as

the one developed by L. Ruff *et al.* [178], who presented Context Vector Data Description (CVDD), a text-specific anomaly detection method that allows working with sequences of variable-length embeddings using self-attention mechanisms. To overcome the limitations of CVDD, J. Mu *et al.* [134] proposed tadnet, a textual anomaly detection network that uses an adversarial training strategy to detect anomalous texts in Social Internet of Things. In addition, thanks to the capture of the different semantic contexts of the texts, both models achieve interpretability and flexibility, allowing to detect which parts of the texts have caused the anomaly.

Other authors, instead of developing text-specific AD models, make use of well-known AD and NLP techniques to design architectures that solve the problem. B. Song *et al.* [191] propose to analyze the accident reports of a chemical processing plant to detect anomalous conditions, defined as unexperienced accidents that occur in unusual conditions. The authors work directly with the original text extracting the meaningful keywords of the reports using the term frequency-inverse document frequency (TF-IDF) index. Based on this, and using the local outlier factor (LOF) algorithm, they identify anomaly accidents in terms of local density clusters, finding four major types of anomaly accidents. Working with the original texts and not with embeddings they achieve a certain interpretability in the results. In another approach presented by S. Seo *et al.* [188], a framework is proposed to identify unusual but noteworthy customer responses and extracting significant words and phrases. The authors use Doc2Vec to vectorize customer responses to which LOF is applied to identify unusual responses and, based on a TF-IDF analysis and the distances in the embedding space, visualize useful information about the results through a network graph.

In some of the works described above, as well as in most of the published works, the detection of fake reviews focuses on detecting type 1 reviews, that is, reviews that positively or negatively describe a product but whose intention is not genuine since they do not come from a real buyer [181, 246, 212]. Other works seek to detect infrequent reviews to detect interesting aspects of their products and services [191, 188], while other approaches do not focus exactly on reviews but try to find anomalies in texts in a global way [178, 134]. In most of these works, moreover, explainability is not a major issue. Since there are several works based on these scenarios, in this work we focus on detecting type 2 and 3 fake reviews, which refer to reviews that

do not provide information about the product itself, but with a special emphasis on explainability. In this way, the objective of this work is to design a pipeline capable of distinguishing reviews related to a specific product (normal reviews) from reviews that do not and therefore do not provide information to the users that read them (anomalous reviews), for example, because they wrongly describe other products or because they are too generic. In addition, the classifications carried out by the system are explained through an analysis process based on NLP techniques.

5.2. Background

This section introduces the theoretical foundations as well as the ideas taken as the basis for the proposed pipeline.

5.2.1. The MPNet model

Although there are machine learning models capable of dealing directly with images or text, the majority of models are usually designed to be trained using numerical vectors that represent the sample data (tabular data). This input format of the data is the usual one for classic anomaly detection methods. In the NLP area, there are multiple techniques to represent texts using vectors of real numbers, which are known as embeddings. These techniques allow the generation of vector spaces that try to represent the relationships and semantic similarities of the language, so that, for example, two synonymous words will be found at a shorter distance in the vector space than two unrelated words.

Embeddings can be calculated independently for each word of the language (word embeddings), which led to models such as word2vec [128] or Global Vectors (GloVe) [158]. The representation of a sentence (sentence embedding) or a document (document embedding) will therefore be the sum of all the individual representations of the terms that make it up. To obtain representations of fixed length, it is usual to perform operations such as the mean. In certain cases, these operations between embeddings can worsen or even invalidate the final embedding, so specific models have been developed capable of understanding and representing a text as a whole, instead

of just encoding it word by word. Among these models are those based on transformers [210], such as BERT [42], XLNet [228], GPT-3 [25] or GPT-4 [150], which have been trained over large-scale datasets and can solve different tasks, including sentence embedding.

Since it inherits the advantages of the BERT and XLNet models while overcoming their limitations, in this work we used the pretrained MPNet [192] model to calculate the embeddings of the reviews. MPNet combines Masked Language Modeling (MLM) and Permuted Language Modeling (PLM) to predict token dependencies, using auxiliary position information as input to enable the model to view a complete sentence and reduce position differences. MPNet maps sentences and paragraphs to a 768 dimensional dense vector space $\mathbf{e} \in \mathbb{R}^{768\times1}$, providing fast and quality encodings. Computing time is a critical aspect in the area in which this work is framed, since in e-commerce platforms (and online reviews in general) we can find a huge number of products and reviews to deal with. Models like GPT-3 or GPT-4 are more advanced, but in addition to not being open source, they demand much higher computational resources, which may be unaffordable or excessive.

5.2.2. The DAEF network

As we have already seen, anomaly detection is a field with a large number of algorithms that solve the problem of distinguishing between normal and anomalous instances in very diverse ways [29, 93]. Among the reconstruction-based methods are autoencoder (AE) networks [214], one of the most widely used models. As we discussed in the previous chapter (Chapter 4), AE is a type of autoassociative neural network whose output layer seeks to reproduce the data presented to the input layer after going through a dimensional compression phase. In this way, they manage to obtain a representation of the input data in a space with a lower dimension than the original, learning a compact representation of the data, retaining the important information, and compressing the redundant one.

One of the objectives of our pipeline is to generate a normality score for each review. This score allows us, among other things, to order the different reviews by their level of normality. When AE networks are used in anomaly detection scenarios, the classification is usually carried out based on the reconstruction error that they

emit to reproduce in its output the embeddings of the reviews it receives as inputs, which represents the level of normality of the evaluated instance. This reconstruction error can be used as the normality score we are looking for. Due to its flexibility and quickness of its training, we have decided to use DAEF (Deep AutoEncoder for Federated learning) [147] as our anomaly detection model. Therefore, the proposed pipeline uses DAEF to classify the embeddings of the reviews, being able to issue a normality score associated with each of the review classifications.

5.3. The proposed pipeline

The purpose of the proposed pipeline is, given the text reviews of an item, to classify them as normal if they refer to it, and as anomalous if they describe different products or if they are so generic that they do not provide useful information to consumers. These classifications will be accompanied by a normality score and explanations that justify their classification as normal or anomalous. Figure 5.1 shows the three modules that are part of the proposed pipeline to solve this task: (1) Text encoding; (2) Anomaly detection; (3) Explainability. In the first phase, the text reviews of the target product are encoded using a pretrained MPNet transformer [192]. In the second phase, a DAEF autoencoder [147] is trained using these embeddings to learn the anomaly detection task. Using the reconstruction errors issued by the network and a predefined threshold error, the model can classify reviews as normal (error greater than the threshold) or anomalous (error less than the threshold). For the last phase, we propose a method based on the most frequent normal terms to generate an explanation associated with each classification.

In the following, the architecture modules and operation flow are described. To illustrate this process, we will use a concrete example where the product considered

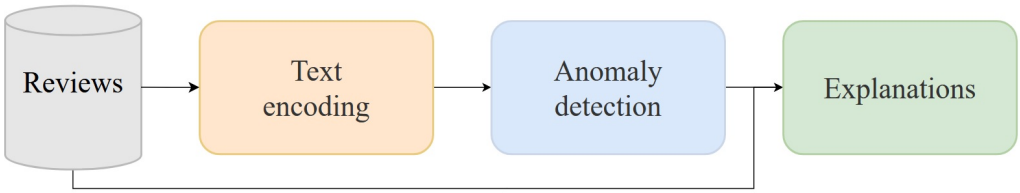


Figure 5.1: General modules that form the proposed pipeline.

as normal corresponds to "Kind chocolate bars" (see Figure 5.2).

5.3.1. Text encoding

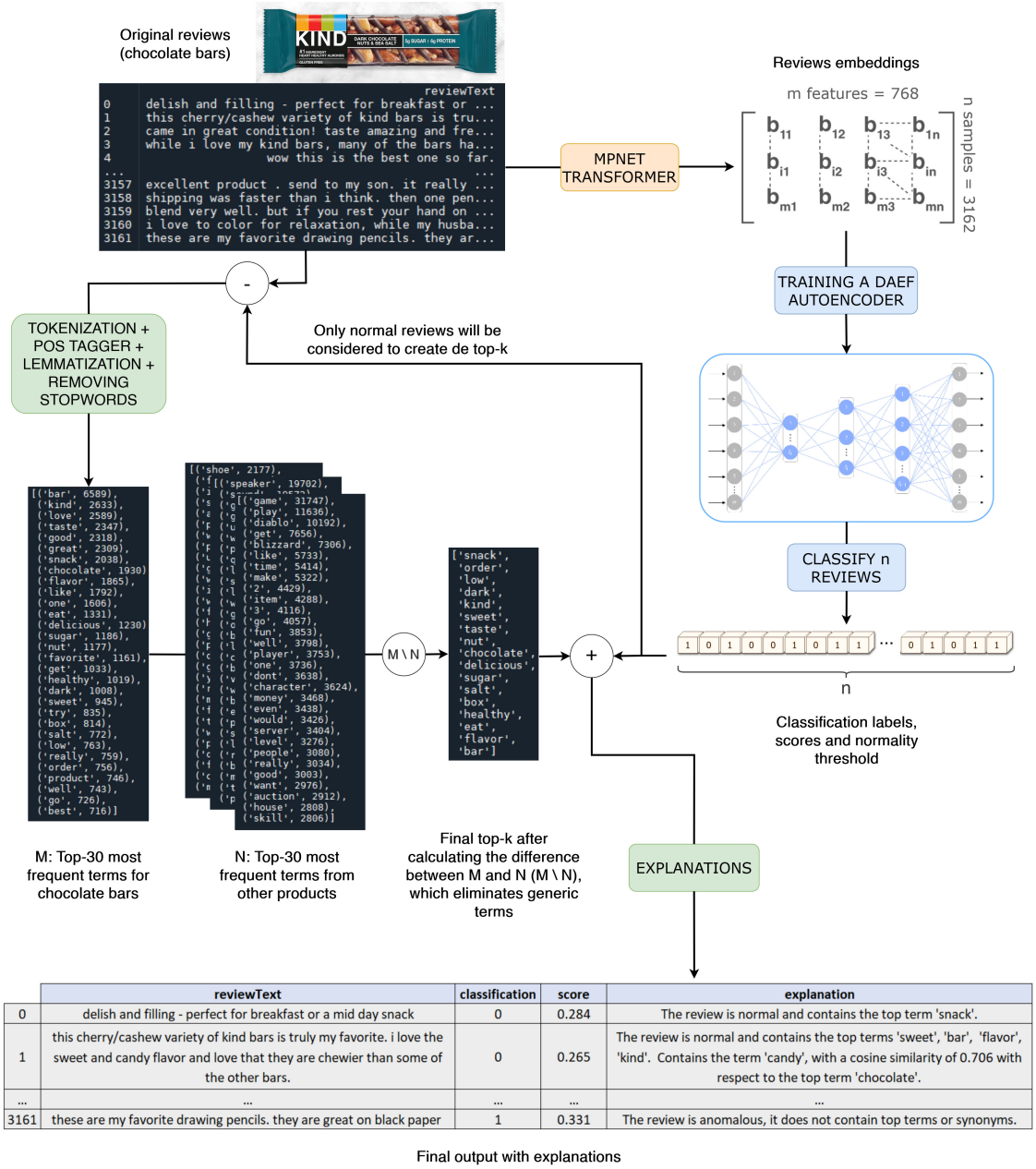
The pipeline starts from the list of reviews associated with the product considered as normal, in our example scenario the product "Kind chocolate bars". Given the set of reviews in text format, the objective of the first module is the representation of these reviews using the MPNet transformer model. As discussed in the previous section, MPNet allows mapping sentences and paragraphs into a 768 dimensional dense vector space to be used for different downstream tasks, such as information retrieval, clustering, or sentence similarity. In our case, we use them to perform the anomaly detection task.

In order to carry out this transformation of the textual reviews, we propose the use of the Hugging Face library [78]. By default, MPNet allows working with texts up to 384 words or tokens. If the input texts contain more than 384 words they will be truncated, although this is not a problem in the context of e-commerce, as most reviews are smaller than this limit. Furthermore, if the environment in which the embeddings are computed has a CUDA-compatible device such as a GPU, this operation will be significantly accelerated.

5.3.2. Anomaly detection

Once the embeddings of the target product reviews are collected, the anomaly detection model is trained. In this work we propose the use of the DAEF autoencoder network [147]. As with any other autoencoder, it is based on the assumption that all, or at least most of the training samples belong to the normal class, even though they are not labeled.

After the network training, for each input example, the network issues a reconstruction error at the output. In this work, reconstruction errors are calculated using Mean Squared Error (MSE). This error quantifies the normality of the review as it passes through the network. The most normal reviews will emit lower reconstruction errors, while the most anomalous reviews will produce the highest values. By



	reviewText	classification	score	explanation
0	delish and filling - perfect for breakfast or a mid day snack	0	0.284	The review is normal and contains the top term 'snack'.
1	this cherry/cashew variety of kind bars is truly my favorite. i love the sweet and candy flavor and love that they are chewier than some of the other bars.	0	0.265	The review is normal and contains the top terms 'sweet', 'bar', 'flavor', 'kind'. Contains the term 'candy', with a cosine similarity of 0.706 with respect to the top term 'chocolate'.
...	...	...	...	...
3161	these are my favorite drawing pencils. they are great on black paper	1	0.331	The review is anomalous, it does not contain top terms or synonyms.

Final output with explanations

Figure 5.2: Proposed pipeline considering the product “Kind chocolate bars” as the normal class.

setting a threshold error we can classify the reviews as normal or anomalous. The way to define this threshold error is very varied. If we know about the percentage of anomalies in the training dataset, we can use various percentiles to calculate it. If we do not have any information, we can use automatic techniques such as the interquartile range (IQR) of the reconstruction errors of the training examples, as explained in the previous chapter (Section 4.6.1). In any case, it is recommendable to experiment with different values to adapt the behaviour of the system to our needs.

Using this anomaly detection module for every review its classification as normal (0) or anomalous (1) will be available together with the corresponding reconstruction error.

5.3.3. Explanations

Many explainability techniques base their operation on determining which characteristics of the dataset have most influenced the predictions. For example, in industrial scenarios, it is common for the features of the datasets to come directly from the physical aspects measured by the sensors of the machines, giving rise to variables such as temperatures, pressures, or vibrations. By quantifying the influence of each of these variables on the output of the system we can achieve very useful explanations.

However, in the scenario proposed in this work, the data received by the anomaly detection model as input are the reviews' embeddings, these being numerical vectors. The variables that make up the embeddings are not associated with aspects understandable by human beings such as temperatures or pressures, so determining which ones have influenced the most would not provide us with useful information.

To overcome this problem, in this work we propose an approach based on a statistical analysis of the product reviews. After the classification of the training reviews, the original list is filtered by removing those classified as anomalous by the model. These normal reviews are the ones used to elaborate the explanations of the system. Based on the definitions of normal and anomalous review presented in Section 5.1, our hypothesis assumes that normal reviews always refer directly

or indirectly to the target product so that there will be a list of terms used very frequently among normal texts. The appearance of one or more of these "normal" terms in a review would justify its classification by the system as normal. In the same way, anomalous reviews may be justified with the non-presence of said terms.

To determine which are these normal terms, their frequency is analyzed. To do this, the texts of the reviews are processed using different NLP techniques including tokenization, post tagging, lemmatization, and stop word removal. This task is carried out using the NLTK library [18]. From these processed text reviews classified as normal, we can calculate the list of most frequent terms. For our specific example (see Figure 5.2), we can observe the M ranking of the 30 most frequent terms and their number of occurrences. As we can see, among the terms referring to the normal product there are also generic terms that could be common in products of all types, such as *love*, *good*, or *like*. To eliminate these generic terms, we also built the lists (N) of the most frequent terms for other items belonging to different categories from that of the item being explained (e.g., "fashion", "electronics", "video games", etc). The terms present in those lists are subtracted from the target ranking (M), giving rise to the final list of frequent terms specific to our normal product.

Employing this final term list and the classifications made by the anomaly detection model the explanations are generated. For each text review, if the review is classified as normal, it is checked if it contains any of the frequent terms from the list or any possible semantically related term, such as a synonym. The cosine similarity calculated by MPNet is used to calculate the similarity between these terms. If the review to be processed was classified as anomalous, it is explained as the non-appearance of normal terms.

Using the classifications and reconstruction errors output by the model, as well as the resulting explanations, the final output of the system is constructed (see Figure 5.2).

5.4. Evaluation

In this section, several experiments are presented to show the behavior of the proposed pipeline in real scenarios. The section is divided into two parts: the evaluation

of the anomaly detection task (Section 5.4.1), and the evaluation of explanations (Section 5.4.2). In the first one, the capability of the pipeline to detect anomalous reviews is evaluated using products from the Amazon platform. The second part discusses the problems derived from evaluating explainability techniques in this same scenario. In addition, a human study is presented to evaluate the benefits of adopting such explanations, both those generated by the explainability technique proposed in this work and other state-of-the-art techniques.

5.4.1. Evaluating the anomaly detection task

The anomaly detection task that solves the proposed pipeline can be evaluated following the usual methodology in the field of anomaly detection. In this first part of the evaluation we will describe the test scenario used, the methodology employed and, finally, we will discuss the results obtained.

5.4.1.1. Experimental setup

The objective of this study is to evaluate the capacity of the pipeline in a real anomaly detection scenario. Although in Section 5.3.2 we propose the use of DAEF as the anomaly detection method, different alternatives were compared. Specifically, a second non-iterative implementation of autoencoder networks, two boundary-based methods and a density-based method were employed. These methods are Online Sequential Extreme Learning Machine (OS-ELM) [114], One-Class Support Vector Machine (OC-SVM)[218], Isolation Forest (IF)[118], and Local Outlier Factor (LOF) [22] respectively.

For the evaluation, we employed datasets obtained from the large Amazon database [6], which collects reviews of various products from the years 1996 to 2018. The main problem with this dataset is that the reviews that compose it are not labeled as normal or anomalous. Because of this, to simulate a scenario similar to the one described throughout the work, we have decided to select the reviews corresponding to several of the most demanded products. Thus, for a given test we could consider the reviews of one product as normal, and introduce reviews from other products as anomalous. Seven different product categories were selected, for which the two

products with the highest number of reviews were used, resulting in a total of 14 products. Table 5.1 summarizes its characteristics. In all cases, MPNet was used as the model to encode the text reviews.

Besides, we considered two types of tests based on the products used:

- **1vs.6 - Far products**: For each of the seven categories, in this type of tests the product with the most reviews from one of the categories has been considered as the normal class, while the product with the highest number of reviews from each of the other six categories was considered the anomalous class. The fact that the product considered normal belongs to a different category should facilitate its distinction.
- **1vs.1 - Near products**: For each of the seven categories, in these tests the two products with the most reviews within the same category have been selected. One is considered the normal class and the other the abnormal one. The fact that both products belong to the same category should make it more difficult to distinguish them since they may have common characteristics.

The anomaly detection algorithms were trained using only normal data (the product considered as normal), while the test phase included data from both classes in a balanced manner (50% normal and 50% anomalies).

To evaluate the performance of each algorithm with each combination of hyperparameters a 10-fold was used. The normal data were divided into 10-folds so that, at each training run, 9 folds of normal data were used for training, while the remaining fold and the anomaly set were used together in a balanced manner for testing.

In this work, reconstruction errors at the output of the anomaly detector were calculated using Mean Squared Error (MSE). To establish the threshold above which this error indicates a given instance corresponds to an anomaly, among the various methods available, we employed the interquartile range (IQR) of the reconstruction errors of the training examples. In addition, throughout the tests, we also tested other thresholds using fixed percentiles (Q_{95}, Q_{90}, Q_{80}, Q_{70}, Q_{60} and Q_{50}), since *a priori* it is not easy to figure out which one can provide the best results, so it is considered as an additional hyperparameter to be taken into account in the

Table 5.1: Characteristics of the products used.

Product	Category	Reviews
Chocolate bars	Grocery and Gourmet Food	11526
Anise seeds	Grocery and Gourmet Food	9083
Colored pencils	Office Products	14340
Ergonomic cushion	Office Products	11942
Gaming mouse	Video Games	6462
PS4 membership	Video Games	5135
Bluetooth speaker	Electronics	28539
Wi-Fi range extender	Electronics	20873
Foot insoles	Amazon Fashion	4384
Yoga leggings	Amazon Fashion	3889
Hamilton album	Amazon Music	3411
Partners album	Amazon Music	3243
Hygrometer	Industry and Scientific	14331
Vacuum	Industry and Scientific	12182

case of DAEF and OS-ELM autoencoders. The OC-SVM method constitutes an exception as it automatically assigns a score to each input instance and decides its classification based on an internally calculated value. In the case of IF and LOF, the parameter contamination is used to define the threshold, so it was also considered as an additional hyperparameter.

To measure the performance of the algorithms the F1-score metric was used (Equation 2.1). Finally, the combinations of hyperparameters chosen for each algorithm, as well as the error thresholds, were selected using a grid search and are available in Appendix A (see Tables A.3 and A.4).

All the evaluation tests were performed in a machine equipped with an Intel Core i7-11700k processor and 64GB of RAM.

5.4.1.2. Evaluating the anomaly detection task

Table 5.2 shows the results of the 1vs.6 - Far products experimentation. Statistical tests were carried out to compare the performance of the five approaches. A Kruskal–Wallis and Tukey's HSD test [151, 137] with a significance level of 5% was used for each dataset to highlight in bold the models that rank first. As can be seen, the performance of the anomaly detection algorithms in general is very close. OS-ELM ranks as the best model for five of the seven datasets, while DAEF

and LOF each rank first twice. A Nemenyi statistical test [41, 57] was carried out to compare the global performance of the algorithms. Using a significance level of 5% and the F1-scores of the algorithms for the different datasets, the five methods rank in the same position, represented graphically by Figure 5.3. In light of the results, we can affirm that the embeddings produced by the MPNet model provide an encoding with sufficient quality for the anomaly detection models to be able to differentiate between the evaluated products.

Table 5.2: Average test F1-score ± standard deviation for the 1vs.6 datasets. Best values according to the statistical test have been highlighted in bold.

Normal class	Anomalous class	DAEF	OS-ELM	OC-SVM	IF	LOF
Chocolate bars	[Colored pencils, Gaming mouse, Bluetooth speaker, Foot insoles, Hamilton album, Hygrometer]	93.3±0.9	96.8±0.1	95.4±0.1	97.2±0.1	**97.6±0.1**
Colored pencils	[Chocolate bars, Gaming mouse, Bluetooth speaker, Foot insoles, Hamilton album, Hygrometer]	**96.2±0.5**	**96.1±0.2**	94.5±0.1	94.7±0.2	95.9±0.1
Gaming mouse	[Chocolate bars, Colored pencils, Bluetooth speaker, Foot insoles, Hamilton album, Hygrometer]	93.2±0.7	**95.0±0.1**	93.3±0.2	93.9±0.2	94.4±0.1
Bluetooth speaker	[Chocolate bars, Colored pencils, Gaming mouse, Foot insoles, Hamilton album, Hygrometer]	**94.9±0.6**	**94.8±0.2**	93.9±0.1	93.8±0.2	94.4±0.2
Foot insoles	[Chocolate bars, Colored pencils, Gaming mouse, Bluetooth speaker, Hamilton album, Hygrometer]	96.4±1.1	96.5±0.1	95.2±0.05	93.9±0.4	**96.9±0.1**
Hamilton album	[Chocolate bars, Colored pencils, Gaming mouse, Bluetooth speaker, Foot insoles, Hygrometer]	94.3±0.7	**95.6±0.2**	94.6±0.2	93.8±0.2	93.9±0.1
Hygrometer	[Chocolate bars, Colored pencils, Gaming mouse, Bluetooth speaker, Foot insoles, Hamilton album]	92.7±1.0	**94.4±0.1**	92.5±0.1	92.6±0.5	92.9±0.1

Table 5.3 collects the results of test 1vs.1 - Near products. Again, a Kruskal–Wallis and Tukey's HSD test [151, 137] with a significance level of 5% have been used for each dataset to highlight in bold the models that rank first. In this case, for some products such as the two music albums, the differences between the performance of the algorithms are more remarkable. Once again, OS-ELM continues to rank first for most times (6/14), followed in this case by LOF (5/14), DAEF (3/14), IF (1/14), and OC-SVM (0/14). A Nemenyi statistical test [41, 57] was carried out to compare the global performance of the algorithms. Using a significance level of 5% and the F1-scores of the algorithms for the different datasets, OS-ELM, LOF,

DAEF, and OC-SVM are placed in the first position, represented graphically by Figure 5.3, while IF is in a lower rank. The overall performance of the anomaly detection methods is still good, although it has been slightly reduced concerning the previous tests, possibly because the task is a little more complicated as the products are semantically closer to each other. Nevertheless, once again, the quality of the embeddings allows a proper differentiation between products. In this type of scenario, new reviews appear over time, even when the model is already in production. This would mean retraining the algorithm from scratch if we want to incorporate it into the model. This problem can be solved by using DAEF or OS-ELM, as they are two of the few anomaly detection models that allow what is known as online or incremental training, so its use can be beneficial. The main difference between both is that DAEF enables the use of deep architectures, while OS-ELM employs autoencoders with a single hidden layer.

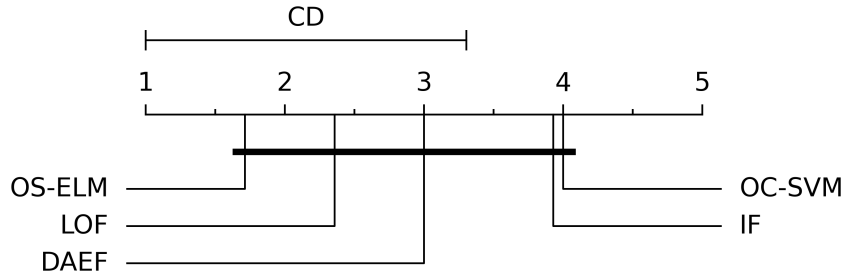


Figure 5.3: Graphical representation of Nemenyi test with $\alpha = 0.05$ for the 1vs.6 tests. The critical distance (CD) obtained was 2.31.

Table 5.3: Average test F1-score ± standard deviation for the 1vs.1 datasets. Best values according to the statistical test have been highlighted in bold.

Normal class	Anomalous class	DAEF	OS-ELM	OC-SVM	IF	LOF
Chocolate bars	Anise seeds	91.6±2.0	92.2±0.1	90.3±0.1	91.6±0.6	**92.8±0.1**
Anise seeds	Chocolate bars	**92.3±0.6**	90.1±0.1	91.4±0.1	89.9±0.3	90.4±0.1
Colored pencils	Ergonomic cushion	96.3±1.1	96.9±0.1	94.9±0.1	97.0±0.2	**97.0±0.1**
Ergonomic cushion	Colored pencils	96.4±0.8	96.4±0.1	94.4±0.1	96.3±0.2	**96.6±0.1**
Gaming mouse	PS4 membership	93.4±1.1	**94.1±0.1**	92.0±0.1	93.7±0.2	93.9±0.1
PS4 membership	Gaming mouse	92.6±1.2	**94.0±0.2**	92.0±0.2	90.0±0.3	90.3±0.1
Bluetooth speaker	Wi-Fi range extender	90.5±2.5	90.8±0.1	91.3±0.1	**93.1±0.2**	92.7±0.1
Wi-Fi range extender	Bluetooth speaker	92.8±0.6	**93.7±0.1**	92.5±0.1	92.1±0.3	91.9±0.1
Foot insoles	Yoga leggings	**91.2±0.5**	90.4±0.1	90.8±0.1	90.6±0.2	**91.2±0.1**
Yoga leggings	Foot insoles	90.2±1.8	**91.2±0.2**	90.7±0.3	87.9±0.6	88.4±0.1
Hamilton album	Partners album	82.5±1.3	**83.4±0.3**	80.3±0.3	62.4±3.4	72.0±0.5
Partners album	Hamilton album	84.1±1.6	**86.9±0.1**	83.8±0.1	75.9±1.3	80.7±0.2
Hygrometer	Vacuum	**91.8±0.4**	89.8±0.1	90.5±0.1	88.5±0.6	89.3±0.1
Vacuum	Hygrometer	93.6±1.1	93.8±0.1	92.6±0.1	93.6±0.2	**93.9±0.1**

5.4.2. Evaluating the explanations for model classifications

In Section 5.3.3, we proposed an explainability method to support the model decisions based on the occurrence of frequent terms, relying on the hypothesis that normal reviews will tend to use certain terms regularly, while anomalous reviews will not. In this section, this approach is compared, qualitatively through user surveys, with two other popular alternative approaches to achieve such explainability, specifically SHAP [121] and GPT-3 [25].

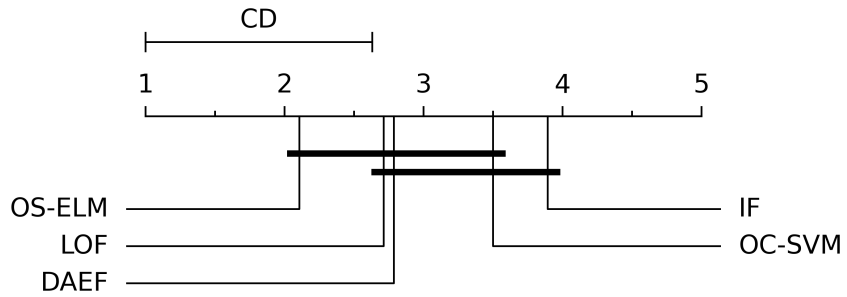


Figure 5.4: Graphical representation of Nemenyi test with $\alpha = 0.05$ for the 1vs.1 tests. The critical distance (CD) obtained was 1.63.

5.4.2.1. Explanations based on SHAP

Some explainability techniques have been adapted to deal with texts. In the case of NLP models, specifically transformers, SHAP (SHapley Additive exPlanations) [121] is one of the most widely used techniques. SHAP is a game theory-based approach to explain the output of any machine learning model. SHAP also assigns each feature an importance value for a particular prediction, but it has been extended to provide interpretability to models that use embeddings as input. In these cases, it can quantify the importance of each word of the original text in the prediction, which generates a quite understandable interpretation for a human being.

In this work, we considered that SHAP could be a good alternative to our proposal to generate the explanations associated with the classification of the reviews. Figure 5.5 shows the explanation generated by SHAP for a review classified as normal, where it can be seen that each term of the review has an associated score representing its influence on the classification. Despite this, for the evaluation of explanation techniques and to facilitate the understanding of the explanation by the final users (many of them unfamiliar with these techniques), during this work the explanations generated by SHAP were simplified, showing only the five most influential terms, instead of all.

5.4.2.2. Explanations based on GPT-3

There is no doubt that GPT-3 has started a technological revolution at all levels. Its availability to the general public has led to the discovery of a large number of unimaginable features before its release. The original idea was to create a high-level conversational bot, trained with a large amount of text available on the web, such as books, online encyclopedias, or forums. However, its deep understanding of language has far exceeded the preset idea of a chatbot. GPT-3 is capable of successfully carrying out tasks that go beyond writing a joke or summarizing a novel, GPT-3 is capable of analyzing and developing code in multiple programming languages, generating SEO positioning strategies, or carrying out NLP tasks traditionally solved by ad hoc models, such as sentiment analysis.

Due to its enormous potential, in this work, we have decided to study GPT-3

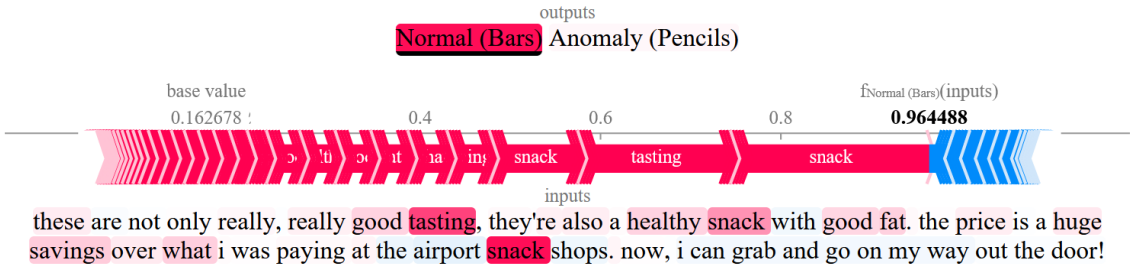


Figure 5.5: Explanation generated by SHAP for the anomalous reviews detection problem raised in this work. In this scenario, the reviews to be analyzed correspond to the product "chocolate bars" (Bars). The review of this example was correctly classified as normal. The most influential terms in its classification as normal are marked in red, while the terms that promote the opposite class are highlighted in blue, in this case practically none since we are dealing with an obvious case. Greater intensity implies greater influence. In this case, the review terms that have most influenced its classification as normal are "snack" and "tasting".

as an explainability model. To do this, first of all, we have introduced a prompt in which we describe the task that our anomaly detection model is carrying out, as well as the format with which we would like to work in future prompts (see Figure 5.6). After this, and following the predefined format, GPT-3 is ready to generate the explanations (see Figure 5.7). Although GPT-3 does not have direct knowledge of the anomaly detection model, its ability to generate consistent and intuitive responses can be of great help to humans reading the explanations.

We should note that since GPT-3 trains on a wide range of data from the internet, including data that may contain biases, there is a risk that the generated

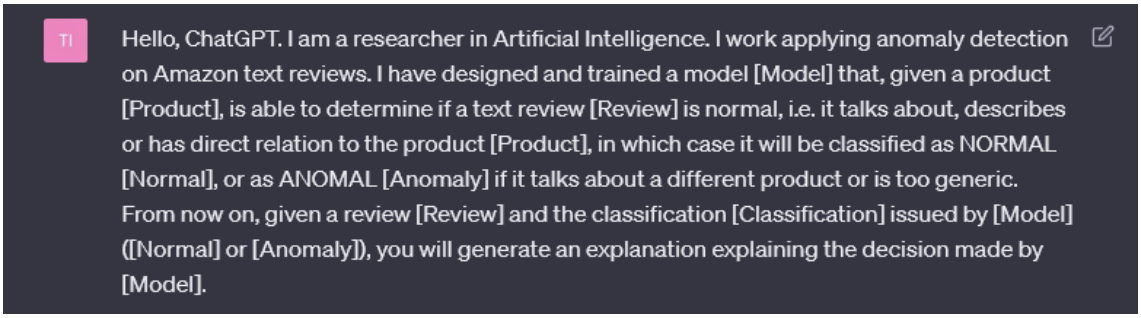


Figure 5.6: Initial prompt to present the anomaly detection problem to be solved to GPT-3.

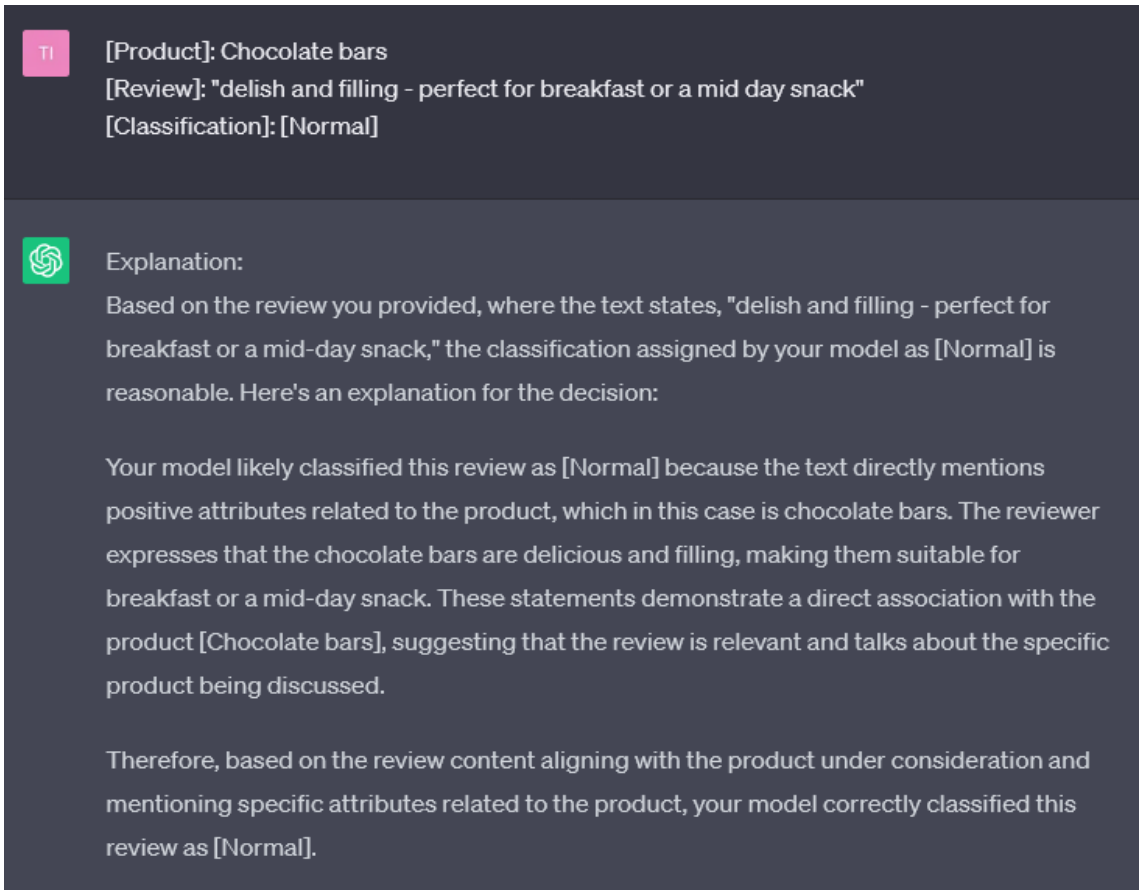


Figure 5.7: Prompt in which the classification of a review is explained by GPT-3. The product considered as normal is chocolate bars and the review has been classified as normal by the anomaly detection model.

explanations may reflect or amplify those biases. It is essential to exercise caution and perform a critical analysis of the explanations generated, considering their context and possible inherent biases.

5.4.2.3. How to evaluate explainability

Unlike other tasks such as anomaly detection, the quality of the explanations generated to provide certain interpretability to a model is not easily measurable. In a scenario like the one described in this work, the subjectivity of the users plays a strong role when determining whether an explanation is appropriate or not. Due to this, we have decided to carry out a comparative study of the three explainability techniques using a survey. This survey was disseminated through the students, professors, and research and administrative staff of our university, giving rise to a total of 241 participants. Table 5.4 shows the number of participants by area of knowledge. To build the survey, reviews from the "1 vs. 6" scenario described in Table 5.2 were used, considering "chocolate bars" as the normal class, and using DAEF as an anomaly detection method.

Table 5.4: Area of knowledge of the survey participants.

Knowledge area	Participants
Engineering and Architecture	87 (36.1%)
Social and Legal Sciences	77 (32.0%)
Natural Sciences	33 (13.7%)
Arts and Humanities	23 (9.5%)
Health Sciences	17 (7.1%)
Others	4 (1.7%)

The survey consists of two different tests: (1) Forward simulation [67], which allows measuring the effect of explanations on users; (2) Personal utility, which allows measuring the utility of the explanations based on the personal preferences of the users.

5.4.2.4. Forward simulation

This test is inspired by the work carried out by P. Hase et al. [67] and is divided into four phases: a Learning phase, a Pre-prediction phase, a Learning phase with explanations, and a Post-prediction phase. To begin, users are given 20 examples from the model's validation set with reviews and model predictions with no explanations. Then they must predict the model's output for 10 new reviews. Users cannot access the example reviews from the learning phase while they are in this prediction phase. Next, they return to the same learning examples, now with explanations included. Finally, they predict the behaviour of the model again on the same instances as in the first round of prediction. The classes of reviews chosen for each of the phases described are balanced between normal and abnormal. By design, any improvement in user performance in the Post prediction phase is attributable only to the addition of explanations that helped the user to better understand the behaviour of the model.

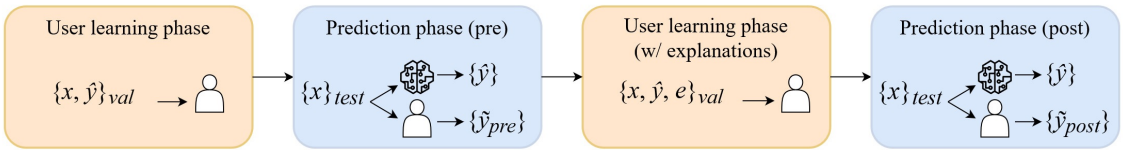


Figure 5.8: Forward simulation test procedure to measure human users' ability to understand and predict model behavior. To isolate the impact of explanations, baseline accuracy is measured first, followed by accuracy measurement after users have access to explanations of the model's behavior. The explained examples are different from the test instances.

Figure 5.8 represents this procedure, where x represents a review, $\widehat{y}$ is the class predicted by the model, and $\tilde{y}$ the class predicted by the human simulation. Taking this into account, the Explanation Effect can be calculated as follows:

$$Explanation\,Effect = Post\ Accuracy - Pre\ Accuracy \tag{5.1}$$

where the pre and post accuracies area calculated comparing the user's prediction against the model's prediction. In order not to bias the results, we have decided

that each person surveyed will only participate in the forward simulation of a single explainability technique, so that the total number of participants was divided randomly into three groups, each one associated with one of the three techniques: Terms frequency (78), SHAP (89), and GPT-3 (77). The three groups of participants faced the same reviews throughout the four phases of the test, only the explanations presented in the third phase of the test (learning with explanations) varied depending on the assigned group/technique. Throughout the survey, we have warned users several times that they should try to simulate the behavior of the anomaly detection model, instead of ranking the reviews using their personal criteria.

Tables 5.5 and 5.6 show the results of this test. As can be seen in the first table, the average explanation effect of the three explainability techniques has not been very remarkable. The high standard deviation suggests a high variability between the different study participants, both in the initial (Pre) and subsequent (Post) classifications and therefore in the explanation effect. The initial difference between the groups in the Pre-accuracy and the closeness of the three techniques in the explanation effect does not allow us to opt for any of the three options.

In Table 5.6 we can see the results of the test based on the area of knowledge of the participants. The results obtained using the different explainability techniques have been aggregated. The purpose of this comparison is to analyze whether there is any relationship between the technical background of the respondents and their performance on the test. As can be seen, there are slight differences between the values of pre and post accuracy, with the group of respondents belonging to the area of natural sciences standing out, whose explanation effect was the only positive one (4.5%).

Table 5.5: Forward simulation tests. Average accuracy and Explanation effect (± standard deviation) for each explainability technique.

Technique	Pre Accuracy	Post Accuracy	Explanation Effect
Terms frequency	76.2 ±13.0	72.8±13.8	-3.4±12.7
SHAP	72.4±14.9	71.9±14.9	-0.5±13.8
GPT-3	69.7±15.2	70.1 ±17.1	0.4±14.3

Table 5.6: Forward simulation tests. Average accuracy and Explanation effect (± standard deviation) broken down by participants' area of knowledge.

Area of knowledge	Pre Accuracy	Post Accuracy	Explanation Effect
Engineering and Architecture	76.3 ±14.5	73.9±16.1	-2.4±12.9
Social and Legal Sciences	72.2±13.0	69.5±15.8	-2.7±13.0
Natural Sciences	70.9±14.7	75.4 ±12.8	4.5±13.5
Arts and Humanities	68.7±17.7	67.8 ±14.8	-0.9±19.0
Health Sciences	72.0±14.2	71.3 ±10.6	-0.7±10.3

Analyzing the results we can affirm that the initial accuracy (pre-accuracy) is quite high, which indicates that the respondents tend to successfully reproduce the behavior of the model even if they do not have the explanations. This could be because, in this case, both the input data to the AD model (in natural language) and the problem it solves are easily understandable to a human, which means that the respondents can solve the classification problem by themselves. The standard deviation accompanying the pre-accuracy results is notable but does not become too high, which reaffirms the previous argument.

After supplying the respondents with the explanations associated with the reviews, the post-accuracy obtained by them presents values very similar to the previous scores (pre-accuracy). We can therefore affirm that in general terms the effect of the explanations, in this case, has not been beneficial for the users during this test.

The non-improvement may be due to several reasons. One of them may be the presence of reviews that show a certain degree of ambiguity, which not only makes their classification difficult for the respondents but also for the AD models. However, in real scenarios the occurrence of reviews whose normality score is around the threshold value would be something to be expected, not all events are easily classifiable. Another possible reason may be that the tendency of some users throughout the survey has been to classify the reviews using their personal criteria, rather than trying to simulate the behavior of the anomaly detection model.

5.4.2.5. Personal utility

After completing the first test, the participants were given a second exercise, common to all participants, regardless of the group to which they were assigned in the previous phase. This consists of a subjective evaluation of the three explainability techniques. The idea is to present the participant with a review, its classification by the model, and an explanation generated by each of the three explainability techniques. The participant must order the explanations based on how useful it is to understand the reasoning behind the model's decision (ties were allowed between explanations). This process was repeated with a total of eight reviews. Tables 5.7 and 5.8 show the final average rankings of the explainability techniques.

Table 5.7: Personal utility tests. Average ranking (± standard deviation) for each explainability technique.

Technique	Position
Terms frequency	1.6 ±0.4
SHAP	2.1±0.5
GPT-3	1.7±0.5

As can be seen in Table 5.7, the explanations based on Term frequency (1.6) and GPT-3 (1.7) are in very close positions, both being above the third method SHAP (2.1). The preference of respondents for the first two methods may be due to the fact that the format of their explanations is more accessible and descriptive for a larger part of the population. Grouping the results by areas of knowledge (see Table 5.8), we can see that the general trend is maintained for most areas. We can highlight the case of Social and Legal Sciences and Health Sciences, areas in which the positions in the ranking of Terms frequency and GPT-3 techniques are slightly inverted, with GPT-3 being the preferred option.

Table 5.8: Personal utility tests. Average ranking (± standard deviation) for each technique and area of knowledge.

Area of knowledge	Technique	Position
Engineering and Architecture	Terms frequency	1.6 ±0.4
	SHAP	2.2 ±0.5
	GPT-3	1.7 ±0.5
Social and Legal Sciences	Terms frequency	1.7±0.4
	SHAP	1.9±0.4
	GPT-3	1.6±0.5
Natural Sciences	Terms frequency	1.6±0.4
	SHAP	2.0±0.5
	GPT-3	1.7±0.5
Arts and Humanities	Terms frequency	1.5±0.4
	SHAP	1.9±0.5
	GPT-3	1.8±0.5
Health Sciences	Terms frequency	1.8±0.3
	SHAP	2.1±0.5
	GPT-3	1.6±0.5

5.5. Conclusion

In this work, we have proposed a pipeline to detect anomalous reviews associated with Amazon products, which can be directly extrapolated to other online review platforms or scenarios with similar characteristics. The representation of the reviews using MPNet embeddings has enabled the training of classical anomaly detection algorithms that have achieved a high performance in terms of F1-score. These have been evaluated using reviews from different products and categories, and the score they emit allows us to sort the reviews based on their normality.

A technique based on the occurrence of frequent terms has been proposed to generate explanations associated with the classifications of the reviews. This technique has been compared with SHAP, one of the reference post-hoc techniques in the field of explainability, and with GPT-3, due to its high power and versatility. To evaluate this aspect of the pipeline, we conducted a two-part survey in which 241

members of the university community participated.

From the first part of the explainability test we can conclude that, in general terms, the effect of the explanations has not been beneficial for the users. In any case, these tests allow us to reflect on the difficulty of using explainability and evaluation techniques in borderline scenarios where subjectivity plays an important role, such as the one presented in this work or in other fields of NLP, as well as in areas such as image or audio generation.

Regarding the second part of the explainability test, we have been able to conclude that respondents preferred explanations that presented a more natural and familiar appearance over more condensed and concise explanations such as those provided by SHAP, regardless of the explanation effect they provide. Explanations based on term frequency analysis have been preferred by respondents along with GPT-3, however, our approach presents significantly lower computational costs and both its use and the explanations produced are simpler for the users.

However, it is necessary to mention some of the limitations of the proposed architecture. The main constraint of the presented approach is that, given the nature of the problem, it is necessary to train and maintain an anomaly detection model for each product of the platform, which in some cases could involve problems such as system upscaling.

Moreover, in this type of situation, new reviews come in over time even after the model is already in production. This would mean retraining the algorithm from scratch if we want to incorporate them into the model. This problem is solved with the use of DAEF or OS-ELM as they are two of the few anomaly detection models that allow what is known as online or incremental training. The main difference between both is that OS-ELM employs autoencoders with a single hidden layer, while DAEF enables the use of deep architectures, which can be beneficial in certain scenarios.

Finally, the calculation of review embeddings using transformer models can be a slow operation if we are faced with a scenario of a certain magnitude, for example, millions of reviews for each product, which can become a limitation if sufficient hardware resources are not available.

In future work, it would be interesting to evaluate GPT-3 and other large lan-

guage models carrying out the complete process followed by the pipeline proposed in this work, instead of being tested only in the explainability module. We have not performed this test due to the high computational cost that would be involved in processing the thousands of reviews to be evaluated. Since it is common for text reviews to be accompanied by images, it would also be of great interest to employ multimodal models for their analysis as a whole, which could be very beneficial to understanding user opinions at a deeper level [164]. If we had information about users in relation to their purchases, it would also be interesting to test the use of recommender systems to discern between legitimate and illegitimate comments from the user's point of view.

Regarding explainability, another interesting possible line of future work would be to broaden the scope of the survey, both in terms of the number of products involved and the number of reviews, in order to clarify the conclusions reached at the forward simulation stage. It would be very useful to try to present the explanations issued by SHAP in a more familiar and natural format for the end user, so that we can see if their level of preference is increased for the general public. Finally, it would be interesting to develop an explainable-by-design anomaly detection algorithm, thus avoiding the need to use post-hoc explainability techniques.

Chapter 6

Conclusions and future work

The massive computational power of today's devices, combined with the huge availability of data, provides the basis for an exciting era of research and application in the field of artificial intelligence. The rapid and vast growth of the technological scenario in which we find ourselves is in many ways comparable to the major revolutions of the past. At the dawn of this new era, it becomes imperative to harness both the great interest of the general public, companies and investors to improve the quality of life of our society. However, this new age also raises new risks and challenges that place humanity at the center of attention. Ethical considerations, privacy concerns, and responsible development must take the lead. If a new technological winter has to come, it will, but we have the opportunity not to be the direct cause.

In this thesis, we have studied and explored different aspects of anomaly detection, its application to different scenarios, and the challenges that they present. In Chapter 3 we have observed the great difficulty of dealing with machine learning with high dimensionality datasets. This problem is accentuated when learning methods operate with the training data in the original space, as a convex hull could do. With the proposed OCENCH method we solve this problem by making use of the aforementioned random projections, which allow us to very effectively reduce the dimensionality of the data. In our particular case two-dimensional spaces, which greatly facilitates the calculation of the convex hulls. In this chapter, we have also explored the difficulties that these methods present when the distributions of the

data are not as expected. We have been able to observe how a proper pruning and subdivision methodology can adapt some of these methods to new scenarios, as in the case of OCENCH. Its performance in the different tests carried out is remarkable, positioning itself as an alternative for both convex and non-convex problems. In addition, OCENCH is easily configurable through its parameters, is a lightweight method in memory requirements, and can be run in parallel. As future work, it would be interesting to develop a new version of the algorithm that could detect the existence of empty regions inside dense regions to be able to fit more precisely in these scenarios, such as rings or concentric rings. Another possible line of work would be the implementation of a forgetting mechanism to extend the method to online learning environments, and to adapt and test the algorithm in edge computing and federated learning environments.

In this thesis, we have also faced the high computational times and costs of iterative neural networks. These heavy trainings, despite being effective in terms of the performance achieved by the models, hinder their applicability to environments where computational power is reduced, as is the case of edge computing and federated learning. That is the reason why in Chapter 4 we present DAEF, a deep autoencoder that solves this problem by performing a non-iterative training. This not only allows its application in edge computing scenarios but also a reduction in the emissions associated with its use, which is desirable in any case. The way it is designed also allows its use in federated environments, preserving the privacy of the nodes that form the federated network, a fundamental aspect in this domain. In both scenarios, the method presents a high and robust performance during the tests carried out, which makes this method an interesting alternative. During this chapter we have also explored the pros and cons of the different federated architectures that can be adopted by models such as DAEF, proposing a semi-supervised architecture for its use that employs MQTT as the communication protocol. As future work, it would be interesting to test the algorithm in real edge computing or federated learning environments using real physical devices that act as independent nodes, instead of simulating them. Another possible line of work would be to implement other communication protocols, such as Secure MQTT [190], and redesign the architecture to be completely decentralized, thus dispensing with the coordinator and turning it into peer-to-peer, as well as the application of blockchain technology to preserve the integrity of the global model collaboratively [163].

Finally, in Chapter 5 we explored the use of anomaly detection on text data. We have encountered great difficulties in defining the concept of anomaly in these scenarios, being always context-dependent. In our case we have decided to deal with text reviews of e-commerce products, so the definitions presented here may vary for other scenarios. In addition, the low availability of datasets and the non-existence of benchmarks for the problem we have addressed have meant that both the decisions associated with the treatment of the dataset and its evaluation fall directly on us. In this chapter we have also explored the use of explainability techniques on these scenarios composed of unstructured data, being aware of the enormous subjectivity involved in these processes. To evaluate the explainability module, a study was conducted to compare three explainability techniques involving a total of 241 participants. The aim of this study was both to measure the impact of the explanations on the reproducibility of the classification model by the respondents, and the usefulness of these explanations in scenarios where subjectivity plays such an important role as the one posed in this work. As future work, it would be interesting to evaluate other large language models carrying out the complete process followed by the pipeline proposed in this work, instead of being tested only in the explainability module. Since it is common for text reviews to be accompanied by images, it would also be of great interest to employ multimodal models for their analysis as a whole, which could be very beneficial to understanding user opinions at a deeper level [164]. If we had information about users in relation to their purchases, it would be valuable to use recommender systems approaches to discern between legitimate and illegitimate comments from the user's point of view. Regarding explainability, another possible line of future work would be to broaden the scope of the survey, as well as to develop an explainable-by-design anomaly detection algorithm, thus avoiding the need to use post-hoc explainability techniques.

In this thesis, we have explored quite different areas, each of which could be the subject of study for a long time. However, as with everything, we must establish certain limits. That is why, as lines of future work, in addition to those already proposed, and given the enormous range of possibilities presented by anomaly detection, we propose to work on new types of data, specifically images and time series.

Anomaly detection in images [226] has a wide range of applications such as video surveillance [155], defect segmentation [231], quality control [138] or medical

imaging [52]. On the other hand, detecting anomalies in time series [19] can have critical implications, such as identifying financial fraud [101], predicting disease outbreaks [170] or ensuring efficient operation of industrial systems [84]. In both cases, the characteristics of the scenarios are completely different from those presented here, and therefore require specific methods for their resolution. Images are usually treated using deep learning techniques that can extract patterns in a very efficient way, such as convolutional networks [108], while time series can be handled using methods that preserve the temporal factor of the underlying data, such as recurrent networks [127] or transformers [221].

The future of anomaly detection holds great opportunities for innovative research, pushing the boundaries of what is possible. It is a journey marked by adaptability and innovation, opening the door to exciting discoveries in this field. So let's keep working on it.

Bibliography

[1] M. Ahmed. Collective anomaly detection techniques for network traffic analysis. *Annals of data science*, 5(4):497–512, 2018. pages 12

[2] M. Ahmed and A. N. Mahmood. Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection. *Annals of Data Science*, 2(1):111–130, 2015. pages 12

[3] M. Ahmed, A. N. Mahmood, and M. R. Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, 2016. pages 2, 10, 25

[4] W. L. Al-Yaseen, A. K. Idrees, and F. H. Almasoudy. Wrapper feature selection method based differential evolution and extreme learning machine for intrusion detection system. *Pattern Recognition*, 132:108912, 2022. pages 66

[5] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming. Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0305–0310. IEEE, 2019. pages 28

[6] Amazon customer reviews dataset. https://nijianmo.github.io/amazon/index.html. pages 104, 114

[7] Amazon targets fake review fraudsters on social media. https://www.aboutamazon.com/news/policy-news-views/amazon-targets-fake-review-fraudsters-on-social-media. pages 103

[8] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1):1–18, 2015. pages 23

[9] A. Asuncion and D. Newman. Uci machine learning repository, 2007. https://archive.ics.uci.edu/ml/index.php. pages 56

[10] D. Bank, N. Koenigstein, and R. Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020. pages 23

[11] D. Bank, N. Koenigstein, and R. Giryes. Autoencoders, 2021. pages 3, 30, 37, 56, 163, 166

[12] D. Bank, N. Koenigstein, and R. Giryes. Autoencoders, 2021. pages 16, 67

[13] C. B. Barber, D. P. Dobkin, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996. pages 39

[14] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. pages 6, 33

[15] B. Barz, E. Rodner, Y. G. Garcia, and J. Denzler. Detecting regions of maximal divergence for spatio-temporal anomaly detection. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1088–1101, 2018. pages 12

[16] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, 16(1):303–336, 2013. pages 2, 10, 25

[17] E. Bingham and H. Mannila. Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, page 245–250, New York, NY, USA, 2001. Association for Computing Machinery. pages 45

[18] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. pages 113

[19] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021. pages 12, 134

[20] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019. pages 4, 32

[21] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997. pages 19

[22] M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *ACM Sigmoid International Conference on management of data*, pages 93–104. ACM SIGMOD Record, 2000. pages 56, 114, 164, 168

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000. pages 22

[24] R. Bro and A. K. Smilde. Principal component analysis. *Analytical methods*, 6(9):2812–2831, 2014. pages 24

[25] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. pages 108, 119

[26] K. Cao, Y. Liu, G. Meng, and Q. Sun. An overview on edge computing research. *IEEE Access*, 8:85714–85728, 2020. pages 4, 31, 65, 66

[27] T. P. Carvalho, F. A. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. Alcalá. A systematic literature review of machine learning methods applied

to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019. pages 2, 10, 27

[28] P. Casale, O. Pujol, and P. Radeva. Approximate polytope ensemble for one-class classification. *Pattern Recognit.*, 47(2):854 – 864, 2014. pages 16, 36, 38

[29] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, 2009. pages 2, 10, 11, 12, 16, 19, 25, 66, 108

[30] A. Chatterjee and B. S. Ahmed. Iot anomaly detection methods and applications: A survey. *Internet of Things*, 19:100568, 2022. pages 28

[31] Y. Chen, X. S. Zhou, and T. S. Huang. One-class svm for learning in image retrieval. In *Proceedings 2001 international conference on image processing (Cat. No. 01CH37205)*, volume 1, pages 34–37. IEEE, 2001. pages 24

[32] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau. Autoencoder-based network anomaly detection. In *2018 Wireless telecommunications symposium (WTS)*, pages 1–5. IEEE, 2018. pages 23

[33] A. V. Chernov, I. K. Savvas, and M. A. Butakova. Detection of point anomalies in railway intelligent control system using fast clustering techniques. In *International Conference on Intelligent Information Technologies for Industry*, pages 267–276. Springer, 2018. pages 12

[34] A. Chernyavskiy, D. Ilvovsky, and P. Nakov. Transformers: "the end of history" for nlp? *CoRR*, abs/2105.00813, 2021. pages 5, 33

[35] N. Cohen and Y. Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020. pages 29

[36] A. A. Cook, G. Mısırlı, and Z. Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2019. pages 28

[37] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000. pages 3

[38] S. Dasgupta. Experiments with random projection. *Conference on Uncertainty in Artificial Intelligence*, abs/1301.3849, 2013. pages 45

[39] S. Dasgupta and A. Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. pages 45

[40] L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft. Image anomaly detection with generative adversarial networks. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*, pages 3–17. Springer, 2019. pages 29

[41] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006. pages 57, 94, 117

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. pages 108

[43] S. Ding, N. Zhang, X. Xu, L. Guo, and J. Zhang. Deep extreme learning machine and its application in EEG classification. *Math. Probl. Eng.*, 2015:1–11, 05 2015. pages 68

[44] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74:406–421, 2018. pages 66

[45] V. N. Dornadula and S. Geetha. Credit card fraud detection using machine learning algorithms. *Procedia computer science*, 165:631–641, 2019. pages 2, 10

[46] D. Dua and C. Graff. UCI machine learning repository, 2017. pages 91

[47] M. Duckham, L. Kulik, M. Worboys, and A. Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, 41(10):3224–3236, 2008. pages 35, 38, 39, 45

[48] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. pages 70

[49] N. Elmrabit, F. Zhou, F. Li, and H. Zhou. Evaluation of machine learning algorithms for anomaly detection. In *2020 international conference on cyber security and protection of digital services (cyber security)*, pages 1–8. IEEE, 2020. pages 18

[50] M. Fahim and A. Sillitti. Anomaly detection, analysis and prediction techniques in iot environment: A systematic literature review. *IEEE Access*, 7:81664–81681, 2019. pages 28

[51] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70:447–489, 2019. pages 25

[52] T. Fernando, H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Deep learning for medical anomaly detection–a survey. *ACM Computing Surveys (CSUR)*, 54(7):1–37, 2021. pages 2, 10, 26, 134

[53] O. Fontenla-Romero, B. Guijarro-Berdiñas, and B. Pérez-Sánchez. Regularized one-layer neural networks for distributed and incremental environments. In *IWANN*, volume 12862, pages 343–355. Springer, 2021. pages 72, 78, 79

[54] O. Fontenla-Romero, B. Guijarro-Berdiñas, B. Pérez-Sánchez, and A. Alonso-Betanzos. A new convex objective function for the supervised learning of single-layer neural networks. *Pattern Recogn.*, 43(5):1984–1992, may 2010. pages 72

[55] O. Fontenla-Romero, B. Pérez-Sánchez, and B. Guijarro-Berdiñas. DSVD-autoencoder: A scalable distributed privacy-preserving method for one-class classification. *Int. J. Intell. Syst.*, 36(1):177–199, 2021. pages 70, 78, 79

[56] L. Gao, L. Zhang, C. Liu, and S. Wu. Handling imbalanced medical image data: A deep-learning-based one-class classification approach. *Artificial intelligence in medicine*, 108:101935, 2020. pages 2, 10, 26

[57] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(89):2677–2694, 2008. pages 57, 94, 117

[58] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. pages 94

[59] M. Goldstein and A. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 1:59–63, 2012. pages 21

[60] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016. pages 15

[61] T. Golomb, Y. Mirsky, and Y. Elovici. Ciota: Collaborative iot anomaly detection via blockchain. *arXiv preprint arXiv:1803.03807*, 2018. pages 28

[62] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 46:235–262, 2013. pages 16

[63] O. Gorokhov, M. Petrovskiy, and I. Mashechkin. Convolutional neural networks for unsupervised anomaly detection in text data. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 500–507. Springer, 2017. pages 29

[64] D. Guthrie, L. Guthrie, B. Allison, and Y. Wilks. Unsupervised anomaly detection. In *IJCAI*, pages 1624–1628, 2007. pages 15

[65] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018. pages 4, 32

[66] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem. Attack and anomaly detection in iot sensors in iot sites using machine learning approaches. *Internet of Things*, 7:100059, 2019. pages 28

[67] P. Hase and M. Bansal. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pages 5540–5552, Online, July 2020. Association for Computational Linguistics. pages 123, 124

[68] M. Haselmann, D. P. Gruber, and P. Tabatabai. Anomaly detection using deep learning based image completion. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 1237–1242. IEEE, 2018. pages 29

[69] M. H. Hassan, A. Tizghadam, and A. Leon-Garcia. Spatio-temporal anomaly detection in intelligent transportation systems. *Procedia Computer Science*, 151:852–857, 2019. pages 12

[70] M. Hauskrecht, I. Batal, M. Valko, S. Visweswaran, G. F. Cooper, and G. Clermont. Outlier detection for patient monitoring and alerting. *Journal of biomedical informatics*, 46(1):47–55, 2013. pages 2, 10, 26

[71] A. Hedström, L. Weber, D. Krakowczyk, D. Bareeva, F. Motzkus, W. Samek, S. Lapuschkin, and M. M. M. Höhne. Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and beyond. *Journal of Machine Learning Research*, 24(34):1–11, 2023. pages 33

[72] D. Hendrycks, M. Mazeika, and T. Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018. pages 16

[73] W. Hilal, S. A. Gadsden, and J. Yawney. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429, 2022. pages 104

[74] B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 603–618, New York, NY, USA, 2017. Association for Computing Machinery. pages 83

[75] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933. pages 3, 30, 37

[76] D. Huang, D. Mu, L. Yang, and X. Cai. Codetect: Financial fraud detection with anomaly feature detection. *IEEE Access*, 6:19161–19174, 2018. pages 26

[77] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489–501, 2006. Neural Networks. pages 68

[78] Hugging face. https://huggingface.co/. Accessed: 2023-05-09. pages 110

[79] T. T. Huong, T. P. Bac, D. M. Long, T. D. Luong, N. M. Dan, L. A. Quang, L. T. Cong, B. D. Thang, and K. P. Tran. Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach. *Computers in Industry*, 132:103509, 2021. pages 104

[80] B. Hussain, Q. Du, S. Zhang, A. Imran, and M. A. Imran. Mobile edge computing-based data-driven deep learning framework for anomaly detection. *IEEE Access*, 7:137656–137667, 2019. pages 67

[81] S. Ioffe. Probabilistic linear discriminant analysis. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision - European Conference on Computer Vision 2006*, pages 531–542, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. pages 3, 30, 37

[82] I. Irigoien, B. Sierra, and C. Arenas. Towards application of one-class classification methods to medical data. *The Scientific World Journal*, 2014, 2014. pages 2, 10, 26

[83] R. Ito, M. Tsukada, and H. Matsutani. An on-device federated learning approach for cooperative model update between edge devices. *IEEE Access*, 9:92986–92998, 2021. pages 68, 90, 94, 95

[84] W. Jiang, Y. Hong, B. Zhou, X. He, and C. Cheng. A gan-based anomaly detection approach for imbalanced industrial time series. *IEEE Access*, 7:143608–143619, 2019. pages 134

[85] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, page 219–230, New York, NY, USA, 2008. Association for Computing Machinery. pages 105

[86] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26:189–206, 1984. pages 38

[87] E. Jove, J.-L. Casteleiro-Roca, H. Quintián, J.-A. Méndez-Pérez, and J. L. Calvo-Rolle. A new method for anomaly detection based on non-convex boundaries with random two-dimensional projections. *Information Fusion*, 65:50–57, 2021. pages 36, 38, 56, 163

[88] S. Kamaruddin and V. Ravi. Credit card fraud detection using big data analytics: use of psoaann based one-class classification. In *Proceedings of the international conference on informatics and analytics*, pages 1–8, 2016. pages 2, 10

[89] P. Kamat and R. Sugandhi. Anomaly detection for predictive maintenance in industry 4.0-a survey. In *E3S web of conferences*, volume 170, page 02007. EDP Sciences, 2020. pages 2, 10, 27

[90] L. Kasun, H. Zhou, G.-B. Huang, and C.-M. Vong. Representational learning with ELMs for Big Data. *IEEE Intelligent Systems*, 28:31–34, 11 2013. pages 68, 74, 76

[91] L. Kaupp, U. Beez, J. Hülsmann, and B. G. Humm. Outlier detection in temporal spatial log data using autoencoder for industry 4.0. In *Engineering Applications of Neural Networks: 20th International Conference, EANN 2019, Xersonisos, Crete, Greece, May 24-26, 2019, Proceedings 20*, pages 55–65. Springer, 2019. pages 2, 10, 27

[92] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE, 2014. pages 22

[93] S. S. Khan and M. G. Madden. One-class classification: Taxonomy of study and review of techniques. *The Knowledge Engineering Review*, abs/1312.0049, 2013. pages 108

[94] S. S. Khan and M. G. Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014. pages 2

[95] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, 2019. pages 4, 31

[96] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019. pages 2, 10, 25

[97] Y. Kim, S. Bang, J. Sohn, and H. Kim. Question answering method for infrastructure damage information retrieval from textual data using bidirectional encoder representations from transformers. *Automation in Construction*, 134:104061, 2022. pages 5, 33

[98] A. Kind, M. P. Stoecklin, and X. Dimitropoulos. Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management*, 6(2):110–121, 2009. pages 21

[99] M. Kirlidog and C. Asuk. A fraud detection approach with data mining in health insurance. *Procedia-Social and Behavioral Sciences*, 62:989–994, 2012. pages 26

[100] M. Köppen. The curse of dimensionality. In *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8, 2000. pages 23

[101] Y. Kou, C.-T. Lu, S. Sirwongwattana, and Y.-P. Huang. Survey of fraud detection techniques. In *IEEE International Conference on Networking, Sensing and Control, 2004*, volume 2, pages 749–754. IEEE, 2004. pages 2, 10, 26, 134

[102] R. Kumari and S. K. Srivastava. Machine learning: A review on binary classification. *International Journal of Computer Applications*, 160(7), 2017. pages 10, 16

[103] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015. pages 15

[104] Y.-T. K. Lai, J.-S. Hu, Y.-H. Tsai, and W.-Y. Chiu. Industrial anomaly detection and one-class classification using generative adversarial networks. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1444–1449. IEEE, 2018. pages 2, 10, 27

[105] L. J. Latecki, A. Lazarevic, and D. Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007. pages 21

[106] R. Laxhammar, G. Falkman, and E. Sviestins. Anomaly detection in sea traffic-a comparison of the gaussian mixture model and the kernel density estimator. In *2009 12th international conference on information fusion*, pages 756–763. IEEE, 2009. pages 20

[107] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 25–36. SIAM, 2003. pages 2, 10, 25

[108] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010. pages 134

[109] K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu. Improving one-class svm for anomaly detection. In *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No. 03EX693)*, volume 5, pages 3077–3081. IEEE, 2003. pages 24

[110] L. Li, Y. Fan, M. Tse, and K.-Y. Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020. pages 4, 32

[111] L. Li, R. J. Hansman, R. Palacios, and R. Welsch. Anomaly detection via a gaussian mixture model for flight operation and safety monitoring. *Transportation Research Part C: Emerging Technologies*, 64:45–57, 2016. pages 20

[112] P. Li and O. Niggemann. Non-convex hull based anomaly detection in cpps. *Engineering Applications of Artificial Intelligence*, 87:103301, 2020. pages 3, 30, 37

[113] Y. Li, B. Fang, L. Guo, and Y. Chen. Network anomaly detection based on tcm-knn algorithm. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 13–19, 2007. pages 22

[114] N.-y. Liang, G.-b. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 17(6):1411–1423, 2006. pages 68, 94, 114, 166, 168

[115] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013. pages 2, 10, 25

[116] J. Lin, W. Yu, X. Yang, P. Zhao, H. Zhang, and W. Zhao. An edge computing based public vehicle system for smart transportation. *IEEE Transactions on Vehicular Technology*, 69(11):12635–12651, 2020. pages 4, 31

[117] T. Lin, Y. Wang, X. Liu, and X. Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022. pages 15

[118] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. pages 56, 114, 164, 168

[119] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008. pages 22

[120] P. Liznerski, L. Ruff, R. A. Vandermeulen, B. J. Franks, K.-R. Müller, and M. Kloft. Exposing outlier exposure: What can be learned from few, one, and zero outlier images, 2022. pages 16

[121] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus,

S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. pages 6, 33, 119, 120

[122] T. Luo and S. G. Nagarajan. Distributed anomaly detection using autoencoder neural networks in wsn for iot. In *2018 ieee international conference on communications (icc)*, pages 1–6. IEEE, 2018. pages 28

[123] T. Luo and S. G. Nagarajan. Distributed anomaly detection using autoencoder neural networks in WSN for IoT. In *IEEE ICC*, pages 1–6, 2018. pages 68

[124] A. Mahapatra, N. Srivastava, and J. Srivastava. Contextual anomaly detection in text data. *Algorithms*, 5(4):469–489, 2012. pages 29

[125] A. W. Marshall and I. Olkin. Families of multivariate distributions. *Journal of the American statistical association*, 83(403):834–841, 1988. pages 20

[126] L. Martí, N. Sanchez-Pi, J. M. Molina, and A. C. B. Garcia. Anomaly detection based on sensor data in petroleum industry applications. *Sensors*, 15(2):2774–2797, 2015. pages 2, 10, 27

[127] L. R. Medsker and L. Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001. pages 134

[128] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013. pages 107

[129] B. Mishra and A. Kertesz. The use of MQTT in M2M and IoT systems: A survey. *IEEE Access*, 8:201071–201086, 2020. pages 78

[130] R. K. Mobley. *An introduction to predictive maintenance*. Elsevier, 2002. pages 27

[131] B. Mohammadi, M. Fathy, and M. Sabokrou. Image/video deep anomaly detection: A survey. *arXiv preprint arXiv:2103.01739*, 2021. pages 29

[132] R. Mohawesh, S. Xu, S. N. Tran, R. Ollington, M. Springer, Y. Jararweh, and S. Maqsood. Fake reviews detection: A survey. *IEEE Access*, 9:65771–65802, 2021. pages 105

[133] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021. pages 80

[134] J. Mu, X. Zhang, Y. Li, and J. Guo. Deep neural network for text anomaly detection in siot. *Computer Communications*, 178:286–296, 2021. pages 106

[135] G. Münz, S. Li, and G. Carle. Traffic anomaly detection using k-means clustering. In *Gi/itg workshop mmbnet*, volume 7, 2007. pages 22

[136] N. Naik. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7, 2017. pages 84

[137] A. Nanda, A. Mahapatra, B. Mohapatra, and a. mahapatra. Multiple comparison test by tukey's honestly significant difference (hsd): Do the confident level control type i error. *International Journal of Applied Mathematics and Statistics*, 6:59–65, 01 2021. pages 92, 116, 117

[138] P. Napoletano, F. Piccoli, and R. Schettini. Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors*, 18(1), 2018. pages 133

[139] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab. Machine learning for anomaly detection: A systematic review. *Ieee Access*, 9:78658–78700, 2021. pages 15, 16, 19, 25

[140] R. Nayak, U. C. Pati, and S. K. Das. A comprehensive review on deep learning-based methods for video anomaly detection. *Image and Vision Computing*, 106:104078, 2021. pages 29

[141] M. V. Ngo, T. Luo, and T. Q. S. Quek. Adaptive anomaly detection for internet of things in hierarchical edge computing: A contextual-bandit approach. *ACM Trans. Internet Things*, 3(1), oct 2021. pages 68

[142] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi. DÏot: A federated self-learning anomaly detection system for iot. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 756–767, 2019. pages 67

[143] S. Niknam, H. S. Dhillon, and J. H. Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6):46–51, 2020. pages 4, 32

[144] W. S. Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006. pages 24

[145] D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas. Characterization of point datasets using subdivisible non-convex hulls. In *Proceedings of the XIX Conference of the Spanish Association for Artificial Intelligence*, pages 141–146, Malaga, Spain, September 2021. AEPIA. ISBN 978-84-09-30514-8. pages 46

[146] D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas. A one-class classification method based on expanded non-convex hulls. *Information Fusion*, 89:1–15, 2023. pages 35, 163

[147] D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas. Fast deep autoencoder for federated learning. *Pattern Recognition*, 143:109805, 2023. pages 66, 87, 109, 110, 166, 168

[148] D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas. Explained anomaly detection in text reviews: Can subjective scenarios be correctly evaluated? *Engineering Applications of Artificial Intelligence*, 133:108065, 2024. pages 104

[149] S. Omar, A. Ngadi, and H. H. Jebur. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications*, 79(2), 2013. pages 16

[150] OpenAI. GPT-4 technical report, 2023. pages 108

[151] E. Ostertagova, O. Ostertag, and J. Kováč. Methodology and application of the kruskal-wallis test. *Applied Mechanics and Materials*, 611:115–120, 08 2014. pages 92, 116, 117

[152] G. Pachauri and S. Sharma. Anomaly detection in medical wireless sensor networks using machine learning algorithms. *Procedia Computer Science*, 70:325–333, 2015. pages 2, 10, 26

[153] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel. Deep learning for anomaly detection: A review. *ACM Comput. Surv.*, 54(2), mar 2021. pages 2, 10

[154] J.-S. Park and S. Oh. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of Information Science and Engineering*, 29:379–392, 2012. pages 3, 30, 37

[155] D. R. Patrikar and M. R. Parate. Anomaly detection using edge computing in video surveillance system. *International Journal of Multimedia Information Retrieval*, 11(2):85–110, 2022. pages 133

[156] E. J. Pauwels and O. Ambekar. One class classification for anomaly detection: Support vector data description revisited. In *Advances in Data Mining. Applications and Theoretical Aspects: 11th Industrial Conference, ICDM 2011, New York, NY, USA, August 30–September 3, 2011. Proceedings 11*, pages 25–39. Springer, 2011. pages 25

[157] D. Peña and F. J. Prieto. Multivariate outlier detection and robust covariance matrix estimation. *Technometrics*, 43(3):286–310, 2001. pages 20

[158] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. pages 107

[159] D. Peña and F. J. Prieto. Multivariate outlier detection and robust covariance matrix estimation. *Technometrics*, 43(3):286–310, 2001. pages 56, 164

[160] C. Phua, V. Lee, K. Smith, and R. Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010. pages 26

[161] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal processing*, 99:215–249, 2014. pages 19, 25

[162] T. Pourhabibi, K.-L. Ong, B. H. Kam, and Y. L. Boo. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, 133:113303, 2020. pages 2, 10, 26

[163] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor. Chained anomaly detection models for federated learning: An intrusion detection case study. *Appl. Sci.*, 8(12), 2018. pages 67, 102, 132

[164] P. Pérez-Núñez, J. Díez, O. Luaces, B. Remeseiro, and A. Bahamonde. Users' photos of items can reveal their tastes in a recommender system. *Information Sciences*, 642:119227, 2023. pages 130, 133

[165] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Communications Surveys & Tutorials*, 22(4):2462–2488, 2020. pages 4, 31

[166] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng. A survey of predictive maintenance: Systems, purposes and approaches. *arXiv preprint arXiv:1912.07383*, 2019. pages 27

[167] G. Ratsch, S. Mika, B. Scholkopf, and K. Muller. Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 24(9):1184–1199, 2002. pages 3, 56, 164

[168] S. Rayana. Odds library, 2016. http://odds.cs.stonybrook.edu/, Revised at 23/03/2021. pages 56

[169] S. Rayana. ODDS library, 2016. pages 91

[170] T. Rekatsinas, S. Ghosh, S. R. Mekaru, E. O. Nsoesie, J. S. Brownstein, L. Getoor, and N. Ramakrishnan. Sourceseer: Forecasting rare disease outbreaks using multiple data sources. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 379–387. SIAM, 2015. pages 134

[171] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3009–3017, 2019. pages 12

[172] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. pages 6, 33

[173] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):119, 2020. pages 4, 32

[174] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. pages xxv, 11, 12, 13

[175] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018. pages 2, 10, 25

[176] L. Ruff, R. A. Vandermeulen, B. J. Franks, K.-R. Müller, and M. Kloft. Rethinking assumptions in deep anomaly detection. *arXiv preprint arXiv:2006.00339*, 2020. pages 16

[177] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft. Deep semi-supervised anomaly detection, 2020. pages 16

[178] L. Ruff, Y. Zemlyanskiy, R. Vandermeulen, T. Schnake, and M. Kloft. Self-attentive, multi-context one-class classification for unsupervised anomaly detection on text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4061–4071, Florence, Italy, July 2019. Association for Computational Linguistics. pages 5, 33, 106

[179] L. Ruff, Y. Zemlyanskiy, R. Vandermeulen, T. Schnake, and M. Kloft. Self-attentive, multi-context one-class classification for unsupervised anomaly detection on text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4061–4071, 2019. pages 29

[180] V. Saligrama and Z. Chen. Video anomaly detection based on local statistical aggregates. In *2012 IEEE Conference on computer vision and pattern recognition*, pages 2112–2119. IEEE, 2012. pages 29

[181] J. Salminen, C. Kandpal, A. M. Kamel, S. gyo Jung, and B. J. Jansen. Creating and detecting fake reviews of online products. *Journal of Retailing and Consumer Services*, 64:102771, 2022. pages 105, 106

[182] D. Samariya and A. Thakkar. A comprehensive survey of anomaly detection algorithms. *Annals of Data Science*, 10(3):829–850, 2023. pages 19, 25

[183] R. A. Sater and A. B. Hamza. A federated learning approach to anomaly detection in smart buildings. *ACM Trans. Internet Things*, 2(4), aug 2021. pages 67

[184] D. Savage, X. Zhang, X. Yu, P. Chou, and Q. Wang. Anomaly detection in online social networks. *Social Networks*, 39:62–70, 2014. pages 5, 33

[185] V. Schmidt, K. Goyal, A. Joshi, B. Feld, L. Conell, N. Laskaris, D. Blank, J. Wilson, S. Friedler, and S. Luccioni. CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing, 2021. pages 95

[186] P. Schneider and F. Xhafa. Chapter 9 - anomaly detection, classification and cep with ml methods: Machine learning pipeline for medicine. In P. Schneider and F. Xhafa, editors, *Anomaly Detection and Complex Event Processing over IoT Data Streams*, pages 193–233. Academic Press, 2022. pages 104

[187] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based anomaly detection: a new approach for detecting network intrusions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 265–274, 2002. pages 2, 10, 25

[188] S. Seo, D. Seo, M. Jang, J. Jeong, and P. Kang. Unusual customer response identification and visualization based on text mining and anomaly detection. *Expert Systems with Applications*, 144:113111, 2020. pages 106

[189] K. Shaukat, T. M. Alam, S. Luo, S. Shabbir, I. A. Hameed, J. Li, S. K. Abbas, and U. Javed. A review of time-series anomaly detection techniques: A

step to future perspectives. In *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 1*, pages 865–877. Springer, 2021. pages 12

[190] M. Singh, M. Rajan, V. Shivraj, and P. Balamuralidhar. Secure MQTT for internet of things (IoT). In *2015 Fifth International Conference on Communication Systems and Network Technologies*, pages 746–751, 2015. pages 102, 132

[191] B. Song and Y. Suh. Narrative texts-based anomaly detection using accident report documents: The case of chemical process safety. *Journal of Loss Prevention in the Process Industries*, 57:47–54, 2019. pages 106

[192] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. pages 108, 109

[193] B. Staar, M. Lütjen, and M. Freitag. Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*, 79:484–489, 2019. pages 15

[194] L. Stojanovic, M. Dinic, N. Stojanovic, and A. Stojadinovic. Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In *2016 IEEE international conference on big data (big data)*, pages 1647–1652. IEEE, 2016. pages 2, 10, 27

[195] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2828–2837, 2019. pages 15

[196] W. Sultani, C. Chen, and M. Shah. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6479–6488, 2018. pages 29

[197] S. Tabinda Kokab, S. Asghar, and S. Naz. Transformer-based deep learning models for the sentiment analysis of social media data. *Array*, 14:100157, 2022. pages 5, 33

[198] M. Tailanian, Á. Pardo, and P. Musé. U-flow: A u-shaped normalizing flow for anomaly detection with unsupervised threshold. *arXiv preprint arXiv:2211.12353*, 2022. pages 12

[199] K. Taunk, S. De, S. Verma, and A. Swetapadma. A brief review of nearest neighbor algorithm for learning and classification. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 1255–1260, 2019. pages 22

[200] D. M. Tax and R. P. Duin. Support vector data description. *Machine learning*, 54:45–66, 2004. pages 25

[201] D. M. Tax and R. P. Duin. Support vector data description. *Machine Learning*, 54:45–66, 2004. pages 56, 164

[202] G. R. Terrell and D. W. Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992. pages 21

[203] Z. Y. Thean, V. Voon Yap, and P. C. Teh. Container-based MQTT broker cluster for edge computing. In *2019 4th International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, pages 1–6, 2019. pages 84

[204] J. Tian, M. H. Azarian, and M. Pecht. Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm. In *PHM society European conference*, volume 2, 2014. pages 22

[205] P. H. Tran, K. P. Tran, T. T. Huong, C. Heuchenne, P. HienTran, and T. M. H. Le. Real time data-driven approaches for credit card fraud detection. In *Proceedings of the 2018 international conference on e-business and applications*, pages 6–9, 2018. pages 26

[206] Tripadvisor review transparency report 2021. https://www.tripadvisor.com/TransparencyReport2021. pages 104

[207] H. T. Truong, B. P. Ta, Q. A. Le, D. M. Nguyen, C. T. Le, H. X. Nguyen, H. T. Do, H. T. Nguyen, and K. P. Tran. Light-weight federated learning-based anomaly detection for time-series data in industrial control systems. *Computers in Industry*, 140:103692, 2022. pages 104

[208] M. E. Tschuchnig and M. Gadermayr. Anomaly detection in medical imaging-a mini review. In *Data Science–Analytics and Applications: Proceedings of the 4th International Data Science Conference–iDSC2021*, pages 33–38. Springer, 2022. pages 26

[209] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos. Challenges and opportunities in edge computing. In *2016 IEEE international conference on smart cloud (SmartCloud)*, pages 20–26. IEEE, 2016. pages 4, 31

[210] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. pages 108

[211] S. Vempala. The random projection method. *American Mathematical Society*, 2004. pages 35

[212] D. Vidanagama, A. Silva, and A. Karunananda. Ontology based sentiment analysis for fake review detection. *Expert Systems with Applications*, 206:117869, 2022. pages 105, 106

[213] M. E. Villa-Pérez, M. A. Alvarez-Carmona, O. Loyola-Gonzalez, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K.-K. R. Choo. Semi-supervised anomaly detection algorithms: A comparative summary and future research directions. *Knowledge-Based Systems*, 218:106878, 2021. pages 16

[214] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, dec 2010. pages 108

[215] B. von Helversen, K. Abramczuk, W. Kopeć, and R. Nielek. Influence of consumer reviews on online purchasing decisions in older and younger adults. *Decision Support Systems*, 113:1–10, 2018. pages 103

[216] K.-C. Wang, Y. FU, K. Li, A. Khisti, R. Zemel, and A. Makhzani. Variational model inversion attacks. In *Advances in Neural Information Processing Systems*, volume 34, pages 9706–9719. Curran Associates, 2021. pages 83

[217] Y. Wang, J. Wong, and A. Miner. Anomaly intrusion detection using one class svm. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 358–364. IEEE, 2004. pages 24

[218] Y. Wang, J. Wong, and A. Miner. Anomaly intrusion detection using one class svm. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 358–364, 2004. pages 114, 168

[219] S. Węglarczyk. Kernel density estimation and its application. In *ITM web of conferences*, volume 23, page 00037. EDP Sciences, 2018. pages 21

[220] Q. Wei, Y. Ren, R. Hou, B. Shi, J. Y. Lo, and L. Carin. Anomaly detection for medical images based on a one-class classification. In *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, pages 375–380. SPIE, 2018. pages 2, 10, 26

[221] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022. pages 134

[222] Worldline and the ML Group of ULB. Credit card fraud detection, 2014. pages 91

[223] J. Wu, W. Zhang, G. Li, W. Wu, X. Tan, Y. Li, E. Ding, and L. Lin. Weakly-supervised spatio-temporal anomaly detection in surveillance video. *arXiv preprint arXiv:2108.03825*, 2021. pages 12

[224] M. Xie, J. Hu, and B. Tian. Histogram-based online anomaly detection in hierarchical wireless sensor networks. In *2012 IEEE 11th international conference on trust, security and privacy in computing and communications*, pages 751–759. IEEE, 2012. pages 21

[225] R. B. Yadav, P. S. Kumar, and S. V. Dhavale. A survey on log anomaly detection using deep learning. In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pages 1215–1220, 2020. pages 5, 33

[226] J. Yang, R. Xu, Z. Qi, and Y. Shi. Visual anomaly detection for images: A survey. *arXiv preprint arXiv:2109.13157*, 2021. pages 133

[227] L. Yang, B. Tan, V. W. Zheng, K. Chen, and Q. Yang. Federated recommendation systems. *Federated Learning: Privacy and Incentive*, pages 225–239, 2020. pages 4, 32

[228] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019. pages 108

[229] H. Yar, A. S. Imran, Z. A. Khan, M. Sajjad, and Z. Kastrati. Towards smart home automation using iot-enabled edge-computing paradigm. *Sensors*, 21(14):4932, 2021. pages 4, 31

[230] N. Ye and Q. Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17(2):105–112, 2001. pages 20

[231] J. Yi and S. Yoon. Patch svdd: Patch-level svdd for anomaly detection and segmentation. In *Proceedings of the Asian conference on computer vision*, 2020. pages 133

[232] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. S. Shen. Toward efficient content delivery for automated driving services: An edge computing solution. *IEEE Network*, 32(1):80–86, 2018. pages 4, 31

[233] H. Zhang, J. Bosch, and H. H. Olsson. Federated learning systems: Architecture alternatives. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, pages 385–394, 2020. pages 69

[234] J. Zhang, Y. Xie, G. Pang, Z. Liao, J. Verjans, W. Li, Z. Sun, J. He, Y. Li, C. Shen, et al. Viral pneumonia screening on chest x-rays using confidence-

aware anomaly detection. *IEEE transactions on medical imaging*, 40(3):879–890, 2020. pages 26

[235] Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, 1:43–52, 2010. pages 15

[236] Y. Zhang, Y. Xu, S. Wei, Y. Wang, Y. Li, and X. Shang. Doubly contrastive representation learning for federated image recognition. *Pattern Recognition*, 139:109507, 2023. pages 4, 32, 65

[237] Z. Zhang, Y. Song, and H. Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017. pages 23

[238] M. Zhao and J. Chen. A review of methods for detecting point anomalies on numerical dataset. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 559–565. IEEE, 2020. pages 12

[239] Q. Zhao, E. Adeli, N. Honnorat, T. Leng, and K. M. Pohl. Variational autoencoder for regression: Application to brain aging analysis. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part II 22*, pages 823–831. Springer, 2019. pages 23

[240] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu. Multi-task network anomaly detection using federated learning. In *SoICT 2019*, page 273–279. ACM, 2019. pages 67

[241] P. Zheng, S. Yuan, X. Wu, J. Li, and A. Lu. One-class adversarial nets for fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1286–1293, 2019. pages 2, 10

[242] C. Zhou and R. C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017. pages 23

[243] J. T. Zhou, J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh. Anomalynet: An anomaly detection network for video surveillance. *IEEE Transactions on Information Forensics and Security*, 14(10):2537–2550, 2019. pages 29

[244] H. Zhu, J. Xu, S. Liu, and Y. Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021. pages 100

[245] T. Zonta, C. A. Da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, and G. P. Li. Predictive maintenance in the industry 4.0: A systematic literature review. *Computers & Industrial Engineering*, 150:106889, 2020. pages 27

[246] Şule Öztürk Birim, I. Kazancoglu, S. Kumar Mangla, A. Kahraman, S. Kumar, and Y. Kazancoglu. Detecting fake reviews through topic modelling. *Journal of Business Research*, 149:884–900, 2022. pages 105, 106

Appendix A

Hyperparameters used during training

This appendix contains the methods and hyperparameters used during the experiments carried out in this thesis throughout Chapters 3, 4, and 5.

A.1. Traditional scenarios

- One-class Classification via Expanded Non-Convex Hulls (OCENCH) [146].
  - Number of projections (τ).
  - Maximum edge size (l).
  - Expansion hyperparameter (λ).
  - Subdivision (boolean value)
- Non-Convex Boundary over Projections (NCBoP) [87].
  - Number of projections.
- Autoencoder (AE) [11].
  - Network architecture.
  - Epochs.

  - Contamination of the data set (c).

- Isolation Forest (IF) [118].

  - The number of base estimators in the ensemble.
  - Contamination of the data set (c).

- Local Outlier Factor (LOF) [22].

  - Number of neighbors.
  - Contamination of the data set (c).

- One-Class Support Vector Machine (OCSVM) [167].

  - An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors (ν).
  - Kernel type: Linear, Polynomial or RBF.
  - Kernel coefficient γ (in the case of polynomial and RBF kernels).
  - Degree (in the case of polynomial kernel).

- Robust Covariance (RC) [159].

  - Contamination of the data set (c).

- Support Vector Data Description (SVDD) [201].

  - Positive penalty.
  - Negative penalty.
  - Kernel type: Linear, Polynomial or Gaussian.
  - Width (in the case of gaussian kernel).
  - Degree (in the case of polynomial kernel).

Dataset	OCENCH	NCBoP	AE	IF	LOF	OCSVM	RC	SVDD
Annthyroid	Proj: 500, l: 0.2, Extend: 0, Subdivision: True	Proj: 10000	Layers: [10, 10, 8, 10, 10], Epochs: 100, c: 0.4	Estimators: 100, c: 0.2	Neighbors: 15, c: 0.4	ν: 0.2, γ: 0.17, Kernel: Polynomial, Degree: 5	c: 0.2	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian, Width: 8
Thyroid	Proj: 200, l: 1, Extend: 0.05, Subdivision: False	Proj: 500	Layers: [5, 2, 5], Epochs: 100 , c: 0.3	Estimators: 200, c: 0.05	Neighbors: 15, c: 0.1	ν: 0.5, γ: 8, Kernel: RBF	c: 0.05	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian, Width: 8
Shuttle	Proj: 50, l: 1.5, Extend: 0.001, Subdivision: True	Proj: 50	Layers: [6, 4, 6], Epochs: 100, c: 0.3	Estimators: 200, c: 0.01	Neighbors: 100, c: 0.1	ν: 0.05, γ: 2^{-4}, Kernel: RBF	c: 0.05	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian Width: 8
Telescope	Proj: 200, l: 0.3, Extend: 0.0005, Subdivision: True	Proj: 2000	Layers: [10, 5, 10], Epochs: 100, c: 1	Estimators: 5, c: 0.2	Neighbors: 15, c: 0.2	ν: 0.1, γ: 4, Kernel: poly, Degree: 5	c: 0.6	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian, Width: 8
Pendigits	Proj: 100, l: 20, Extend: 0, Subdivision: False	Proj: 50	Layers: [10, 8, 6, 8, 10] Epochs: 100, c: 0.4	Estimators: 200, c: 0.1	Neighbors: 15, c: 0.1	ν: 0.05, γ: 1, Kernel: RBF	c: 0.35	Pos.: 0.05, Neg.: 0.05, Kernel: Gaussian, Width: 4
Cardio	Proj: 200, l: 20, Extend: 0.3, Subdivision: False	Proj: 50	Layers: [5, 2, 5], Epochs: 100 c: 0.3	Estimators: 200, c: 0.2	Neighbors: 100, c: 0.2	ν: 0.05, γ: 0.06, Kernel: RBF	c: 0.35	Pos.: 0.1, Neg.: 0.1, Kernel: Gaussian Width: 2^{-4}
Ionosphere	Proj: 5, l: 0.9, Extend: 0.1, Subdivision: False	Proj: 10	Layers: [10, 5, 10], Epochs: 100, c: 0.3	Estimators: 15, c: 0.2	Neighbors: 15, c: 0.35	ν: 0.05, γ: 2^{-3}, Kernel: RBF	c: 0.2	Pos.: 0.05, Neg.: 0.05, Kernel: Gaussian Width: 1
Miniboone	Proj: 200, l: 0.2, Extend: 0, Subdivision: True	Proj: 5000	Layers: [10, 5, 10], Epochs: 100, c: 0.3	Estimators: 100, c: 0.2	Neighbors: 100, c: 0.5	ν: 0.05, γ: 2^{-10}, Kernel: RBF	c: 0.8	Pos.: 0.1, Neg.: 0.1, Kernel: Linear
Optdigit	Proj: 20, l: 0.5, Extend: 0.05, Subdivision: True	Proj: 5000	Layers: [10, 7, 5, 7, 10], Epochs: 100, c: 1	Estimators: 50, c0.2	Neighbors: 50, c: 0.1	ν: 0.1, γ: 2^{-9}, Kernel: RBF	c: 0.5	Pos.: 0.1, Neg.: 0.1, Kernel: Linear
MNIST	Proj: 200, l: 1.6, Extend: 0.05, Subdivision: False	Proj: 2000	Layers: [10, 7, 5, 7, 10], Epochs: 100, c: 1	Estimators: 200, c: 0.2	Neighbors: 15, c: 0.1	ν: 0.1, γ: 2^{-4}, Kernel: RBF	c: 0.2	Pos.: 0.4, Neg.: 0.4, Kernel: Linear

Table A.1: Hyperparameters used for training.

A.2. Edge computing and federated learning scenarios

- Deep Autoencoder for Federated learning (DAEF) [147].
  - Architecture: Neurons per layer.
  - λ_{hid}: Regularization hyperparameter of the hidden layer.
  - λ_{last}: Regularization hyperparameter of the last layer.
  - μ: Anomaly threshold.
- Online Sequential Extreme Learning Machine (OS-ELM) [114].
  - Architecture: Neurons per layer.
  - μ: Anomaly threshold.
- Iterative Single Hidden Layer Autoencoder (SHL-AE) [11].
  - Architecture: Neurons per layer.
  - Epochs: Training epochs.
  - μ: Anomaly threshold.
- Iterative Multiple Hidden Layer Autoencoder (MHL-AE) [11].
  - Architecture: Neurons per layer.
  - Epochs: Training epochs.
  - μ: Anomaly threshold.

Dataset	DAEF Ortho.	DAEF Random	DAEF Xavier	OS-ELM	SHL-AE	MHL-AE
Shuttle	Arch: [9, 5, 7, 9], λ_{hid}: 0.01, λ_{last}: 0.8, μ: outlier IQR	Arch: [9, 5, 7, 9], λ_{hid}: 0.8, λ_{last}: 0.8, μ: outlier IQR	Arch: [9, 5, 7, 9], λ_{hid}: 0.8, λ_{last}: 0.3, μ: outlier IQR	Arch: [9, 7, 9], Batch: 100, μ: extreme IQR	Arch: [9, 5, 9] Epochs: 100, μ: extreme IQR	Arch: [9, 7, 5, 3, 5, 7, 9] Epochs: 200, μ: extreme IQR
Covertype	Arch: [10, 6, 8, 10], λ_{hid}: 0.1, λ_{last}: 0.9, μ: P_{80}	Arch: [10, 6, 8, 10] , λ_{hid}: 0.05, λ_{last}: 0.1, μ: P_{80}	Arch: [10, 6, 8, 10], λ_{hid}: 0.005, λ_{last}: 0.8, μ: P_{80}	Arch: [10, 8, 10], Batch: 500, μ: P_{80}	Arch: [10, 6, 10] Epochs: 10, μ: P_{70}	Arch: [10, 8, 6, 4, 6, 8, 10] Epochs: 10, μ: P_{90}
Pendigits	Arch: [16, 5, 10, 16], λ_{hid}: 0.3, λ_{last}: 0.8, μ: P_{40}	Arch: [16, 5, 10, 16], λ_{hid}: 0.01, λ_{last}: 0.8, μ: P_{40}	Arch: [16, 5, 10, 16], λ_{hid}: 0.8, λ_{last}: 0.3, μ: P_{60}	Arch: [16, 12, 16], Batch: 100, μ: P_{80}	Arch: [16, 8, 16] Epochs: 50, μ: P_{80}	Arch: [16, 12, 8, 4, 8, 12, 16] Epochs: 100, μ: outlier IQR
Cardio	Arch: [21, 10, 15, 21], λ_{hid}: 0.9, λ_{last}: 0.9, μ: P_{90}	Arch: [21, 10, 15, 21], λ_{hid}: 0.9, λ_{last}: 0.7, μ: P_{90}	Arch: [21, 10, 15, 21], λ_{hid}: 0.9, λ_{last}: 0.2, μ: P_{80}	Arch: [21, 5, 21], Batch: 100, μ: P_{80}	Arch: [21, 5, 21] Epochs: 100, μ: P_{90}	Arch: [21, 15, 10, 5, 10, 15, 21] Epochs: 90, μ: P_{90}
Credit card	Arch: [29, 15, 20, 25, 29], λ_{hid}: 0.9, λ_{last}: 0.7, μ: outlier IQR	Arch: [29, 15, 20, 25, 29], λ_{hid}: 0.005, λ_{last}: 0.9, μ: P_{90}	Arch: [29, 15, 20, 25, 29], λ_{hid}: 0.01, λ_{last}: 0.9, μ: outlier IQR	Arch: [29, 25, 29], Batch: 100, μ: extreme IQR	Arch: [29, 10, 29] Epochs: 25, μ: outlier IQR	Arch: [29, 20, 10, 20, 29] Epochs: 10, μ: extreme IQR
Ionosphere	Arch: [33, 20, 25, 33], λ_{hid}: 0.005, λ_{last}: 0.9, μ: outlier IQR	Arch: [33, 20, 25, 33], λ_{hid}: 0.01, λ_{last}: 0.8, μ: outlier IQR	Arch: [33, 20, 25, 33], λ_{hid}: 0.005, λ_{last}: 0.8, μ: P_{90}	Arch: [33, 20, 33], Batch: 100, μ: extreme IQR	Arch: [33, 15, 33] Epochs: 100, μ: outlier IQR	Arch: [33, 25, 20, 15, 20, 25, 33] Epochs: 50, μ: extreme IQR
Optdigit	Arch: [62, 20, 30, 40, 50, 62], λ_{hid}: 0.01, λ_{last}: 0.8, μ: P_{40}	Arch: [62, 30, 40, 50, 62], λ_{hid}: 0.01, λ_{last}: 0.9, μ: P_{50}	Arch: [62, 20, 30, 40, 50, 62], λ_{hid}: 0.01, λ_{last}: 0.3, μ: P_{40}	Arch: [62, 20, 62], Batch: 100, μ: P_{60}	Arch: [62, 30, 62] Epochs: 10, μ: P_{80}	Arch: [62 ,50, 40, 30, 20, 30, 40 ,50, 62] Epochs: 25, μ: P_{90}

Table A.2: Hyperparameters used during the experimentation.

A.3. Natural language scenarios

- Deep Autoencoder for Federated learning (DAEF) [147].
  - Architecture: Neurons per layer.
  - λ_{hid}: Regularization hyperparameter of the hidden layer.
  - λ_{last}: Regularization hyperparameter of the last layer.
  - μ: Anomaly threshold.
- Online Sequential Extreme Learning Machine (OS-ELM) [114].
  - Architecture: Neurons per layer.
  - μ: Anomaly threshold.
- One-Class Support Vector Machine (OC-SVM) [218].
  - An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors (ν).
  - Kernel type: Linear, Polynomial or RBF.
  - Kernel coefficient γ (in the case of polynomial and RBF kernels).
  - Degree (in the case of polynomial kernel).
- Isolation Forest (IF) [118].
  - The number of base estimators in the ensemble.
  - Contamination of the dataset (c).
- Local Outlier Factor (LOF) [22].
  - Number of neighbors.
  - Contamination of the dataset (c).

Normal class	DAEF	OS-ELM	OC-SVM	IF	LOF
Chocolate bars	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: outlier IQR	Arch: [768, 400, 768], μ: extreme IQR	ν: 0.1, kernel: linear, γ: scale	Estimators: 1000, c: 0.05	Neighbors: 2000, c: 0.05
Colored pencils	Arch: [768, 550, 650, 768], λ_{hid}: 0.1, λ_{last}: 0.1, μ: outlier IQR	Arch: [768, 500, 768] , μ: extreme IQR	ν: 0.1, kernel: linear, γ: scale	Estimators: 300, c: 0.1	Neighbors: 4000, c: 0.05
Gaming mouse	Arch: [768, 550, 650, 768], λ_{hid}: 0.1, λ_{last}: 0.1, μ: outlier IQR	Arch: [768, 400, 768], μ: extreme IQR	ν: 0.1, kernel: poly, degree: 2, γ: scale	Estimators: 1000, c: 0.1	Neighbors: 2000, c: 0.05
Bluetooth speaker	Arch: [768, 550, 650, 768], λ_{hid}: 0.75, λ_{last}: 0.1, μ: outlier IQR	Arch: [768, 300, 768], μ: outlier IQR	ν: 0.1, kernel: poly, degree: 3, γ: scale	Estimators: 1000, c: 0.1	Neighbors: 4000, c: 0.1
Foot insoles	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: outlier IQR	Arch: [768, 300, 768], μ: extreme IQR	ν: 0.1, kernel: poly, degree: 2, γ: scale	Estimators: 1000, c: 0.05	Neighbors: 2000, c: 0.05
Hamilton album	Arch: [768, 550, 650, 768], λ_{hid}: 0.75, λ_{last}: 0.1, μ: outlier IQR	Arch: [768, 100, 768], μ: outlier IQR	ν: 0.1, kernel: poly, degree: 2, γ: scale	Estimators: 200, c: 0.1	Neighbors: 1000, c: 0.1
Hygrometer	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: outlier IQR	Arch: [768, 200, 768], μ: outlier IQR	ν: 0.1, kernel: poly, degree: 2, γ: scale	Estimators: 1000, c: 0.1	Neighbors: 4000, c: 0.1

Table A.3: Hyperparameters used during the 1vs.6 experimentation.

Normal class	DAEF	OS-ELM	OC-SVM	IF	LOF
Chocolate bars	Arch: [768, 550, 650, 768], λ_{hid}: 0.1, λ_{last}: 0.1, μ: Q_{90}	Arch: [768, 500, 768], μ: outlier IQR	ν: 0.1, kernel: poly, degree: 4, γ: scale	Estimators: 1000, c: 0.1	Neighbors: 4000, c: 0.1
Anise seeds	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: Q_{80}	Arch: [768, 400, 768] , μ: outlier IQR	ν: 0.1, kernel: poly, degree: 4, γ: scale	Estimators: 1000, c: 0.2	Neighbors: 3000, c: 0.2
Colored pencils	Arch: [768, 550, 650, 768], λ_{hid}: 0.25, λ_{last}: 0.25, μ: outlier IQR	Arch: [768, 500, 768], μ: extreme IQR	ν: 0.1, kernel: linear, γ: scale	Estimators: 500, c: 0.05	Neighbors: 4000, c: 0.05
Ergonomic cushion	Arch: [768, 550, 650, 768], λ_{hid}: 0.5, λ_{last}: 0.1, μ: outlier IQR	Arch: [768, 200, 768], μ: extreme IQR	ν: 0.1, kernel: sigmoid, γ: scale	Estimators: 500, c: 0.05	Neighbors: 3000, c: 0.05
Gaming mouse	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: Q_{90}	Arch: [768, 500, 768]], μ: extreme IQR	ν: 0.1, kernel: poly, degree: 2, γ: scale	Estimators: 1000, c: 0.1	Neighbors: 3000, c: 0.1
PS4 membership	Arch: [768, 550, 650, 768], λ_{hid}: 0.1, λ_{last}: 0.1, μ: Q_{90}	Arch: [768, 400, 768], μ: extreme IQR	ν: 0.1, kernel: poly, degree: 2, γ: scale	Estimators: 1000, c: 0.2	Neighbors: 1000, c: 0.2
Bluetooth speaker	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: Q_{90}	Arch: [768, 20, 768] , μ: Q_{90}	ν: 0.1, kernel: poly, degree: 2, γ: scale	Estimators: 1000, c: 0.1	Neighbors: 4000, c: 0.1
Wi-Fi range extender	Arch: [768, 550, 650, 768], λ_{hid}: 0.25, λ_{last}: 0.1, μ: Q_{90}	Arch: [768, 500, 768], μ: outlier IQR	ν: 0.1, kernel: poly, degree: 4, γ: scale	Estimators: 1000, c: 0.1	Neighbors: 4000, c: 0.1
Foot insoles	Arch: [768, 550, 650, 768], λ_{hid}: 0.75, λ_{last}: 0.1, μ: outlier IQR	Arch: [768, 20, 768], μ: extreme IQR	ν: 0.1, kernel: poly, degree: 4, γ: scale	Estimators: 1000, c: 0.2	Neighbors: 500, c: 0.2
Yoga leggings	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: Q_{90}	Arch: [768, 20, 768]], μ: extreme IQR	ν: 0.1, kernel: poly, degree: 4, γ: scale	Estimators: 500, c: 0.2	Neighbors: 300, c: 0.2
Hamilton album	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: Q_{90}	Arch: [768, 100, 768]], μ: Q_{80}	ν: 0.1, kernel: poly, degree: 4, γ: scale	Estimators: 200, c: 0.2	Neighbors: 50, c: 0.2
Partners album	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: Q_{90}	Arch: [768, 50, 768]], μ: Q_{80}	ν: 0.2, kernel: poly, degree: 4, γ: scale	Estimators: 1000, c: 0.2	Neighbors: 300, c: 0.2
Hygrometer	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: Q_{90}	Arch: [768, 100, 768]], μ: Q_{90}	ν: 0.1, kernel: poly, degree: 4, γ: scale	Estimators: 1000, c: 0.2	Neighbors: 4000, c: 0.2
Vacuum	Arch: [768, 550, 650, 768], λ_{hid}: 0.9, λ_{last}: 0.9, μ: Q_{90}	Arch: [768, 300, 768]], μ: outlier IQR	ν: 0.1, kernel: poly, degree: 3, γ: scale	Estimators: 300, c: 0.1	Neighbors: 4000, c: 0.1

Table A.4: Hyperparameters used during the 1vs.1 experimentation.

Appendix B

Publications supporting this thesis

The contents of the present research have been published in the following specialized journals and conferences.

Journal publications

- D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas (2023). A one-class classification method based on expanded non-convex hulls. *Information Fusion*. JCR Q1.

- D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas (2023). Fast deep autoencoder for federated learning. *Pattern Recognition*. JCR Q1.

- D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas. Explained anomaly detection in text reviews: Can subjective scenarios be correctly evaluated?. *Engineering Applications of Artificial Intelligence*. JCR Q1.

International Conferences

- D. Novoa-Paradela, O. Fontenla-Romero, B. Guijarro-Berdiñas, and D. Orellana-Cañas (2023, December). A Federated Learning architecture for Anomaly Detection on the Edge using Deep Autoencoders. 31th IEEE International

Conference on Enabling Technologies Infrastructure for Collaborative Enterprises (WETICE-2023). Paris, France.

National Conferences

- D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas (2020, October). Adaptive Real-Time Method for Anomaly Detection Using Machine Learning. In 3rd XoveTIC Conference, MDPI Proceedings 2020, A Coruña, Spain.
- D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas (2021, September). Characterization of point datasets using subdivisible non-convex hulls. In Proceedings of the XIX Conference of the Spanish Association for Artificial Intelligence (CAEPIA), Malaga, Spain.

Works related to the thesis.

- D. Novoa-Paradela, O. Fontenla-Romero, and B. Guijarro-Berdiñas (2020, July). Online learning for anomaly detection via subdivisible convex hulls. In 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, Scotland, United Kingdom.
- D. Novoa-Paradela, C. Eiras-Franco, O. Fontenla-Romero, and F. Lamas-López (2020, November). Predictive maintenance of ship engines through machine learning. In VIII Spanish National Defense and Security R+D Congress (DESEi+d), León, Spain.
- D. Fernández-Barrero, O. Fontenla-Romero, F. Lamas-López, D. Novoa-Paradela, M. D. R-Moreno, and D. Sanz (2021). Soprene: Assessment of the Spanish Armada's Predictive Maintenance Tool for Naval Assets. *Applied Sciences*. JCR Q3.
- F. Lamas-López, D. Novoa-Paradela, C. Eiras-Franco, and O. Fontenla-Romero (2021, October). Predictive Maintenance of Naval Assets Using Machine Learning Techniques. In The 15th NATO Operations Research & Analysis Conference (OR&A).

Mentions

- Best paper award in the III XoveTIC Congress (2020) with the work "Adaptive Real-Time Method for Anomaly Detection Using Machine Learning". A Coruña, Spain.
- Isdefe I+D+i "Antonio Torres" award for the best paper at the VIII Spanish National Defense and Security R+D Congress (DESEi+d 2020) with the work "Predictive maintenance of ship engines through machine learning". León, Spain.
- Third Doctoral Consortium award at the Artificial Intelligence Summer School (EVIA 2023) for the thesis project "Machine learning for anomaly detection: From surface to deep". Sevilla, Spain.

Appendix C

Resumen extendido

En los últimos años, la sociedad ha presenciado una explosión sin precedentes en la generación, intercambio y almacenamiento de datos. Este inmenso volumen de información, conocido como *Big Data*, ha transformado la forma en que vivimos, trabajamos y nos relacionamos. La aparición de esta tendencia se debe a tres razones principales:

- Mayor capacidad computacional: Los avances tecnológicos han dado lugar a la aparición de dispositivos con una gran potencia de cálculo y capacidad de almacenamiento. Esta mayor capacidad computacional ha permitido el procesamiento eficiente de grandes conjuntos de datos y el desarrollo de algoritmos de aprendizaje automático más sofisticados.
- Aumento en el número de dispositivos: La disminución de los costos de estos dispositivos ha permitido su uso generalizado, lo que ha dado lugar a millones de fuentes de datos de todo tipo y por tanto a oportunidades para aplicaciones de análisis de datos en gran variedad de campos. La diversidad existente entre estas fuentes de datos demanda la adopción de nuevas perspectivas.
- Conectividad global: La expansión de Internet ha generado una red global de generación e intercambio de datos que conecta todos estos dispositivos, generando un flujo constante de información.

Este ecosistema rico en datos es ideal para áreas de la informática como el Apren-

dizaje Automático, cuyos algoritmos se nutren directamente de estas fuentes de información. Es debido a esto que el campo de la Inteligencia Artificial en general y el aprendizaje automático en particular han experimentado un impulso significativo en la última década. Entre los problemas resueltos por estos sistemas se encuentra la Detección de Anomalías.

La detección de anomalías es una rama del aprendizaje automático que se encarga de construir modelos capaces de diferenciar entre datos normales y anómalos. A primera vista, la detección de anomalías puede parecer reducible a un problema de clasificación en dos clases. Sin embargo, dado que las anomalías suelen ser eventos poco frecuentes, en estos escenarios es común que los datos normales predominen, lo que requiere del desarrollo de modelos capaces de ser entrenados exclusivamente con datos normales.

Al aprender los patrones y características inherentes a la clase normal, un modelo de detección de anomalías es capaz de discernir instancias que se desvían significativamente de la distribución considerada como normal de aquellas que no, lo que resulta fundamental en diversos dominios y aplicaciones. Algunas de las más destacadas son la detección de fraudes, la ciberseguridad, la medicina y el mantenimiento de sistemas industriales. En numerosos escenarios del mundo real, la identificación temprana de anomalías es de suma importancia, como en el caso de fallos en sistemas industriales o intrusiones en redes. En otros contextos, como la atención médica y la banca, la privacidad de los datos es una preocupación primordial, mientras que en escenarios que involucran dispositivos IoT (Internet of Things) se busca un bajo costo computacional y de ancho de banda.

A pesar de ser una disciplina exitosa y con un gran uso en la actualidad, la detección de anomalías, al igual que otras áreas del aprendizaje automático, es un área en constante expansión que se enfrenta a numerosos retos. A continuación se presentan los cinco retos principales que motivan esta tesis doctoral:

- Distribuciones no convexas y disjuntas: La forma de la distribución de la clase normal puede ser crítica para la elección de la técnica de detección de anomalías. Cuando esta distribución no es convexa, los métodos que emplean representaciones convexas no pueden caracterizar adecuadamente a la clase normal. Además, los datos pueden contener múltiples regiones separadas en

el espacio, lo que complica su representación con un solo cierre. Asimismo, calcular los límites de los cierres en el espacio original puede llegar a ser una tarea inabordable computacionalmente debido a la alta dimensionalidad de los datos, por lo que es habitual recurrir a técnicas que reduzcan dicha dimensinalidad. Ya sea en el espacio original o en dimensiones reducidas, los modelos de detección de anomalías deben ser capaces de representar regiones no convexas y disjuntas, lo que exige el desarrollo de técnicas específicas para abordar este escenario.

- Computación en el borde: El avance tecnológico ha llevado a un incremento significativo en el número de pequeños dispositivos capaces de realizar tareas antes consideradas inaccesibles, como entrenar modelos de aprendizaje automático, inferir en tiempo real o transferir grandes volúmenes de datos a alta velocidad. Esta proliferación ha dado lugar a un nuevo enfoque de cómputo conocido como computación en el borde, que traslada la computación desde los centros de datos en la nube hasta el borde de la red, acercando los servicios y utilidades al usuario final y sus dispositivos. Esto ofrece ventajas como un procesamiento más rápido, menor latencia y un uso más eficiente del ancho de banda de la red. Sin embargo, la computación en el borde también presenta desafíos, como la limitación de recursos, la heterogeneidad de los dispositivos y redes, la seguridad de los datos, la tolerancia a fallos y la escalabilidad. Para abordar estos desafíos, es esencial adaptar los algoritmos de detección de anomalías existentes o diseñar nuevos métodos para su uso en estos entornos de computación en el borde.

- Aprendizaje federado: Desde el punto de vista del aprendizaje automático, el escenario tecnológico actual plagado de dispositivos es muy adecuado para el uso de aprendizaje federado, un esquema colaborativo de aprendizaje que permite que dispositivos heterogéneos con diferentes conjuntos de datos privados trabajen juntos para construir un modelo global. Los desafíos en el aprendizaje federado abarcan la protección de la privacidad, el coste de la comunicación, la heterogeneidad de los dispositivos involucrados y el manejo de modelos poco confiables. Las preocupaciones sobre la privacidad surgen debido a la posible fuga de información privada durante la comunicaciones de los dispositivos. Además, la eficiencia de estas comunicaciones se vuelve crucial cuando se trata

de redes federadas de gran escala. La heterogeneidad de los sistemas, incluidas las variaciones de hardware y de red, también presenta grandes desafíos como la tolerancia a fallos. Además, evitar la carga de modelos no confiables es esencial para mantener la integridad del aprendizaje federado. Abordar estos desafíos requiere de la adaptación de métodos tradicionales así como el desarrollo de nuevos métodos específicos.

- Procesamiento del lenguaje natural: Aunque existen modelos capaces de tratar directamente con imágenes o texto, los métodos clásicos de aprendizaje automático generalmente están diseñados para trabajar con datos estructurados. En el área del procesamiento del lenguaje natural existen múltiples técnicas para representar texto utilizando vectores de números reales, conocidos como *embeddings*. Estas técnicas permiten generar espacios vectoriales que representan las relaciones semánticas del lenguaje, de modo que, por ejemplo, dos palabras sinónimas se encontrarán a una distancia más cercana que dos palabras no relacionadas. El desarrollo tecnológico de los últimos años ha permitido la construcción de modelos muy potentes, sin embargo, a diferencia de otras tareas, la aplicación de la detección de anomalías en textos no es habitual, probablemente debido a las dificultades a la hora de definir el concepto de anomalía en este tipo de escenarios. Acercar la detección de anomalías a este campo es un ejercicio que puede resultar beneficioso para muchas aplicaciones, como la ciberseguridad, el análisis de redes sociales o de comentarios de clientes.

- Explicabilidad: El uso de modelos de aprendizaje automático en contextos críticos ha disparado la demanda de transparencia para evitar la toma de decisiones basadas en modelos de caja negra. El peligro de estos modelos radica en que sus decisiones no son justificables, legítimas o no permiten explicaciones detalladas de su comportamiento. Las explicaciones que respaldan el resultado de un modelo son cruciales, especialmente en escenarios que pueden afectarnos directamente como humanos, como la medicina o la conducción autónoma. La creciente demanda de transparencia, interpretabilidad y confianza en los sistemas incluye también a los modelos de detección de anomalías, por lo que es necesario incorporar esta característica en el desarrollo de nuevos algoritmos, así como estudiar cómo evaluar la calidad de dichas explicaciones.

En esta tesis, hemos decidido desarrollar métodos de detección de anomalías que aborden los aspectos mencionados anteriormente. En primer lugar, y debido a que el campo de la detección de anomalías presenta ciertas características que lo hacen único en comparación con otras tareas de aprendizaje automático, presentamos un capítulo que sirve de introducción al área. En este se describe en detalle la tarea de detección de anomalías, así como a sus métodos, aplicabilidad y limitaciones. En segundo lugar, proponemos un nuevo método de detección de anomalías basado en cierres no convexos capaz de tratar con formas no convexas e inconexas, además de ser paralelizable. En este capítulo se presentan los fundamentos teóricos del método, así como su implementación, evaluación y conclusiones. En tercer lugar, presentamos una nueva implementación de redes *autoencoder* profundas no iterativas válidas para escenarios de aprendizaje federado y computación de borde. Además, en este capítulo se propone una arquitectura federada semi-descentralizada para utilizar estas redes en los escenarios mencionados. La red propuesta es evaluada tanto en escenarios clásicos como en dichos escenarios de aprendizaje federado. Finalmente, en el último capítulo nos hemos centrado en acercar la detección de anomalías al campo del lenguaje natural proponiendo un *pipeline* para la detección de reseñas anómalas en plataformas de comercio electrónico, utilizando modelos de explicabilidad para justificar las decisiones tomadas por el sistema. Este capítulo se centra en cómo tratar el escenario propuesto como un problema de detección de anomalías, cómo transformar las reseñas de texto originales a un formato adecuado para el uso de algoritmos de aprendizaje automático, cómo generar explicaciones asociadas a las clasificaciones llevadas a cabo por estos modelos y cómo evaluar su calidad.

El primer método presentado en esta tesis es OCENCH (One-class Classification via Expanded Non-Convex Hulls), un método de clasificación de una sola clase y detección de anomalías basado en el uso de proyecciones aleatorias del conjunto de datos normal para reducir su dimensionalidad, seguido de un proceso basado en la triangulación de Delaunay para representar geométricamente a la clase normal en cada uno de estos espacios de menor dimensión. Concretamente, los datos son proyectados a espacios bidimensionales, de manera que, empleando un número suficiente de proyecciones, la información original es preservada. En estos espacios 2D, el cálculo de operaciones como la triangulación de Delaunay a partir de la cual obtenemos los cierres iniciales es muy rápida y eficiente. Partiendo de estos cierres iniciales y mediante un proceso iterativo de poda, subdivisión y expansión, los

límites de los cierres se ajustan a la forma de la distribución normal de manera independiente en cada espacio 2D, pudiendo representar tanto formas no convexas como disjuntas. Una vez el entrenamiento se ha completado, el modelo puede clasificar un nuevo dato como normal o anómalo proyectándolo en cada una de estos espacios 2D y comprobando si en alguno de ellos el dato se encuentra fuera de los límites de la clase normal. Si en alguno de estos espacios el punto se encuentra fuera, el dato será clasificado como anómalo, mientras que si permanece dentro de los límites de la clase normal en todas las proyecciones, el dato será clasificado como normal. La selección del número de proyecciones aleatorias a utilizar, así como los parámetros que manejan el proceso de ajuste de los cierres no presenta grandes problemas para el usuario, y este trabajo está acompañado de un estudio sobre el impacto de estos hiperparámetros en el rendimiento y tiempo de entrenamiento del modelo. El método solamente precisa datos normales para ser entrenado y no es necesario conocer *a priori* el porcentaje de anomalías presente en el conjunto de datos. Los modelos generados con este algoritmo requieren poca memoria ya que no es necesario almacenar todos los datos utilizados para la creación de los cierres no convexos, el modelo solo necesita la información de los vértices de cada cierre final para determinar en producción si los nuevos datos a clasificar se encuentran dentro o fuera de los límites de la clase normal. Además, los cálculos de cada proyección se realizan de forma independiente, por lo que su paralelización permite reducir drásticamente el tiempo de ejecución de las fases de entrenamiento y clasificación. OCENCH permite trabajar con conjuntos de datos no convexos de una forma novedosa, ofreciendo un comportamiento robusto y un rendimiento notable, posicionándose así como una alternativa tanto para problemas convexos como no convexos.

El segundo método propuesto es DAEF (Deep AutoEncoder for Federated learning), un *autoencoder* profundo alternativo a las redes tradicionales debido a su entrenamiento no iterativo. El principal objetivo de DAEF es aprender una representación comprimida de los datos normales conocida como espacio latente y, a partir de dicho espacio reducido, reconstruir la entrada a la salida de la red. El error de reconstrucción proveniente de las diferencia entre la entrada y la reconstrucción nos permite determinar si un dato es normal o anómalo. Ya que el *autoencoder* es entrenado total o mayoritariamente con datos normales, un dato anómalo presentará un error de reconstrucción más elevado que un dato normal. Para llevar este proceso a cabo, DAEF emplea una arquitectura *encoder-decoder* asimétrica. En

primer lugar, el *encoder* de una sola capa reduce la dimensionalidad de los datos de entrada y se ajusta mediante una implementación distribuida de Descomposicion en Valores Singulares. Tras esto, se emplea un *decoder* multicapa para reconstruir la señal de entrada a la salida, entrenado capa a capa mediante un proceso no iterativo inspirado en las Máquinas de Aprendizaje Extremo, también de manera distribuida. Estas características hacen que el entrenamiento de DAEF se pueda llevar a cabo de forma incremental (agregación de modelos), de forma distribuida (entrenamiento compartido entre múltiples nodos), y en paralelo (a nivel de nodo si dispone de varios núcleos), y debido a su formulación matemática, la información que intercambia durante estos procesos no pone en riesgo la privacidad de los datos de entrenamiento. Todo esto hace de DAEF un método útil para escenarios de computación en el borde y aprendizaje federado, siendo capaz de realizar tareas como la detección de anomalías en grandes conjuntos de datos mientras mantiene el rendimiento de los *autoencoders* tradicionales (iterativos) en un menor tiempo. Asimismo, a este trabajo le acompaña una propuesta de arquitectura semi-centralizada para el uso de DAEF en entornos federados. La arquitectura ha sido definida en detalle junto con el protocolo de comunicación a emplear y su interacción. También se llevaron a cabo las pruebas pertinentes para evaluar el rendimiento y escalabilidad del sistema en este tipo de escenarios federados.

En tercer lugar, se ha propuesto un *pipeline* para la detección de reseñas anómalas en productos de Amazon de una manera explicable, pudiendo ser extrapolado directamente a otras plataformas de comercio electrónico u otros escenarios con características similares. La representación de las reseñas se ha llevado a cabo mediante *embeddings* generados con el modelo *transformer* MPNet, lo que ha permitido el entrenamiento de algoritmos clásicos de detección de anomalías que han logrado un muy buen rendimiento. Estos modelos son entrenados únicamente con las reseñas asociadas a un producto objetivo, permitiendo clasificar a las reseñas como normales si tratan del producto a modelar, o como anómalas si en realidad están relacionadas solamente con otros productos diferentes o son demasiado genéricas. Cada clasificación asociada a una reseña está a su vez acompañada por un *score* de normalidad que puede ser de utilidad para llevar a cabo tareas como su ordenación de cara al usuario en la plataforma web. Para dotar de explicabilidad al sistema se ha propuesto una técnica basada en la ocurrencia de términos frecuentes para generar explicaciones asociadas a las clasificaciones. La técnica se basa en la idea de

que las reseñas consideradas como normales para un producto tenderán a emplear ciertos términos con mayor frecuencia. La ocurrencia de dichos términos en el texto de la reseña, o la aparición de términos cercanos semánticamente a estos términos frecuentes, justificarían la clasificación de una reseña como normal o como anómala. Para mejorar la calidad de esta lista de términos frecuentes asociados a un producto eliminamos los términos frecuentes comunes a otros productos, de modo que los términos más genéricos no son utilizados a la hora de explicar las clasificaciones. Esta técnica ha sido comparada con SHAP, una de las técnicas *post-hoc* de referencia en el campo de la explicabilidad, y con GPT-3, por su gran potencia y versatilidad. Para evaluar la explicabilidad del sistema hemos realizado una encuesta compuesta por dos partes en la que participaron 241 miembros de la comunidad universitaria. En dicha encuesta se ha cuantificado el efecto de las explicaciones en los usuarios y su preferencia personal a la hora de escoger un modelo de explicabilidad u otro. A lo largo del trabajo hemos encontrado grandes dificultades a la hora de definir el concepto de normalidad y anormalidad en escenarios de este tipo debido a la alta subjetividad del mismo, al contrario que ocurre en otros entornos como la detección de anomalías en sistemas industrales. Este mismo aspecto también ha hecho de la evaluación de las explicaciones un reto importante que motivó la realización de una evaluación manual (humana) mediante la encuesta antes mencionada.

En conclusión, esta tesis ha contribuido a avanzar en el campo de la detección de anomalías enfrentando desafíos clave y proponiendo soluciones efectivas. El abordaje de los cinco desafíos presentados al comienzo de este trabajo demuestra la capacidad de innovar y encontrar soluciones adaptadas a las cambiantes necesidades de la sociedad y la tecnología desde el ámbito del aprendizaje automático. Los métodos desarrollados durante esta tesis doctoral presentan aplicaciones potenciales en una variedad de dominios y escenarios reales, lo que demuestra su relevancia y utilidad. Este trabajo proporciona por tanto una base sólida para futuras investigaciones y avances en la detección de anomalías, contribuyendo al progreso continuo de esta disciplina esencial en el mundo actual.

Óscar Fontenla Romero
Catedrático de Universidad
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

Bertha Guijarro Berdiñas
Titular de Universidad
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

CERTIFICAN

Que la memoria titulada "*Machine learning for anomaly detection: From surface to deep*" ha sido realizada por D. David Novoa Paradela bajo nuestra dirección en el Departamento de Ciencias de la Computación y Tecnologías de la Información de la Universidade da Coruña, y concluye la Tesis Doctoral que presenta para optar al grado de Doctor en Ingeniería Informática.

En A Coruña, a 28 de febrero de 2023

Fdo.: Óscar Fontenla Romero
Director de la Tesis Doctoral

Fdo.: Bertha Guijarro Berdiñas
Directora de la Tesis Doctoral

Fdo.: David Novoa Paradela
Autor de la Tesis Doctoral