Methodological Advancements in Continual Learning and Industry 4.0 Applications

Davide Dalle Pezze

December 2022

Contents

1 Introduction **11**

2 Industry 4.0 **15**

2.1 Introduction to Industry 4.0 15
2.2 Enabling Technologies 16
2.2.1 Artificial Intelligence (AI) and Machine Learning (ML) . 19
2.2.2 Big data analytics (BDA) 20
2.2.3 Internet of Things (IoT) 21
2.2.4 Cloud computing (CC) 21
2.2.5 Additive Manufacturing 23
2.2.6 Augmented Reality (AR) 23
2.2.7 Robotics and Cyber-Physical System (CPS) 24
2.2.8 Cybersecurity 25
2.2.9 Digital Twin 25
2.3 Research Areas 26
2.3.1 Predictive Maintenance 27
2.3.2 Fault Detection 31
2.3.3 Remaining Useful Life (RUL) 31
2.4 Machine Learning for I4 32
2.4.1 Supervised Learning 34
2.4.2 ML Algorithms in Manufacturing 35
2.4.3 Unsupervised Learning 38
2.4.4 Reinforcement Learning 41
2.4.5 Public data sets for predictive maintenance 42
2.5 Challenges 43
2.5.1 Cybersecurity 44
2.5.2 Financial and Organizational Limits 45
2.5.3 Data Source Limits 46
2.5.4 Human-machine interaction 46
2.5.5 Interpretability 47
2.5.6 Machine Repair Activity Limits 47
2.5.7 Limits in the Deployment of Industrial Predictive Maintenance Models and Data Distribution Shift 48

3 Continual Learning **53**
3.1 Continual Learning . . . 53
3.1.1 Formal Definition . . . 54
3.2 Continual Learning Scenarios . . . 55
3.2.1 Types of Data Distribution Changes . . . 55
3.2.2 Continual Learning scenarios . . . 57
3.2.3 Task Incremental Learning (TIL) . . . 58
3.2.4 Domain Incremental Learning (DIL) . . . 59
3.2.5 Class Incremental Learning (CIL) . . . 59
3.2.6 Conclusions . . . 59
3.3 Related Paradigms . . . 59
3.3.1 Transfer Learning (TL) . . . 59
3.3.2 Multi-Task Learning (MTL) . . . 60
3.3.3 Online Learning (OL) . . . 60
3.3.4 Open World Learning . . . 60
3.3.5 Reinforcement Learning (RL) . . . 61
3.3.6 Meta Learning . . . 61
3.4 Evaluation Metrics . . . 61
3.4.1 Goals of CL . . . 62
3.5 Datasets . . . 67
3.5.1 Classic Datasets . . . 67
3.5.2 CL Datasets . . . 70
3.6 Continual Learning Strategies . . . 74
3.6.1 Upper Bounds and Lower Bounds . . . 75
3.6.2 Regularization-based approaches . . . 75
3.6.3 Rehearsal-based approaches . . . 75
3.6.4 Architecture-based approaches . . . 76
3.7 Regularization-based approaches . . . 76
3.7.1 Elastic Weighted Consolidation (EWC) . . . 76
3.7.2 SI . . . 77
3.7.3 Learning without Forgetting (LwF) . . . 78
3.8 Rehearsal-based approaches . . . 79
3.8.1 Experience Replay (ER) . . . 79
3.8.2 Generative Replay . . . 79
3.8.3 Gradient Episodic Memory (GEM) . . . 80
3.8.4 incremental Classifier and Representation Learning (iCaRL) . . . 80
3.8.5 Maximally Interfered Retrieval (MIR) . . . 81
3.9 Architecture-based approaches . . . 81
3.9.1 Progressive Neural Network (PNN) . . . 81
3.9.2 Expert Gate(EG) . . . 82
3.9.3 PathNet . . . 82
3.9.4 Piggyback Approach . . . 82
3.9.5 PackNet . . . 83
3.10 Applications . . . 83
3.10.1 Computer Vision . . . 83

3.10.2 NLP . . . 84
3.10.3 Audio . . . 84
3.10.4 Robotics . . . 85
3.10.5 Recommendation Systems . . . 85
3.10.6 Industry 4.0 . . . 85
3.10.7 Edge Computing . . . 85
3.10.8 Cybersecurity . . . 86
3.10.9 Smart City . . . 86
3.10.10 Machine Learning Production Systems . . . 86
3.10.11 Conclusions . . . 87
3.11 Continual Learning and Industry 4.0 . . . 87
3.11.1 Manufacturing . . . 88
3.11.2 Conclusions . . . 94

4 Interpretability **95**
4.1 Motivation . . . 95
4.2 Intepretability . . . 96
4.2.1 Intro . . . 96
4.2.2 Related Work . . . 96
4.2.3 SHAP-based Approaches . . . 98
4.3 SHAP . . . 98
4.3.1 KernelSHAP . . . 99
4.3.2 SHAP results visualization . . . 100
4.4 Proposed Approach . . . 100
4.4.1 AcME . . . 101
4.4.2 Global interpretability for regression tasks . . . 102
4.4.3 Local interpretability for regression tasks . . . 104
4.4.4 AcME for classification . . . 107
4.5 Experimental results . . . 107
4.5.1 Synthetic datasets . . . 108
4.5.2 Experiments on a regression task: Boston Housing dataset 111
4.5.3 Experiments on a classification task: Glass dataset . . . 123
4.5.4 Experiments on a classification task with Neural Network 124

5 Replay for Multilabel Setting **129**
5.1 Multilabel . . . 129
5.1.1 Introduction . . . 129
5.1.2 Applications . . . 130
5.1.3 Challenges . . . 130
5.1.4 Evaluation metrics . . . 131
5.2 Alarm Forecasting . . . 132
5.2.1 Motivation . . . 132
5.2.2 Related work . . . 133
5.2.3 Challenges . . . 134
5.3 Proposed Approach for Alarm Forecasting . . . 134

5.3.1 Motivation and Formalization of Alarm Forecasting as Multi-label classification . . . 134
5.3.2 Dataset design in the multi-label classification setting . . 136
5.3.3 FORMULA . . . 137
5.3.4 NN Architectures . . . 138
5.3.5 Loss functions for rare alarms forecasting . . . 138
5.3.6 Benchmark Approaches . . . 141
5.3.7 Experimental Results . . . 142
5.3.8 Conclusions . . . 149
5.4 Replay for Multilabel Setting . . . 153
5.4.1 Motivation . . . 153
5.4.2 Related work . . . 154
5.4.3 Optimizing Class Distribution in Memory (OCDM) . . . 155
5.5 Proposed Approach for Multi-label Replay . . . 156
5.5.1 Continual Learning classifier design . . . 156
5.5.2 Balanced Among Tasks OCDM (BAT-OCDM) . . . 157
5.6 Experimental Setting . . . 160
5.6.1 Metrics . . . 160
5.6.2 Experiment setup . . . 160
5.6.3 Considered Continual Learning approaches . . . 161
5.6.4 Results . . . 162
5.7 Conclusions and future works . . . 166

6 Anomaly Detection for Continual Learning 167
6.1 Anomaly Detection . . . 167
6.1.1 Anomaly Detection . . . 167
6.1.2 Benchmark considered . . . 168
6.1.3 Strategies for Anomaly Detection . . . 168
6.1.4 Image-level Anomaly Detection . . . 170
6.1.5 Pixel-level Anomaly Detection . . . 172
6.2 Compressed Replay . . . 174
6.3 Anomaly Detection for Continual Learning . . . 175
6.3.1 Proposed Framework for ADCL . . . 175
6.4 Our Approach: SCALE . . . 177
6.5 Experimental Settings . . . 180
6.5.1 Considered CL Strategies . . . 180
6.5.2 Considered Architectures in Memory and Anomaly Detection modules . . . 182
6.5.3 Evaluation Metrics for Anomaly Detection . . . 183
6.5.4 Evaluation of image reconstruction quality . . . 183
6.5.5 CL Metrics . . . 183
6.6 Results . . . 184
6.6.1 Quality of the Anomaly Detection . . . 184
6.6.2 Quality of the reconstructed images . . . 187
6.6.3 Results of SR Model . . . 188
6.7 Conclusions . . . 192

7 Conclusions and Future Research **195**
7.1 Conclusions 195
7.2 Future research 196

UNIVERSITY OF PADUA

Department of Computer Science and Engineering
Doctoral Degree

Methodological Advancements in Continual Learning and Industry 4.0 Applications

Davide Dalle Pezze

Abstract

The Fourth Industrial Revolution, also known as Industry 4.0, is built on various technologies, including Artificial Intelligence, the Internet of Things, Cloud Computing, Robotics, and Big Data Analytics. The ultimate goal is to improve efficiency, productivity, and flexibility. For example, Quality Control allows for the discovery and elimination of defective products. Instead, with Predictive Maintenance is possible to predict the equipment's next failure and schedule maintenance in advance. Industry 4.0 has many challenges that will be discussed and faced, like Interpretability.

In particular, a significant challenge in Industry 4.0 is when the manufacturing process is not static but dynamic. The current classic DL setting can only adapt to changing environments if it is retrained from scratch with all data, which ultimately results in high costs and training time to update the model. Continual Learning, on the other hand, allows ML models to be updated and grow their knowledge over time with minimum computation and memory overhead, lowering the costs associated with machine learning model maintenance.

As a result, we concentrate on CL techniques that can be applied in an Industry 4.0 field. Many industrial problems, such as Anomaly Detection and Multi-Label Classification, are outside of the conventional classification problems studied in Continual Learning, as presented in the following text; therefore, little research has been conducted. Finally, CL can improve the performance of industrial machine learning models, making them more adaptable to new environments. Moreover, they can be updated with less resource use than would otherwise be necessary.

Acknowledgments

I want to thank everyone who has helped me and supported me, personally and professionally, during this long and exciting journey. I express my deep gratitude to my thesis advisors Prof. Gian Antonio and Chiara Masiero, who generously provided knowledge and expertise. Especially a big thank you for all the support, patience, and meetings discussing projects and works. Their support and guidance helped me to refine my research and improve the quality of my writing.

Additionally, this experience would not have been possible without the generous support from Statwolf Data Science, Fondazione Cariparo, Intesa Sanpaolo, and Unismart, which financed my research. Special thanks to Statwolf for his assistance and expertise in industrial applications, as well as for providing me with intriguing industrial data to work on and develop my research. A special thanks for the time spent together also to the employees of Statwolf, David, Luca, and Chiara. Of course, a big thank is due to all my colleagues at the University of Padua who have accompanied me on this long journey, which names are too many to be written here. Thank you for the time spent together and for the constant and stimulating scientific discussions

I want to thank my family for all the support: Arianna, Roberto, Diego, Giorgio, Fabrizia, Stella, and grandmother Renata. Special thanks go to my family for the unconditional love and support they have shown me, especially during times of stress and hardship. A particular thanks to my parents; their belief in me has kept my spirits and motivation high during this process. Thank you so much to my grandmother Elda and grandfather Ezio, who is no longer with us; their faith in me was unwavering, and their love and guidance will be remembered forever.

Lastly, I would like to thank my friends for their support and encouragement: Andrea, Giorgio, Matteo, Alvise, Anna, Roberto, Fabio, Francesco, Luca, and Piero. They have been a constant source of inspiration and motivation throughout this journey, and I am deeply grateful for their support.

I thank Prof. Irina Rish for generously hosting me at the Mila research institute in Montreal. The experience was truly enriching, both in terms of the educational and professional growth opportunities it provided. I would also like to express my appreciation to the many individuals I met during my stay, which greatly enhanced my experience through their knowledge and friendship: Timothee, Alexei, Lucas, Karina, Mathilde, Pascal, Francesco, Luca, Martin, Jan, Mirco, Piero. Thank you for making it such a memorable and valuable experience.

The research presented in this thesis would not have been possible without the support of all those acknowledged above.

Chapter 1

Introduction

The Fourth Industrial Revolution, also known as Industry 4.0, is built on a variety of technologies, including Artificial Intelligence, the Internet of Things, Cloud Computing, Robotics, and Big Data Analytics. The ultimate goal is to improve efficiency, productivity, and flexibility. For example, Robotics and machine can significantly minimize waste and defects, while data analytics can help detect bottlenecks and inefficiencies in the manufacturing process. Moreover, with Predictive Maintenance is possible to predict the equipment's next failure and schedule maintenance in advance.
Industry 4.0 has many challenges that will be discussed and faced, like Cybersecurity, Interpretability and Data Distribution Shift.
In particular, a significant challenge in Industry 4.0 is when the manufacturing process is not static but dynamic, i.e., there is a data distribution shift. Machine learning has advanced the state of the art in solving numerous research and industry problems in recent years. In particular, deep neural networks (DNNs) improved the state of the art in many domains like Computer Vision, Natural Language Processing, and Audio. However, the current classic DL setting can only cope with changing environments if it is retrained from scratch with all data, which ultimately results in high costs and training time to update the model. Indeed, applying a simple fine-tuning of the model using only new data will result in what is known as Catastrophic Forgetting (CF), where all previous knowledge is quickly forgotten.
Continual Learning (CL) proposes to consider a setting where new data arrive at different moments in the form of a sequence of tasks. Continual Learning enables a DL model to learn and adapt to new tasks without forgetting previously learned patterns. In other words, CL allows ML models to be updated and grow their knowledge over time with minimum computation and memory overhead, lowering the costs associated with machine learning model maintenance.
As a result, we concentrate on CL techniques that can be applied in an Industry 4.0 field. Most of the work in CL is developed in a multi-class classification setting. However, it is common to have problems to solve in Industry 4.0 that shift from the classic multi-class classification setting. For example, it would

be expected in a real scenario to not have supervision and to be in an Unsupervised Learning paradigm. For instance, Anomaly Detection is an important problem since, in practice, often the labels are not provided. Another realistic industrial scenario would be the weakly supervised learning scenario or the case where there are missing features in the input samples. These and many other examples are often not considered in CL. However, it is not immediate to assume that the CL approaches that work in the classic multi-class classification scenario can be extended automatically to other scenarios. Therefore, more research on this front should be developed. As a result, we concentrate on CL techniques that can be applied in an Industry 4.0 field. Many industrial problems, such as Anomaly Detection and Multi-Label Classification, are outside of the conventional classification problems studied in Continual Learning, as presented in the following text; therefore, little research has been conducted. Finally, CL can improve the performance of industrial machine learning models, making them more adaptable to new environments. Moreover, they can be updated with less resource use than would otherwise be necessary.

The contributions presented in the thesis can be listed in the following original work:

- [298] *"Alarm Logs in Packaging Industry (ALPI)"* on *IEEE DataPort* (2020) - Discussed in Chapter 5

- [69] *"FORMULA: A Deep Learning Approach for Rare Alarms Predictions in Industrial Equipment"* published on Journal *IEEE Transactions on Automation Science and Engineering* (2021) - Discussed in Chapter 5

- [228] *"A Multi-label Continual Learning Framework to Scale Deep Learning Approaches for Packaging Equipment Monitoring"* submitted to Journal *"Control Engineering Practice"* - Discussed in Chapter 5

- [229] *"Continual Learning Approaches for Anomaly Detection"* submitted to Journal *"Engineering Applications of Artificial Intelligence"* - Discussed in Chapter 6

- [70] *"AcME — Accelerated model-agnostic explanations: Fast whitening of the machine-learning black box"* published on the Journal *"Expert Systems with Applications"* (2023) - Discussed in Chapter 4

In this dissertation, we present an overview on the themes of Industry 4.0 and Continual Learning. The original contributions presented in the following text are numerous and can be summarized in the following points:

- To foster research in the field of Predictive Maintenance we introduce a new public dataset called ALPI (Alarm Logs in Packaging Industry).

- A novel approach called FORMULA to solve Alarm Forecasting (AF) formalized as a multi-label classification (MLC) problem is proposed

- We propose a new method to select the optima subset of samples to keep in memory in the Replay strategy in the Multi-label setting. Some idea will be taken from FORMULA and the validity of approach is confirmed using a real-world industrial scenario
- We also present an overview of CL approaches applied for Industry 4.0 showing the increasing interest of CL in this field
- We propose a framework to study approaches of Anomaly Detection in the Continual Learning (ADCL)
- We study the ADCL using as benchmark a complex dataset designed for AD and evaluating using many AD approaches.
- We propose a novel approach to perform Compressed Replay, where the images are compressed in memory to keep more samples and to have a distribution more similar to the original one.

The following text is how the manuscript is structured. In Chapter 2, the theme of Industry 4.0, including enabling technologies, applications, and research areas, will be discussed. Furthermore, various Industry 4.0 challenges will be discussed, such as Cybersecurity, Data Distribution Shift, Interpretability, and other issues. The Chapter 3 will provide an overview of Continual Learning, presenting numerous concepts such as the different scenarios in CL, the differences between CL and related paradigms, and common metrics and datasets used in the field. It will also provide an overview of the most well-known CL approaches and explain the various families of methods. Furthermore, it will be provided with a detailed list of CL applications, focusing on real-world industrial scenarios. In Chapter 4, a novel solution to Interpretability will be described, a critical problem of Industry 4.0 when models must be deployed in a real context. The chapter 5 will highlight the significance of Alarm Forecasting (AF) for Predictive Maintenance (PdM) as well as an innovative solution for a real-world case from the packaging industry that has been defined as a multi-label classification problem. Furthermore, this strategy is extended to work in the context of Continual Learning, and a new method for performing Replay in Multi-label is proposed. A framework for evaluating Anomaly Detection approaches in the Continual Learning environment is presented in Chapter 6. We provide a benchmark considering several AD approaches and a complex dataset designed specifically for AD to foster research in the field A novel approach to Compressed Replay is also presented. The Chapter 7 will give the dissertation's conclusions, as well as explore the issues of Continual Learning and Industry 4.0.

Chapter 2

Industry 4.0

Industry 4.0 is a new, so-called, industrial revolution which focuses primarily on machine learning, fuelled by real-time data monitoring and processing, interconnectivity between machines and computers in the autonomous systems

2.1 Introduction to Industry 4.0

The Fourth Industrial Revolution, or Industry 4.0, is a term used to describe the current automation and data exchange trend in manufacturing technologies. Industry 4.0, also known as Smart Manufacturing, is a revolution in manufacturing, and it brings a whole new perspective to the industry on how manufacturing can collaborate with new technologies to get maximum output with minimum resource utilization [131].

The First Industrial Revolution lasted from the late 18th century through the early 19th century. It witnessed the evolution of mechanization. A process that shifted the emphasis from agriculture to industry as the basis of society's economic structure. Other key drivers for this transformation included utilizing steam power and other new metal know-how. At the end of the nineteenth century, significant technological advances introduced new energy sources: electricity, gas, and oil. This new paradigm was called the Second Industrial Revolution (Industry 2.0). People began to use electricity in their daily lives and factories. This improved the industrial operations' efficiency. In addition, the assembly line concept was implemented to increase production. Additionally, new methods of communication were discovered with the invention of the telegraph and the telephone. Finally, the introduction of the automobile and the airplane at the turn of the twentieth century altered transportation systems. Industry 3.0 evolved in the second half of the twentieth century. As a result, nuclear energy emerged as a new source of energy. It also saw the rise of electronics, with the transistor and microprocessor. This revolution ushered in an era of high-level industrial automation and the employment of robotics in manufacturing operations. Computers and digital technology are used to develop

and control manufacturing processes. Indeed, an important aspect is the integration of information and communication technologies (ICT) into industrial processes. Overall, Industry 3.0 has significantly increased manufacturing and other industrial sectors' productivity and efficiency. It has also opened the way for Industry 4.0, which symbolizes the next stage of the industrial digital revolution. Nowadays we are currently living in the Fourth Industrial Revolution. Industry 4.0 is the new phase of the industrial revolution which focuses primarily on machine learning, fuelled by real-time data monitoring and processing, interconnectivity between machines and computers in the autonomous systems [328] . Industry 4.0 is not only using new machines and new technologies. It is about using the right technology to improve efficiency and revolutionizing the way the entire business operates and grows.

Integrating modern digital technology into the manufacturing sector is called Industry 4.0. More specifically, it is distinguished by incorporating sophisticated technology into industrial processes such as artificial intelligence, the Internet of Things (IoT), cloud computing, robotics, and big data analytics. These technological breakthroughs aim to improve efficiency, productivity, and flexibility in Industry 4.0. Manufacturing businesses constantly strive to improve their production yield, uptime, and throughput to improve product quality while lowering costs. Nowadays, modern industries collect an enormous amount of data, thanks to the increased availability of sensors in the production lines. These data can provide useful information about many quantities of interests whose knowledge would dramatically impact both productivity and cost management. Overall, Industry 4.0 has the potential to revolutionize manufacturing and supply chain management, enabling companies to respond more quickly and effectively to changing market conditions and customer needs. Industry 4.0 can enhance industrial efficiency and productivity by automating procedures and optimizing operations. Robotics and machine learning, for example, can significantly minimize waste and defects, while data analytics can help detect bottlenecks and inefficiencies in the manufacturing process. Additionally, 3D printing and other advanced manufacturing technologies can enable businesses to offer customized products on demand [131]. The market for Industry 4.0 technologies is projected to grow in the coming years as more and more companies adopt these technologies to improve their operations and stay competitive [32], as visible from Fig. 2.2.

2.2 Enabling Technologies

One of the key features of Industry 4.0 is the integration of physical and digital technologies, which allows for the creation of "smart factories" that can operate with minimal human intervention. This involves using sensors, robotics, and other advanced technologies to automate and optimize manufacturing processes, as well as data analytics and cloud computing to enable real-time monitoring and control.

Various technologies have been used for implementing Industry 4.0 applica-

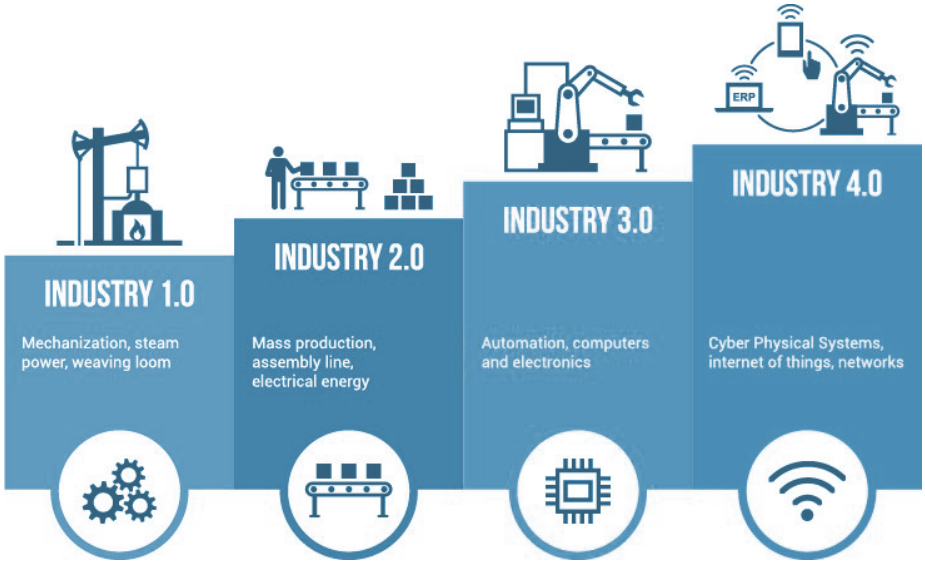


Figure 2.1: Evolution of Industry. Image from [328]

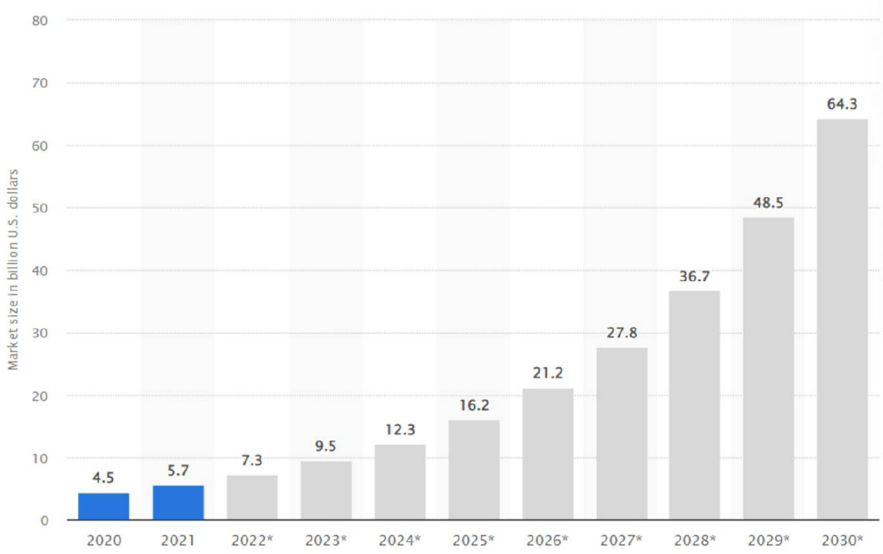


Figure 2.2: Size of Maintenance 4.0 market worldwide in 2020 and 2021 with forecast for future years (from 2022 to 2030) [32]

Figure 2.3: Enabling technologies involved with Industry 4.0

Table 2.1: Notation used in this chapter

Nomenclature	**Symbol**
Artificial Intelligence	AI
Artificial Neural Network	ANN
Augmented Reality	AR
Condition-Based Maintenance	CBM
Cyber-Physical Systems	CPS
Deep Learning	DL
Denial of service	DoS
Digital Twin	DT
Human Machine Interface	HMI
Industrial Internet of Things	IIoT
Internet of Things	IoT
Machine Learning	ML
Predictive maintenance	PdM
Preventive maintenance	PM
Prognostics and Health Management	PHM
Return on investment	ROI
Remaining Useful Life	RUL
Virtual Reality	VR

tions. These technologies include IoT, Cloud Computing, AI, Big Data, and other related technologies [337]. This section focuses on emerging technologies that will play a significant role in Industry 4.0 in the coming years. It should be noted that this list is not exhaustive and does not include all emerging technologies that may impact Industry 4.0.

2.2.1 Artificial Intelligence (AI) and Machine Learning (ML)

Artificial intelligence was founded as an academic discipline in the 1950s. The field was founded on the assumption that human intelligence can be precisely described and human-like intelligence through artificial approaches is possible [271]. AI is concerned with building smart machines capable of performing tasks that typically require human intelligence [271].

Industry 4.0 incorporates the digital revolution into the physical world, providing a promising new direction for artificial intelligence [278, 141]. The artificial intelligence field has encountered a turning point mainly due to the recent advancements in industrialization and digitalization. Artificial intelligence is much more than a research field. It is a future technology with the potential to redefine many tasks in Industry 4.0 [271]. These technologies enable machines to learn and adapt to new situations, enabling them to perform tasks that would be difficult or impossible for humans to do.

AI technologies are becoming an important technical component of Industry 4.0, helping to perform many challenging tasks in industrial operations. AI has the potential to transform many different industries and has already been integrated into a variety of applications, including image and speech recognition, natural language processing, self-driving cars, and personal assistants. In recent years, AI technologies have experienced a resurgence in industrialization [108, 195]. The advancements in machine learning and deep learning are generating significant impacts in many sectors of industries.

The problem domain of AI research includes reasoning, knowledge representation [238], planning, learning, and natural language processing [132]. Approaches include machine learning [121] and others. Many techniques are used in AI, Deep Learning, genetic algorithm [364], knowledge management [85], intelligent systems [99], knowledge-based systems [265], and others.

In [153], study applications of AI under the Industry 4.0 concept. In [24] is presented the AI technologies in manufacturing systems related to Industry 4.0, including AI-based techniques for production monitoring, optimization, and control. The study by [86] discusses the importance of AI methods in the control and operation monitoring of dynamic measurements in automation systems in Industry 4.0. The study [199] shows how AI techniques can assist the manufacturing process in Industry 4.0 environment. According to [124], AI and machine learning will put Industry 4.0 forward, with the prediction that much more fundamental changes in business processes and business models caused by the integration of AI and other new technology eventually will lead to the fact that business and industry are changing the way of operation [271].

2.2.2 Big data analytics (BDA)

Big data refers to data sets that are so large or complex that they are difficult to process using traditional data processing applications. Big data also often have high variety, which means that it includes data from a wide range of sources and in various formats, such as text, images, audio, and video. This can make it difficult to structure and process the data meaningfully.

Big data analytics and technologies support real-time data collection from many different sources, comprehensive data analysis, and real-time decision-making. This leads to improved manufacturing flexibility, product quality, energy efficiency, and improved equipment service through predictive maintenance [155, 131]. Big data analytics has been widely used in manufacturing for process monitoring, and fault finding supports new capabilities like predictive analytics [155, 313].

Collecting, processing, and analyzing real-time Big Data from cyber-physical systems is a strategic phase for the intelligent transformation of the maintenance function, especially concerning failure prediction, planning, and risk management [315]. This is done through the planning and optimization of interventions using artificial intelligence tools and techniques such as machine learning and deep learning or through statistical models and approaches based on the data and information collected by the different sensors [272].

2.2.3 Internet of Things (IoT)

The IoT is a new industrial ecosystem that combines intelligent and autonomous machines, advanced predictive analytics, and machine-human collaboration to improve productivity, efficiency, and reliability [296, 333]. In other words, the Internet of Things refers to a network of physical items equipped with sensors, software, and connectivity to collect and exchange data. Examples of IoT devices range from simple sensors that collect data from the environment, such as temperature and humidity, to more complex devices that can interact with their surroundings, such as smart thermostats and smart locks. IoT technologies are utilized in Industry 4.0 to connect machines, devices, and systems in production environments, allowing them to communicate and share data in real-time. In Industry 4.0, IoT technologies connect machines, devices, and systems in manufacturing environments, enabling them to communicate and share data in real-time. The Internet of Things (IoT) enables real-time sensing and actuation, as well as fast data and information transmission, allowing for remote operation of manufacturing activities and efficient collaboration among stakeholders.

The increasing use of sensors on physical products allows them to capture, process, and communicate data with humans and other physical systems. IoT has the potential to transform many industries by enabling the collection and analysis of large amounts of data from a wide range of sources. There is enormous potential for creating sensor-based applications, as these sensors provide real-time data that may be used for predictive maintenance by detecting equipment wear and tear, for better capacity planning, and to assess the usage and functionality of products [165]. It can also track the location and status of products and materials within the factory, improving supply chain visibility and reducing waste. In conclusion, IoT in Industry 4.0 aims to create a more connected and automated manufacturing environment, increasing efficiency, productivity, and innovation.

2.2.4 Cloud computing (CC)

Cloud computing delivers computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet. Some key benefits of cloud computing include scalability, cost, reliability, and performance. With Cloud Computing is possible to scale up or down the amount of computing resources depending on the needs. Therefore, the cost is restricted to the computing resources effectively used. In general, in the cloud computing paradigm, users get high-quality services at a lower cost [236]. Moreover, cloud providers have invested heavily in their infrastructure to make it highly available, reliable, and secure. Regarding performance, Cloud computing resources are usually faster and more powerful than what companies can afford.

Due to the potential and practical benefits to society and economy, cloud computing paradigm has attracted enormous attention from academia and industry. In particular, Cloud Computing is an important enabler of Industry 4.0, because it allows companies to store and process vast amounts of data from con-

nected devices and systems in real-time, enabling them to make better, faster decisions and optimize their operations.

Cloud connects all manufacturers and customers. Customers communicate with each other and manufacturers to effectively integrate social demands and production capacities by using computers or mobile devices.

All manufacturing resources and capabilities are virtualized and encapsulated as services to be managed, allocated, and on-demand used through cloud [289, 290]. Combining IoT, CPS, BDA, and CC technologies enables smart manufacturing [174].

Cloud computing enables manufacturers to store and process data remotely, rather than on local servers or devices. This enables them to access and analyze large amounts of data from anywhere, at any time.

Industry 4.0 systems often rely on cloud computing to store, process, and analyze data and provide computing resources on demand. This allows for greater flexibility and scalability and the ability to access data and services from anywhere.

More in detail, cloud computing can be used to:

- Analyze data from sensors and other Internet of Things (IoT) devices to optimize production and maintenance processes.
- Use artificial intelligence and machine learning to predict equipment failures and improve quality control.
- Connect and coordinate different parts of the supply chain in real-time, enabling companies to respond faster to changing market conditions.
- Develop and deploy new applications and services quickly and at low cost, enabling companies to be more agile and innovative.

The manufacturing system in a cloud environment allows for higher utilization without increasing investment or degrading performance, and it frees manufacturers and users from many details. However, many problems still limit the expansion of smart manufacturing, such as overfull bandwidth, unavailability, latency, data validity, security and privacy, and inefficient interaction [236]. Data generated by various manufacturing resources, which may be geographically distributed, is experiencing explosive growth. These data are conveyed over the network to the cloud, where data processing is carried out [100]. Data's increasing volume and velocity require high bandwidth, which is very expensive. When network congestion is severe, some data may be lost. Although the data stored in the cloud can be accessed from anywhere at any time, the user relies heavily on the availability of an internet connection and the servers [321]. Therefore, when data cannot be accessed due to network unavailability, the power of the cloud becomes unusable. Moreover, some real-time and concurrent scenarios require time synchronization, bringing real-time issues [39]. Data validity refers to the large set of insignificant data (e.g., redundancy, noise, temporary data, etc.) conveyed to the cloud, which wastes resources.

Strictly related to Cloud Computing, there is the theme of Edge Computing. To extend the smart manufacturing applications in a cloud environment and offer perspectives for implementing solutions that require very low and predictable latency, a reference architecture based on edge computing for smart manufacturing should be considered. The definition of edge computing is a distributed computing paradigm that brings computation and data storage closer to the location where it is needed to improve response times and save bandwidth [271]. Edge computing has the goal of minimizing latency. Edge computing does this by processing and storing data outside the core network. The time and distance for data transmission are reduced by processing the information closest to its source, and the connection speed dramatically increases. Moreover, after integrating with 5G, the power of edge computing can increase dramatically. For the IoT, edge computing helps process much more information, and the processing speed can be much faster.

2.2.5 Additive Manufacturing

Machining techniques such as cutting, drilling, grinding, and sanding used in traditional manufacturing processes are referred to as subtractive manufacturing, in which parts and components manufacturing relies on the removal method [131]. These parts and components are then assembled to form the final products. In contrast,3-D printing relies on additive manufacturing, which forms the final products by building up successive layers of materials, thus avoiding the need for parts and component assembly. A computer-aided design (CAD) software is used in 3-D printing to generate a digital model, followed by creating (printing) a three-dimensional object in a 3-D printer from raw materials in either liquid or particle form. Thin layers of raw material are deposited microscopically by the printer so that the deposition of successive layers materializes in the formation of the final product.

Additive manufacturing allows for the production of complex and customized objects using a wide range of materials, including plastics, metals, ceramics, and even living cells. It can revolutionize manufacturing by allowing small batches of customized products and on-demand production.
With open source kits the barriers to entry for designers and inventors started to fall. While the price of 3D printers has fallen rapidly in recent years, the accuracy of 3D printing has significantly improved, and designers are no longer limited to printing with plastic. It has applications in many sectors as diverse as healthcare, aerospace, and parts replacement [189]. Although 3-D manufacturing systems are still in the initial stages, organizations are expected to use it more broadly in Industry 4.0. In [57] the need for 3-D printing technology as an enabler for smart manufacturing is asserted.

2.2.6 Augmented Reality (AR)

AR is an emerging technology developed based on virtual reality (VR), which generates three-dimensional virtual information through a computer system, in-

cluding virtual scenes, virtual objects, etc. [320], and then superimposes this information into the real scene to realize the function of real-world enhancement and improve the user's perception of the real world [4].
In general, augmented reality can guide human workers through unfamiliar tasks and visualize information. It has a wide range of applications in the manufacturing industry, such as interactive training and onboarding for new employees, finding spare components in a warehouse, as well as providing real-time guidance and instructions to technicians conducting maintenance and repair jobs [217]. This technology can help increase manufacturing process efficiency, accuracy, and productivity and ensure high-quality work and near-zero error rates in numerous industrial processes.

2.2.7 Robotics and Cyber-Physical System (CPS)

Robotics is a field of engineering that involves the design, construction, operation, and application of robots. A robot is a machine capable of carrying out a complex series of actions automatically, especially by being programmed by a computer. Robotics technologies are used in various applications, including manufacturing, transportation, healthcare, and the military.

Modern robots are characterized as systems offering autonomy, flexibility, and cooperation. It is predicted that the robots will soon start interacting with one another and work safely with humans and even learn from them. Moreover, these robots will offer a cost advantage and an excellent range of capabilities, performing most of the processes in the intelligent factory [93].

It allows for automation and the integration of digital technologies in industrial processes. Advanced robotics technologies are being used in Industry 4.0 to automate repetitive, dangerous tasks or those that require a high level of precision, such as welding, painting, and assembly. Industrial robots can be programmed to perform a wide variety of tasks. To improve their performance and capabilities, they can be integrated with other digital technologies, such as sensors and the Internet of Things (IoT).

Robotic technologies can automate and optimize various aspects of the manufacturing and production process, such as material handling, assembly, inspection, and Maintenance. Robots can be used in material handling to move materials around a factory or warehouse, freeing up human workers for other tasks. They can also be used to assemble products with high precision and speed, improving efficiency and quality. In addition, robots can be used to inspect products for defects and ensure that only high-quality products are shipped to customers. Finally, robots can be used for maintenance tasks such as cleaning and lubricating machinery, reducing downtime, and improving safety.

A programmable dual-arm robot is proposed by [207] for material distribution in the assembly line. Further, to ensure the safe operation of the robot and monitor the environment, if any disturbance, such as a person or equipment such as an automated guided vehicle, enters the virtual space, the system stops the robot's movement with a unique sound, anticipating some complicity. The operator has to remove the obstacle before the robot starts working again. As

manufacturing tasks become more individual and flexible, smart factory machines must do variable tasks collaboratively without reprogramming.

Cyber-Physical System (CPS) is one of the core foundations of Industry 4.0. CPS presents a higher level of integration and coordination between physical and computational elements. In CPS, physical and software components are deeply intertwined, each operating on different spatial and temporal scales and interacting with each other. With the introduction of CPS, machines can communicate with each other, and decentralized control systems will be able to optimize production [271]. CPS are physical systems integrated with computational elements to sense, actuate, and process data. CPS technologies are used in various applications, including manufacturing, transportation, healthcare, and the military.

Robotics and CPS technologies often overlap and are often used together in various applications. For example, a robotic arm in a manufacturing plant may be controlled by a CPS that monitors and adjusts the production process in real time.

2.2.8 Cybersecurity

The industrial data is highly sensitive, encapsulating various aspects of the industrial operation, including information about products, business strategies, and companies [332]. As Industry 4.0 involves the exchange of large amounts of data and the integration of advanced technologies, it also raises cybersecurity concerns. Industry 4.0 systems are often complex and interconnected, making them more difficult to secure and vulnerable to attacks. Researchers are developing techniques and technologies to protect against cyber threats, such as data breaches and cyber attacks. Since Cybersecurity can be considered a technology to solve security and privacy problems, this part will be discussed further in the Section of the Challenges of Industry 4.0.

2.2.9 Digital Twin

A Digital Twin (DT) is a digital or virtual copy of physical assets or products. A digital twin is a virtual model of a physical system or process that can be used to simulate and analyze the performance of the system or process.

The term was initially coined by Dr. Michael Grieves in 2002 [130]. Digital twins are created using data from sensors, historical records, and other sources. As a result, they can be used to predict how the physical system or process will behave in different scenarios. DTs connect the real and virtual worlds by collecting real-time data from the installed sensors. The collected data is either locally decentralized or centrally stored in a cloud. The data is then evaluated and simulated in a virtual copy of assets [119]. Integrating data into real and virtual representations helps optimize the performance of real assets. They can be used to improve the efficiency, reliability, and safety of physical systems, and they have the potential to revolutionize the way we design, operate, and maintain complex systems and processes.

Digital twins can be utilized to enhance and improve manufacturing processes in various ways in Industry 4.0. They can, for example, be used to simulate and evaluate manufacturing processes to discover bottlenecks and inefficiencies and optimize them for increased performance. For example, digital twins can predict when equipment will break, allowing maintenance to be arranged in advance, minimizing downtime, and increasing efficiency. Furthermore, digital twins can be applied for product design and development, simulating and testing new items and processes, eliminating the need for physical prototypes and accelerating the development process.

2.3 Research Areas

Machine learning is a key technology in Industry 4.0, as it allows for the automation and optimization of manufacturing processes. Some of the most relevant research areas in the use of machine learning for Industry 4.0 and manufacturing include:

- **Predictive maintenance**: Machine learning algorithms can be used to analyze data from sensors and other sources to predict when equipment is likely to fail, allowing preventive maintenance to be scheduled before the failure occurs. This can help reduce downtime and improve the overall reliability of manufacturing systems.
- **Quality control**: Machine learning can be used to analyze data from sensors and other sources to identify defects in products in real-time, allowing for quick corrective action to be taken. This can help improve the overall quality of products and reduce waste.
- **Production optimization**: Machine learning algorithms can be used to optimize production schedules and identify bottlenecks in the production process. By analyzing data from sensors and other sources, machine learning can help companies identify opportunities for improvement and increase efficiency.
- **Supply chain optimization**: Machine learning can also optimize the supply chain by analyzing data from multiple sources to identify patterns, trends, and correlations. This can help companies make more informed decisions about inventory management, transportation, and other aspects of the supply chain.
- **Asset management:** This involves managing and tracking the performance and maintenance of equipment and other assets in order to optimize operations and reduce costs.
- **Fault Detection**: Fault detection refers to the process of identifying failures or abnormalities within a system. There are several different approaches to fault detection, including model-based, rule-based, and data-driven methods.

- **Sensor Fusion**

  Sensor fusion aims to improve the accuracy and reliability of the information being collected by combining data from multiple sensors. Using sensor fusion, it is possible to overcome the limitations of individual sensors, such as their accuracy, range, and sensitivity, and combine the strengths of different sensors to create a more comprehensive and reliable picture of the environment or system being monitored. Sensor fusion can achieve various goals, depending on the specific application.

- **Virtual Sensing (VS)** Virtual sensing involves using computer simulations and models to replicate the behavior of physical sensors, while sensor fusion involves combining data from multiple sensors to improve the accuracy and reliability of the information being collected. Virtual sensing can be used as an alternative to physical sensors in some applications, or it can be used in combination with physical sensors to supplement the data being collected. For example, virtual sensing can be used to simulate the behavior of sensors on a manufacturing production line to optimize the production process, while physical sensors can be used to confirm the accuracy of the simulated data.

The use of machine learning in Industry 4.0 and manufacturing has the potential to significantly improve efficiency, productivity, and reliability.

In literature, some terms are used interchangeably or with conflicting definitions, though different terms lead to different results. In the following text, we will consider Fault Detection and Diagnosis when the monitoring is in real-time, i.e., in the present. Instead, when talking of Predictive Maintenance or Fault Prediction, we are referring to the future of the system.

In other terms, differentiation is made among them based on the desired results. Detection is typically used to recognize imminent failures before the system fails. With Diagnosis, we also want to determine the fault state. Instead, with Prognosis, Fault Prediction, and PdM, we are interested in the long-term behavior of the equipment in terms of failure. Moreover, we are considering RUL as a possible approach to solving PdM.

2.3.1 Predictive Maintenance

PdM is inevitable for sustainable smart manufacturing in I4.0. The development of technology aims to increase productivity in areas such as production, maintenance, and quality in enterprises. Factors such as ineffective periods due to malfunctions in production and defective products affect productivity significantly. A maintenance strategy that is predetermined and implemented at the right time is an essential factor in increasing efficiency. Maintenance strategies, also called maintenance policies in the literature, include maintenance activities such as the parts' replacement, renewal, and repair required to ensure the continuity of the health status of the assets in the enterprise throughout their life and to fulfill the operational functions. Maintenance strategies have been

classified in different ways by many researchers. However, the literature mentions four general maintenance strategies: preventive; predictive; corrective and prescriptive maintenance [371, 136]. PdM is the process of planning maintenance activities and performing maintenance using various forecasting methods for potential failures before the failure occurs. PdM activities use data science to predict when equipment might fail. Based on the data, the fault point is estimated and maintenance activities can be planned before this point. The aim is to ensure the system's sustainability by planning the maintenance process at the most appropriate moment before the life of the equipment expires [372, 94].

This involves scheduling maintenance activities before equipment failures occur to prevent downtime and improve the overall reliability of manufacturing systems. Some of the main problems that are typically considered in the context of predictive maintenance include:

- **Equipment failures:** Predictive maintenance aims to identify potential equipment failures before they occur, in order to prevent downtime and avoid costly repairs.
- **Downtime:** Downtime can be costly for manufacturing companies, as it can disrupt production and lead to lost revenue. Predictive maintenance can help reduce downtime by identifying and addressing potential equipment failures before they occur.
- **Maintenance costs:** Predictive maintenance can help reduce maintenance costs by enabling preventive maintenance to be scheduled before equipment failures occur, rather than reactive maintenance after the fact.
- **Quality:** Predictive maintenance can help improve the overall quality of products by identifying and addressing potential equipment failures before they result in defects.

Overall, predictive maintenance is an important aspect of Industry 4.0, as it helps improve the reliability and efficiency of manufacturing systems, as well as reduce downtime and maintenance costs.

Types of Maintenance

Techniques for maintenance policies can be categorized into the following main classifications [349, 284, 283, 5, 209] and summarized in Figure 2.4.

1. **Run 2 Failure (R2F)**: also known as corrective maintenance or unplanned maintenance. It is the most straightforward maintenance technique performed when the equipment has failed. This approach can also be risky, as equipment failures can result in unplanned downtime, lost productivity, and increased repair costs. In addition, equipment that is allowed to run until it fails may also cause additional damage to other systems or components, which can further increase the cost of repairs.

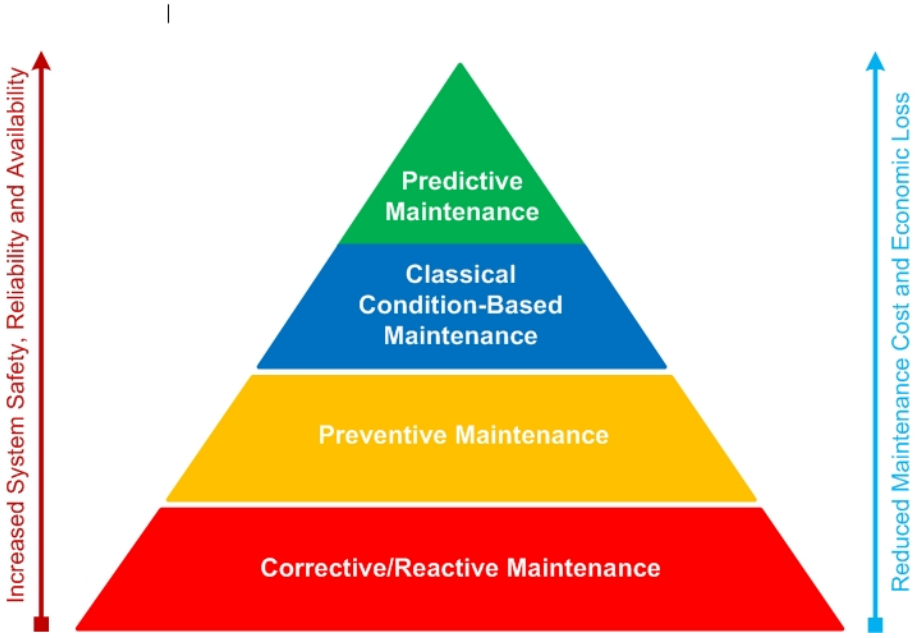


Figure 2.4: Types of Maintenance. Image from [329]

Run-to-failure maintenance can be an effective strategy in some situations, such as when equipment is inexpensive to repair or replace or when the cost of performing maintenance is significantly higher than the cost of equipment failure.

2. **Preventive Maintenance (PvM)**: also known as scheduled maintenance or time-based maintenance. PvM refers to periodically performed maintenance based on a planned schedule to anticipate failures. The frequency of preventive maintenance tasks is typically based on the type and complexity of the equipment, as well as its operating conditions and environment. For example, equipment that operates in harsh environments or under heavy loads may require more frequent preventive maintenance than equipment that operates in more benign conditions. It sometimes leads to unnecessary maintenance, which increases operating costs. The main aim here is to improve the efficiency of the equipment by minimizing the failures in production [249].

3. **Condition-based Maintenance (CBM)**: This maintenance method is based on machine or equipment monitoring of process health, which can only be executed when required The maintenance actions can only be carried out when the actions on the process are taken after one or more degradation conditions. CBM is intended to optimize maintenance activities and reduce the overall cost of maintenance by avoiding unnecessary

repairs and downtime. However, CBM usually cannot be planned.

4. **Predictive Maintenance (PdM)**: known as Statistical-based maintenance: maintenance schedules are only taken when needed. It involves using data and analytics to predict when equipment or systems will likely fail. It is based on the continuous monitoring of equipment like CBM. PdM aims to identify potential problems before they occur and to schedule maintenance activities in advance rather than reacting to problems as they occur. It utilizes prediction tools to measure when such maintenance actions are necessary. Therefore, maintenance can be scheduled. Furthermore, it allows failure detection at an early stage based on historical data by utilizing those prediction tools such as machine learning methods, integrity factors (such as visual aspects and coloration different from the original), statistical inference approaches, and engineering techniques.

It is required that any maintenance strategy minimize equipment failure rates, improve equipment condition, prolong the equipment's life, and reduce maintenance costs [64]. PdM captivates the attention of the industries; hence it has been applied in the era of I4.0 because it can optimize the use and management of assets [49, 151].

Components of a PdM system

By using PdM, organizations can improve equipment reliability, reduce downtime, and lower maintenance costs. There are several key components to a PdM program:

- **Monitoring**: This involves using sensors, instrumentation, and other diagnostic tools to gather data on the condition of equipment or systems. This data is used to identify potential problems before they occur and to determine when maintenance is needed.

- **Data analysis**: The data collected from monitoring is analyzed to identify patterns and trends that may indicate an impending failure.

- **Predictive analytics**: Predictive analytics tools and techniques are used to analyze the data and make predictions about when equipment is likely to fail.

- **Maintenance planning**: Based on the predictions made through predictive analytics, maintenance activities can be planned in advance to avoid downtime and keep equipment running smoothly.

By using PdM, organizations can improve equipment reliability, reduce downtime, and lower maintenance costs.

2.3.2 Fault Detection

Fault detection involves using sensors and other monitoring technologies to detect failures or potential failures in equipment. The goal of fault detection is to identify and fix problems before they cause equipment to fail or result in unplanned downtime. By identifying and addressing problems before they become critical, organizations can reduce the frequency and cost of repairs, improve equipment reliability and availability, and increase overall productivity.
There are several different approaches to fault detection, including model-based, rule-based, and data-driven methods. Model-based fault detection involves using a mathematical model of the system to predict its behavior and identify deviations from expected performance. Rule-based fault detection involves defining specific rules or thresholds that trigger an alert when certain conditions are met. Data-driven fault detection involves machine learning techniques to analyze sensor data and identify patterns that may indicate a fault. In the case of data-driven usually are considered two cases:

- Anomaly Detection in Unsupervised Learning: Assuming that faults are not present in the training dataset or are so rare that it is not possible to train a classification model because of the imbalance.

- Anomaly Detection in Supervised Learning using classification if there are labels that can be used

Fault detection is often implemented in combination with fault diagnosis, which involves identifying the root cause of a detected fault,

2.3.3 Remaining Useful Life (RUL)

The remaining useful life (RUL) is the period that a piece of equipment is likely to operate before it needs to be repaired or replaced. Depending on the system, this period can be represented in days, miles, cycles, or any other quantity. RUL can be an important factor in determining when maintenance or repair work should be performed or when a replacement should be considered. Therefore, it enables maintenance planning, optimizes operating efficiency, and avoids unplanned downtime [221].

RUL is an essential concept in various fields, including engineering, maintenance, and asset management. It is used to plan and budget maintenance and repair work, optimize the use of resources, and ensure the safe and reliable operation of systems and equipment.
RUL is often used in the context of predictive maintenance. In predictive maintenance, RUL can identify when maintenance or repair work should be performed to extend the useful life of the equipment or system [237]. This can be done by monitoring the performance and condition of the equipment or system over time and using data and analytics to predict when it will fail or require maintenance. Identifying potential problems before they occur makes it possible to take proactive action to prevent failures and extend the useful life of the

equipment or system.
The estimation of RUL is a real challenge because the relevance and effectiveness of maintenance actions depend on the accuracy and precision of the results obtained $RUL = t_f - t_c$. A confidence measure should also be constructed to indicate the degree of certainty of the RUL.

Generally, two types of methods are used to predict equipment life: physics-based and data-based methods [361].

The physics-based approach uses mathematical principles such as statistics and probability to create accurate mathematical models from prior knowledge to predict equipment life.

However, relying on prior knowledge can lead to poor generalization. Data-driven methods aim to explore the potential relationship between monitored sensor data and their RUL value based on historical data.
The RUL of equipment is a random variable that depends on the equipment's current age and health information. It gives a machine, component, or system the remaining time before it is no longer functional. The research has shown interest in using remaining life estimates within industries [370].
Machine learning can be used to estimate the remaining useful life (RUL) of equipment or systems. In addition, these techniques can analyze data from sensors or other sources to identify patterns and trends that may be used to predict the future performance of the equipment or system.
One common technique for estimating RUL using machine learning is to use regression algorithms. Regression algorithms are designed to predict a numerical output based on input features. For example, a regression algorithm could be trained to predict the RUL of a piece of equipment based on data about its usage, maintenance history, and operating conditions.
Another common machine learning technique for estimating RUL is classification. Classification algorithms are designed to predict a categorical output based on input features. For example, a classification algorithm could be trained to predict whether a piece of equipment will likely fail within a certain period based on data about its usage, maintenance history, and operating conditions.
In particular, Deep Learning techniques can also be used to estimate RUL. These techniques can analyze complex data and identify patterns and trends that may not be detectable using other methods. In addition, Deep Learning algorithms can be trained on large amounts of data and often make more accurate predictions than machine learning techniques.

2.4 Machine Learning for I4

Artificial intelligence (AI) as an intelligence exhibited by machines has been an effective approach to human learning and reasoning [213]. Machine learning and data mining have recently become the center of attention and the most popular topics among the research community [306]. This section presents the various machine learning algorithms considered in the literature in the context of Industry 4.0. For each algorithm, we provide a brief overview of its functioning

and some examples of its use in the context of Industry 4.0. Please note that these are just a few examples, and much more research has been conducted in this field.

ML uses increased computing power and various software to gain meaningful information and knowledge from big data collected from the environment [23]. The most important techniques that are used for learning, classified by the available feedback, are supervised, unsupervised, and reinforcement learning methods [205].

Machine learning (ML) techniques have emerged as a promising tool in PdM applications for smart manufacturing in I4.0; thus, it has increased the attraction of authors in recent years. In general, ML applications provide some advantages, which include maintenance cost reduction, repair stop reduction, machine fault reduction, spare-part life increases and inventory reduction, operator safety enhancement, increased production, repair verification, an increase in overall profit, and many more. These advantages also have a tremendous and strong bond with the procedures of maintenance [49, 226, 264, 30, 312]. As of today, ML techniques have been widely applied in several manufacturing areas (such as maintenance, optimization, troubleshooting, and control) [335]. The rapid development of technologies interconnected with ICT and IoT enables manufacturing growth, leading to Industry 4.0. The implementation of CPS combined with IoT can provide intelligent, flexible systems capable of self-learning, which presents the core of Industry 4.0 [213]. As a part of an intelligent system in Industry 4.0, ML is broadly implemented in various fields of manufacturing where its techniques are designed to extract knowledge from existing data. The new knowledge (information) supports the decision-making process of a manufacturing system. But the ML techniques' end goal is to detect patterns among the data sets or regularities that describe the relationships and structure between those sets [336].

ML is a technology by which the outcomes can be forecasted based on a model prepared and trained on past or historical input data and its output behavior [194]. ML approaches have tremendous advantages, as they can handle multivariate, high dimensional data and extract hidden relationships within data in complex, dynamic, and chaotic environments [49, 335, 295]. However, the performance and advantages might differ depending on the ML approach chosen. As of today, ML techniques have been widely applied in several manufacturing areas (such as maintenance, optimization, troubleshooting, and control) [335].

From the manufacturing perspective, various types of big data sets can be captured, collected, extracted, and analyzed to improve the traditional manufacturing systems [213, 90]. Finding the knowledge in big data and transforming it into information is done by Knowledge Discovery in Databases (KDD) with the help of ML techniques [213]. According to escobar2017machine, the primary objective of ML applications in combination with big data analytics is the achievement of defect-free and fault-free processes. Most manufacturing difficulties are related to classification issues, where the experts in the industrial field have to determine a class label for a specific object or situation based on the big data set [231].

The applications of ML in manufacturing refer to pattern recognition in existing sets of data. That is beneficial for the development of foreseeing the future behavior of the manufacturing system with the end goal being to detect the present behavior patterns or regularities that describe relations between data [213], [336]. Also, supervised learning is employed for investigating the decision-making and process-planning problems in manufacturing [231].
Nowadays, ML algorithms have wide utilization in different manufacturing areas such as optimization, troubleshooting, and quality control [23]. Moreover, the result of the scientific research has shown that ML techniques are considered a powerful tool for permanent quality improvement in a large and complex process, e.g., semiconductor manufacturing [231].

2.4.1 Supervised Learning

In supervised learning, a system is trained with data that has been labeled. The labels categorize each data point into one or more groups. Then the system learns how this training data is structured and uses this to predict which categories to classify new output. The final goal of supervised learning process is that the outputs are close enough to be useful for all given input sets.

The most common supervised machine learning assignments are classification, and regression [336]. In classification assignments, the program has to learn how to predict the most likely category, class, or label for discrete output values from one or more input data sets. Similar to classification, regression problem also requires supervised learning techniques. The difference in regression problems is that programs must foresee and predict the value of a continuous output by themselves [241]. According to [336] as well as [129], supervised learning is the most commonly used ML technique because majority of applications can provide labeled data.
Supervised learning is a type of machine learning in which an algorithm is trained on a labeled dataset, meaning that the data has been labeled or classified in some way. Supervised learning aims to make predictions or classify new data based on the patterns and relationships learned from the training data.
For example, in Industry 4.0, supervised learning could predict equipment failures or maintenance needs by training a model on a dataset of historical equipment performance data labeled with failure or non-failure instances. This could allow a company to schedule maintenance and prevent unexpected downtime proactively. Supervised learning could also be used to optimize production processes by predicting the yield of a manufacturing process based on various input variables, such as raw material quality and equipment performance. This could allow a company to optimize its production and reduce waste. Supervised learning could also improve supply chain management by predicting demand for products or materials based on historical data, allowing a company to optimize its production and inventory management.

2.4.2 ML Algorithms in Manufacturing

Support vector machine (SVM)

A supervised learning algorithm is used for linear and non-linear problems, such as classification and regression [259]. SVMs work by finding the "hyperplane" that maximally separates the data points of different classes. The hyperplane is chosen such that it maximally separates the data points of different classes and has the largest margin, or distance, between the data points and the hyperplane. The data points closest to the hyperplane are called "support vectors," and they play a critical role in determining the position of the hyperplane. Once the SVM has been trained on a labeled dataset, it can classify new data points by determining which side of the hyperplane they fall on. SVMs are popular because they are effective at handling high-dimensional data and relatively simple to implement. The biggest drawbacks of SVM are its slow learning speed and its lack of explanation ability to humans [147].
A SVM model is used in [234] to identify failures in automotive transmission boxes. In this study, four gearboxes are tested at two different speeds and load conditions, and then the resources are extracted from the vibration signals acquired to train a SVM. Another work proposing a SVM model for PdM is [288]. In this work, an ion filaments prediction module is built. The proposed model is based on the decision limit provided by the SVM model. The used data are synthetic and generated using Monte Carlo simulation. As last example, a SVM is employed in [160] to predict alarm faults in a bearing of a rail network.

Random Forest

A decision tree (DT) is a tree-like model of decisions, with an internal node representing a feature, the branches representing the possible decisions based on that feature, and the leaf nodes representing the final decision or prediction. To create a decision tree, the algorithm starts at the root node and splits the data into subsets based on the feature that maximally separates the data into different classes or values.
A random forest is an ensemble learning method that combines the predictions of multiple decision trees to make a more accurate and stable prediction. RF was developed by Breiman [37]. It works by training a large number of decision trees on randomly selected subsets of the data and then averaging their predictions to make the final prediction. This helps to reduce the overfitting that can occur when training a single decision tree on the entire dataset. According to [28], RFs have shown good performance when the number of variables is larger than the number of samples (observations).
In [235] a generic method for predicting repairs to various components of commercial vehicles. However, the method is evaluated using only air compressors. In [45], RF generates dynamically predictive models. This work proposes an improvement of the paper [154], where monitoring of wind turbines is performed. To do this, status data (activated and disactivated alarms) and operational data (about the performance of the wind turbines) are employed to design the RF

model. The main contributions of the paper [45] are speed in data processing, scalability, and automation. In [281] it proposed a real-time predictive fault detection system to perform hard disk drive faults. As last example, [150] apply a RF model to detect the presence or absence of an issue in refrigeration and cold-storage systems.

XGBoost (eXtreme Gradient Boosting)

XGBoost was developed by [56], a scalable tree boosting system widely used by data scientists that provides state-of-the-art results on many problems. It is a gradient boosting algorithm that can be used for classification and regression tasks. It works by training a sequence of weak decision trees and combining their predictions through an ensemble method to make a final prediction. XGBoost is known for its efficiency and effectiveness and has achieved state-of-the-art performance in many machine learning competitions.
For instance, research on predicting a printing machine's downtime was reported based on real-time predictions of imminent failures [359]. Their study used unstructured historical machine data to train the ML classification algorithms, including RF and XGBoost, to predict machine failures. For Predictive Maintenance, in [302] used XGBoost for Data-wind turbines.

k-Nearest Neighbour (KNN)

Supervised machine learning algorithm that can be used for classification or regression. It stores all available cases and classifies new ones based on a similarity measure (e.g., distance functions). Classification is done by a majority vote of its k nearest neighbors. For example, if k=3, the new case would be classified by a simple majority vote of its 3 nearest neighbors. One of the main advantages of k-NN is that it is a simple, easy-to-implement algorithm. It is also relatively fast and can handle high-dimensional data well. However, one of the main disadvantages of k-NN is that it can be computationally expensive to find the k nearest neighbors for each new data point, especially if there is a large training dataset. Moreover, it is important to keep in mind that the performance of the k-NN algorithm can be sensitive to the choice of k and the distance metric. Finding the k nearest neighbors for each new data point can be computationally expensive.
The k-NN (k-nearest neighbor) algorithm can be used in Industry 4.0 for predictive maintenance, quality control, and fault detection. Based on sensor data, k-NN can classify equipment as "healthy" or "failing" in predictive maintenance. In quality control, k-NN can classify products as "defective" or "non-defective" based on data collected during manufacturing. In fault detection, k-NN can classify equipment as "normal" or "faulty" based on data collected from the equipment. A real study of kNN was performed in [314] for Fault Diagnosis on rolling bearings.

Naive Bayesian

It is a supervised machine learning algorithm based on the Bayesian theorem, which states that the probability of an event occurring is equal to the prior probability of the event occurring multiplied by the likelihood of the event occurring given certain observations or evidence. In machine learning, Naive Bayes can be used for classification tasks. It makes predictions based on the probability of an event occurring, given certain features or attributes. One of the main advantages of Naive Bayes is that it is a simple, easy-to-implement algorithm that can be used for a wide range of classification tasks. It is also relatively fast and can handle large amounts of data well. However, one of the main disadvantages of Naive Bayes is that it assumes independence between features, which is often not true in real-world data. This can lead to poor performance on some tasks. A Naive Bayesian Classifier for Remaining Useful Life Prediction was used considering degrading bearings [78]. Naive Bayesian Classifier was used in [118] for Predictive Maintenance in Oil and Gas.

Deep learning

The Deep Learning (DL) concept appeared for the first time in 2006 as a new field of research within machine learning [306]. Deep learning is inspired by the structure and function of the brain, specifically the neural networks that make up the brain. It is called "deep learning" because it involves training artificial neural networks with many layers, typically consisting of multiple interconnected "hidden" layers between the input and output layers. The ANN allows an artificial system to perform supervised, unsupervised and reinforcement learning assignments [336]. Deep learning has achieved state-of-the-art performance on several tasks, including image and speech recognition, natural language processing, and machine translation. It has also been applied to other domains, including healthcare, finance, and self-driving cars. One of the key advantages of deep learning is its ability to automatically learn features from raw data without manual feature engineering. This makes it well-suited to tasks where the relationships in the data may be complex and difficult to specify explicitly. However, training deep learning models can be computationally intensive, requiring a large amount of labeled training data to perform well. Additionally, it can be challenging to interpret the decisions made by deep learning models, as they are typically composed of many layers of interconnected neurons, making it difficult to understand how the model arrived at a particular decision.

- **Feedforward neural networks**: These are the most common type of neural networks, in which the data flows through the network in one direction, from the input layer to the output layer, without looping back.

- **Convolutional neural networks (CNNs)**: These are specialized neural networks for processing data with a grid-like structure, such as an image. They are composed of convolutional layers, which apply a set of filters to

the input data to extract features, and pooling layers, which reduce the dimensionality of the data.

- **Recurrent neural networks (RNNs)**: These are neural networks that are designed to process sequential data, such as time series or natural language. They are composed of recurrent layers, which allow the network to maintain a state that depends on the previous input.

- **Generative adversarial networks (GANs)**: These are a type of neural network that consists of two networks: a generator network and a discriminator network. The generator network generates synthetic samples, while the discriminator network attempts to distinguish the synthetic samples from real samples. The two networks are trained simultaneously, with the generator network trying to generate samples that the discriminator network cannot distinguish from real samples.

Artificial Neural Networks (ANNs) are among the most common and applied ML algorithms. They have been proposed in many industrial applications, including soft sensing [276] and predictive control [270]. [31] employs an ANN. It was developed as bench test equipment designed to mimic the operational condition of a wind turbine to monitor its conditions. This procedure enables fault recognition in the critical components of the wind turbine. The authors collected vibration data in a healthy condition and a deteriorated condition. In [146] ANNs detect faults in industrial equipment for anode production in real-time, using process sensor data from operation periods. In [218], Convolutional Neural Network (CNN), a class of deep learning algorithms, is proposed to predict faults in acoustic sensors and photovoltaic panels, respectively.

2.4.3 Unsupervised Learning

Unsupervised Learning represents learning where the evaluation of the action is not dependent, provided or supervised, because there is no expert [336]. Unlike Supervised Learning, Unsupervised Learning does not learn from labeled data. Instead of that, it discovers patterns in the data. The main goal of Unsupervised Learning is to discover the unknown relationships between samples using clustering analysis [336]. Another unsupervised learning task is dimensionality reduction [129]. It represents the process of discovering the relationships between input data sets and can be used for visualizing. Considering that some problems might contain thousands and thousands of input data, a problem with big data is that it becomes impossible to visualize [23].

- **Dimensionality Reduction**: In unsupervised learning, dimensionality reduction is often used as a tool for exploring and understanding the underlying structure of the data. By reducing the dimensionality of the data, it can be easier to visualize and identify patterns or trends in the data.

Dimensionality reduction can also extract meaningful features or representations of the data, which can be used as inputs to other machine learning algorithms.

- **Clustering**: The assignment is to discover groups of related observations of the input data, namely clusters [23]. Such observations within groups have cognition based on some similar measurements where similar points are grouped together. Clustering is a common approach used in unsupervised learning. Clustering aims to identify natural groups or patterns in the data that may not be immediately obvious.
- **Anomaly detection**: Anomaly Detection (AD) is an important and challenging problem in the field of Machine Learning (ML). Anomalies are patterns characterized by a noticeable deviation from the so-called normal data, where normal means compliance with some typical or expected features. In order to effectively detect anomalies, it is important to clearly understand what constitutes normal behavior and how to differentiate it from abnormal patterns.
- **Auto-Encoders**: These are a type of neural network that can be used for dimensionality reduction and feature extraction.
- **Association rule learning**: This involves identifying relationships between variables in the data, such as finding items that are frequently purchased together.

Auto-Encoder(AE)

Auto Encoder (AE) is an unsupervised learning algorithm that extracts features from input data without label information needed. It mainly consists of two parts, an encoder and a decoder. This type of neural network can be used for dimensionality reduction. The encoder can perform data compression, especially in dealing with input of high dimensionality, by mapping input to a hidden layer [252]. The decoder can reconstruct the approximation of input. For example, suppose the activation function is linear and we have less hidden layers than the dimensionality of input data [319].
Several variants of AE have been developed and listed as follows:

- **Vanilla autoencoder**: This is the simplest form of autoencoder, which consists of an encoder and a decoder network, with the goal of reconstructing the input data from a reduced-dimensional representation.
- **Convolutional autoencoder**: This type of autoencoder is designed to process data that has a grid-like structure, such as an image. It uses convolutional layers in the encoder and decoder networks to capture the spatial relationships between pixels in the input data.
- **Denoising autoencoder**: This type of autoencoder is trained to reconstruct the original input data from a corrupted version of the input. The

goal is to learn a robust representation of the data that is robust to noise and other forms of corruption.

- **Sparse Auto Encoder (SAE)**: SAE makes the most of the hidden unit's activations close to zero by imposing sparsity constraints on the hidden units, even the number of hidden units is large.
- **Contractive Auto Encoder (CAE)**: In order to force the model resistant to small perturbations, CAE encourages learning more robust representations of the input x.
- **Variational autoencoder (VAE)**: This probabilistic model learns a continuous latent space, which can be used to generate new samples that are similar to the training data. The key difference between a VAE and a traditional autoencoder is that the latent space learned by a VAE is continuous and has a well-defined probability distribution, typically a Gaussian distribution. This allows VAEs to generate new samples by sampling from the latent space and passing them through the decoder network. As a result, VAEs have many applications, including image generation, text generation, and anomaly detection.

Clustering

There are several different approaches to clustering, including k-means, hierarchical, and density-based. K-means clustering involves dividing the data into a predetermined number of clusters based on the distance between the data points. Hierarchical clustering involves creating a hierarchy of clusters, with each cluster divided into smaller subclusters. Density-based clustering involves identifying clusters based on the density of the data points, with clusters being defined as areas of high density surrounded by areas of lower density. These different approaches can be applied in various contexts, including Industry 4.0, to gain insights and improve decision making.

K-means is a clustering algorithm used to partition n observations into k clusters, where each observation belongs to the cluster with the nearest mean. This popular approach involves dividing the data into a predetermined number of clusters based on the distance between the data points. The k-means model is a popular clustering algorithm that uses an unsupervised strategy to determine a set of clusters [76]. The main aim is to find the k partitions (or clusters) of the dataset so that "close" samples to each other are associated with the same cluster, and "far" samples from each other are associated with different clusters [36]. The k-means algorithm is easy to implement. In addition, it presents good performance and handles large data sets (as long as the number of clusters k is small), and it can change the centers of the clusters with retraining when new samples are available. Another important feature of the k-means algorithm is that it minimizes inter-class variance.
It is commonly used in the industry for data analysis and data mining tasks. In the context of Industry 4.0, k-means can be used to cluster data from sensors

and other sources of data in order to gain insights and improve decision-making. For example, k-means could be used to cluster data from sensors on a manufacturing line to identify patterns and improve efficiency.
For example, in [87] is employed k-means to analyze the behavior of wind turbines by using vibration signal analysis. In [89], k-means is applied to automatically extract groups (clusters) in dissolved gas data in the insulating oil of a transformer. The aim was to identify the characterization of each cluster that induces a fault or an alert for possible maintenance actions. The k-means clustering algorithm was developed using Euclidean distance as a similarity criterion. In [303], it is proposed to use a k-means algorithm to identify clusters using data (i.e., sensors of the platform temperature, oxygen percentage in the process chamber and process chamber pressure) from a selective laser melting machine tool.

Conclusions

To summarize, Unsupervised learning is a type of machine learning where the data is not labeled or annotated, and the goal is to discover patterns or relationships in the data. Unsupervised learning algorithms do not have a specific target or output they are trying to predict. Instead, they must rely on the structure of the data to identify patterns or relationships.

In the context of Industry 4.0, unsupervised learning could be used to improve various manufacturing and supply chain processes. For example, unsupervised learning can detect anomalies in production data, such as unexpected changes in output or quality, which could indicate a problem with the manufacturing process. Unsupervised learning could also be used to identify patterns in customer behavior, such as purchasing patterns or preferences, which can optimize marketing and sales efforts.

Unsupervised learning could also improve supply chain management by identifying patterns in the flow of materials and finished products through the supply chain. This could allow a company to optimize its logistics and transportation operations and identify potential bottlenecks or inefficiencies.

2.4.4 Reinforcement Learning

Reinforcement learning is a type of machine learning in which an agent learns by interacting with its environment and receiving feedback in the form of rewards or punishments. The goal of reinforcement learning is to learn a policy that maximizes the agent's cumulative reward over time.

In reinforcement learning, the agent learns by trial and error, gradually improving its performance as it receives more and more feedback. The agent learns through a process of exploration and exploitation, trying out different actions to see what works best and then relying on the actions that have proven most effective in the past.
One potential use of reinforcement learning in Industry 4.0 is optimizing production processes. For example, an industrial robot could use reinforcement

learning to learn how to perform tasks more efficiently by receiving rewards for completing tasks and punishments for mistakes. The robot could then use this learning to improve its performance continually. Another potential use of reinforcement learning in Industry 4.0 is optimizing supply chain management. For example, a company could use reinforcement learning to learn how to effectively manage the flow of materials and finished products through its supply chain by receiving rewards for meeting delivery deadlines and punishments for delays.
In process control, to circumvent a conventional model-based approach and an online adaptation to continuous process modifications, [215] initially developed a deep RL approach to control the liquid levels of multiple connected tanks. The controller minimized the target state difference and adjusted inlet flow rates between multiple tanks accordingly. In production scheduling, [166] proposed a study. High uncertainties regarding customized products, shutdowns, or similar cause the complexity of production scheduling. To cope with the complexities and to reduce human-based decisions, [166] proposed a multi-class DQN approach that feeds local information to schedule job shops in semiconductor manufacturing. Based on the edge framework, the DQN demonstrated superior performance and reduced makespans and average flow times. Another application for RL is personalized production, it has an enormous impact on the complexity of production control due to individual product configuration options. Depending on the customer requirements, the products must be dispatched to where they can be processed, under consideration of several technical and logistic constraints and optimization variables [325]. To meet the requirements in wafer fabrication dispatching, [12] implemented a single agent DQN that processed 210 data points as a single state input (such as machine loading status or machine setup). This enabled the DQN to meet strict time constraints better than competitive heuristics (TC, FIFO) while reaching predefined work-in-progress (WIP) targets as a secondary goal. Regarding maintenance, the interaction of several linked machines in a serial production line was considered by [122]. Based on a large state space containing buffer levels, operating inputs, and fault indicators for each machine, the algorithm made decisions about which machines needed to be turned off at a given time for service. A review of CL approaches applied in production systems is proposed in [219].
Reinforcement learning has the potential to be a powerful tool for improving industrial processes and supply chain management in the context of Industry 4.0 by allowing systems to learn and adapt to changing conditions and optimize their performance over time.

2.4.5 Public data sets for predictive maintenance

Public datasets are useful in research because they enable the comparison and evaluation of various research approaches. They aid research efforts' progress and robustness by offering a shared base for comparison. Public datasets can help to establish standards and best practices for data collection and contribute to the repeatability and dependability of the research. Furthermore, public datasets allow researchers to test their models and algorithms on a

Table 2.2: A short list of public datasets for predictive maintenance.

Reference	**Description of Dataset**
Lopes & Camarinha-Mato, 1999	Force and torque measurements to detect robot failures
Lopes & Camarinha-Mato, 2009	Failure data of a generic gearbox
Lindgren & Biteus, 2016	Operation data and failures of a pressure pressurizing system of a truck
Tarapore et al., 2017	Failure data in a simulated swarm of 20 e-puck robots (mobile robot with differential wheels)

broader and more diverse set of data, which can aid in identifying potential biases and limitations as well as improving the generalization of the study. Moreover, they can stimulate the sharing of fresh work and ideas to advance the field.

On this premise, some examples of public datasets for Predictive Maintenance in the context of Industry 4.0 are discussed in [50] and reported in the Tab. 2.2.

The first data set, proposed by Lopes and [177], aims to detect robot failures using force and torque measurements. It consists of 463 samples and 30 attributes. The second data set, proposed by [176], aims to detect faults and estimate magnitudes for a gearbox using accelerometer data and information about bearing geometry. The third data set, proposed by [169], aims to detect component failures in an air pressure system for trucks. It consists of 76000 samples and 171 attributes. Finally, the fourth data set, proposed by [291], aims to detect faults in robot swarms. For further information about the data set, see their references. Another interesting list of public datasets is provided by [135].
The public datasets can be used to test and evaluate PdM methodologies in different scenarios. However, it is important to highlight that a PdM methodology is unique and depends on the application [50]. It depends on the environment, the produced data, and the equipment. And if any of these entities changes, then, in most cases, the PdM methodology also needs to be changed. These public data sets can support new researchers in developing, testing, and comparing different ML techniques in PdM applications.

2.5 Challenges

Even though adopting predictive maintenance in an industrial context is inevitable, it is surrounded by challenges that hinder the application and collective adoption of this smart maintenance approach.

Table 2.3: Notation used in this chapter

Nomenclature	**Symbol**
Artificial Intelligence	AI
Artificial Neural Network	ANN
Augmented Reality	AR
Condition-Based Maintenance	CBM
Cyber-Physical Systems	CPS
Deep Learning	DL
Denial of service	DoS
Digital Twin	DT
Human Machine Interface	HMI
Industrial Internet of Things	IIoT
Internet of Things	IoT
Machine Learning	ML
Predictive maintenance	PdM
Preventive maintenance	PM
Prognostics and Health Management	PHM
Return on investment	ROI
Remaining Useful Life	RUL
Virtual Reality	VR

2.5.1 Cybersecurity

Cybersecurity is the set of practices to protect computers, servers, mobile devices, electronic systems, networks, and data from digital attacks, theft, and damage. It involves using technologies, processes, and policies to secure networks, devices, and data from unauthorized access, use, disclosure, disruption, modification, or destruction.

Cloud-based systems, the IoT, and the interconnectedness of smart industries have significantly increased unexpected security breaches [198]. Due to digitalized and connected business processes, Industry 4.0 is more vulnerable to cyber espionage or cyber sabotage. Currently, we are witnessing the development of well-organized groups of cyber criminals with excellent skills and accustomed to targeting specific industries to hack sensitive information and intellectual property.

The problem associated with this phenomenon is not limited to its impact on sales but also includes damage to the organization's image, loss of know-how and reduction in the level of competitiveness of affected organizations [224] [134].

One of the characteristics of Industry 4.0 is the ability to connect across organizational environments, which has the potential to make the AP provisioning chain more efficient. However, supply chain systems have inherent security vulnerabilities that attackers exploit.

One such security vulnerability is at the supplier level, which is vulnerable to

phishing attacks and theft of privileged credentials, resulting in massive data exposure. Security awareness, access control through authentication mechanisms, cryptographic processes, and behavioral analysis are the security mechanisms that can help prevent supply chain hacking [25].

Another type of attack is Denial of service (DoS), which makes a system or application unavailable. For example, a DoS attack can be achieved by bombarding a server with a large number of requests to consume all available system resources, passing malformed input data to the server that can crash a process, infiltrate a virus, or destroy or disable a sensor in a system, not allowing it to function normally [224]. Industry 4.0 relies on a large number of interconnected systems and processes, and DoS attacks are a very significant threat in such environments [222].

The transition to Industry 4.0 is a monumental task impacting many areas of today's manufacturing industry, including security. Most manufacturing companies are unaware of the security risks associated with adopting the Industry 4.0 paradigm [198]. Typically, they only address security issues when a serious incident occurs. Therefore, it is critical and essential that organizations adopt the development of a strategy to deploy and manage the security compliance processes that Industry 4.0 requires, including reducing the organization's exposure and effectively managing the mitigation process [25]. In other words, individuals and organizations must stay up-to-date on the latest cybersecurity threats and best practices to protect themselves and their sensitive information.

2.5.2 Financial and Organizational Limits

Despite the availability of predictive maintenance algorithms, companies that want to benefit from Industry 4.0 still have to trade off the opportunities of predictive maintenance against the capital expenditure required to purchase necessary instrumentations, software, and expertise. This disadvantage is more important in the early stages of predictive maintenance development when actual data on normal and abnormal equipment behavior is lacking or scarce, and in the case of new systems, when there is no experience with their operation.

Predictive maintenance efforts, such as sensor installation, information retrieval, model preparation and maintenance, and maintenance activities, generate costs for the companies in which predictive maintenance methods are introduced. These costs can vary depending on multiple factors, such as the type and complexity of equipment and corresponding sensors, the cost of consulting, installation, and knowledge extraction, and whether the necessary expertise can be found internally or externally [327]. Moreover, companies may face challenges finding and retaining skilled workers familiar with Industry 4.0 technologies. This can be particularly challenging in industries not traditionally associated with technology.

Companies may hesitate to invest in Industry 4.0 technologies if they are not convinced that the benefits outweigh the costs. This can be especially challenging if the benefits of these technologies are difficult to quantify or are not immediately apparent. One method of assessing whether the introduction

of predictive maintenance can be beneficial is to create a projected return on investment (ROI) [79]. The projection of the predictive ROI should consider the value of predictive maintenance results, the payback time, and the described costs.

2.5.3 Data Source Limits

The availability of relevant data is essential for creating a production process management model. However, companies rarely have all the relevant data at the beginning of the introduction of production process management [185]. After using the already available data, it is necessary to identify the gaps and aim to resolve them.

Furthermore, the quality of the available data can vary significantly, and poor quality data can limit the usefulness of the insights that can be derived from it. For example, data that needs to be completed, corrected, or updated can lead to incorrect conclusions or decisions. If only part of the data is affected by unsatisfactory quality, this can be overcome during data preparation, as long as the amount of data points is sufficient to achieve statistical significance and as long as defect detection can successfully isolate machine-critical points [137].

In Industry 4.0, various sources often generate data, including sensors, devices, and systems. Integrating this data can be challenging, as different sources may use different formats, protocols, and standards.

The company using predictive maintenance methods may then face challenges when the necessary confidence in the data does not hold true, i.e., if sensors, controllers, or other data sources provide inaccurate measurements. This can result in incorrect predictions and missed maintenance urgency or false alarms. Another challenge for sensor technology is that sensors currently operate offline without contributing to online data. In addition, sensors are subject to downtime, instrument degradation, noise, or simply the sensor may fail. It is then important to clean the data before applying the predictive maintenance algorithm to predict the true reality and not distort the results.

Other challenges are connected to Data Privacy, Industry 4.0 involves the collection and analysis of large amounts of data, which raises concerns about data privacy and the potential for misuse of personal information. Moreover, it is also necessary to focus on the regulations and standards for using such data, which can create uncertainty for companies looking to adopt these technologies.

2.5.4 Human-machine interaction

Human intelligence and intervention play a key role due to the safety, security, and social aspects and the uncertainties posed by these autonomous and intelligent systems. In addition, in parallel with the advanced technologies of these intelligent systems, the role of humans has evolved from low-level operations that can be dangerous, difficult, and boring to highly specialized and safe tasks [212]. However, humans may feel they are easily changed due to technology

implementation. In addition to technical skills, it has been pointed out that human work in the manufacturing sector increasingly requires technological skills, social and communication skills, as well as team skills and self-management skills [245] [308]. The essential skills to perform the tasks must be identified and training provided to meet the requirements. The person should have more opportunities for autonomous decision making, diversity in the workplace, and social interaction [222]. In other words, Industry 4.0 technologies often involve close interaction between humans and machines, which can raise concerns about job displacement and the safety of workers. Ensuring that the interaction between humans and machines is safe and efficient is essential.

2.5.5 Interpretability

In machine learning, Interpretability entails comprehending how a model produces predictions. This ability is crucial in many fields, especially for Industry 4.0. Some ML methods, like linear regression and simple decision trees, are intrinsically interpretable and provide a feature importance score for each of the input variables. However, many ML models are known as "black-box" since they cannot explain how these predictions are produced. Many advanced models nowadays are black-box models, not only neural networks but also other models like random forests. These models are becoming a growing challenge in Industry 4.0 because ML models are frequently connected to real-world equipment installed in a real-world environment. As a result, understanding how a model produces predictions is critical because any error or bias in the model might have serious implications. For example, to avoid edge cases in robotics, it is essential to comprehend why machines respond differently based on diverse inputs. Another example is using a PdM model to determine when a breakdown will occur and the reasons behind that. Moreover, to improve human-machine interaction is necessary to have a complete understanding of the model that is used.

2.5.6 Machine Repair Activity Limits

By being able to predict the remaining life of a component, maintenance times can be determined, but the actual maintenance of a component still faces challenges related to the dependence on human interactions and the lack of self-maintenance [206].

The effectiveness of maintenance depends on the quality of human management and skills, given that machine components currently depend on human operators for control and maintenance. Repairing complex machines often requires specialized technical knowledge and skills. Therefore, companies may need to invest in training and education in order to ensure that their technicians have the necessary knowledge and expertise to perform repairs. Moreover, repairing machines can involve working in hazardous environments or with hazardous materials, and it is important to ensure that appropriate safety measures are in place to protect technicians and other workers.

Industrial machines particularly work by executing commands in a reactive manner and do not question the plan for them. However, human task planning is based on data and experience, which the machine might also be able to retrieve. Thus, an intelligent component could autonomously propose or even initiate actions that are beneficial to system health, asset throughput, or product quality. However, currently, industrial machines do not have this level of self-awareness and self-maintenance.

2.5.7 Limits in the Deployment of Industrial Predictive Maintenance Models and Data Distribution Shift

Phases of a ML project

Today, manufacturing is facing an increment of challenges related to complexity and dynamic behaviours [23] while adding the fact that the manufacturing is affected by uncertainty [336]. In other words, the constant enlargement of big data and its availability, high-dimensionality, variety as well as homogeneity represents the main challenges in manufacturing environment because the knowledge cannot be extracted [23].
Usually a typycal ML project can be splitted in the following points:

- **Data acquisition**: during this phase one or more datasets are collected using several acquisition tools available in the system under exam. It is possible to distinguish two different procedures for the acquisition process:
  - A Design of Experiment (DOE) can be realized and a series of tests will be performed with the only purpose to acquire data
  - The data are constantly acquired during the normal functioning of the system
- **Data Analysis**: the acquired datasets are analyzed using graphical tools or unsupervised methods with the purpose to discover some features that better represent the relationship between the input data and output data.
- **Feature Extraction**: features are a series of quantities which represent the information extracted from raw data. This step can be performed using automatic tools like PCA (principal component analysis) or manually extracting some significant characteristics of the data
- **Modeling**: After the previous step, all extracted features are collected in a matrix called model matrix, which is used to train the desired models. Usually, performance evaluation and comparison is performed between different models using methods such as K-fold or Monte-Carlo [212].
- **Roll out**: The most suitable model is chosen and deployed in the real system. Its behavior is then monitored to asses its quality over time and the correct functioning. The model is available for use in a limited or controlled way, typically to a small group of users or in a specific location.

This is often done as a way to test the model's performance and gather feedback before making it more widely available.

When an ML model learns from the training data, it means that the model learns the underlying distribution of the training data to leverage this learned distribution to generate accurate predictions for unseen data, data that are not shown during training. The assumption is that the unseen data comes from a stationary distribution that is the same as the training data distribution. If the unseen data comes from a different distribution, the model might not generalize well [282].
However, Roll out of a model is not the end of the process. Deploying a model, on the other hand, refers to the process of making a model available for use in a production environment. This typically involves integrating the model into an application or system and making it available to a larger group of users. Once a model has been deployed, we still have to continually monitor its performance to detect issues and deploy updates to fix these issues. The assumption is that the unseen data comes from a stationary distribution that is the same as the training data distribution.
Generally, there are three challenging steps after developing failure prediction models: their **integration, monitoring, and updating**.

Integration

To be deployed in the real world often is necessary also an integration step. Model integration in the industry is a challenge because an information technology (IT) team often performs this task, which is usually dissociated from the team of researchers and developers who developed the predictive maintenance models. Building such an IT infrastructure to maintain the data pipelines can be laborious and is not usually factored into the project planning.
This part is very relevant and challenges of this phase are associated to software system failures. Software system failures are failures that would have happened to non-ML systems. Some examples of software system failures are:

- Dependency failure: a software package or a codebase that your system depends on breaks, which leads your system to break. This failure mode is common when the dependency is maintained by a third party and especially common if the third-party that maintains the dependency no longer exists.

- Deployment failure: failures caused by deployment errors, such as when you accidentally deploy the binaries of an older version of your model instead of the current version or when your systems don't have the right permissions to read or write certain files.

- Hardware failures: when the hardware that you use to deploy your model, such as CPUs or GPUs, doesn't behave the way it should. For example, the CPUs you use might overheat and break down.

- Downtime or crashing: if a component of your system runs from a server somewhere, such as AWS or a hosted service, and that server is down, your system will also be down.

To understand the importance of this aspect, it should be observed that they reviewed data from over the previous 15 years to determine the causes and found out that 60 out of these 96 failures happened due to causes not directly related to ML.

Monitoring

When a machine learning model is in production, one of the main problems that can occur is a shift in the data distribution, also known as "concept drift." This can happen when the statistical properties of the data that the model was trained on change over time, resulting in a mismatch between the model's expectations and the new data is being applied to. This can lead to a decrease in the model's performance and accuracy.
In the monitoring phase of a machine learning system in production, the system is continuously monitored to ensure it performs as expected and to identify any potential issues or problems. This can involve monitoring model performance, monitoring the accuracy and performance of the machine learning model over time to ensure that it is performing as expected. This may involve tracking key metrics such as precision, recall, and F1 score, as well as monitoring the model's output to ensure that it is making accurate predictions.
Instead of model performance is possible to monitoring also data quality. It is important to ensure that the data used by the machine learning system is of high quality and represents real-world situations the system will encounter. This may involve monitoring the data for any issues or anomalies, and taking steps to address them as needed.
Measuring, tracking, and making sense of metrics for complex systems is a non-trivial task, and engineers rely on a set of tools to help them do so. There are many different tools and techniques that can be used to monitor a machine learning system in production [123]. Some common ones include:

- Logging: Logging can be used to track the performance and behavior of the machine learning system over time, and to identify any issues or problems that may arise.

- Alerting: Alerting systems can be set up to notify users or system administrators when certain conditions are met, such as when the model's performance falls below a certain threshold or when certain system resources are running low.

- Dashboards: Dashboards can be used to display real-time or near-real-time data about the performance and behavior of the machine learning system, and can be used to monitor the system in real-time.

Updating and Data Distribution Shift

There are a few reasons why data distribution shift can occur:

- Changes in the underlying process: The process that generates the data may change over time, leading to a shift in the data distribution. For example, if a model was trained on data from a manufacturing process, and the process was modified, the model may no longer be able to accurately predict the output of the new process.
- Changes in the environment: The environment in which the data is collected may change, leading to a shift in the data distribution. For example, suppose a model was trained on data from a specific location and the climate or other environmental conditions changed. In that case, the model may no longer be able to predict outcomes accurately.
- Changes in the data collection process: The data collection process itself may change, leading to a shift in the data distribution. For example, if a model was trained on data collected using a certain set of equipment and the equipment was replaced with newer models, the data collected with the new equipment may have different statistical properties.

As said, the updating of the models is necessary in order to avoid the conceptual drift phenomenon that affects machine learning models. This continuous retraining of the prediction models has several disadvantages.
There are several costs associated with retraining a model from scratch. One of the main costs is the time and computational resources required to retrain the model (i.e., its efficiency). Retraining a model from scratch can be time-consuming, especially if the dataset is large or the model is complex. It also requires significant computational resources, such as processing power and memory, to perform the training. These costs can be especially high if the model is retrained multiple times, as each retraining requires an increasing amount of time and resources with respect to the initial training. Consider a scenario in which a model is retrained from scratch multiple times, with each retraining being considered one update. Suppose there are a total of T updates, and each update involves adding new data of size N to the dataset. In that case, the overall complexity of the retraining process will be quadratic in T.
To see why, let's consider the complexity of each update. If the computational cost of training the model is the same for each update, the complexity of the i-th update will be $O(N \cdot i)$. The overall complexity of T updates will then be the sum of the complexities of each update, which is $\sum_{i=1}^{T} O(N \cdot i) = O(\frac{T \cdot (T+1)}{2} \cdot N) = O(N \cdot T^2)$. This means that the difficulty of the process grows rapidly as T increases, which can make the process computationally expensive to run, especially for large values of T. Retraining as new data is introduced means that the data used for training may be retained for a longer period than in traditional machine learning approaches. It is essential to ensure that data is deleted or anonymized when it is no longer needed to protect privacy. This means that keeping confidential or important data has several problems connected to data

sharing and access to sensitive data. In other words, it is important to ensure that access to this data is controlled and that only authorized parties can access it. At the same time, it is essential to have clear data sharing agreements in place and to ensure that data is used ethically and responsibly. The optimal scenario would be to have a model that can update itself using only new data and at the same time perform correctly on the old data.
When a model is retrained from scratch, it is necessary to discard the knowledge gained from the previous model and start over. This can be wasteful if the previous model can be exploited, leading to better performance on a given task because the model can retain and build upon its previous knowledge. This can be especially useful when the data is highly correlated across updates, as the model can transfer its knowledge from one task to another.

Chapter 3

Continual Learning

3.1 Continual Learning

In recent years, machine learning models have been reported to exhibit or even surpass human-level performance on individual tasks, such as Atari games or object recognition [72]. The current dominant paradigm in ML is to train an ML model on a given dataset to generate a model which is then applied in the real-world. Without the ability to accumulate and use past knowledge, an ML algorithm typically needs a large number of training examples in order to learn effectively. In contrast, humans never learn in isolation or from scratch. We always retain knowledge learned in the past and use it to help future learning and problem solving [61].
As a practical example, let's consider an application to categorize the different 100 products of a company automatically. Such a simple model will be useful to speed up asset management operations. Therefore, given a set of images, the model is trained to classify the products correctly, and the resulting performance is very satisfying. However, ten new products are added at some point, and such an event is known to happen each month.
The naive way to handle such a problem would be to fine-tune using the new data and add in the output of the new classes. Fine-tuning is a common method to adapt a deep neural network (DNN) model trained on a source domain to perform well on a target domain. In other words, it leverages the previously learned knowledge of the source domain to adapt to the target domain. This process can effectively adapt the model to the target domain, and it has the advantage of needing less data from the target domain.

However, when a DNN is naively fine-tuned on a new dataset, it loses most of the representations of old classes. In other words, a NN forgets learned patterns upon being presented with new patterns for a certain period. This effect is also known as Catastrophic Forgetting (CF). A possible explanation of this effect is that when a NN trained on task A is trained on a new task B, the weights in the network that are important for task A are changed to meet the

objectives of task B. In other words, the new task will likely override the weights learned in the past and thus degrade the model's performance for the past tasks. From another point of view, CF is strictly connected to Plasticy-Stability Dilemma. Plasticity is associated with the ability to adapt to new information, while stability refers to preserving previously learned information. The NN must remain plastic when presented with significant or useful information but remain stable when presented with irrelevant information. So, in other words, if the network is too plastic, older memories will quickly be overwritten; however, if the network is too stable, it cannot learn new data. The plasticity-stability dilemma refers to the balance between a neural network's ability to adapt to new tasks and the stability of its existing knowledge [3]. As the model is trained on new tasks and data, the trade-off between preserving the knowledge of past tasks (stability) and adapting to new ones (plasticity) becomes evident.
A possible solution would be to retrain the model from scratch using all data seen so far. However, this could be unfeasible for many reasons. The first reason is simply the computing power, retraining from scratch each time new data arrives can scale quickly in terms of computing power and associated costs. In other words, updating the model using only the new data would be much more efficient. Other reasons could be the security and privacy of data.
A research field known to handle the problem of catastrophic forgetting in a DL model is Continual Learning (CL), which enables a DL model to learn and adapt to new tasks without forgetting previously learned patterns. CL methods aim to balance the two conflicting objectives of plasticity and stability, enabling the model to continually learn and adapt without forgetting previously learned patterns.

3.1.1 Formal Definition

In order to provide a formal definition for the CL setting, we will follow a similar approach to that used in previous works such as [178]. This will allow us to define the CL framework in a clear and consistent manner, making it easier to understand and define metrics for CL proposed in the literature. It should be noted that many works propose the same concepts but sometimes with a slight change in the formulas. We are going to take a formal definition similar to the one used in works such as [178]. We focus on a stream of tasks $\mathcal{T}$ of length T . Each task $t \in \mathcal{T}$ has associated a dataset D_t. Therefore, each sample in D_t can be represented by the triplet (x_i, t, y_i), which is formed by a feature vector $x_i \in \mathcal{X}_{t_i}$, a task descriptor $t \in \mathcal{T}$, and a target vector $y_i \in \mathcal{Y}_{t_i}$. For simplicity, it is assumed that every triplet (x_i, t, y_i) satisfies $(x_i, y_i) \overset{iid}{\sim} P_t(X, Y)$. Therefore, the goal is to learn a predictor f, which can be queried at any time to predict the target vector y. Such test pair can belong to a task that we have observed in the past, the current task, or a task that we will experience in the future. Moreover, observe that, based on the kind of scenario, the information of the task label could be hidden from the model during the training, testing, or both, as explained in Section 3.2.

3.2 Continual Learning Scenarios

3.2.1 Types of Data Distribution Changes

Classic machine learning algorithms often assume that the data are drawn, i.i.d. from a stationary probability distribution. However, typically in the real world, such an assumption is only valid for a limited period. When examining the data distribution, one issue is the observed changes in the data statistics as time progresses.
Data distribution shift refers to the phenomenon in supervised learning where the data a model works with changes over time. This causes this model's predictions to become less accurate as time passes [123]. They can happen suddenly for some specific event, like in the case of a recommendation system where a new item receives much attention for some news or advertising campaign. It can also happen gradually because of gradual changes in social norms, languages, trends, and industries.
There are multiple types of drifts, and it is unlikely that any specific approach can work well for all of them [158]. Therefore, it is possible to characterize CL algorithms by their ability to learn under specific distribution drifts associated with different CL settings.
Changes in the data distribution over time are commonly referred to as concept drift. It is fundamental to distinguish the different types of drifts since they will not affect the learning process similarly. We distinguish the following drifts, a visual representation is also shown in Fig. 3.1:

1. **Covariate shift or Domain drift**: Drift in $P(x)$ without affecting $P(y|x)$ nor $P(y)$. Data points sampled after the drift are new, i.e. new data of the same label space. It is one of the most widely studied forms of data distribution shift. In statistics, a covariate is an independent variable that can influence the outcome of a given statistical trial, but which is not of direct interest. In supervised ML, the label is the variable of direct interest, and the input features are covariate variables. In production, covariate shift usually happens because of major changes in the environment or in the way your application is used.

2. **Virtual Concept Drift or Label shift**, also known as prior shift, prior probability shift or target shift, is when $P(Y)$ changes but $P(X|Y)$ remains the same. You can think of this as the case when the output distribution changes but for a given output, the input distribution stays the same. Data points sampled after the drift have new labels. In supervised learning, it involves observing new data from new classes y. Specifically, the joint distribution $P_t(y, x)$ shifts through $Pt(y) \neq P_{t+1}(y)$ while $P_t(x|y) = P_{t+1}(x|y)$.

3. **Real concept drift**: Concept drift is when the input distribution remains the same, but the conditional distribution of the output given an input changes i.e., drift in $P(y|x)$ with $P(x)$ fixed. Real concept drifts

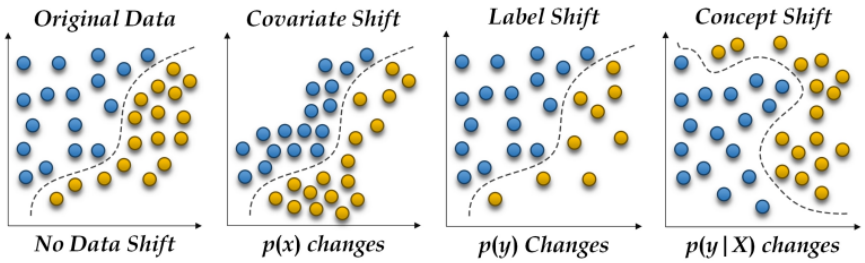


Figure 3.1: Different type drifts: Covariate Shift, Label Shift, Real Concept Drift. Figure taken from [227]

happen when the relation between the input and the target changes. It is problematic since it leads to conflicts in the classification, for example, when a new but visually similar class appears in the data: this will, in any event, have an impact on classification performance until the model can be re-adapted accordingly. Specifically, $P_t(y|x) \neq P_{t+1}(y|x)$ while $P_t(x) = P_{t+1}(x)$, with $P_t(y|x)$ being the probability $P(Y = y|X = x)$ at time t.

Another possible way to consider the shift is the temporal change. For example, concept drift can be gradual or abrupt. A list of possible temporal changes is shown in Figure 3.2 and can be categorized in the following 5 terms [98].

1. **Sudden drift**: This refers to a sudden and significant change in the data distribution used to train a machine learning model. In other words, it assumes that the underlying distribution change from one to another in a punctual way, i.e., the moment of change is precise. This type of drift can significantly impact the model's performance, as it may be less able to predict outcomes accurately based on the new data distribution. An example of such drift could be the replacement of a sensor with another sensor with a different calibration in a chemical plant.

2. **Gradual drift**: This refers to a slow and incremental change in the data distribution over time. This type of drift may be harder to detect as the changes are smaller and occur over a longer period. However, if left unaddressed, gradual drift can also significantly impact the performance of a machine learning model. For example, relevant news topics change from dwelling to holiday homes, while the user does not switch abruptly but instead keeps going back to the previous interest for some time [98].

3. **Incremental drift**: This refers to a series of small changes in the data distribution that occur over time. In other words, it consists of many intermediate concepts in between. An example could be a sensor slowly

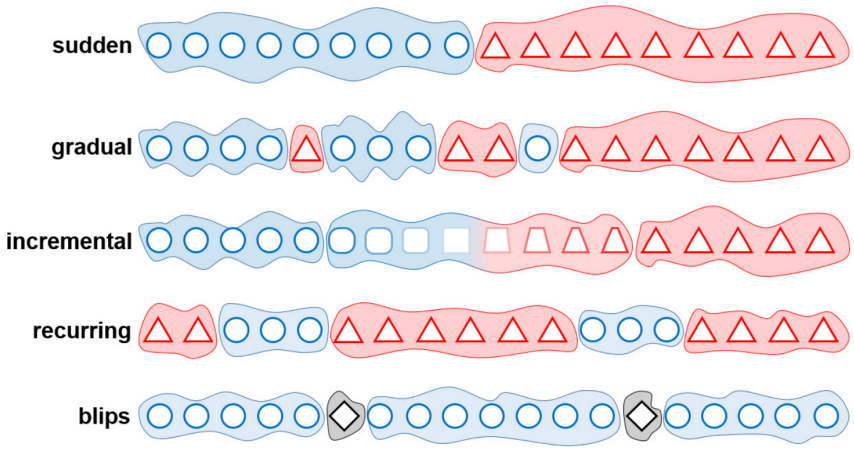


Figure 3.2: Different type of temporal changes: sudden, gradual, incremental, recurring, blips [98]

wearing off and becoming less accurate. This can be due to various factors, such as physical wear and tear, exposure to harsh environments or materials, or the passage of time.

4. **Recurring drift**: This refers to a periodic change in the data distribution that occurs at regular intervals. This type of drift can be managed by retraining the model at regular intervals to account for the changes in the data. This type of drift can be seen as different concepts are seen multiple times among tasks. For example, images of cats could be seen in the first task, never appear again for multiple tasks, and reappear only when visiting the fifth task.

5. **Blips**: This refers to a temporary change in the data distribution that quickly returns to its previous state. This type of drift may not significantly impact the model's performance, as the change is short-lived.

The one most seen in CL is the sudden drift. This is because the tasks are divided very precisely, without ambiguity. This assumption might be seen as a weakness of CL approaches since it makes them dependent on a specific type of temporal change. However, such an assumption helps to reduce the search space and ease the learning process.

3.2.2 Continual Learning scenarios

A well-adopted framework for the context of CL is proposed in [304], where they split the possible CL scenarios into three categories: Task Incremental Learning (TIL), Domain Incremental Learning (DIL), and Class Incremental Learning (CIL). In many continual settings, a period within which a context

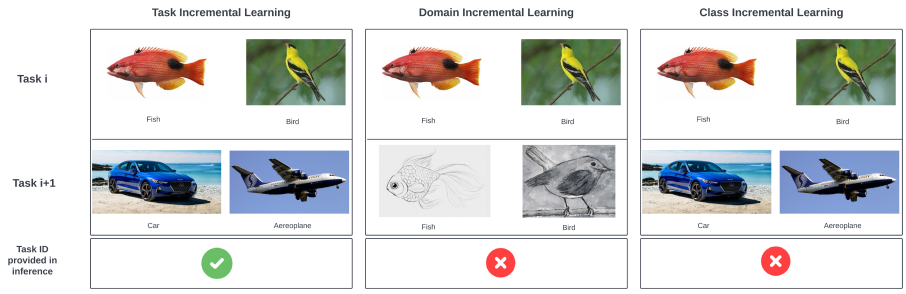


Figure 3.3: Different types of CL scenarios.

remains fixed (corresponding to a stationary data distribution) is called a task and usually corresponds to a learning experience determined by some specific learning purpose. Switching from one task to another is usually associated with a change in the learning process [158].
In classical machine learning, models depend on the type of supervision available, Supervised Learning relies on label supervision and reinforcement learning on reward supervision. In Continual Learning, algorithms do not only rely on those types of supervision but also on task labels. So in CL, the task label is a piece of additional information given to the model. However, based on the scenario, such a label could be provided in some phases and not in others. The task index might be available (1) for training and testing, (2) for training, or (3) not provided at all. Case (1) is associated with Task Incremental Learning, usually to have also the task label during inference allows dedicating a subset of weights of the model to that specific task. For example, a dedicated and separated head for each task will help to decrease the inference. However, how much such a scenario represents a real-world application is questionable. Indeed, knowing the specific task to be solved is not always possible. Case (2) is represented by Domain Incremental Learning (DIL) and Class Incremental Learning. The last case is often called task-agnostic learning.

3.2.3 Task Incremental Learning (TIL)

This refers to a CL scenario where the model is required to learn and adapt to new tasks over time, and a task label is provided during training and inference. This is the easiest continual learning scenario since task identity is always provided. In this scenario, it is possible to train models with task-specific components. A typical network architecture in this scenario has a multi-head output layer, meaning each task has its output units. However, the rest of the network is (potentially) shared between tasks [304]. Knowing the task label during inference allows to allocate some weights as task-specific. Many of the CL approaches that belong to the family architecture-based can perform mainly in this scenario.

3.2.4 Domain Incremental Learning (DIL)

Domain Incremental Learning refers to a CL scenario where the model is required to learn and adapt to new data from the same domain over time. For example, a DIL model might be trained to classify images of cats and then be required to continuously learn and adapt to classify as new images of breeds arrive. In this case, the task identity is not available at test time. Examples of this scenario are cases where the structure of the tasks is always the same, but the input distribution is changing.

3.2.5 Class Incremental Learning (CIL)

This refers to a CL scenario where the model is required to learn and adapt to new data classes over time. For example, a CIL model might be trained to classify images of dogs and cats and then be required to continuously learn and adapt to new classes of animals, such as lions and zebras. In such a scenario, the models should be able to solve all the tasks they have encountered thus far and determine the specific task being presented to them.

3.2.6 Conclusions

The scenarios described assume that during training there are clear and well-defined boundaries between the tasks to be learned. Suppose there are no such boundaries between tasks, for example, because transitions between tasks are gradual or continuous. In that case, the scenarios we describe here no longer apply, and the continual learning problem becomes less structured and potentially a lot harder [304].

3.3 Related Paradigms

It is important to clearly explain related paradigms to Continual Learning and how they differ from CL to avoid misunderstanding and ambiguity. Doing so will make it easier to understand the specific concept of CL and how it differs from other related ideas. Furthermore, by providing this context, readers will better understand CL's unique characteristics and how it fits within the broader field of machine learning. Some possible sources that describe related paradigms are [175, 72, 208].

3.3.1 Transfer Learning (TL)

Transfer learning involves taking knowledge learned by a model in one context and applying it to improve its performance in a different context. This is based on the premise that certain features or patterns learned by a model in one domain or task may be useful in a different domain or task [326]. Though in CL, the main focus is on avoiding Catastrophic Forgetting, one of CL's characteristics is the leverage of previous knowledge to learn quickly and better the

new task. However, TL is not concerned with continuous learning or knowledge accumulation. TL typically involves two domains and the transfer of information from a source to a target. Though CL adapts to the new task, there is no constraint to remembering the original task. In other words, it does not retain the transferred knowledge. Moreover, TL uses only two tasks instead of a sequence of tasks that arrive over time, assuming that the source domain and target domain are very similar.

3.3.2 Multi-Task Learning (MTL)

Multi-task learning (MTL) trains a machine learning model to perform multiple tasks simultaneously. It aims for better generalization and reduced overfitting using shared knowledge extracted from the related tasks. By using data from multiple tasks during learning, forgetting does not occur because the network weights can be jointly optimized for performance on all tasks. Joint learning of all tasks simultaneously represents an MTL baseline. There is no adaptation after the model has been deployed, as opposed to continual learning [72]. So while MTL learns from a set of tasks, CL learns from a sequence of tasks that arrive over time. Because Catastrophic Forgetting makes CL challenging, literature is often considered a reasonable upper bound for CL approaches. In conclusion, MTL and CL differ in the training process and data availability.

3.3.3 Online Learning (OL)

In online learning (OL), the model is trained on one sample at a time and the model's parameters are updated after each sample. This allows the model to continuously learn and adapt to changes in the data distribution. Its goal is the same as classic learning, i.e., to optimize the performance of the given learning task. It is normally used when it is computationally infeasible to train over the entire dataset for hardware constraints or data availability. In traditional offline learning, the entire training data must be made available before learning the task. On the contrary, online learning studies algorithms that learn to optimize predictive models over a stream of data instances sequentially. Although online learning deals with the streaming of data, its objective is very different from continual learning. Online learning still performs the same learning over time, but its objective is rather to learn more efficiently when a new piece of data becomes available. In particular, online learning assumes an i.i.d data sampling procedure and considers a single task domain, which sets it apart from continual learning. Instead, CL considers a sequence of tasks, with the additional constraint of remembering the previously learned knowledge.

3.3.4 Open World Learning

An effective open world recognition system must efficiently perform four tasks: detect unknowns, choose which points to label for addition to the model, label the points, and update the model [208]. Open World Learning deals with the

problem of detecting new classes at test time, hence avoiding wrong assignments to known classes. When those new classes are then integrated into the model, it meets the problem of incremental learning. As such, open world learning can be seen as a subtask of continual learning [72].

3.3.5 Reinforcement Learning (RL)

Reinforcement learning (RL) is a type of machine learning in which an agent learns by interacting with its environment and receiving feedback in the form of rewards. The goal of reinforcement learning is to learn a policy that maximizes the agent's cumulative reward over time. In RL, the agent learns by trial and error, gradually improving its performance as it receives more and more feedback. The agent learns through exploration and exploitation, trying out different actions to see what works best and then relying on the actions that have proven most effective. RL aims to learn an optimal policy that maps states to actions that maximize the long-run sum of rewards. However, in most cases, learning is limited to one task and one environment without the explicit accumulation of knowledge to help future learning tasks. Moreover, environments are often stationary, losing the need for more specific and explicit incremental learning and adaptation algorithmic capabilities [175].

3.3.6 Meta Learning

Meta Learning is most commonly understood as learning to learn. It is a learning process that uses meta-data about past experiences to improve its capacity to learn from new experiences. During base learning, an inner learning algorithm solves a task defined by a dataset and objective. During meta-learning, an outer algorithm updates the inner learning algorithm such that the model improves an outer objective [208]. It aims to learn a new task with only a few training examples using a model trained on many other very similar tasks. It is commonly used for one-shot or few-shot learning. The goal is to design models for learning new skills (tasks) or quickly modifying to new environments with minimal training examples. While these ideas seem quite close to continual learning, Meta Learning still follows the same offline training assumption, with data randomly drawn from a task training distribution and test data being tasks with few examples [72]. Hence, it is not capable, alone, of preventing forgetting the previous tasks.

3.4 Evaluation Metrics

Continual learning is an important aspect of AI where the model needs to adapt and improve their performance over time. The framework involves training a model on a series of tasks over time rather than training it on a single and fixed task. Therefore, some specific metrics need to be designed to measure parameters usually not considered in the classic DL setting. These metrics will be

used to evaluate a model's performance in CL. By measuring the performance of a model using various metrics, researchers can better understand the model's capabilities and limitations and identify areas for improvement.
However, there is not a clear consensus on which metrics should be used to evaluate a CL model. In some cases, different formulas could be associated with the same metric, though they indicate the same concept. This can make it difficult to compare results across different studies. As a result, researchers often have to decide which metrics to use based on their judgment or subjective preferences. Therefore, the following text will describe a set of metrics that should be considered useful for evaluating a CL model from various perspectives. These metrics will provide a foundation for understanding the model's performance and identifying areas for improvement and can help guide the development of more effective CL systems. Some works that define CL metrics used in literature are [178, 80, 138].
Many of the following metrics will be calculated based on metrics collected during the entire stream of tasks in training. In other words, many metrics used to evaluate a CL algorithm are composed of a combination of metrics used in the classic DL setting. Such metrics are used to evaluate the performance of a model on a given task. Many different types of metrics can be used, depending on the specific characteristics of the task and the model. Some common examples of metrics include accuracy (for classification tasks), mean squared error (for regression tasks), and F1 score (for tasks like Anomaly Detection).
Let's assume that we consider the classification problem in CL and select the accuracy as a metric. In CL setting, at the end of a task i we evaluate the model's performance on all tasks. In such a way, we obtain a matrix $R \in R^{TxT}$ where the value R_{ij} refers to the test classification accuracy obtained by the model after being trained on task i and evaluated on the dataset associated to task j. Where T indicates the total number of tasks present in the stream. Based on the type of metric considered, the following formulas could be slightly changed, for example, if the objective for the metric is to minimize instead of maximize, like for accuracy.

3.4.1 Goals of CL

Continual learning allows ML models to update and expand their knowledge over time, with minimal computation and memory overhead [72]. Thus, an effective CL solution is expected to have low forgetting, require low memory consumption and be computationally efficient. By such a definition, it is clear that though forgetting is an important parameter to evaluate, the performance of a CL algorithm is not the only one. Indeed, retraining from scratch is a solution to CF, but at the cost of high memory and computing power. Therefore, when considering a CL algorithm, multiple variables should be considered. This means that other variables, such as the memory used to retain samples of old tasks, the memory of the model used and the computing power, should be taken into account when considering a CL algorithm. In particular, a CL algorithm could be chosen wrt another one based on what a project needs. For example,

if we have a system with a lot of memory and low computing power, we will choose a CL algorithm with such characteristics. Different papers like [208, 80] describe which measures should be considered for a CL algorithm. Below, we will formally present some measures to consider when evaluating a CL approach.

Average Accuracy (ACC)

Let's assume that a CL approach will be evaluated on a total of T tasks. Given the accuracy matrix $R \in \mathbb{R}^{T \times T}$. ACC considers the average accuracy by considering the values on the last row of R. In other words, it is just evaluated the final performance on all tasks after the training on the last task of the sequence:

$$ACC = \frac{1}{T} \sum_{i=1}^{T} R_{T,i} \tag{3.1}$$

Forgetting (F)

We are going to use the definition of forgetting as defined by [54]. We define forgetting for a particular task as the difference between the maximum knowledge gained about the task throughout the learning process in the past and the knowledge the model currently has about it. This estimates how much the model forgot about the task given its current state. Following this, for a classification problem, we quantify forgetting for the j-th task after the model has been incrementally trained up to task $k > j$ as:

$$f_j^k = \max_{l \in \{1, \cdots, k-1\}} R_{l,j} - R_{k,j}, \quad \forall j < k \tag{3.2}$$

Note, $f_j^k \in [-1, 1]$ is defined for $j < k$ as we are interested in quantifying forgetting for previous tasks.
Moreover, by normalizing against the number of tasks seen previously, the average forgetting at k-th task is written as:

$$F = \frac{1}{k-1} \sum_{j=1}^{k-1} f_j^k \tag{3.3}$$

Lower F value implies less forgetting on previous tasks.

Backward Transfer (BWT)

It is the influence that learning a task t has on the performance of a previous task. Positive backward transfer exists when learning about some task t increases the performance on some previous task k. Instead, negative backward transfer exists when learning about some task t decreases the performance on some preceding task k. Large negative backward transfer is also known as (catastrophic) forgetting. In other words, Backward Transfer (BWT) measures the influence of learning a task on previous tasks' performance [178]. In a multi-task

or data stream setting, it is important for an agent to be able to continually improve and maintain its performance over time rather than degrading. Following the equation proposed in [178], BWT can be mathematically defined as:

$$\text{BWT} = \frac{1}{T-1} \sum_{i=1}^{T-1} R_{T,i} - R_{i,i} \tag{3.4}$$

A modified version of such metric is defined in [80] that split the metric in two separated effects called REM and BWT^{+}. The backward transfer metric is an important measure of the ability of an agent to learn new tasks without forgetting what it has learned previously, and is often used to compare the effectiveness of different continual learning algorithms.

Forward transfer (FWT)

It is the influence that learning a task t has on the performance of a future task. In other words, it measures the extent to which knowledge gained from learning one task helps an agent improve its performance on a different but related task. Let $\bar{b}$ be the vector of test accuracies for each task at random initialization. We can define FWT as:

$$\text{FWT} = \frac{1}{T-1} \sum_{i=2}^{T} R_{i-1,i} - \bar{b}_i \tag{3.5}$$

A high forward transfer score indicates that the agent can use knowledge from one task to improve performance on a different task. In contrast, a low score indicates that the agent cannot transfer its knowledge effectively. If two models have similar ACC, the one with larger BWT and FWT is preferable.

Model size efficiency

Let's indicate with θ the parameters of the model and with θ_i the parameters at task i. The memory size of the model is quantified in terms of parameters θ at each task $i, \text{Mem}\,(\theta_i)$, should not grow too rapidly with respect to the size of the model that learned the first task, $\text{Mem}\,(\theta_1)$. Model size (MS) is thus:

$$MS = \min \left(1, \frac{\sum_{i=1}^{N} \frac{\text{Mem}(\theta_1)}{Mem(\theta_i)}}{N} \right) \tag{3.6}$$

The model size efficiency metric measures the trade-off between the size of a model (the number of its parameters) and its performance on a series of tasks. It is an important measure of the efficiency of continual learning algorithms, as it indicates how well the agent can learn new tasks using a fixed amount of model capacity. The ideal goal would be to obtain a MS equal to 1 (or close to 1). In this way, the model's size at the end of the training would be equal (close) to the size of the initial model.

Samples storage size efficiency

Many CL approaches save training samples as a replay strategy for doesn't forget. The memory occupation in bits by the samples storage memory $M, \text{Mem}(M)$, should be bounded by the memory occupation of the total number of examples encountered at the end of the last task, i.e., the cumulative sum of all datasets encountered called lifetime dataset D. So Samples Storage Size (SSS) efficiency can be defined as:

$$SSS = 1 - \min\left(1, \frac{\sum_{i=1}^{N} \frac{\text{Mem}(M_i)}{\text{Mem}(D)}}{N}\right) \tag{3.7}$$

The samples storage size efficiency metric in continual learning measures the trade-off between the amount of data an agent stores (i.e., its memory size) and its performance on a series of tasks.

Computational efficiency

Since the computational efficiency (CE) is bounded by the number of multiplication and addition operations for the training set D_i associated to the task i, we can define the average CE across tasks as:

$$CE = \min\left(1, \frac{\sum_{i=1}^{N} \frac{Ops{\uparrow\downarrow}(Tr_i)\cdot\varepsilon}{Ops(Tr_i)}}{N}\right) \tag{3.8}$$

Where $\text{Ops}\,(\text{D}_i)$ is the number of (mul-adds) operations needed to learn D_i, and $Ops\ {\uparrow\downarrow}\ (\text{D}_i)$ is the number of operations required to do one forward and one backward (backprop) pass on D_i.

Aggregate metric

In [80] is proposed to aggregate several metrics that represent different concepts into a single final metric that should summarize the performance of a CL algorithm.

In order to assess a CL algorithm, each criterion $c_i \in \mathcal{C}$ (where $c_i \in [0, 1]$) is assigned a weight $w_i \in [0, 1]$ where $\sum_i^{\mathcal{C}} w_i = 1$. Each c_i should be the average of r runs. Therefore, the final CL_{score} to maximize is computed as:

$$CL_{\text{score}} = \sum_{i=1}^{\#\mathcal{C}} w_i c_i \tag{3.9}$$

where each criterion c_i that needs to be minimized is transformed to $c_i = 1 - c_i$ to preserve increasing monotonicity of the metric (for overall maximization of all criteria in $\mathcal{C}$). On the point of view of vectors we can imagine to have a vector W such as $W = [w_A, w_{MS}, w_{SSS}, w_{CE}, w_{BWT}, w_{Rem}, w_{FWT}]$ and a vector C that for each criterion has the associated value. The aggregated value

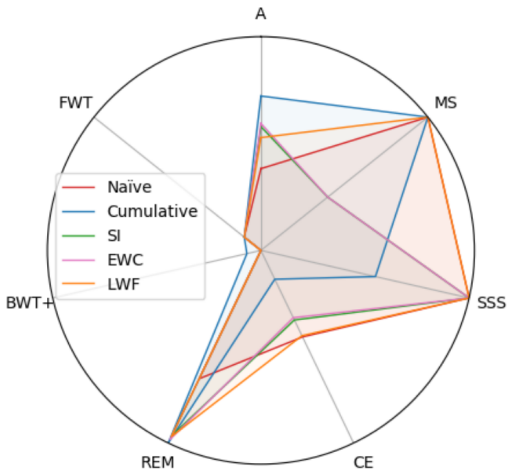


Figure 3.4: Spider chart representing some CL algorithms and the value associated to each criterion. Image taken from [80]

can be calculated scalar product between W and C i.e. $CL_{\text{score}} = 1 - W \cdot C$. For simplicity, a possible configuration of criteria weights could be to evaluate all metrics equally i.e. each $\frac{1}{7}$ since there are 7 metrics. Note that though such vector consider equally all metrics is not necessary the only choice of values. This is because based on a specific CL scenario or problem it could be more important have some optimal parameters and ignore others. For example, in the case of a real world application on the Edge where the computation power is limited but not the memory, then it is going to consider more important a method with low CE and it will weight less high values for SSS and MS. Such results can also be represented as in Fig. 3.4 using a spider chart.

Conclusion

In general, the main metrics to consider when evaluating the performance of a CL algorithm are forgetting, backward transfer (BWT), forward transfer (FWT), and accuracy (ACC) i.e., the performance of a model in the CL setting (included the ability to avoid catastrophic forgetting and to apply transfer learning to new tasks). However, these metrics are not the only ones to consider when evaluating and choosing a CL algorithm. Other metrics, such as model size efficiency and sample storage size efficiency, may also be important when choosing between different CL algorithms. Even though two CL algorithms may perform similarly on the main metrics, one algorithm could still be preferred over another based on its performance on these additional metrics.

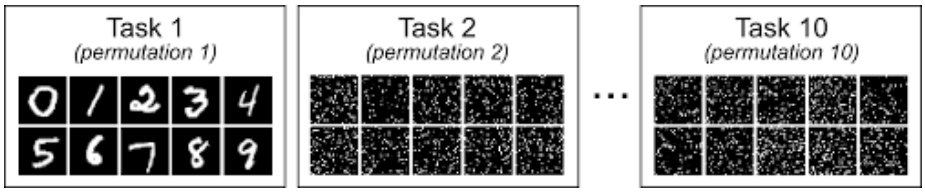


Figure 3.5: Example of PermutedMNIST Dataset. Image taken from [304]

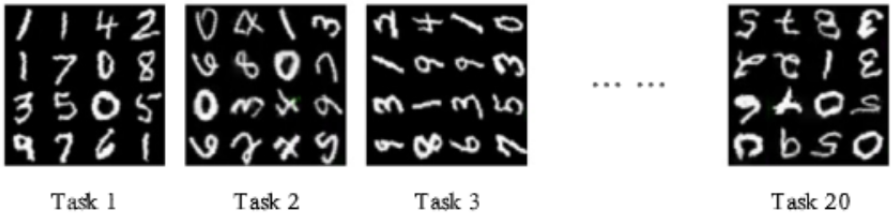


Figure 3.6: Example of RotatedMNIST Dataset. Image taken from [127]

3.5 Datasets

A variety of datasets are commonly used to evaluate and benchmark continual learning algorithms. It should also be noted that many datasets are artificially transformed in datasets for the CL setting instead to have inherent properties like a temporal axis. For example, MNIST, CIFAR-100, SVHN and ImageNet are all well-known CV datasets that can be transformed to work in the CL under different scenarios. In other cases, the dataset inherently has a temporal order associated with the samples.

3.5.1 Classic Datasets

Here will give a brief review of some of well known datasets used in classic DL setting.

- **MNIST**: MNIST is a widely used dataset for training and evaluating machine learning models in the field of computer vision. It consists of a training set of 60,000 28x28 grayscale images and a test set of 10,000 images, each labeled with the digit it represents.

- **CIFAR-10 and CIFAR-100**: CIFAR-10 consists of 60,000 32x32 color training images and 10,000 test images, labeled into 10 classes. The 10 classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. CIFAR-100 is similar to CIFAR-10, but it has 100 classes instead of 10.

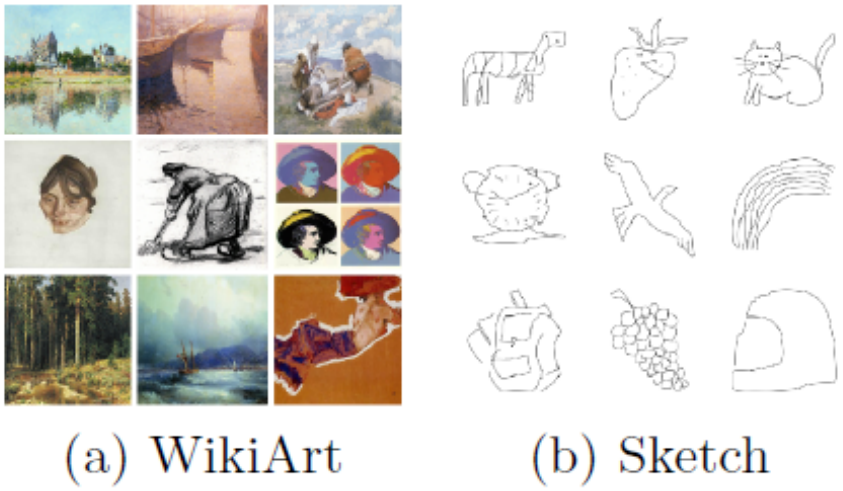


Figure 3.7: Example of images belonging to domains different from photorealistic images like ImageNet. Image taken from [300]

- **CUB-200**: The CUB-200 dataset (Caltech-UCSD Birds-200-2011) is a dataset of images of birds. It consists of 11,788 images of 200 different bird species, with a roughly equal number of images per class. The images were collected from the web and annotated with the presence or absence of certain attributes, such as head markings, wing markings, etc. The CUB-200-2011 dataset is a well-known benchmark in the field of computer vision and has been widely used in a number of research studies.

- **SVHN**: The Street View House Numbers (SVHN) dataset is a large dataset of images of house numbers collected from Google Street View images. The dataset consists of over 600,000 images, which have been labeled with the house numbers that appear in the images.

A lot of other datasets are considered to be used in Continual Learning such as: Stanford cars, Oxford owers, Caltech-256, GTSR, MIT scenes, Flower, FGVC-Aircraft, Letters and Places365 [61]. Datasets of images belonging to domains different from photorealistic images like ImageNet are WikiArt Paintings [257] and Human Sketches [88]. Examples of these datasets are shown in Fig. 3.7.
Other interesting datasets could be MSCOCO and NUSWIDE. The MSCOCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset [168]. The dataset consists of 328000 images, each of which has been annotated with a variety of labels, including object categories, object attributes, and image captions. NUS-WIDE (National University of Singapore - Web Image Database)

is a large-scale dataset of images and associated labels that was developed by researchers at the National University of Singapore [63]. The dataset contains over 269,000 images that have been annotated with one or more of 81 different labels, including object categories, scenes, and image attributes.
In general, most of works of CL are performed in the CV domain. However, there are other domains like audio for example. In this case the AudioSet can be used for CL as in the case of [138]. It is a large-scale collection of human-labeled 10-sec sound clips sampled from YouTube videos [102]. In another work [323], they used self-supervised learning to classify new sound classes using UrbanSound8K, DCASE TAU19 and and VGGSound. UrbanSound8K is a dataset of audio samples collected from urban environments. It consists of 8732 audio clips in total, each lasting 4 seconds. The UrbanSound8K dataset is divided into 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer, siren, and street_music. Each audio clip in the dataset is labeled with one of these classes, indicating the type of sound event that is present in the clip. The dataset was created for the 2019 Task on Acoustic Scene Classification and Sound Event Detection. The DCASE TAU19 dataset consists of audio recordings from six different urban environments, each with a distinct acoustic character. It consists of 10-seconds audio segments from 10 acoustic scenes: Airport, Indoor shopping mall, Metro station, Pedestrian street, Public square, Street with medium level of traffic ,Travelling by ,a tram, Travelling by a bus, Travelling by an underground metro, Urban park. Recordings were made with three devices that captured audio simultaneously. Each acoustic scene has 1440 segments (240 minutes of audio) recorded with device A (main device) and 108 segments of parallel audio (18 minutes) each recorded with devices B and C. The dataset contains in total 46 hours of audio. VGG-Sound is a large-scale dataset of audio signals, it consists of more than 210k videos for 310 audio classes. It consists of short clips of audio sounds, extracted from videos uploaded to YouTube. The VGG Sound Dataset is an extensive collection of audio recordings that cover a wide range of challenging acoustic conditions and noise types that are commonly encountered in real-world applications. The audio clips in the dataset were recorded in diverse environments and are accompanied by corresponding video footage, ensuring that the sound source is visually identifiable. Each segment in the dataset is 10 seconds long and includes both audio and video data [55]. About the NLP domain, possible datasets are GLUE and SUPERGLUE benchmarks that track performance on eleven and ten language understanding tasks respectively, using existing NLP datasets. Along the same line, in [196] presented the Natural Language Decathlon (DECANLP) benchmark for evaluating the performance of models across ten NLP tasks. The DECA NLP benchmark consists of ten different NLP tasks, including machine translation, summarization, question answering, and others. The tasks are designed to cover a broad range of languages and language-related problems, and the benchmark includes data for over 100 different languages. Similar to DECANLP, a recently proposed Cross-lingual TRansfer Evaluation of Multilingual Encoders (XTREME) benchmark also uses a diverse set of NLP tasks and task-specific measures to evaluate

the performance of cross-lingual transfer learning [120]. XTREME consists of nine tasks derived from four different categories and uses zero-shot cross-lingual transfer with the English language as the source language for evaluation [29].

3.5.2 CL Datasets

Many of the datasets described above like MNIST, CIFAR-100, SVHN and ImageNet can be transformed to use them in the Continual Learning framework. Since these datasets don't have any temporal order is necessary to propose some transformation. However, the transformation depends onon factors like the type of CL scenario that must be studied, the total number of tasks that want to be considered, and the number of classes for each task.

CIL

In CIL, the model is trained to learn a series of tasks in a specific order, each task involving a new set of classes. In the case of CIL, many datasets can simply be transformed by splitting samples with different classes in different tasks. For example, in the case a dataset with 100 classes can be created a dataset composed of 10 tasks with 10 classes for each task. However, if we want to consider a longer stream we can consider a set of 50 tasks with two classes for each task. Some examples of possible datasets are the following:

- **Split MNIST**: Each new session learned contains a set of new classes. It is tested a model's ability to sequentially learn new classes is tested.
- **Split CIFAR-100**: Split into 20 disjoint subsets, 5 classes for each task, which are randomly sampled without replacement.
- **Split CUB**: 200 categories into 20 disjoint subsets of classes
- **Split mini ImageNet**: 20 disjoint subsets, subset of ImageNet with 100 classes and 600 images per class.
- **iCIFAR-100**: Split into 10 disjoint subsets, 10 classes for each task.

DIL

In the Domain Incremental Scenario (DIL), while the set of possible classes remains fixed, it changes the underlying input distribution.
A well-known dataset and one of the first to be proposed is called **PermutedMNIST** [106] in Fig. 3.5. The advantage of such a dataset is that the number of tasks is potentially very large. Indeed, for each task, it is chosen a rearrangement of the pixel of the images is kept fixed during the task and changes when a new task arrives. This method allows for a very large set of tasks that are also not related to each other. Such a dataset can be useful to measure if a model can maintain the memory of old tasks. Still, since there is no relation among tasks, evaluationing methods that exploit task correlation is not possible.

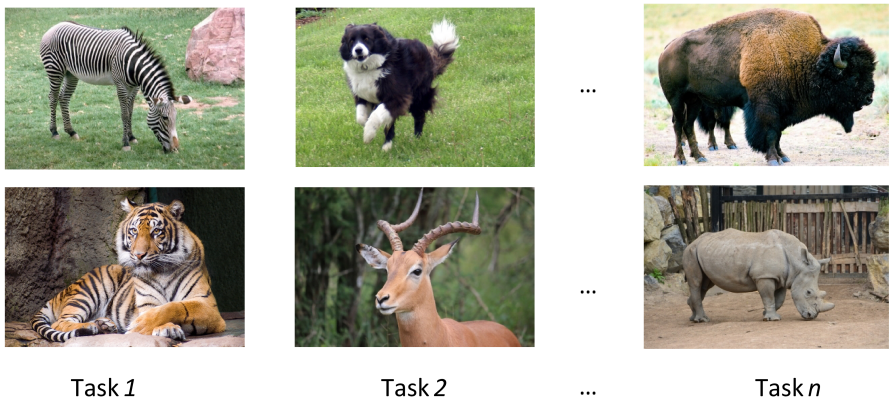


Figure 3.8: Example of Class Incremental Learning setting. Each new task corresponds to a new set of images.

Another very used dataset in CL literature is **RotatedMNIST** [178]. The RotatedMNIST Dataset is a variant of the MNIST Dataset that consists of images that are rotated by an angle chosen uniformly by a random angle between 0 and 360 degrees. So each task corresponds to a rotation degree that remains fixed for all the images of the task and changes among tasks. An example is shown in Fig. 3.6. As for the case of PermutedMNIST, it is possible to obtain a high number of tasks. The main difference is that PermutedMNIST is formed by completely independent tasks, while RotatedMNIST has correlated tasks. This means that while in the first case, it is interesting only to avoid forgetting, in the second case, it would be optimal that the model can also apply some transfer learning and exploit the previous knowledge for the new task.
Another DIL example would be splitting the datasets based on some other condition. For example, a dataset could have multiple classes like orange, car, flower, etc and be split based if the images are real photos, images from sketches, images from cartoons and so on.

Complex Datasets

A simple way to obtain new datasets could be a stream of well-known datasets instead of subsets of such datasets. For example, a stream of tasks for CIL could be created where the first task is the entire dataset MNIST with ten classes. Then the second dataset could be SVHN with ten classes, the third could be CIFAR10, etc. For example, a benchmark of this kind is proposed in [244] called **Visual Domain Decathlon**. The goal of this challenge is to solve ten image classification problems representative of very different visual domains.

The data for each domain is obtained from the following image classification benchmarks: ImageNet, CIFAR-100, Aircraft, Daimler pedestrian classification, Describable textures, German traffic signs, Omniglot, SVHN, UCF101 Dynamic Images, VGG-Flowers.
In another work, they propose considering the following sequence of datasets as tasks: Flower, Scenes, Birds, Cars, Aircraft , Action, Letters, SVHN. A sequence of eight object recognition datasets:

1. **Oxford Flowers**: for fine-grained flower classification (8,189 images in 102 categories)
2. **MIT Scenes**: for indoor scene classification (15,620 images in 67 categories)
3. **CUB-200** (Caltech-UCSD Birds): for fine-grained bird classification (11,788 images in 200 categories)
4. **Stanford Cars**: for fine-grained car classification (16,185 images of 196 categories)
5. **FGVC-Aircraft**: for fined-grained aircraft classification (10,200 images in 70 categories)
6. **VOC actions**: the human action classification subset of the VOC challenge 2012 (3,334 images in 10 categories)
7. **Letters**: the Chars74K datasets for character recognition in natural images (62,992 images in 62 categories)
8. **SVHN**: Google Street View House Number SVHN dataset for digit recognition (99,289 images in 10 categories)

Such an approach allows it to scale quickly in terms of data and the number of tasks.
Moreover, while the proposed benchmarks have a dataset that belongs to only the CV domain, it is also possible to consider a multi-modal dataset. For example, a possible benchmark could be the composition of a stream of tasks where each task belongs to a different domain, like images, audio, text, etc.
A very recent benchmark proposed is the Never Ending VIsual-classification Stream (NEVIS'22), consisting of a stream of over 100 visual classification tasks, sorted chronologically and extracted from papers sampled uniformly from computer vision proceedings spanning the last three decades [34]. Despite being limited to classification, the resulting stream has a rich diversity of tasks such as OCR, texture analysis, crowd counting, scene recognition, etc. The diversity is also reflected in the wide range of dataset sizes, spanning over four orders of magnitude. Overall, NEVIS'22 poses an unprecedented challenge for current sequential learning approaches due to the scale and diversity of tasks, yet with a low entry barrier as it is limited to a single modality and each task is a classical supervised learning problem.

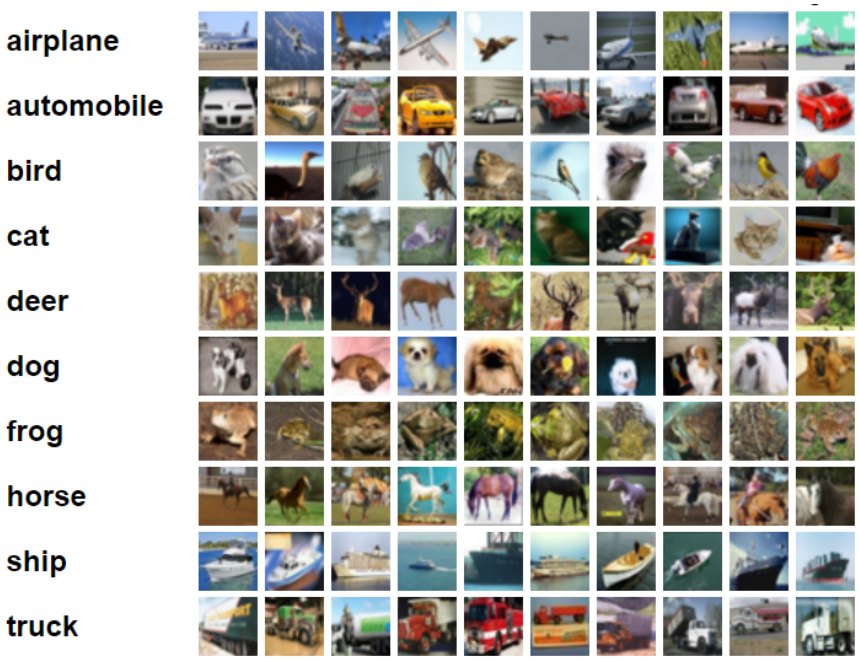


Figure 3.9: Images belonging to the dataset CIFAR-10

Reinforcement Learning

Though most of the CL literature is about Supervised Learning in the CV domain, other scenarios like RL are also considered. Reinforcement learning (RL) is a type of machine learning in which an agent learns by interacting with its environment and receiving feedback in the form of rewards.
In Continual Learning on Reinforcement Learning, different environments were used for evaluation [61]. Atari games [203] are among the most popular ones, which are used in [145, 254]. The Atari games are well-suited for RL because they provide a rich, interactive environment where an RL agent can learn to make decisions and take actions to maximize a reward. Some other environments include Adventure Seeker [170], CartPole-v0 in OpenAI gym [40], and Treasure World [188]. Other possible datasets are VizDOOM built around Doom II, a first-person shooter game [175]. For example, we can consider environmental changes like light, textures, and objects in the 3D VizDOOM Maze. For all environments, changes are not gradual but happen at specific points. Other research platforms are DeepMind Labs and Malmo, which allowed researchers to explore new research directions to scale reinforcement learning. In [331] is proposed Continual World, a benchmark consisting of realistic and meaningfully diverse robotic tasks built on top of Meta-World [353].

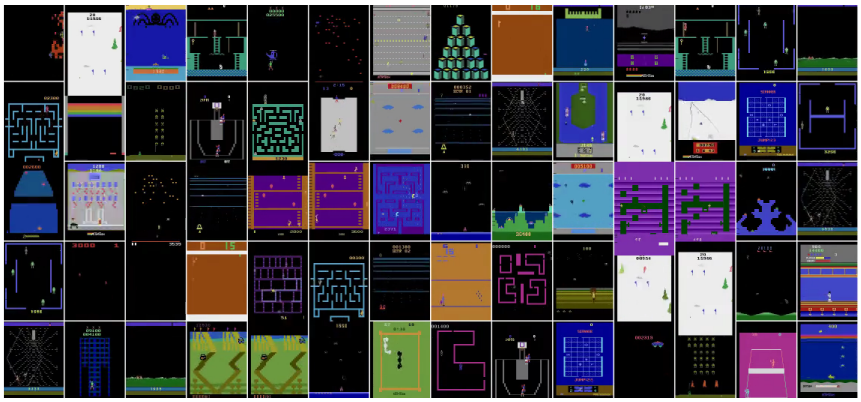


Figure 3.10: Pong, Alien, Centipede, Boxing, Hero, James Bond, Seaquest, Riverraid, asterix, krull, crazy_climber, riverraid, jamesbond, breakout, gopher, kangaroo, kung_fu_master, fishing_derby, enduro, pong, star_gunner, demon_attack, boxing, asterix, road_runner, defender, freeway, krull, space_invaders. Image taken from [67]

Final Considerations

Another additional parameter not described previously is the order of tasks. The order of the tasks is also very important when evaluating an algorithm on a dataset. For example, a specific configuration of how the data are presented can alter the model's performance. A possible solution, though computationally expensive, would be to test the CL algorithm on multiple instances of the CL dataset where the order of tasks is different. New datasets can be useful for continual learning in several ways. First, new datasets can provide a more realistic and diverse range of tasks for the model to learn, which can help better evaluate the model's capabilities and identify areas for improvement. Second, new datasets can help expose weaknesses or limitations in current continual learning algorithms. For example, a dataset that requires the model to learn many tasks may be more challenging for current algorithms and could help identify new approaches or techniques needed to improve performance.

3.6 Continual Learning Strategies

Continual learning involves the ability to learn from a non-stationary data stream and accumulate knowledge over time, while successfully dealing with the catastrophic forgetting problem [95]. A taxonomy can be an important tool for organizing the CL approaches and provides a clear and systematic framework for grouping different concepts. In the case of continual learning, the strategies can be organized into different categories based on the underlying method used to address the challenge of retaining previously learned knowledge while learning new tasks. The literature often summarizes the CL approaches in 3 groups as

depicted in Fig. 3.11, regularization-based [145] [363], architecture-based methods [254][350] and replay-based [53] [250]. It is important to remember that there is no universal solution for CL, and different approaches may be more or less appropriate depending on the specific problem and environment.

3.6.1 Upper Bounds and Lower Bounds

Usually, when evaluating CL approaches is often made a comparison with some upper bounds to understand how much CL approaches are far from the ideal case. When considering upper bounds, there are three possible strategies: Cumulative, Multi-Task, and Single Model. The Cumulative strategy, called also Full Rehearsal, assumes learning a sequence of tasks as in a CL setting. However, it considers no constraints in terms of memory or computing power. This means that during the second task, the Cumulative approach will train the model using all the data from the first and second tasks and a number of epochs equal to what would be done in the classic DL scenario. Another possible upper bound is Multi-Task (MT), the model is trained using all the data from all tasks. The main difference between Cumulative and MT is that MT is trained directly on all the samples, while such a thing happens with Cumulative only at the end of the training when it has already seen all the tasks except the last. Therefore, MT can be considered an upper bound, assuming that training on multiple tasks allows generalization and improves performance on all tasks. Finally, Single Model refers to the scenario where a separate model is trained for each task. This could represent the ideal case if it is assumed that the set of tasks could interfere with each other. In the same way, to understand how far we are from the worst case, we can compare the CL approaches with approaches considered lower bounds. An example of such an approach is Fine-Tuning (FT). It would be the naive approach where the model is updated using only the data of the current task, which causes Catastrophic Forgetting (CF).

3.6.2 Regularization-based approaches

These approaches focus on designing loss terms to retain the representations for old tasks. In practice, a possible example is Elastic Weight Consolidation (EWC). Such a method prohibits drastic updates to important parameters. Based on the same idea, other approaches like SI, MAS, and IMM exist. Another regularization-based approach is Learning without Forgetting (LwF), which uses distillation to retain previously acquired knowledge.

3.6.3 Rehearsal-based approaches

The idea is to store or generate data to augment new training batches that help the network retain its old representations. The previous task's data is either used again as input for rehearsal or to restrict the optimization of the new task's loss to prevent interference from the previous task. Many approaches belong to the category such as Gradient Episodic Memory (GEM), Experience Replay

(ER), Generative Replay (GR), Latent Replay (LR), and Maximally Interfered Retrieval (MIR) [267].

3.6.4 Architecture-based approaches

These approaches involve modifying the model's architecture to avoid Catastrophic Forgetting. Specific architectures, layers, activation functions, and weight-freezing strategies are generally used to mitigate forgetting in this family of approaches. Architecture-based approaches include Progressive Neural Network (PNN), ExperGate, and PathNet. This family dedicates different model parameters for each task to prevent any possible forgetting. However, when no constraints are applied to the model size, there is the risk that it loses scalability. Some approaches expand the size of the network over time as new tasks arrive. Instead, in other approaches, the architecture remains static, with fixed parts allocated to each task. One of the main disadvantages of such a family of approaches is that they generally require the task label during inference, which makes such approaches unfeasible in some real-world applications where such information cannot be acquired. The task label in inference is necessary to activate parameters specific to the task, like masks, heads, or parts of the network. In other words, architecture-based approaches are usually considered for Task Incremental Learning, where the task is known during inference.

3.7 Regularization-based approaches

3.7.1 Elastic Weighted Consolidation (EWC)

It uses the training data to build a Fisher Information Matrix (FIM) that determines the importance of each weight in the network for the classification task it just learned. It employs a regularization scheme that redirects plasticity to the weights that are least important to previously learned sessions. In other words, it mitigates forgetting by forcing parameters that contribute a lot to old tasks to stay fixed. Each parameter's importance to the change in the loss function is used as a quadratic penalty term to compose a surrogate loss.
The intuition is given by assuming that the model needs to optimize for two tasks, A and B. Many configurations of a model of parameters θ will result in the same performance. Over-parameterization makes it likely that there is a solution for task B with optimal parameters θ^*_B, that is close to the previously found solution for task A with optimal parameters θ^*_A. Therefore, it is necessary to maintain memory of the parameters important for task A, and keeps them unchanged and allows the free parameters to adapt to the new task.
So the importance for a weight θ_k of the network after training on task A can be calculated as :

$$\Omega_k = \mathbb{E}_{(x,y)\sim D}\left[\left(\frac{\delta L}{\delta \theta_k}\right)^2\right] \tag{3.10}$$

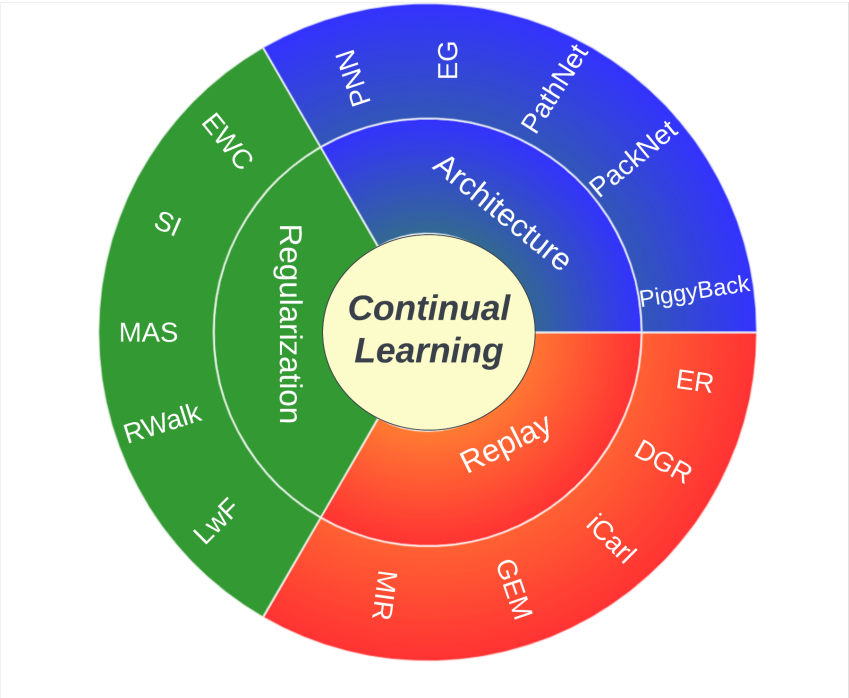


Figure 3.11: Taxonomy of CL approaches

During training on the new task B, the loss can be calculated in the following way:

$$L(\theta) = L_B(\theta) + \sum_k \frac{\lambda}{2} \Omega_k \left(\theta_k - \theta^*_{A,k}\right)^2 \tag{3.11}$$

Where $L_B(\theta)$ measure the loss of the new task and the second term avoid that the new solution shifts too much from task A.

3.7.2 SI

Synaptic Intelligence (SI) replaces Fisher Information Matrix-based Importance and it was introduced as a variant of EWC [360]. The authors argued that the computation of FIM is too expensive. So they proposed to calculate weight importance online during SGD. Therefore, it breaks the EWC paradigm of determining the new task importance weights in a separate phase after training, they maintain an online estimate during training [72]. Assuming a set of previous tasks $t < N+1$. It is possible to determine the new task importance weight

Ω_k^{N+1} in a separate phase after training, maintaining an online estimate ω_k^{N+1} during training to eventually attain:

$$\Omega_k^{N+1} = \sum_{t=1}^{N+1} \frac{\omega_k^{N+1}}{\left(\Delta\theta_k^t\right)^2 + \xi} \tag{3.12}$$

Where $\Delta\theta_k^t = \theta_k^t - \theta_k^{t-1}$ the task-specific parameter distance, and damping parameter ξ avoiding division by zero.
In another work, it is proposed Memory Aware Synapses (MAS). They use averaged L2-Norm of per-parameter gradient as the importance measure [10]. In a different study, it is proposed Riemannian Walk, it combines the SI path integral with an online version of EWC to measure parameter importance [54].

3.7.3 Learning without Forgetting (LwF)

Learning Without Forgetting (LWF) is a regularization approach that tries to control forgetting using Knowledge Distillation (KD) [162].
In short, knowledge distillation involves the transfer of knowledge, usually from a large pre-trained model called teacher to a smaller model called student. When training a smaller model, it tries to imitate the behavior of the larger pre-trained model [116], to improve efficiency and potentially improve performance. A possible application is to be used as a form of model compression, which could be helpful to deploy a large deep neural network model on edge devices with limited memory and computational capacity. This is achieved by using the larger model to generate soft targets for the training data, which the smaller model is then trained to predict. These soft targets contain knowledge about the data and can be useful for the student model to learn from. The smaller model is trained to minimize the difference between its own predictions and the soft targets produced by the larger teacher model.
Lwf records the network's response to old tasks, which is used in the KD Loss Term. It uses them to encourage the network to make similar predictions after being trained for new tasks. The idea is to optimize the new task, with the constraint that the predictions on the new task's examples do not shift too much.
However, the success of this method depends heavily on the new task data and how strong it is related to prior tasks. It has been shown that this strategy is vulnerable to domain shift between tasks [72]. Distribution shifts concerning the previously learned tasks can result in a gradual error build-up to the prior tasks as more dissimilar tasks are added [72]. This error build-up also applies in a class-incremental setup.

3.8 Rehearsal-based approaches

3.8.1 Experience Replay (ER)

In experience replay is stored a small amount of previously seen data and using them to rehearse or review old tasks. This can help the model retain its ability to perform previously learned tasks while learning new ones.
In more detail, during the training of the new task, for each batch of the new task, it also loaded a batch of data from memory based on certain criteria. Then the model performs training on both the new task and the data from memory. This allows the model to adapt to the new data and, at the same time to keep the memory of old tasks. Multiple algorithms were proposed as a strategy for the optimal selection of the subset of samples to keep in memory. The most simple strategy will just select a subset in a random way from the dataset, also known as Random Sampling (RS). Other strategies could be based on the model prediction (e.g. entropy) or the feature embeddings (e.g. kmeans). A review of such methods can be found in [344].
In general, the replay strategy was found to be one of the most effective solutions in CL [343, 223, 42, 140, 200]. The disadvantage of such an approach is that with an increasing number of tasks, the memory may not be large enough to contain all the previous tasks. Additionally, a subset of data has too few risks to accurately represent the original distribution, leading to catastrophic forgetting. Moreover, using data from previous tasks to train requires additional computation. Other concerns are about data privacy and the problem of keeping an additional memory that based on the device could be a huge constraint.

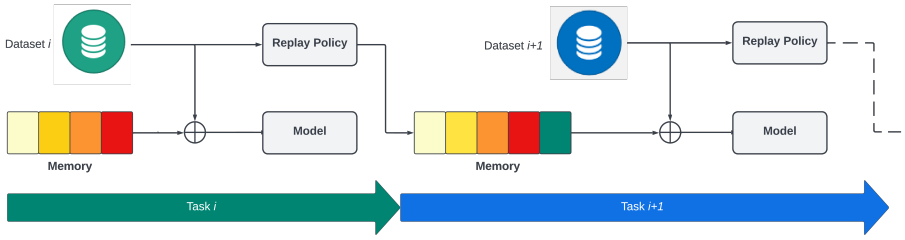


Figure 3.12: Scheme of Experience Replay approach

3.8.2 Generative Replay

Unsupervised learning is a vast subject in machine learning, and it has the advantage wrt Supervised Learning that labels are not necessary to train these models. Generative models belong to the area of Unsupervised Learning. They are a particular type of models designed to reproduce the input data distribution. The goal is to learn to generate data from the original data distribution, similar but not identical to the data of the training set. The generative model is generally a GAN [105] or a Variational Auto-Encoder (VAE), though in general

can be considered model-agnostic [157].
Generative Replay, also known as Pseudorehearsal, allows the network to revisit previous memories during the training of a new task without the need to store previous training examples, which is more memory efficient. It is inspired by the suggestion that hippocampus is better paralleled with a generative model than a Replay Buffer. Generative Replay approaches differ from Rehearsal approaches in that they model the past by training a generative model on the data distribution rather than relying on a few samples. This allows them to generate new examples from past experiences when learning new data.
A classical method implementing a generative replay typically makes use of dual models [157]. One frozen model generates samples from past experiences, and another learns to generate and classify current samples in addition to the regenerated ones. When a task is over, we replace the frozen model with the current one, freeze it, and initialize a new model to learn the next task.
In [269], given a sequence of tasks, a scholar model containing a generator and a solver is learned and retained. Such a scholar model holds the knowledge representing the previous tasks and thus prevents the system from forgetting previous tasks.

3.8.3 Gradient Episodic Memory (GEM)

The memory of samples can also be used for regularization purposes and not just replayed from time to time, along with new data in the learning process. Specifically, GEM aims to reduce forgetting and make positive backward possible [178]. The key idea is to only constrain new task updates to not interfere with previous tasks. This is achieved by projecting the estimated gradient direction on the feasible region outlined by previous task gradients through first-order Taylor series approximation.

3.8.4 incremental Classifier and Representation Learning (iCaRL)

iCaRL can be seen as a mixture of a regularization and rehearsal approaches. For each class, it stores a subset of exemplars per class, where an exemplar set is a subset of all examples of the class, aiming to carry the most representative information of the class i.e. they best approximate class means in the learned feature space. At test time, the class means are calculated the nearest mean classification based on all exemplars. The idea is that the last layers are the most task-dependent. Therefore these layers are the most influenced by the addition of new tasks. They use the first layers of a DNN, which are the most stable, for representation learning and use the DNN as a feature extractor. Therefore, using LwF a DNN can be trained continuously. The base idea is that instead of using a DNN for classification, iCarl uses it for supervised representation learning.

3.8.5 Maximally Interfered Retrieval (MIR)

In [11] is proposed a controlled sampling of memories for replay. They retrieve the samples which will be maximally interfered by the new task data, i.e. whose prediction will be most negatively impacted by the foreseen parameters update. This approach also takes some motivation from neuroscience where replay of previous memories is hypothesized to be present in the mammalian brain [197], but likely not random. In the CL setting and given a limited computational budget, an approach can't replay all buffer samples each time. Therefore, it becomes crucial to select the best candidates to be replayed. MIR proposes a better strategy than random sampling in improving the learning behaviour and reducing the interference.

3.9 Architecture-based approaches

3.9.1 Progressive Neural Network (PNN)

It proposes a progressively growing neural network, which is extended by an additional column when trained on a new task. By utilizing lateral connections to the new column and freezing the previous columns, the system can prevent catastrophic forgetting while reusing previously learned knowledge for the current task [254]. The idea is to keep a pool of pre-trained models as knowledge and use lateral connections between them to adapt to the new task. It is a combination of parameter freezing and network expansion. For each new task encountered, a new neural network (or a new column) is created, and its lateral connections with all previous ones are learned. In progressive neural networks, each task t is associated with a neural network, which is assumed to have L layers with hidden activations $h_i^{(t)}$ for the units at layer $i \leq L$. The set of parameters in the neural network for task n is denoted by $\Theta^{(n)}$. When a new task $N+1$ arrives, the parameters $\Theta^{(1)}, \Theta^{(2)}, \ldots, \Theta^{(N)}$ remain the same while each layer $h_i^{(N+1)}$ in the task $N+1$'s neural network takes inputs from $(i-1)$ th layers of all previous tasks' neural networks.

$$h_i^{(N+1)} = \max\left(0, W_i^{(N+1)} h_{i-1}^{(N+1)} + \sum_{n<N+1} U_i^{(n:N+1)} h_{i-1}^{(n)}\right) \tag{3.13}$$

where h_0 is the network input,$W_i^{(N+1)}$ denotes the weight matrix of layer i in neural network $N+1$. The lateral connections are learned via $U_i^{(n:N+1)}$ to indicate how strong the $(i-1)$ th layer from task n influences the i th layer from task $N+1$. Unlike pretraining and fine-tuning, progressive neural networks do not assume any relationship between tasks, which makes them more practical for real-world applications. The lateral connections can be learned for related, orthogonal, or even adversarial tasks [61]. To avoid catastrophic forgetting, settings of parameters $\Theta^{(n)}$ for each task n, where $n \leq N$ are frozen, while the new parameter set $\Theta^{(N+1)}$ is learned and adapted for the new task $N+1$.

As a result, the performance of existing tasks does not degrade. The main disadvantage of such an approach is the significant amount of memory required, making such an approach not scalable.

3.9.2 Expert Gate(EG)

It is proposed as a network of experts. Each expert model is trained given a specific task. Assuming EG saw already N tasks, we define for each task k, an autoencoder model A_k and an expert model E_k where $k \in \{1, \ldots, N\}$. When a new task $N+1$ and its training data D_{N+1} arrive, D_{N+1} will be evaluated against each autoencoder A_k to find the most relevant tasks. The expert models of these most relevant tasks are used for fine-tuning or learning-without-forgetting (LwF) to build the expert model E_{N+1}. At the same time, A_{N+1} is learned from D_{N+1}.
When there are many tasks, a system needs to know what model to load when making the prediction on a test example. An algorithm is used to determine the relevance of tasks and only load the most relevant for inference. When predicting a test example, the expert models whose corresponding autoencoders best describe the sample are loaded in memory and used to make the prediction. However, the size of the entire architecture tends to explode with an increasing number of tasks. Therefore for some applications could be unfeasible for scalability problems.

3.9.3 PathNet

PathNet is a modular deep neural network with L layers, each consisting of M modules, where each module is a neural network. Each task at the end of training will be represented as a pathway, i.e., for each layer a subset of modules and all of them together represent the pathway for the task. For each layer, the outputs of the modules in that layer are summed before being passed into the active modules of the next layer. When a new task arrives, it uses an evolutionary algorithm to find the optimal path through such a large DNN, then freezes the weights along the path. A new task can use previously learned modules for an old task or create a new one. An advantage is the memory remains fixed over time. One of the disadvantages is that it is computationally expensive, and it is necessary to know the task label to select the path associated with the task. Moreover, the sequence of tasks is limited since every task will use some new modules until they don't run out.

3.9.4 Piggyback Approach

A method for adapting a single, fixed deep neural network to multiple tasks without affecting performance on already learned tasks [186]. A major drawback of finetuning is the phenomenon of catastrophic forgetting. So they propose not to change the weights of the network. The idea is based on network quantization, a binary mask is learned, the mask "piggyback" on an existing network to

provide good performance on a new task. The advantages of such a method are that the overhead is very low, only 1 bit for each parameter of NN (per task), and that performance is agnostic to the task order. However, the choice of the initialization of the backbone network is crucial for obtaining good performance. Moreover, only the features learned for the initial task, such as the ImageNet pre-training, are re-used and adapted for new tasks. It needs to know from which domain (i.e., task) belongs the sample. Additionally, there is no scope for added tasks to benefit from each other.

3.9.5 PackNet

It identifies weights important for prior tasks through network pruning and keeps the important weights fixed after training for a particular task [187]. Additional information is stored per weight of the network to indicate which tasks it uses. However, a disadvantage of such a method is that performance begins to drop as more tasks are added to the network due to a lack of available free parameters, and the total number of tasks that can be added is ultimately limited due to the fixed size of the network.

3.10 Applications

When considering many real-world applications, there is to consider that at some point in time, a data distribution shift is going to happen. CL moves the usual paradigm of DL towards networks that can continually accumulate knowledge over different tasks without the need to retrain from scratch. Below are some examples of domains where continual learning may be beneficial or has already been applied. However, it should be noted that this is not an exhaustive review but rather a list of potential applications. The applications consider cases where the data is available over time and that require an update of the model in an efficient way. For some applications could also be the additional constraint of real-time and so the data cannot be stored and reprocessed.

3.10.1 Computer Vision

Over the last few years deep learning methods have been shown to outperform previous state-of-the-art machine learning techniques in several fields, with computer vision being one of the most prominent cases. Computer vision (CV) is a field of artificial intelligence (AI) that enables computers to take meaningful information from images and videos. Some research fields of CV are image classification, object detection, semantic segmentation, image restoration, activity recognition and human pose estimation [311].
A possible application of CL in CV could be the necessity to adapt their behaviors over time. For example, in the context of object detection, CL may be used to update a model using only new images of new classes, avoiding retraining from scratch and improving its object detection capabilities. In general, given

that much information in a lot of fields can be represented in terms of images, the field of CV is considered very important. Because of such importance, most of the work in CL is performed considering the CV domain. It is important to note that CV is a multifaceted field and therefore, its use can include a lot of real-world applications described below. Some examples are healthcare, manufacturing and robotics [301].

3.10.2 NLP

Natural Language Processing (NLP) is a field of artificial intelligence that deals with how computers can understand, interpret, and generate human language [62]. NLP is an important area of research and development that has the potential to transform how we interact with computers and each other. Some research fields are text classification, text summarization, machine translation and sentiment analysis.
CL algorithms can learn and adapt to new tasks or domains without forgetting what they have learned previously. This can be useful for tasks that require systems to adapt to changing patterns or user needs over time. As for CV, NLP is a multifaceted field and therefore, its uses can include a lot of real-world applications like healtcare, recommendation systems, virtual assistants and robotics. For example, in virtual assistants such feature could be very useful. Indeed, CL algorithms can adapt to the specific needs and preferences of individual users, allowing NLP systems to provide more personalized and effective language processing services. In particular, a training more light could allow an adaptation based on user needs directly on the device, avoiding issues of data privacy. Moreover, CL algorithms can transfer knowledge learned from one NLP task or domain to another, allowing NLP systems to leverage existing knowledge to learn new tasks more efficiently.

3.10.3 Audio

Deep learning has significant applications in the field of audio, as proved in recent years. Some examples are speech recognition, audio classification, audio generation, and music generation. Continual learning is particularly relevant in the field of audio, where the model needs to be able to adapt to changes in the data distribution. One way to apply continual learning to audio is by training a model to recognize and classify different audio events, such as speech, music, or ambient noise. The model can be initially trained on a large dataset of audio signals and then continually fine-tuned as it is exposed to new audio events or variations in the data distribution. Another application of continual learning in audio is the development of personalized speech recognition systems, which can learn and adapt to an individual's unique voice and speaking style over time.

3.10.4 Robotics

Autonomous robots can make judgments and conduct activities on their own without direct human guidance. This is especially beneficial in situations when a human operator is not practical, such as space exploration or disaster response operations. Autonomous robots have numerous applications, including manufacturing, construction, transportation, agriculture, and healthcare. CL in this field can be useful because real-world robots may require the ability to adapt when confronted with new scenarios. For example, CL might be used in autonomous navigation. CL enables robots to adapt to changing environments and improve their capacity to navigate unexpected terrain.

3.10.5 Recommendation Systems

A recommender system is a type of AI application that uses data and algorithms to make personalized recommendations to users. Recommendation systems are used in various contexts, including e-commerce, social media, and entertainment. CL can be an essential part of a Recommender System, because it can help improve efficiency and customer satisfaction. It can help users discover new items they may be interested in and make it easier for them to find what they want.

3.10.6 Industry 4.0

Industry 4.0 refers to the integration of advanced digital technologies into the manufacturing sector. Manufacturing industries are in a continuous effort to improve their production yield, uptime and throughput, to increase production quality and minimize costs. Complex manufacturing can use Continual Learning to adapt to changes in the manufacturing process and improve efficiency over time. There are a few reasons why data distribution shift can occur in Industry 4.0 such as changes in the underlying process, changes in the environment and data collection process. For example, if a model was trained on data from a manufacturing process, and the process was modified, the model may no longer be able to accurately predict the output of the new process. In another case, the environment in which the data is collected may change, leading to a shift in the data distribution. For example, if a model was trained on data from a specific location and then some environmental conditions changed.

3.10.7 Edge Computing

Edge computing is a paradigm that aims to improve response times and conserve bandwidth by bringing computation and data storage closer to the location where it is needed. This is typically done by placing small, low-power computing devices at the edge of a network, near to machines or where users will use them. So such devices can reduce the amount of data that needs to be transmitted over the network and reduce the latency. Such features can be especially important

in situations where real-time data processing is required.
In comparison to cloud computing, edge computing typically has more limited computing power and data storage capabilities. As a result, it is often only practical to apply models in the inference phase, rather than training them using a lot of data. Continual learning can help to address this challenge by allowing models to update and adapt to the new information that arrives over time, rather than requiring retraining from scratch using all data seen so far. This can help to reduce the computing power needed for training, as only an update of the model is required. For example, consider an edge computing system that is deployed in a manufacturing location and is used to process data received from the machines connected thanks to IoT. The data collected by the sensors may change over time, for example, due to changes in the environment or the deployment of new sensors. With continual learning, the edge computing system would be able to be retrained from scratch every time, which may not be practical due to the limited computing resources available at the edge.

3.10.8 Cybersecurity

Cybersecurity protects computers, servers, electronic systems, networks, and data from digital attacks. It involves using technologies and policies to avoid attacks and unauthorized access to networks and data. So CL can be used to improve many cases of cybersecurity such as malware detection, spam filtering and network security. For example, in the case of network security, CL can be used to improve security by enabling systems to adapt to changes in network traffic patterns and identify suspicious activity.

3.10.9 Smart City

Smart cities utilize many technologies like sensors and artificial intelligence to gather and analyze data about the city. This information is then used to improve citizens' quality of life and enhance the efficiency of services offered by the city.
Continual learning can be useful in the context of smart cities to enable systems to adapt to changing conditions and improve their performance over time. Some potential applications of CL in smart cities are traffic management, energy management and public safety. For example, CL can optimize traffic flow in real-time based on changing conditions such as accidents. Considering energy management, CL can be used to predict energy consumption in different locations in buildings and understand where there can be problems by adapting to changing demand patterns.

3.10.10 Machine Learning Production Systems

Data could be changing because of trends, or because of different actions made by the users. For example, let's consider a recommendation system for buying new items. Let's assume that a user could be interested in some hobbies, and

after some time, it changes user behavior and type of hobbies. So, the system needs to retrain the model and recommend new items associated to the new hobbies to the customer. With a model in production, this can happen often and it needs to be retrained periodically.

Being able to fast train and deploy new prediction models over time becomes essential to provide up-to-date and always improving services. Another example could be Anomaly Detection Systems in Continual Learning, in this scenario, may substantially reduce the computational burden incurred by such systems in retraining models from scratch every time at a massive scale with a direct impact on resources occupation and energy consumption.

3.10.11 Conclusions

Continual learning is a type of machine learning that enables models to adapt to changing data and environments over time. This is particularly valuable in applications that need machine learning models to be more flexible and improve their performance as they encounter new data and situations.
The presented applications are meant to give a general idea of some ways that CL can be utilized, but they only represent part of the scope of its potential use cases. In other words, there are likely many more ways CL can be applied beyond the examples mentioned.
CL is a valuable capability for machine learning models, as it allows them to adapt and improve their performance over time, even as the data and environments they are working with change. This can make machine learning models more effective and valuable in various domains.

3.11 Continual Learning and Industry 4.0

Industry 4.0 takes the manufacturing or operating a factory at a new level. It interconnects the multiple digital technologies with each other and shares the data by using IoT (Internet of Things). The IoT allows the user as well as a control unit to access the real-time data of all the systems running in the factory. This allows better collaboration of digitally connected devices with each other as well the control unit. This gives an instant boost to productivity, improve a process, and drives growth and safety.
When considering about CL in the context Industry 4.0 there are many possible domains like Manufacturing and many possible fields such as Predictive Maintenance. Continual learning, also known as lifelong learning or incremental learning, refers to the ability of a machine learning model to learn and adapt to new information or tasks over time, without forgetting previous knowledge. This is in contrast to traditional machine learning models, which are trained on a fixed dataset and are not able to adapt to new information or tasks.
In the context of Industry 4.0, continual learning can be particularly useful for applications that involve dynamic environments or changing data. For example, in a manufacturing setting, a machine learning model that is able to

continually learn and adapt to new production data can be more flexible and responsive to changes in the manufacturing process. Similarly, in a predictive maintenance application, a machine learning model that is able to continually learn and adapt to new data from sensors and other sources can improve the accuracy of its predictions over time.

3.11.1 Manufacturing

In the work [319] is pointed out that one of the disadvantages is that DL models need to rebuild the model from scratch and the existing knowledge may be difficult to utilize. Therefore, it is necessary to enable deep learning with incremental learning capabilities [319] with the capacity of transfer learning.
In [274] is discussed of the potential of CL for Manufacturing and how continual learning concepts can be exploited for the development of anomaly detection systems. Distribution drifts refer to changes in the distribution of data over time, which can cause problems in the performance of machine learning models in manufacturing (especially anomaly detection systems). These drifts can be caused by changes in the manufacturing process and can lead to a decrease in the overall effectiveness of equipment (OEE), as well as slower manufacturing processes. For example, in the case of Anomaly Detection, OEE decreases when workpieces produced under the new distribution are wrongly categorized as faulty workpieces, slowing down the manufacturing process by unnecessary and excessive manual quality measurements. To avoid these problems, machine learning systems must be constantly evaluated and potentially retrained and redeployed to account for changes in the data distribution. However, this requires a manual and time-demanding workflow. Continual learning approaches allow for the continuous acquisition of new information from a continuous data stream and may offer a solution for developing systems that can adapt to distribution drifts in real-world manufacturing data.
In practice we can find many studies on CL to solve problems of Manufacturing and Industry 4.0.
For example, considering Manufacturing, the authors of [191] propose a study of CL approaches applied on a real industrial metal forming dataset collected on a hydraulic press. It consists of data from eight pumps applying pressure on a shared oil reservoir. Due to this setup, anomalous behavior of one pump is compensated by other pumps. This hides such behavior from the operator, because initially no problems occur. However, the other pumps experience increased wear, which makes an early detection of the described anomalous behavior desirable. The challenge is further increased by frequent alterations of the production process, either by improvements such as new molds or changes of the manufactured product. Every alteration causes a change of the process' characteristics, which require anomaly detection algorithms to be retrained. Therefore they propose to use approaches belonging to a specific family of CL known as regularization-based and perform Anomaly Detection on data. In the dataset used in this study, pressure data labeled 'normal' or 'anomalous' from the production of fifteen different products each being produced hundreds of

times are included
In [156] they consider CL applied to Fault Classification. They consider the problem where stream data collected by manufacturing processes have characteristics that vary over time. These sequentially non-stationary hallmarks of process data confuse model stability and give rise to catastrophically forgetting the previous features. When temporal structured data samples change over time, changes in data distribution can be observed. To overcome such problem they propose in the context of CL a pseudo-rehearsal approach where they use a GAN as generative model to produce the samples of old tasks to use in conjunction of samples of the current task during the training of the current model. Moreover, the model uses GAN to balance fault data during generation. To validate the approach a dataset of the UCI Machine Learning Repository with unbalanced six faults is used, showing the positive effect of a balancement.
In [334] is studied CL in the domain of Industrial chemical processes. Industrial chemical processes are a vital part of many industries and play a critical role in the production of a wide range of products. Nowadays, industrial chemical processes are becoming more and more complex, and the scale is increasing. The article proposes a new strategy that it is the combination of the proposed improved residual neural network (DResnet), enhanced loss function (AM-Softmax loss function and distillation loss function), and enhanced training set is used to alleviate catastrophic forgetting They use the enhanced loss function to constrain the neural network to learn new knowledge and retain old knowledge. The proposed method is compared to the well-known iCaRL approach showing an improvement in performance. Experiment demonstrates the effectiveness of the algorithm for mitigating catastrophic forgetting by using the Tennessee-Eastman (TE) chemical process dataset.
In terms of Fault Prediction and Manufacturing, in [192] they consider the manufacturing process for lithium-ion batteries, which it is complex and requires specialized equipment and expertise. In recent years, the use of lithium-ion batteries has greatly expanded into products from many industrial sectors, e.g. cars, power tools or medical devices. Previous knowledge of the time a component's failure will occur allows for countermeasures and reduces the risk to a system's safety and security [1]. An early prediction and robust understanding of battery faults could therefore greatly increase product quality in those fields. However, taking those lithium-ion batteries as an example, their fault mechanisms and usage behaviors are so complex, that traditional, model-driven approaches do not suffice. While current approaches for data-driven fault prediction provide good results on the exact processes they were trained on, they often lack the ability to flexibly adapt to changes, e.g. in operational or environmental parameters. To obtain such flexibility they perform a study using Continual learning approaches, allowing for an automatic adaption of previously learnt knowledge to new tasks. More in detail, the paper discusses different continual learning approaches from the group of regularization strategies and evaluated on a real battery wear dataset.
Another work was performed by [193] using the Turbofan Engine Degradation Simulation Data Set (TEDSDS) by NASA [260]. It is based on the Commercial

Modular Aero-Propulsion System Simulation developed by [214] and includes four simulated datasets, each one containing several dozens of individual engines' time series. They are interested in CL to automatically adapting to different usage contexts, in such way, it would greatly expand the usefulness of current predictive maintenance solutions. As CL approach they considered Elastic Weight Consolidation (EWC).
In [368] performed an interesting study of Fault Diagnosis on the bearings. Rolling element bearings play vital roles in mechanical equipment, they support rotating bodies to reduce friction and guarantee rotational accuracy. Bearings normally work in complex and harsh conditions and they are prone to failure that shuts down the entire machine. However, about 40Bearing monitoring and diagnosis ensure safety and reduce economic losses. In recent years, various types of signals have been used for bearing faults diagnosis, including vibration. Deep-learning schemes have thus been used to diagnose bearing faults [14–16]. However, given the complex and changeable bearing work environments, the distributions of collected data vary greatly. As deep-learning algorithms are generally data-driven, performance of the model decreases when the data distribution changes (i.e. catastrophic forgetting). Therefore they consider as solution Continual Learning and propose a new approach called Repeated Replay Using Memory Indexing (R-REMIND). Such approach is inspired to the original REMIND since they also perform product quantization and store the features in indices. The performance was assessed using a dataset from Paderborn University [32]. The dataset was derived using ball bearings of type 6203 and includes information on artificial damage caused by electric discharge machining, drilling, etc.
In [128] they propose the study of CL in the context of Induction Motors (IMs). IMs support most of the production process in the modern industry's daily life due to their straightforward construction, reliability, and relatively low cost. However, IMs operate for long uninterrupted working periods, are exposed to the elements, and minimum preventive maintenance. These operative conditions raise unexpected faults that can show up at any time, causing lower productivity and economic losses. Thus, early motor failure detection and correction are challenging problems that catch many researchers' attention. In past years, deep learning (DL) architectures have been used in fault diagnosis. However, modifications in the operative conditions can generate patterns from new failures that differ from those detected by the current model. This issue forces existing methods to learn a new model considering unknown failure conditions, but current models for automatic detection can learn new faults at the cost of forgetting concepts previously learned. Therefore in [citation], the authors propose a CL approach to overcom this problem in the context of early detection of fault events of electromechanical systems, which is considered one of the most critical data challenges in modern industry. Similarly to the well-known CL approach ICaRL, the authors also use the Nearest Centroid Classifier (NCC) to classify samples instead of relying on the final output of the neural network. In conjunction they also apply experience replay (ER) to mantain the knowledge of old tasks. They provide evidence to support their claim that their approach

is effective in identifying and addressing faults in a continuous learning setting. They validate their approach using two public benchmark datasets: the first belongs to the domain of motor common fault diagnosis and the second is in the domain of bearing fault diagnosis. Specifically, they used two public benchmark datasets are: asynchronous motor common fault (AMCF) [1] and Case Western Reserve University (CWRU) [25].

In the paper [347] it is considered Fault Diagnosis of Control Valve. Deep neural network learning is a commonly used method for fault diagnosis of the control valve. However, the catastrophic forgetting problem of deep learning in multi-task affects the fault diagnosis accuracy. This paper proposes a fusion of elastic weight consolidation algorithm and residual shrinkage network method, sharing common feature layers. The results indicate that this method can effectively alleviate the problem of the catastrophic forgetting for the condition identification of the control valve.

Fault data are distributed in a continuous flow of constantly generated information and new faults will inevitably occur in unconsidered submachines, which are also called machine increments. Therefore, adequately collecting fault data in advance is difficult. Limited by the characteristics of DL, training existing models directly with new fault data of new submachines leads to catastrophic forgetting of old tasks, while the cost of collecting all known data to retrain the models is excessively high. DL-based fault diagnosis methods cannot learn continually and adaptively in dynamic environments. In [33], it is proposed a dual-branch adaptive aggregation residual network (DAARN) where two types of residual blocks are created in each block layer of DAARN: steady and dynamic blocks. In addition, a feature-level knowledge distillation loss function is proposed to further overcome catastrophic forgetting. The effectiveness of the method is verified by experiments on the three bearing datasets. The results show that it outperforms other continual learning methods and has satisfactory robustness.

To characterize machine health for predictive maintenance activities on smart factory, it is necessary to consider data-driven algorithms. Since data collection can be expensive, datasets may not cover all the possible fault conditions. Research has focused ondetecting unknown conditions rather than integrating unknown conditions into future predictions. Therefore, Continual Learning solutions should learn to classify new conditions and use improved representations that minimize the need for future fine-tuning. Meta-learning approaches like Few-Shot Prototypical Networks (FSPN) regularize base-task learning to find these more generalizable representations. In [253], experiments on a motor data set demonstrate that FSPN with only 5 or 10 examples of the novel fault consistently outperforms static, fine-tuning, and Elastic Weight Consolidation (EWC). To evaluate the proposed method, a SpectraQuest Machinery Fault Simulator (MFS) Magnum simulated eight motor operating conditions: normal, faulted bearings, bowed rotor, broken rotor, misaligned rotor, unbalanced rotor, phase loss, and unbalanced voltage.

In [363] they consider two datasets that resemble manufacturing processes. They propose a novel approach that is the conjunction of Probabilistic Slow Feature

Analysis (PSFA) and Elastic Weight Consolidation(EWC) called PFSA-EWC. PFSA is an unsupervised machine learning method for extracting slow-varying features from a stream of high-dimensional data. . The first considered dataset is a continuous stirred tank heater (CSTH), a type of industrial equipment used to heat a liquid or slurry in a tank by stirring it with a heating element. The second one is a practical coal pulverizing system, a method of crushing coal into a fine powder, so that it can be burned more efficiently and used more effectively in power plants and other industrial processes.

For multimode processes, one generally establishes local monitoring models corresponding to local modes. However, the significant features of previous modes may be catastrophically forgotten when a monitoring model for the current mode is built. In study [362], a modified PCA algorithm is built with continual learning ability for monitoring multimode processes, which adopts elastic weight consolidation (EWC) to overcome catastrophic forgetting of PCA for successive modes. Such method is called PCA-EWC. A practical industrial system in China are employed to illustrate the effectiveness of the proposed algorithm
Industrial systems generally operate under multiple modes due to changing of raw materials, market demands, etc. Multimode process monitoring is increasingly significant as industrial systems generally operate in varying operating conditions. However, most researches focus on multiple local monitoring models for complex multimode processes and assume that data of all possible modes are available and stored before learning. When similar or new modes arrive, local models are rebuilt corresponding to each mode and the model's capacity would increase with the continuous emergence of modes.
The previous method called PCA–EWC [362] assumes that data follow multivariate Gaussian distribution and are stationary in each mode. Therefore, in the paper [363], they consider sparse dynamic inner principal component analysis (SDiPCA). By adopting the concept of intelligent synapses in continual learning, a loss of quadratic term is introduced to penalize the changes of mode–relevant parameters, where modified synaptic intelligence (MSI) is proposed to estimate the parameter importance. The resulting approach is called SDiPCA–MSI). The effectiveness and superiorities of the proposed method are demonstrated by a continuous stirred tank heater case and a practical industrial system. For industrial systems, such as large–scale power plants and chemical systems, the proposed method has the ability to monitor successive dynamic modes.
In [294] was proposed a CL study about Quality Prediction for Manufacturing. Deep learning-based predictive quality enables manufacturing companies to make data-driven predictions of the quality of a produced product based on process data. In the current state of research, there are already many examples that successfully demonstrate the feasibility of deep learning based predictive quality in various manufacturing processes such as deep drawing, hydrocracking, lasermachining, or additive manufacturing. However, a central challenge is that production processes are subject to continuous changes. For example, as soon as a new product is manufactured or a process is reparameterized, the process behavior changes and with it the relationships between process and quality

data, with the result that previously trained models may no longer perform well in the process. This strongly limits the sustainable use of deep learning in the production context, especially since the collection of representative process data is costly and time-consuming. Moreover, often in the production domain are that, due to limited hardware capacities or corporate policies, long-term process data cannot be stored or accessed and model training must be carried out in a resource-friendly manner. Such problems are addressed and demonstrate the feasibility of application of Continual Learning for a real use case in injection molding. They consider some CL approaches and also propose a variant of the approach Memory Aware Synapses (MAS). More in detail, it is used a neural network for numerical prediction of product quality based on machine parameters and validate using a real-world regression problem in injection molding.
In was performed a study in the field of oil about the Distributed Virtual Flow Meter. A distributed flow meter can be defined as a metering solution that measures volume, velocity, and the fraction of fluid components in every location in the pipe. A distributed reading of the flow meter is essential because it can provide accurate information about the location of interest on where the intervention might be necessary. With this information, more effective action can be taken and the reaction time can be significantly reduced, which will translate to an increase in production well productivity within the multi-trillion-dollar oil industry. The distributed measurements in this work are acoustic signals acquired using a technology called distributed acoustic sensing (DAS). It was shown that using a deep neural network (DNN) can be used to estimate the phase-fraction of the fluids from the acoustic data. Unfortunately, the accuracy of such approach degrades rapidly when the model tries to infer from the data outside the training distribution, which is a common situation for real-world applications of modeling the DAS data. In [17], they study several well known CL approaches applied to a real-world distributed sensor dataset collected from an oilfield, and also propose a novel technique to do Compressed Replay. The novel sample compression algorithm allows for the rehearsal training to store representative data with less memory than the usual size required to store a sample. The algorithm uses the inverse cutout technique to compress and isolate the most relevant parts of the input data, and use them for rehearsal. Industrial robots (IRs) are mainly used for handling and welding tasks that do not exert high forces on them. The paper [299] presents a CL framework for data-driven learning of the dynamics model of a 6-degree-of-freedom serial industrial robot. This model can be used for model-based control algorithms, without the need for extensive identification of robot specific parameters such as mass inertia, and can additionally model complex effects such as friction. Furthermore, using CL, it can adapt to changes of the robot e.g., due to wear or new tasks. With the help of CL, the ML-based dynamics model is continually fed new data and improves over the operating period of the robot.
Employers are obliged to have specific work equipment inspected to guarantee a safe workplace. All work equipment is subject to a maintenance obligation and some work equipment must undergo an inspection. The employer is obliged to determine which work equipment must be inspected and must also provide

written evidence for the regulatory agency. To improve the asset management, based on customer experiences that registration assets is a time-consuming activity, they propose to determine the type of asset based on the image. Based on the asset type, the brand and object type can also be determined. Retraining on the data set, including the new images of already familiar classes, might increase the model's performance on these classes since training on more data often results in more accurate performance. In addition, because the model must also predict the newly added classes, the model must occasionally be updated. A solution to the problem that training from scratch each time is resource inefficient, is a Continual Learning model, so in [251] they propose to apply CL in an asset management context.

3.11.2 Conclusions

Continual learning is a type of machine learning that enables models to adapt to changing data and environments over time. This is particularly valuable in the context of Industry 4.0, as it allows machine learning models to be more flexible and improve their performance as they encounter new data and situations.
The above approaches mentioned in the original text are just a few examples of how CL can be applied in Industry 4.0. These examples are meant to give a general idea of some of the ways that CL can be utilized in this context, but they do not represent the full scope of its potential use cases. In other words, there are likely many more ways that CL can be applied in Industry 4.0 beyond the examples mentioned.
Overall, CL is a valuable capability for machine learning models in Industry 4.0, as it allows them to adapt and improve their performance over time, even as the data and environments they are working with change. This can help to make machine learning models more effective and useful in a variety of industrial settings.
The above approaches are just a few examples of how Continual Learning can be applied in Industry 4.0, and do not represent the full scope of its potential use cases. These examples are intended to give a general idea of some of the ways CL can be utilized in this context.
Overall, continual learning is a valuable capability for machine learning models in Industry 4.0, as it allows them to adapt to changing data and environments over time, improving their flexibility and performance.

Chapter 4

Interpretability

In the following chapters, we will examine research questions that are inspired by industrial problems. Many of the studies we will discuss were conducted in the context of Continual Learning, as this approach can be beneficial for addressing many industrial challenges. However, there are other important challenges in Industry 4.0, such as Interpretability, which also play a significant role. For example, thanks to Interpretability is possible to understand how a machine learning model is making decisions and predictions. This is important in industries where the consequences of incorrect decisions can be significant.

4.1 Motivation

With the increase of computational power and the growing availability of data, Machine Learning (ML) approaches have been successfully applied to many scientific, technological, and business areas [129, 133, 275, 305, 163, 112]. In particular, ML has paved the way for more effective Decision Support Systems (DSS). To make the most from the advanced data analytics capabilities enabled by ML, analysts should be able to get explanations about model predictions so they can make faster, better, and more reliable decisions based on the information processed by ML algorithms [268, 6]

On the one hand, some ML methods (e.g. linear regression, simple decision trees) are intrinsically interpretable and provide a feature importance score for each of the input variables. On the other, the most advanced ML models that allow for better performance (not only neural networks but also advanced classic models such as those based on ensembles of trees) work just like "black boxes" [111, 75, 348]: they provide predictions to the users, but no hint about how such predictions are derived. As a consequence, often there is a lack of trust that hinders the adoption of ML-based solutions. This is particularly detrimental in the context of DSS [83] and other socio-technical systems [14]. The importance of being able to explain this type of model, widely used in multiple sectors, generated great interest and value in the search for methods of interpretability.

4.2 Intepretability

4.2.1 Intro

Interpretable Machine Learning is a research field that aims at shedding light on how black-box ML model predictions depend on input features [210]. ML interpretability methods can be classified into *model-specific* and *model-agnostic* ones. The former, also known as *ad-hoc* methods, are designed for a specific class of models [204, 46]. The latter, also known as *post-hoc* methods, can be applied on top of every ML model [255].

In this work, with DSS in mind, we focus on model-agnostic approaches. Indeed, in many practical scenarios, different ML models can be used to address many data processing tasks needed to distil the information to be exposed through the DSS. However, users may lack a background in ML, so it can be difficult for them to deal with different feature importance indicators produced by different methods. For them, having to deal with a unique set of feature importance indicators, such as those computed by a model-agnostic explainability approach, is undoubtedly a great advantage. Indeed, less training is needed and, given a task (e.g. regression or classification), users are always confronted with the same model evaluation interface, independently from the actual ML model implemented to solve it.

From the perspective of the level of granularity, interpretability algorithms can be further divided into two classes: *global* interpretability methods, aimed at providing explanations of the model as a whole, and *local* interpretability methods, aimed at providing explanations associated with individual predictions. Of course, both global and local interpretability are useful in DSS. On the one hand, global interpretation can be used to build trust in the predictive model; on the other, local interpretation provides users with contextual information that can be relevant in the decision-making process (e.g. in root cause analysis).

While it is apparent that in human-in-the-loop applications interpretability insights must be provided to the user in a reasonable amount of time, most state-of-the-art interpretability approaches require time-consuming procedures for computation that do not allow for 'on-the-fly' operation.

4.2.2 Related Work

Among model-agnostic approaches, Partial Dependence Plots (PDPs) [96] are one of the most well-known: PDPs simply make each variable vary in a range and show the annexed predictions in a plot. While PDPs provide a rich description of the effect each feature has on the prediction, such approach requires a non-negligible effort by the user to analyse and compare a potentially large number of plots (one for each feature). Moreover, PDPs focus on global interpretability only. An alternative approach is given by Permutation Importance, introduced for Random Forests in [38], and its variants, such as the Conditional Permutation Importance described in [279]. These approaches are based

on random permutations of values for each feature and produce a ranking of features according to the resulting importance score. Results are very easy to understand, but they provide only global level interpretability.

As for local interpretability, a key contribution was given in [247], where authors propose Local Interpretable Model-agnostic Explanations (LIME) approach. LIME is one of the interpretability techniques that take advantage of data disturbances, such as EXPLAIN and IME [280].

To explain a prediction of a complex model (target model), LIME chooses a model from a class of interpretable surrogate models with low complexity that locally minimizes the distance between the model to be interpreted and the surrogate model predictions. Examples of LIME usage can be found in healthcare [71] and industrial [263].

Another model-agnostic approach designed to explain local predictions is Anchors [248]; Anchors produces decision rules in the form of *if-then* statements, and it can be applied to different domains, e.g., tabular data, images and text. However, since rules are computed by means of reinforcement learning and beam search, this approach has a high computational burden that makes it not suitable for the deployment within a DSS. Among *a posteriori* explainability techniques that can provide both global and local interpretation, one that has received a lot of interest in recent years is SHapley Additive exPlanation (SHAP) [182]. SHAP has been applied in a wide variety of research fields. In [104], a new global interpretable method based on Shapley values was proposed: the approach combines predictive accuracy with explainability and it is applied to the financial problem of predicting bitcoin prices. In [220], SHAP is used to intepret an XGBoost model that detects the occurrence of accidents using a set of real time data. SHAP has also been extended in different ways. Recently, approaches such as Asymmetric Shapley values [97] and Shapley Flow [318] have been proposed to include prior knowledge about causal relationships among data in the interpretation of the models. In [15], the authors use Kernel SHAP to tackle an unsupervised task, i.e., to explain anomalies detected by an autoencoder. Despite the great success, SHAP has some drawbacks. First, SHAP is computationally burdensome, as it will be discussed in the next section. Second, it is a complex approach, not only for its theoretical basis, but also in terms of the resulting plots, that can be difficult to read, especially for users with no ML background. About the methodological foundations, they seem to be not fully understood by many users, potentially leading to misuse of the tool [152]. While the last issue could be eased by a better training on the theoretical aspects (even though many end-users may lack the technical background to grasp them or the time to invest in such training), the computational complexity could be a bottleneck in situations where multiple models have to be compared, or limited computational resources are available, or fast reaction based on model prediction is required. As for the computational speed, it is nowadays a challenge, particularly in the Web-of-Things context, where the amount of data is considerable, often in the form of live streams with extremely fast update. Besides, a fast update of data leads to a more frequent retrain of the ML models, and when a model changes, it is necessary to rerun the entire ML

interpretability procedure, which worsens the computational burden associated to SHAP.

4.2.3 SHAP-based Approaches

4.3 SHAP

SHAP

As mentioned in 4.2.2, SHAP [182] is a framework for interpreting predictions produced by a ML model based on the concept of Shapley value from cooperative game theory. Given a trained model f and a generic data point $\mathbf{x}$, represented by a p-dimensional feature vector (i.e. $\mathbf{x} \in \mathbb{R}^p$), SHAP computes, for each feature, a real-valued quantity that represents the contribution of the feature to the prediction $f(\mathbf{x})$. The main idea is that the prediction can be explained by treating it as a "payout" that needs to be distributed across features, which act as "players" in a coalition. As many other interpretability methods, SHAP relies on the definition of an *explanation model* g which is simpler than the predictive model f to be explained, while being a good approximation thereof at least locally. Specifically, the explanation model g is a linear function of binary variables z_j, indicating the presence or absence of the corresponding feature in the sampled coalition, and takes the form

$$g(\mathbf{z}) = \phi_0 + \sum_{j=1}^{p} \phi_j z_j. \tag{4.1}$$

The p-dimensional vector of ones and zeros $\mathbf{z} = [z_1, \ldots, z_p]$ represents the sampled coalition: $z_j = 1$ denotes presence of the j-th feature in the coalition, while $z_j = 0$ indicates that the j-th feature is not in the coalition. At the core of the SHAP method is the estimation of the linear model g given K sampled coalitions $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(K)}$, which allows for a straightforward computation of the feature attribution coefficients ϕ's, i.e. the Shapley values.

Notice that, in light of the nature of the underlying explanation model (4.1), SHAP can be considered as an additive feature attribution method. The most popular model-agnostic version of SHAP is *KernelSHAP*, characterized by high portability as it can be used on any ML model that has tabular data as input. Model-specific versions have been proposed for tree-based models (*TreeSHAP* [181]) and for Deep Learning models (*DeepSHAP* [182]), both with the aim of leveraging the internal structure of the model at hand to improve the computational performance of the algorithm. Since our main goal is the design of a completely model-agnostic interpretability method for tabular data, in the remainder of this work we mainly consider KernelSHAP as benchmark for comparisons.

Age	Weight	Height	Sex		Age	Weight [kg]	Height [cm]	Sex
1	1	1	1	$\longrightarrow$	24	88	185	M
1	0	0	0	$\longrightarrow$	24	**60**	**193**	**F**
1	1	0	0	$\longrightarrow$	24	88	**178**	**F**
1	1	1	0	$\longrightarrow$	24	88	185	**F**

Table 4.1: An example of mapping from the space of coalition vectors to the original feature space, with absent features replaced with feature values (in **bold**) randomly sampled from the dataset.

4.3.1 KernelSHAP

At the core of the *KernelSHAP* method is the estimation of the linear explanation model (4.1) by means of the optimization of a squared loss function in which the contribution of each sampled coalition $\mathbf{z}$ is weighted according to a specific weighting kernel (whose rationale and properties are discussed below). The procedure is summarized in Algorithm 4.1.

Algorithm 1 KernelSHAP

for $i = 1$ to N **do** ▷ For each point in the dataset

for $k = 1$ to K **do**:

- $\mathbf{z}^{(k)} \in \{0,1\}^p, k \in \{1,\dots,K\}$ ▷ Sample the k-th coalition
- $\pi_x(\mathbf{z}^{(k)}) = \frac{(p-1)}{\binom{p}{|\mathbf{z}|}|\mathbf{z}|(p-|\mathbf{z}|)}$ ▷ Compute the weight for coalition $\mathbf{z}^{(k)}$

end for

- $\mathbf{X}' = h_{\mathbf{x}}(\mathbf{Z})$ ▷ Map coalitions to the original feature space
- f: $\widehat{y}' = f(\mathbf{X}')$ ▷ Compute predictions
- $L(f, g, \pi_x) = \sum_{\mathbf{z} \in Z}[f(h_{\mathbf{x}}(\mathbf{z})) - g(\mathbf{z})]^2 \pi_x(\mathbf{z})$ ▷ Fit the linear explanation model g
- $\phi_j = \beta_j$ for $j = 1,\dots,p$ ▷ Return the Shapley values

end for

- $I_j = \sum_{i=1}^{N} |\phi_j^{(i)}|$, for $j = 1,\dots,p$ ▷ Return the global importance score

Figure 4.1: Algorithm KernelSHAP

We recall that $h_{\mathbf{x}}$ is a function from $\{0,1\}^p$ to $\mathbb{R}^p$ mapping 1s into the original feature values of data point $\mathbf{x}$ and 0s into feature values randomly sampled from the dataset (see the example in Table 4.1);

The authors of the method suggest to use $K = 2048 + 2 \cdot p$ coalitions [180] to obtain the best results in the majority of situations. The higher the number of evaluated coalitions, the higher the computational time, due to the larger dimension of the matrix to be inverted in the estimation of the linear explanation model. On the other hand, a larger number of coalitions could ensure the estimation of a more accurate explanation model, assumed the sampled coalitions are sufficiently informative. For this purpose, the coalitions are not chosen at random: the weight $\pi_x(\mathbf{z})$ is exploited to select top K most informative ones, i.e. those whose associated weights are the highest. Notice that the number

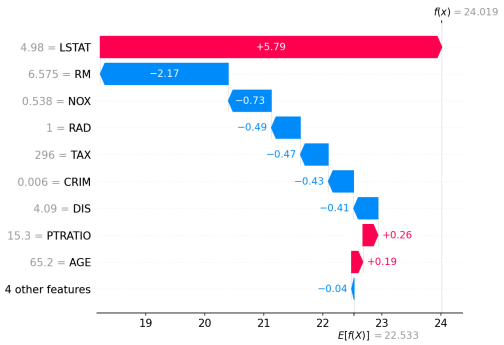
of sampled coalitions (according to the authors' suggestion) increases with the number of features. In addition, the number of linear models that have to be estimated increases with the number of data points in the dataset. As a consequence, the computational cost does not scale well with the dataset dimensions and this is the reason why *KernelSHAP* is computationally burdensome, especially in big data regimes. In Section 4.5.2 we will analyze different solutions the authors provide to lower the computational time, highlighting the impact such trade-offs have on the quality of the produced explanations.

4.3.2 SHAP results visualization

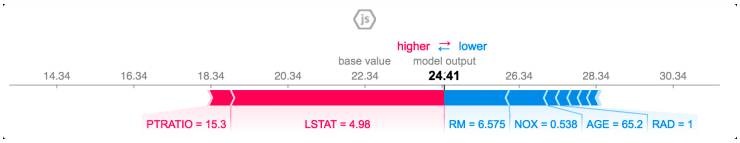
Before introducing AcME, we briefly introduce the most frequently encountered visualizations of SHAP results, that will be useful to better assess the visual efficacy of the novel approach detailed in the following sections. SHAP global importance is a jitter plot that shows both feature effects and feature importance. The y-axis has the feature names sorted by decreasing importance score, while on the x-axis there are the Shapley values. The color map, from blue to red, represents the value of the feature from low to high. Overlapping points are jittered in y-axis direction to get a sense of the distribution (Figure 4.5b). As for local interpretation, SHAP provides two different plots: the waterfall plot (Figure 4.2(a)) and the force plot (Figure 4.2(b)). They are both based on the concept that each SHAP value is a 'force' that either increases or decreases the estimation: the prediction starts from a baseline, that is the average of all predictions, and it is then modified by such force. In local importance plots, each positive Shapley value is an arrow that increases the prediction, while negative values decrease it. Balancing each other, the arrows point to the actual prediction for the selected observation. The difference is that, while the force plot has all the arrows on the same row divided in positive values (on the left) and negative values (on the right), the waterfall plot has one row for each arrow, ordered by impact. Both visualizations are very different from the one used for global interpretation. One shortcoming is that they do not provide any hint about features distributions to contextualize the values of current observation.

4.4 Proposed Approach

We draw inspiration from the computational simplicity behind PDPs, the versatility and some excellent visualization of SHAP, and we propose AcME, a new model-agnostic approach for both global and local interpretability. By proposing a different methodology, with the goal of speeding up computational time, AcME manages to provide an importance score for each feature and a plot inspired by SHAP's summary plot, in which we see not only each feature's importance but also the effect on the predictions. Despite being much faster, the proposed approach proves to be comparable to SHAP in terms of feature impact evaluation. Also, it brings the same effectiveness to visualization for local interpretability.



(a) SHAP waterfall plot.



(b) SHAP for local importance.

Figure 4.2: SHAP local importance plot : Waterfall and Force plot

4.4.1 AcME

In this work, we propose a new approach to model explainability, AcME[1], aimed at analyzing the role played by each feature at both global and local scale. The importance scores provided by AcME rely on perturbations of the data based on quantiles of the empirical distribution of each feature. These perturbations are performed with respect to a reference point in the input space, which we call *baseline vector* (denoted $\mathbf{x^b}$). First we deal with regression tasks. The procedures exploited to get global and local importance scores, described in Section 4.4.2 and Section 4.4.3, respectively, mainly differ in the choice of the baseline vector and the produced visualizations. In Section 4.4.4, we extend the proposed approach to classification tasks. As supported by the experimental results in Section 4.5, the rationale behind AcME allows for a substantial reduction in computational time, while retaining a quality of explanations comparable to state-of-the-art interpretability method SHAP. Besides, AcME provides very similar visualizations both for global and local interpretability. They are reminiscent of SHAP global importance visualization, but simpler, provided the number of considered quantiles in score evaluation is low. Moreover, local visualization can be used as *what-if* analysis tools to assess how changes in each feature values would impact model prediction for a specific observation.

[1]A Python implementation of the proposed approach is made available at the following link https://github.com/dandolodavid/ACME

4.4.2 Global interpretability for regression tasks

When dealing with global interpretability, we consider the mean vector $\bar{\mathbf{x}}$, i.e. the p-dimensional vector whose components are the mean values of the features (computed over the whole dataset), as the baseline vector $\mathbf{x^b}$. As previously mentioned, the baseline vector represents the point with respect to which perturbations (and corresponding predictions) are computed. To evaluate the importance of each feature $j \in 1, \ldots, p$, we create a variable-quantile matrix $\mathbf{Z}_j \in \mathbb{R}^{Q \times p}$ whose rows are identical to the baseline vector except for the j-th component, which is substituted by the Q quantiles of the empirical distribution of the processed feature j. Notice that the number of rows in $\mathbf{Z}_j$ can be easily tuned by changing the number of selected quantiles Q. By comparing the predictions associated with the rows of the variable quantile matrix $\mathbf{Z}_j$ with the prediction associated with the baseline vector, an importance score representing the relevance of the j-th feature can be computed. Specifically, the whole procedure for the computation of the global importance score for feature j can be summarized as follows:

1. Compute the baseline vector:
$$\mathbf{x^b} = \bar{\mathbf{x}} = [\bar{x}_1, \ldots, \bar{x}_{j-1}, \bar{x}_j, \bar{x}_{j+1}, \ldots, \bar{x}_p]^T. \tag{4.2}$$

2. For each $q \in \{0, 1/(Q-1), 2/(Q-1), \ldots, 1\}$, create the new vector $\mathbf{z}_{j,q} \in \mathbb{R}^p$. This is obtained from $\bar{\mathbf{x}}$ by substituting $\bar{x}_j$ with $x_{j,q}$ i.e., the value of quantile q for the j-th variable:
$$\mathbf{z}_{j,q} = [\bar{x}_1, \ldots, \bar{x}_{j-1}, x_{j,q}, \bar{x}_{j+1}, \ldots, \bar{x}_p]. \tag{4.3}$$
To be more robust, we can avoid the use of quantile 0 and 1, limiting the range of q to an interval that excludes the possible outliers; in this case, an appropriate interval could be for example from $q = 0.1$ and $q = 0.9$. For simplicity, in the rest of the paper we will use the complete range of quantiles, however the procedure remains unchanged.

3. Create the variable-quantile matrix for feature $j \in \{1, \ldots, p\}$:
$$\mathbf{Z_j} = \begin{bmatrix} \mathbf{z}_{j,0} \\ \mathbf{z}_{j,1/(Q-1)} \\ \vdots \\ \mathbf{z}_{j,\,1} \end{bmatrix} = \begin{bmatrix} \bar{x}_1 & \bar{x}_2 & \ldots & x_{j,\,0} & \ldots & \bar{x}_p \\ \bar{x}_1 & \bar{x}_2 & \ldots & x_{j,\,1/(Q-1)} & \ldots & \bar{x}_p \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \bar{x}_1 & \bar{x}_2 & \ldots & x_{j,\,1} & \ldots & \bar{x}_p \end{bmatrix} \tag{4.4}$$

4. Compute predictions associated with the variable-quantile matrix rows:
$$\hat{\mathbf{y}}_j = \begin{bmatrix} \hat{y}_{j,0} \\ \hat{y}_{j,1/(Q-1)} \\ \vdots \\ \hat{y}_{j,1} \end{bmatrix} = \begin{bmatrix} f(\mathbf{z}_{j,0}) \\ f(\mathbf{z}_{j,1/(Q-1)}) \\ \vdots \\ f(\mathbf{z}_{j,1}) \end{bmatrix} \tag{4.5}$$

5. Calculate the standardized effect:

$$\Delta_{j,q} = \frac{\hat{y}_{j,q} - f(\mathbf{x^b})}{\sqrt{var(\hat{\mathbf{y}}_j)}}(max(\hat{\mathbf{y}}_j) - min(\hat{\mathbf{y}}_j)). \tag{4.6}$$

6. The global feature importance score for the generic feature j can be computed by averaging the magnitude of standardized effects over the quantiles:

$$I_j = \frac{1}{Q}\sum_{q=1}^{Q} |\Delta_{j,q}|. \tag{4.7}$$

The first multiplicative factor in Equation (4.6) is similar to the well-known standard score: the objective of this standardization is to achieve scale invariance in the evaluation of the change in prediction caused by using the quantile q (in place of the baseline value). The second multiplicative factor, instead, accounts for the overall impact of the variable in terms of how the applied perturbations spread out predictions. Indeed, it is reasonable that the wider the range of changes in the prediction, the more relevant the variable is for the model.

Notice that for categorical variables, the mode is used in place of the mean value as baseline. Then, the M distinct values the feature could assume the role of quantiles.

AcME has in general lower computational complexity than KernelSHAP. Indeed, AcME only needs to apply the model on $Q \times p$ observations, corresponding to the vectors $\mathbf{z}_{j,q}$. The number of observation N in the dataset that is used to get explanations only affects the computational burden required to calculate the quantiles (for efficient approaches to compute approximate quantiles in large scale data, we refer the reader to [60]). On the other hand, as detailed in Section 4.3, to provide global interpretability, KernelSHAP requires to train a linear model N times, once for each point in the evaluation dataset. First, for each data point x_i, $i \in \{1, \ldots, N\}$, KernelSHAP samples K coalitions. Then, it applies the model that has to be interpreted to each coalition. Finally, KernelSHAP uses the K predictions as a training dataset to fit a local linear model. Notice that, according to the documentation, the choice of K should also depend on p.
The effectiveness of AcME is amplified when explanations of high-dimensional datasets are required in real-time, a scenario in which computationally intensive algorithms are not viable.

In addition, the use of the quantiles enhances robustness against the presence of outliers (not rare in large datasets), and could be a solution to the lack of information in small datasets. In fact, with few observations the estimated density could allow to generate unobserved values, instead of using only observed values permutation.

Results visualization For the visualization of global feature importance scores calculated by AcME, we propose two different kinds of plots. The first

one is simpler, while the second one is more informative.
The former is just a bar plot that shows the feature scores computed according to Equation (4.7), in decreasing order. An example is depicted in Figure 4.3(a). This high-level visualization is designed to figure out quickly which features are the most relevant for the model. Thus, we can use this visualization as a diagnostic tool, for example, when comparing different models.
As for the second visualization, we draw inspiration from SHAP global interpretability plots, since they are concise and effective. An example of AcME global importance detailed score visualization is given in in Figure 4.3(b): on the y axis the features are sorted in decreasing order of importance according to Equation (4.7), while the standardized effects for each element of the variable-quantile matrix are plotted along the x axis. In other words, each horizontal line (associated with a specific feature) represents the standardized effects for the Q perturbations based on quantiles. The color represents the quantile level of the feature from low (marked in blue) to high (marked in red). Moreover the ACME visualization provides a black dashed line, corresponding to the prediction for the base point, to separate positive effects, i.e. those pushing the prediction to higher values, from negative effects, i.e. those pushing the prediction to lower values.

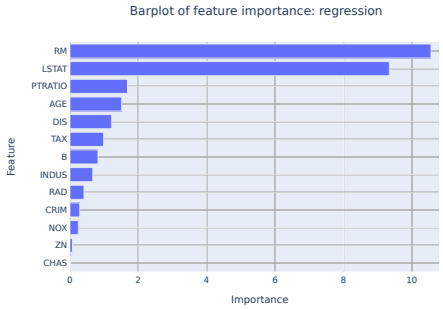
Compared to PDPs visualization (see the example in Figure 4.4), AcME offers a remarkable improvement as it provides a condensed visualization that makes the analysis of results immediate and effective. We recall that the PDP method produces p different plots (one for each feature) and the time and human effort required for the analysis could easily become unsustainable as the dimensionality of data grows. Indeed, the complexity of the visualization (in terms of the number of displayed plots) may collide with the limited human ability to elaborate simultaneous information, that has been proven to be restricted to a set of seven univariate simultaneous stimuli [201].

4.4.3 Local interpretability for regression tasks

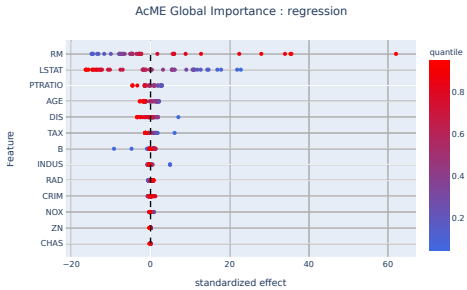
When the scope of the analysis is the interpretation of individual predictions, we set the baseline vector $\mathbf{x^b}$ equal to the specific data point to be explained $\mathbf{x}^*$, instead of setting $\mathbf{x^b} = \bar{\mathbf{x}}$ as in the scenario of global interpretability.

The procedure to get importance scores is similar to that outlined in Section 4.4.2, but in the local case it only serves to order features in the displayed plot, which is meant to convey a different kind of information compared to the global case. The visualization strategy adopted for local interpretability is described in the next paragraph.

Results visualization In AcME, we produce a local interpretability visualization that is reminiscent of its counterpart used for global interpretability, but at the same time we introduce fundamentals modifications aimed at making its interpretation more intuitive and its usage more actionable. Specifically, in the local case we do not display standardized effects but the actual predictions associated with the perturbed data points (based on the selected Q quantiles).



(a) AcME global importance scores



(b) AcME global importance detailed visualization

Figure 4.3: AcME on a regression task (Boston Housing dataset, RF model, see Section 4.5.2 for details)

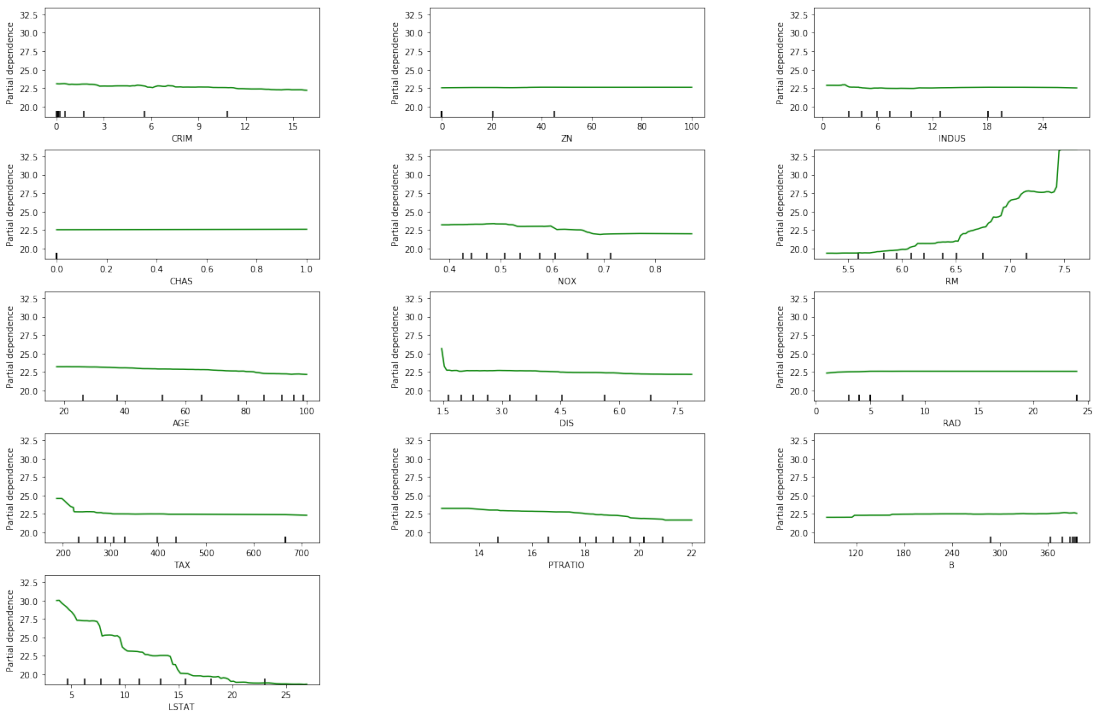


Figure 4.4: PDP calculated for a Random Forest with 100 trees on the Boston Housing dataset (see Section 4.5.2 for details).

Such a design choice allows for immediate understanding since the underlying *what-if* approach is well-aligned with human tendency to reason in counterfactual terms. A dashed line is placed in correspondence of the prediction associated with the original observation $\mathbf{x}^*$, so that it is clear which variables are pushing to increase (or decrease) the prediction. The advantage of AcME visualization for local interpretability, if compared to the SHAP force plot, is that also information about feature distributions is implicitly provided by means of quantile levels. In particular, the knowledge of the quantile corresponding to each feature value in the original observation makes it possible to understand how much the value of a specific feature can be reduced or increased, and how the corresponding prediction will be affected. Figure 4.13 shows an example of local interpretability provided by AcME, which is further detailed in Section 4.5.2.

4.4.4 AcME for classification

In Section 4.4.2 we described AcME for regression tasks. As for classification, by considering the problem of estimating the probability assigned to each class instead of the problem of assigning a label, we can easily resort to a regression task. Thus, the procedure detailed above is still valid. In particular, for global feature importance, we consider, for each class l and each feature j, the standardized effects computed by evaluating the changes in the predicted probability of class l under perturbations of feature j (as usual, perturbations are obtained by replacing the original feature value with the selected Q quantile). The only drawback of this approach is that we should produce as many plots as the number of distinct classes in the dataset, as shown in Figure 4.17. The barplot simplified visualization can circumvent the problem, evolving to a stacked barplot: each feature is assigned a bar partitioned in as many blocks as the number of distinct classes and the length of each block is proportional to the standardized effect associated with the specific class. Figure 4.14 depicts a comparison between this plot and the on provided by SHAP. Similar considerations hold for the local interpretability. For each class, AcME displays the predicted probabilities under perturbations with respect to the original observation feature values (see, for example, Figure 4.15). An application of AcME to a classification task is described in Section 4.5.3.

4.5 Experimental results

In this section, we describe the experiments carried out to assess the effectiveness of AcME. It is worthwhile to notice that interpretability problems are non-supervised tasks, meaning that, in general, there is no ground-truth available to assess feature ranking and importance scores. To overcome this issue, in Section 4.5.1, we resort to synthetic dataset generation that gives us the chance to have a ground truth, so it enables a more objective evaluation. Then, to assess the efficiency of AcME, we compare it with KernelSHAP, a model-agnostic vari-

ant of SHAP, by considering both computational time and explanation quality. For the sake of reproducibility, experiments were conducted on well-known, publicly available ML datasets that are paradigmatic for regression (Section 4.5.2) and classification tasks (Section 4.5.3).

4.5.1 Synthetic datasets

We created two synthetic datasets with controlled characteristics in order to assess the robustness of AcME w.r.t. such conditions. Specifically, we study a regression task and consider the data points generated by the following linear model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{4.8}$$

where $\mathbf{y} = [y_1, \ldots, y_N]^T \in \mathbb{R}^N$ is the vector of responses, $\mathbf{X} \in \mathbb{R}^{N \times p}$ is the design matrix, $\boldsymbol{\beta} \in \mathbb{R}^p$ the vector of model's coefficients and $\boldsymbol{\epsilon} \sim \mathcal{N}_p(\mathbf{0_p}, \boldsymbol{\Sigma})$. The vector of model coefficients $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]^T$ is designed so that some variables have a real effect ($\beta_j \neq 0$), while others are not important ($\beta_j = 0$). In both the experiments detailed below, we generate $N = 200$ observations according to Equation (4.8), then we fit a linear model to the obtained data. We highlight that interpretability approaches are designed to explain model predictions, not existing causal relationships among input features and the output. By considering a data generation process known in advance and by training a model that can learn the generation mechanism, we can gauge the effectiveness of the interpretability approaches. That is why it makes sense to consider also synthetic data in the experiments. In particular, what we expect is that both KernelSHAP (with default parameters) and AcME (with $Q = 50$) succeed to identify the correct feature importance ranking. Notice that this is known in advance, since we can compute it based on vector $\boldsymbol{\beta}$. To certificate the quality of the obtained results, we use Normalized Discounted Cumulative Gain (NDCG) [322]. NDCG sums the true scores ($\boldsymbol{\beta}$) ranked in the order induced by the predicted scores (the AcME features importance) after applying a logarithmic discount. Then it divides by the best possible score (ideal Discounted Cumulative Gain, obtained for a perfect ranking) to obtain a normalized score between 0 and 1. In addition, to test the effectiveness of the model trained on the synthetic data, we used the normalized mean squared error.

Experiment #1: variables with the same scale

In the first experiment, we analyze the simplest scenario, where variables all share the same scale. In particular, the dataset is obtained by setting the following parameters for the linear model in Equation (4.8) and generating $N =$

Experiment	True ranking	AcME ranking	NDCG
1	$[x_2, x_3, x_1, x_5, x_8, x_4, x_6, x_7]$	$[x_2, x_3, x_1, x_5, x_8, x_7, x_6, x_4]$	0.9998
2	$[x_1, x_2, x_3, x_5, x_8, x_4, x_6, x_7]$	$[x_1, x_2, x_3, x_5, x_8, x_7, x_4, x_6]$	0.9998

Table 4.2: NDGC score for Experiment 1 and 2. It is strictly near to 1 in both of them, showing how the ranking produced by AcME is consistent with the real one.

200 observations:

$$\begin{aligned} \boldsymbol{\mu} &= [10, 10, 10, 10, 10, 10, 10, 10]^T \\ \boldsymbol{\Sigma} &= 10 * \mathrm{I}_8 \\ \mathbf{X}_i &\sim \mathcal{N}_8(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \boldsymbol{\beta} &= [10, 20, -10, 0.3, 1, 0, 0, -0.5]^T \\ \boldsymbol{\epsilon}_i &\sim \mathcal{N}_8(\mathbf{0}_8, \boldsymbol{\Sigma}). \end{aligned} \tag{4.9}$$

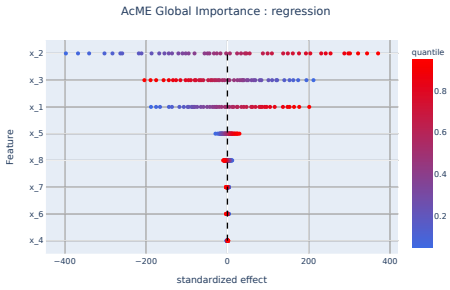
Notice that such a design choice for the vector of model coefficients $\boldsymbol{\beta}$ leads to specific considerations with regard to features ranking: x_1, x_2 and x_3 are the variables that have the largest impact on the response y (x_2 being the most relevant), while x_4, x_6, x_7 and x_8 have negligible or no effect. Moreover, x_3 and x_8 affect the model response in the opposite direction w.r.t. the other variables, resulting in a negative input-output correlation. As can be seen in Figure 4.5, both AcME and KernelSHAP can correctly identify which features are actually important for the fitted model. Also, both methods attribute a negative effect (on the predicted output) to the highest quantiles for features x_3 and x_8, reflecting the fact that $\beta_3 < 0$ and $\beta_8 < 0$. Features x_4, x_6 and x_7 exhibit the lowest feature importance scores, in accordance with the true model specifications. The main difference is in the computational time: AcME requires less than a second, while KernelSHAP requires about 4 minutes in the tested hardware[2]. The NDCG calculated on the feature ranking produced by AcME is reported in Table 4.2. It is very close to 1, showing that the feature ranking returned by AcME is always similar to the expected one, as the model was able to infer the actual underlying data generation process, and AcME could explain how the model maps input to outputs.

Experiment #2: variables with different scale

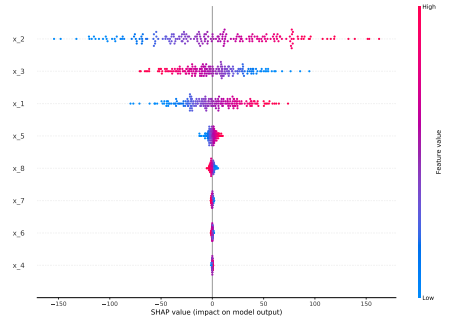
In this experiment, we aim at studying the effect of exploiting variables with different scales. The data-generating model is instantiated as previously, with the exception of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, now set as follows:

$$\begin{aligned} \boldsymbol{\mu} &= [100, 10, 10, 10, 100, 10, 10, 100]^T \\ \boldsymbol{\Sigma} &= diag(100, 10, 10, 10, 100, 10, 10, 100). \end{aligned} \tag{4.10}$$

[2]As a reference, the experiment reported in this work were achieved on a Macbook with CPU i5 2,9 GHz, SSD 256 GB and 8GB RAM.



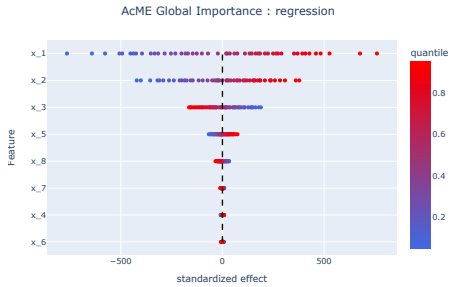
(a) AcME



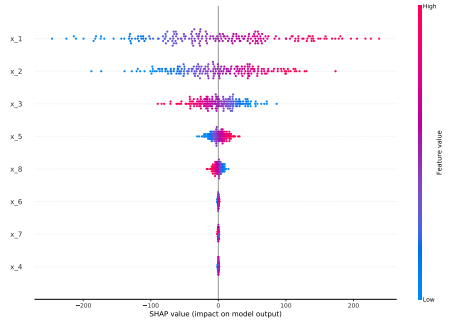
(b) KernelSHAP

Figure 4.5: [Experiment #1] Result comparison of AcME and KernelSHAP. AcME elapsed time: 0.9864 s; KernelSHAP elapsed time: 242.0958 s. Model NMSE : 0.00176

What we expect is that the impact of x_1 will be very higher than before, surpassing x_2 and x_3. Moreover, we expect that the importance of x_8 with $\beta_8 = -0.5$ will be comparable more with the importance of x_4, x_6, x_7, even if the variable receives a great increase both in terms of position and variance. This expected behaviour is reasonable even if the user performs data normalization in the preprocessing step, in fact larger scale variables effects would be captured by an associated larger β, and as demonstrated in the previous experiment, AcME correctly describes that situation. In Figure 4.6, we could see that the proposed approach behaves as expected, supporting the quality of the results. As for experiment #1, the most obvious difference between the two methods is the elapsed time: approximately 1 second for AcME and 4 minutes for KernelSHAP. As happened in the previous experiment, the ranked list metric is near to the upper limit (Table 4.2).



(a) AcME



(b) KernelSHAP

Figure 4.6: [Experiment #2] Result comparison of AcME and KernelSHAP. AcME elapsed time: 0.9450; KernelSHAP elapsed time: 242.1215. Model NMSE: 0.00007

4.5.2 Experiments on a regression task: Boston Housing dataset

The goal of this experiment is to show the results obtained by AcME on a regression task, and to compare them with those achieved by KernelSHAP. To this aim, we consider a well-known publicly available dataset, UCI Boston Housing dataset[3]. The Boston Housing dataset is composed of 506 observations with 14 variables describing houses in the area of Boston Mass. The task is predicting the median value of owner-occupied homes.
KernelSHAP and AcME are compared in terms of both evaluated feature importance and computational time.

Efficiency comparison between KernelSHAP and AcME

As seen in Section 4.3, what makes *KernelSHAP* so computationally burdensome is:

1. the high number of linear models that are estimated;

[3]https://archive.ics.uci.edu/ml/machine-learning-databases/housing

	Number of samples	Elapsed Time (in seconds)
AcME	complete	0.36
KernelSHAP	5	357.23
KernelSHAP	10	425.61
KernelSHAP	20	875.85
KernelSHAP	100	1855.65

Table 4.3: [Boston Housing Dataset] Elapsed time for KernelSHAP with different dataset sampled.

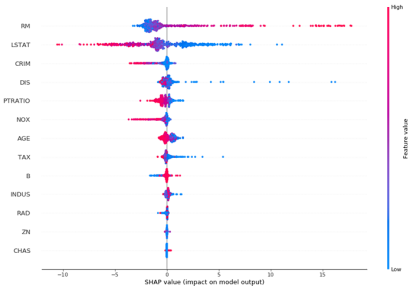
$\widetilde{N}$ (# samples)	Ranking	Kendall Tau (full)	Kendall Tau (top 5)
5	RM,LSTAT,CRIM,DIS,PTRATIO,NOX,AGE,TAX,B,INDUS,RAD,ZN,CHAS	0.231	0.2
10	RM,LSTAT,CRIM,PTRATIO,DIS,NOX,AGE,TAX,B,INDUS,RAD,CHAS,ZN	0.692	0.0
20	TAX,AGE,B,ZN,LSTAT,INDUS,CRIM,RAD,PTRATIO,RM,DIS,CHAS,NOX	0.103	-0.2
100	LSTAT,RM,CRIM,DIS,PTRATIO,NOX,AGE,TAX,B,INDUS,RAD,ZN,CHAS	0.359	0.8
complete	LSTAT,RM,DIS,NOX,PTRATIO,CRIM,AGE,TAX,B,INDUS,RAD,CHAS,ZN	1	1

Table 4.4: [Boston Housing Dataset] Kendall Tau score obtained with the comparison of KernelSHAP ranking list with the full dataset with KernelSHAP rankings obtained using only 5,10,20,100 samples from the dataset.

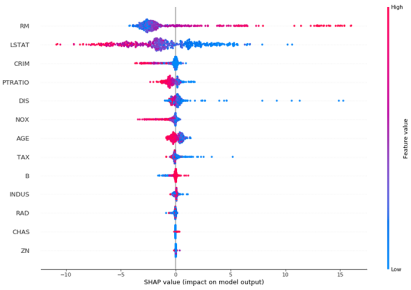
2. the computational complexity of the matrix inversion, that increases with the number of coalitions.

To overcome the first problematic, the authors of [180] suggest to reduce the number of rows in the dataset, sampling $\widetilde{N}$ rows from the original ones. For the second issue, the only option is to reduce the number of coalitions K. In this section, we will explore these solutions, evaluating the quality of results with respect to changes in parameters $\widetilde{N}$ and K in terms of their similarity to results obtained with the complete original dataset and the suggested number of coalitions. Our experiments suggest that using dataset sampling and coalitions reduction to lessen the computational time required by KernelSHAP may translate into unreliable results. Moreover, elapsed time for KernelSHAP is still much higher than that required by AcME.

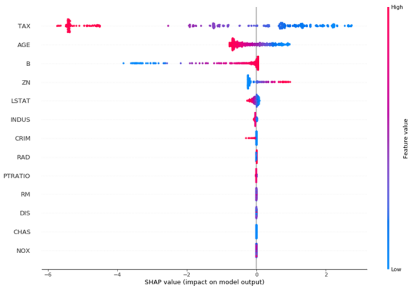
Reducing KernelSHAP computational burden by sampling observations We compute KernelSHAP by sampling 5, 10, 20, and 100 rows at random from the original dataset, keeping the default number of coalitions. In Table 4.3, we show the computational time of AcME run on full dataset compared to that required by KernelSHAP when considering the sampled rows only. It is clear that AcME is always much faster. In addition, even though the computational time required by the sampled KernelSHAP is much lower than the original version when using lower values of $\widetilde{N}$, the output is not reliable, as depicted in Figure 4.7. For instance, when we use $\widetilde{N} = 20$, not only the feature order based on computed importance is different, but also the model behaviour is not correctly detected: for example the Shapley values of `TAX, RM, AGE` are completely different from all the other cases. The instability in the results is caused by the sampling procedure on the original dataset, that does not extract rows that are



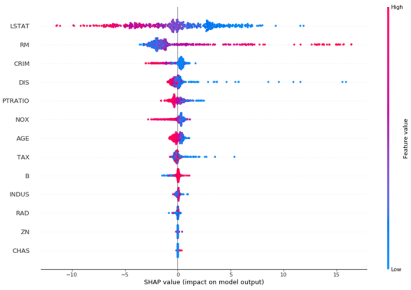
(a) SHAP with sampling $g = 5$ rows



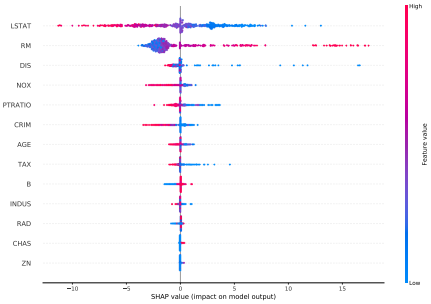
(b) SHAP with sampling $g = 10$ rows



(c) SHAP with sampling $g = 20$ rows



(d) SHAP with sampling $g = 100$ rows



(e) SHAP calculated on the full dataset

Figure 4.7: [Boston Housing Dataset] KernelSHAP summary plot for the Random Forest model with 100 tree, using a sample in order to reduce the number of rows of the input dataset. As clear from the plots, the results are not stable in term of feature behaviour and importance compared to the results obtained using the full dataset.

	Number of coalitions K	Elapsed Time (in seconds)
AcME	complete	0.36
KernelSHAP	10	92.40
KernelSHAP	25	221.61
KernelSHAP	50	327.65
KernelSHAP	100	506.31

Table 4.5: [Boston Housing Dataset] Elapsed time for KernelSHAP with different number of coalitions used in the estimation of the Shapley values.

truly representative of the dataset.

To measure the uncertainty in the ranking induced by dataset sampling, we use Kendall's Tau metric, a measure of the correspondence between rankings; being a rank correlation coefficient, it will be high for two lists with similar ranking (with the maximum in 1 when perfectly identical), whereas it will be low for two lists very differently sorted (with the minimum in -1 when all position pairs are discordant). In Table 4.4 Kendall's Tau values are reported for each ranking generated by KernelSHAP with different number of samples, compared with KernelSHAP ranking obtained with the full dataset, both for entire lists and for the top five elements of each list. While the correlation values for the full lists are strictly positive, the same doesn't happen by considering the top five lists. In addition, it should be noted that there isn't a monotonous increasing trend: in fact, worst results are obtained with $g = 20$. Both KernelSHAP and AcME use perturbations of the input dataset to obtain model explanation. However, as for KernelSHAP, analysing the similarity among the rankings corresponding to different sampling strategies suggests that sampling may have a detrimental effect on the quality of explanations, especially when features distribution is not trivial. Instead, AcME relies on quantiles: the estimation of these statistics is robust to outliers, while the sampling strategy implemented by SHAP seems to be not, in general.

With $\widetilde{N} = 100$ the results appear similar to those on the entire dataset, however the elapsed time is already significantly high.

In this experiment, we extracted rows at random. Alternatively, smarter sampling techniques could be used: for instance, a suggestion could be the usage of clustering to detect groups of observations to sample from. However, the use of clustering, and more generally of others advanced sampling techniques, requires the tuning of hyper parameters, like the number of clusters. The correct choice of the parameters is relevant because it could heavily affect the quality of the results, and it needs *ad hoc* analysis. For instance, it is not possible to determine a rule-of-thumb to suggest the correct number of lines to sample, because this parameter is strongly linked to the intrinsic complexity of the dataset.

Reducing KernelSHAP computational burden by reducing the number of coalitions In Table 4.5, we report how the computational time required by the KernelSHAP procedure varies with the number of sampled coali-

K (# coalitions)	Ranking	Kendall Tau (full)	Kendall Tau (top 5)
10	LSTAT,RM,ZN,CRIM,INDUS,CHAS,NOX,DIS,AGE,PTRATIO,TAX,RAD,B	-0.05	-0.2
25	LSTAT,RM,DIS,CRIM,NOX,PTRATIO,AGE,B,TAX,INDUS,ZN,CHAS,RAD	0.231	0.06
50	LSTAT,RM,CRIM,DIS,PTRATIO,NOX,AGE,TAX,B,INDUS,RAD,ZN,CHAS	0.359	0.8
100	LSTAT,RM,DIS,CRIM,PTRATIO,NOX,AGE,TAX,B,INDUS,RAD,CHAS,ZN	0.821	0.6
default	LSTAT,RM,DIS,NOX,PTRATIO,CRIM,AGE,TAX,B,INDUS,RAD,CHAS,ZN	1	1

Table 4.6: [Boston Housing Dataset] Kendall Tau score obtained with the comparison of KernelSHAP ranking list with the default number of coalitions and KernelSHAP rankings obtained using only 10,25,50,100 coalitions.

tions, by considering 10, 20, 50 and 100 coalitions, using the entire dataset. As previously stated, reducing the number of sampled coalitions can really speed up the computation, but the results are once again unstable. Besides, results exhibit higher variance when the number of considered coalitions is lower. In Table 4.6 we reported the Kendall's Tau measure calculated on the ranked lists obtained reducing the number of coalitions and the ranked list obtained using the default number of coalitions. This is strictly related to the approximation used in the computation of the Shapley values, which uses a lower number of permutations to estimate the true values. Again, feature distribution complexity is an important aspect: with simple distribution a lower number of permutations could already lead to decent results, but with more complex distributions this is not always true.

Feature ranking evaluation and comparison between KernelSHAP and AcME

To study the importance scores assigned to each feature by AcME and KernelSHAP (calculated on the full dataset with default parameters), we fit three different types of models: *Linear Regression*, *Random Forest* and *CatBoost Regression*. To evaluate the generated feature ranking, we estimated other two different models for each model type. We trained the former using only the five features with the highest importance according to AcME. Instead, the latter considers all other features only. We expect that the first model will have a Mean Squared Error (MSE) closer to the value obtained using all the features, whereas the MSE for the second model will be much higher. There is no ground truth that we can use for a fair comparison. However, this procedure allows us to assess the interpretability performance quantitatively. As detailed in Table 4.7, experimental results reflect the expectations, confirming that AcME correctly identified the features that are most relevant for the model.

Table 4.8 reports the computing time for AcME and KernelSHAP, while Figure 4.9 - Figure 4.12 depict the results of the two methods. In Table 4.8, we also report the elapsed time using the TreeSHAP procedure, which is much faster than the KernelSHAP on the same model, but in this case, is usable only with *CatBoost* and *Random Forest*. In all four situations, the results are similar both in terms of importance scores rank and feature-to-output map behaviour, but the computing time is much lower for AcME, which turns out to be extremely

Model	MSE	Top 5 features	All features except top 5
Linear Regression	21.89	26.01	55.15
Random Forest	1.54	2.40	4.42
CatBoost	0.52	0.86	3.80
XGBoost	2.37	4.09	14.54
LightGBM	1.54	2.66	13.37

Table 4.7: [Boston Housing Dataset] Mean Square Error for the 3 model used for AcME ranking evaluation. The best model here is the CatBoost, followed by the Random Forest.

Method	Model	Elapsed Time (in seconds)
KernelSHAP	Linear Regression	3651.8556
KernelSHAP	Random Forest	5639.9273
KernelSHAP	CatBoost Regression	4578.0989
KernelSHAP	XGBoost	1938.70
AcME	Linear Regression	0.2610
AcME	Random Forest	0.4107
AcME	CatBoost Regression	0.9764
AcME	XGBoost	0.24
TreeSHAP	Random Forest	4.7886
TreeSHAP	CatBoost Regression	3.1525

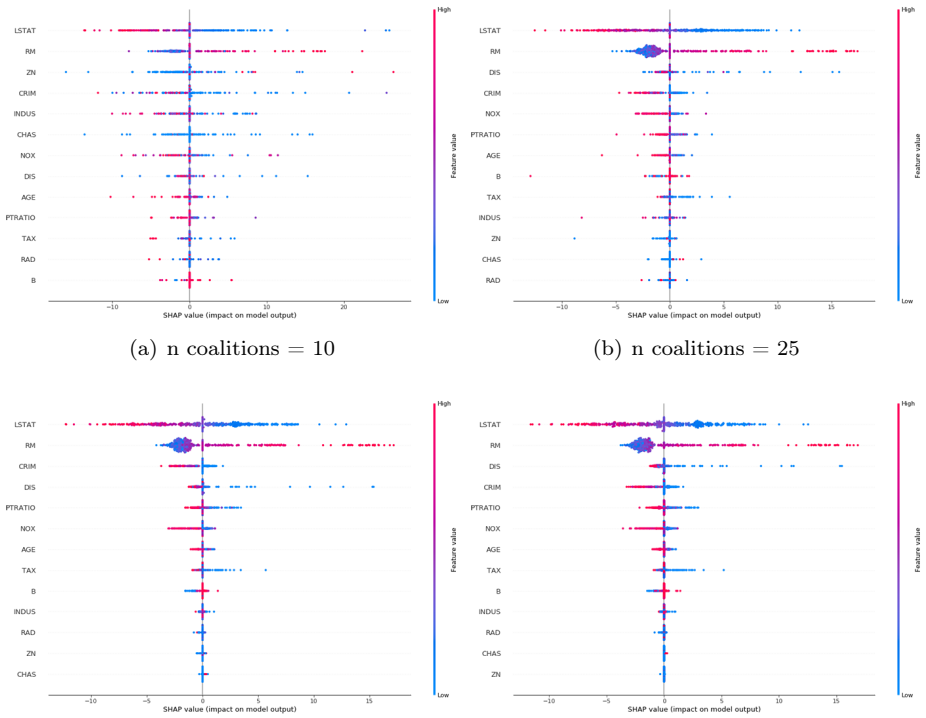
Table 4.8: [Boston Housing Dataset] Computing time for the various tested models and the different methods of model explanation on the Boston dataset.

faster than each version of SHAP. In particular:

- Linear Regression: both AcME and KernelSHAP recognise `LSTAT` as the most important variable, and both methods mostly agree on how the model uses the variables. The most important difference is in the importance score for RAD: while KernelSHAP ranks it at the third place, our method put RAD at sixth place. This happens for construction of the $\Delta_{j,k}$ (Equation (4.6)), because we give greater importance to the variables with high impact on predictions range. In fact, instead of `RAD`, AcME prefers `RM` that has the bigger variation among all variables.

- Random Forest: here the two methods agree on the first two variables regarding importance. Then, KernelSHAP gives approximately the same importance to `DIS`, `NOX`, `PTRATIO`, `CRIM`, `AGE` and `TAX`, while AcME gives higher importance to `DIS`. This happens because, for very low quantiles of `DIS`, the prediction is pushed to very high values and this does not occur for the other variables, as shown in Figure 4.10(a) and Figure 4.10(b).

- CatBoost Regression: this is the best model in terms of MSE. In this case, as in the Random Forest, the only relevant difference between AcME and KernelSHAP is the impact of `DIS`, that AcME considers larger than what KernelSHAP does. Besides, the other importance scores and the model's behaviour explanations given by the two methods are very similar.



(a) n coalitions = 10

(b) n coalitions = 25

(c) n coalitions = 50

(d) n coalitions = 100

Figure 4.8: [Boston Housing Dataset] KernelSHAP summary plot for the Random Forest model using a lower number of coalitions of the model to accelerate the procedure.

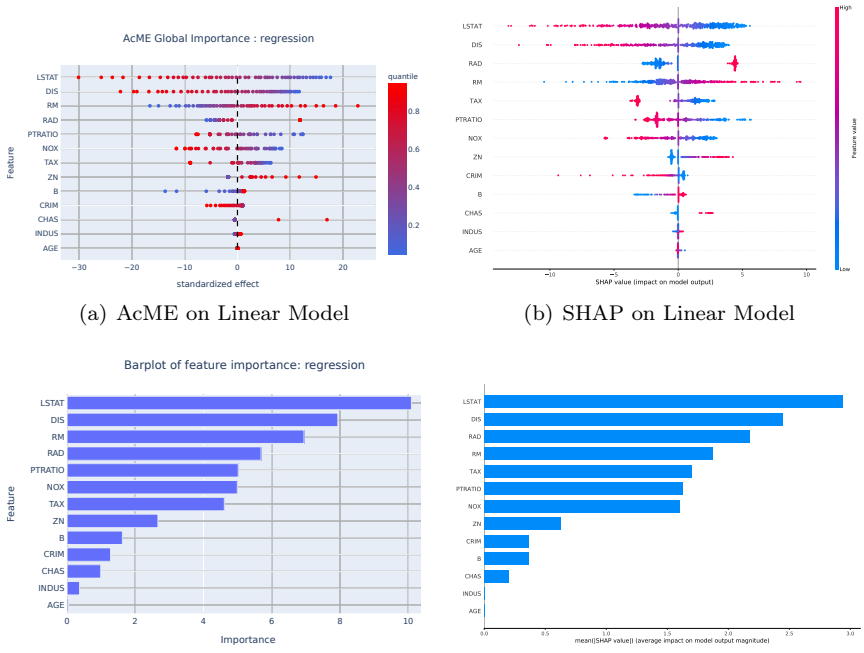
- XGBoost: once again, the results of KernalSHAP and AcME are very similar. There are minor differences in ranking of medium-low importance variables, the only relevant difference between AcME and KernelSHAP is the impact of `DIS`, that AcME considers lower than what KernelSHAP does.

Additional experiments with other real-world datasets are reported in the Appendix.

Local interpretability

Figure 4.13 depicts local interpretability results for observation with $ID = 200$, when considering a Random Forest model. From the plot, we can understand how each feature impacts prediction and explore how variations in input values can affect estimates using a *what-if* approach:

- variable `RM` is the number of rooms, it has a high value (big red bubble),



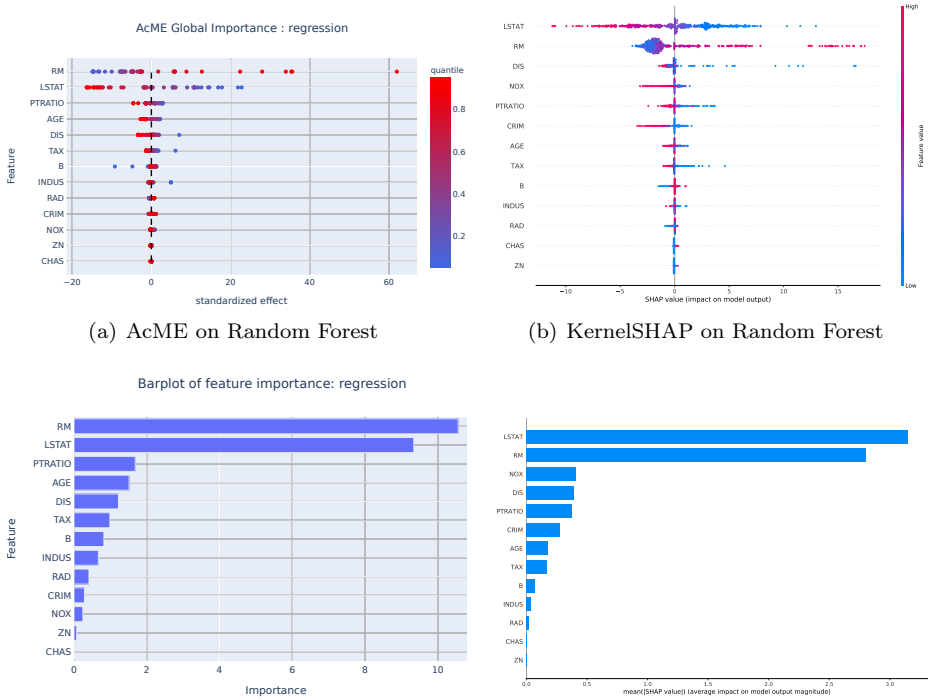
(a) AcME on Linear Model

(b) SHAP on Linear Model

(c) AcME Feature Importance

(d) SHAP Feature Importance

Figure 4.9: [Boston Housing Dataset] LINEAR REGRESSION: Result comparison of KernelSHAP and AcME.



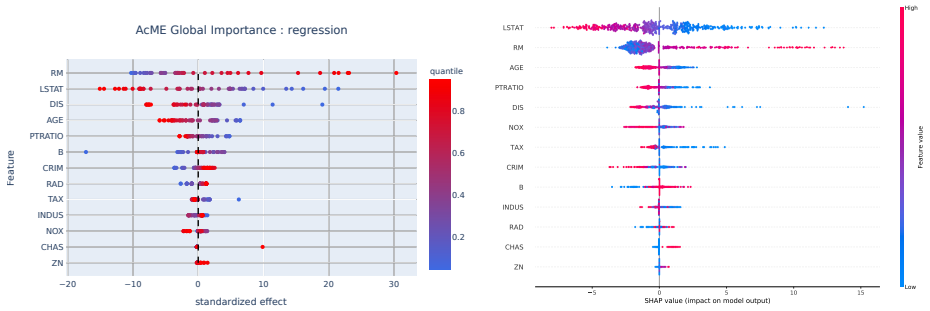
(a) AcME on Random Forest

(b) KernelSHAP on Random Forest

(c) AcME Feature Importance

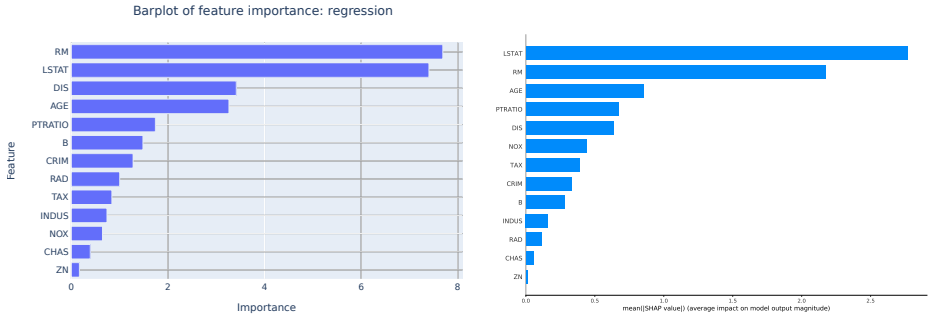
(d) KernelSHAP Feature Importance

Figure 4.10: [Boston Housing Dataset] RANDOM FOREST: Result comparison of KernelSHAP and AcME



(a) AcME on Catboost Regression

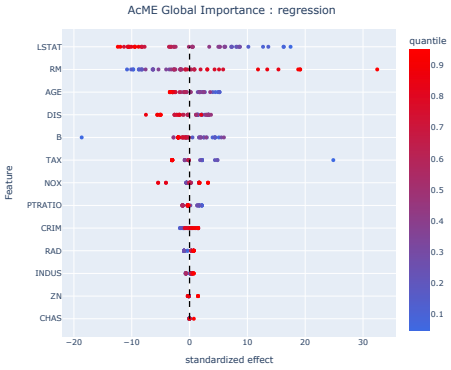
(b) SHAP on Catboost Regression



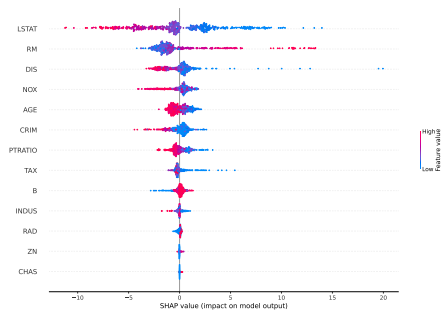
(c) AcME Feature Importance

(d) SHAP Feature Importance

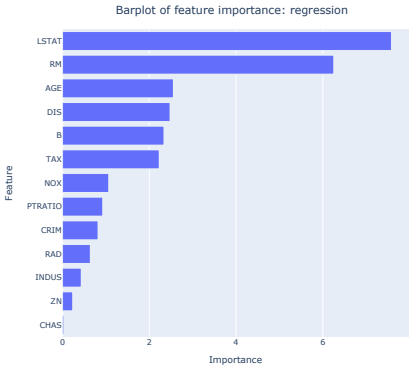
Figure 4.11: [Boston Housing Dataset] CatBoost REGRESSION: Result comparison of KernelSHAP and AcME.



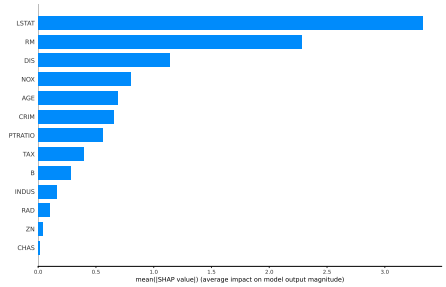
(a) AcME on XGBoost



(b) SHAP on XGBoost



(c) AcME Feature Importance



(d) SHAP Feature Importance

Figure 4.12: [Boston Housing Dataset] XGBoost: Result comparison of KernelSHAP and AcME.

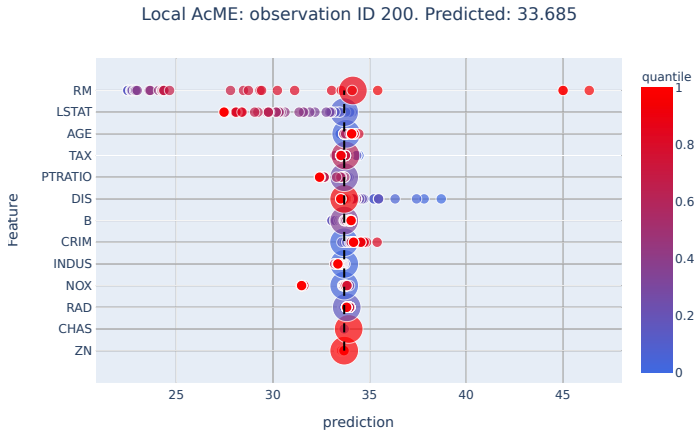


Figure 4.13: AcME result plot for single observation. The underlying model is a Random Forest with 100 tree, trained on the Boston Housing dataset.

and this increases the estimated value of the house. We could easily see from the plot that a house with the same characteristics but with a lower number of rooms will have a lower price, going from actual 33.685 to less than 25;

- variable `LSTAT` represent the proportion of the population that is of lower status (low education or low income). The estimated value is higher due to the low value of this feature. It can be clearly seen that, as the value of this proportion increases, the estimated price lowers considerably.

- variable `DIS` is the weighted distance to five Boston employment centres, and for this house it is near to the highest encountered in the dataset. For the model, this is a negative factor that reduces the house estimated value, and as we can see, a house with the same characteristics could have a much higher value by reducing the distance: from the actual value of 33.685 it could go near to 38-39.

As for most perturbation-based interpretability methods, it is possible that new data points are created by AcME in the local interpretability procedure. This can be problematic since new data points might be very different from the ones provided during model training. Nevertheless, the proposed approach to local interpretability is still a relevant contribution. In particular, (i) local AcME is suitable in the scenario of Decision Support Systems, where other model-agnostic approaches, such as SHAP, are not viable due to the high computational cost that is not acceptable for the application; (ii) it allows for similar visualizations both for local and global interpretability, which cannot be said for all interpretability methods.

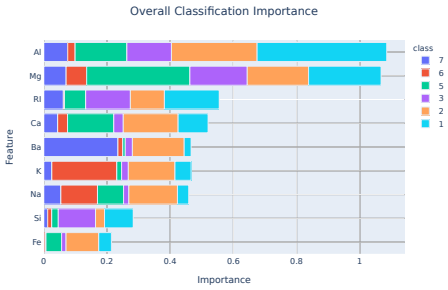
4.5.3 Experiments on a classification task: Glass dataset

This experiment aims at evaluating the performance of AcME on a classification task. To this aim we consider a well-known publicly available dataset, Glass[4]. The goal is to classify six different types of glass based on their chemical features. To evaluate the model accuracy, we split the dataframe in train and test with 70%-30% proportion. The model used to resolve the multiclass classification problem is CatBoost classifier, and the accuracy obtained is 0.84375. Then, we compare AcME with KernelSHAP. From Figure 4.14 we cans see that AcME (with $K = 20$) and KernelSHAP provide similar explanations about global importance for input features. However, while AcME runs in 0.58 seconds, it takes 186.5 seconds to obtain the results from KernelSHAP.

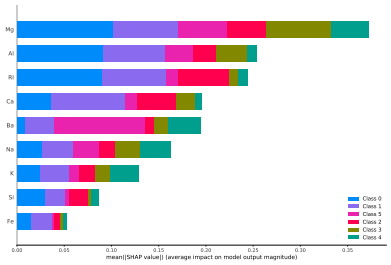
In Figure 4.17, instead, we can see how the detailed visualizations provided by AcME can be adapted to multi-class classification tasks by considering each class predicted probability, in the same vein as SHAP.

Finally, in Figure 4.15 we show how local interpretability can be used as a *what-if* analysis tool. For example, we see that observation with $ID = 100$ has probability 0.21 of belonging to class 1, according to the CatBoost model. Everything else remaining fixed, this probability would grow over 0.5 if we decrease the aluminium content (Al) from quantile 0.55 (bigger bubble corresponding to the current observation value) to quantile 0.47, corresponding to the rightmost dot first row of the chart. Instead, if we reduced Refractive Index (RI) from current value (quantile 0.59) to quantile 0.33, the resulting class 1 probability would be 0.04.

[4]https://archive.ics.uci.edu/ml/datasets/glass+identification



(a) AcME Feature Importance



(b) KernelSHAP Feature Importance

Figure 4.14: [Glass dataset] Comparison of feature importance provided by KernelSHAP and AcME on Catboost. Classifier with accuracy: 0.84375

4.5.4 Experiments on a classification task with Neural Network

After studying the behavior of AcME with classification problems, we proceed to its use with a more complex and, certainly, not self-explanatory model: a Multi-layer Perceptron classifier. We apply the model to the same dataset of Section 4.5.3, with the same train test split setup. The number of hidden layer chosen is 4, with sizes $9, 15, 15$ and 6 respectively, with an obtained accuracy of 0.6094. The results of AcME are once again very similar to the results of KernelSHAP (Figure 4.18). In this case, the order of importance of the features is exactly the same, and the observable effect on the various classes is also quite comparable. Nevertheless, the computational times of the two methods are extremely different: AcME complete the estimation in 0.54 seconds while KernelSHAP requires 104.85 seconds.

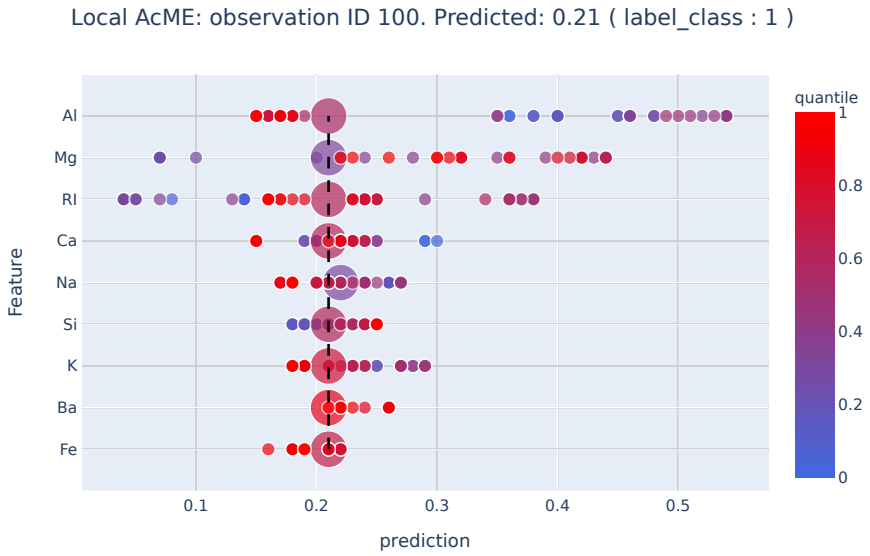
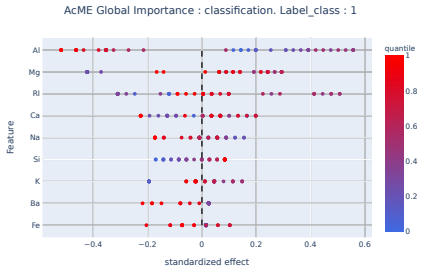
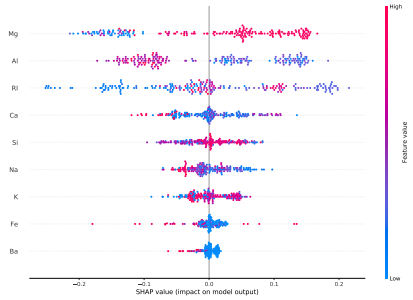


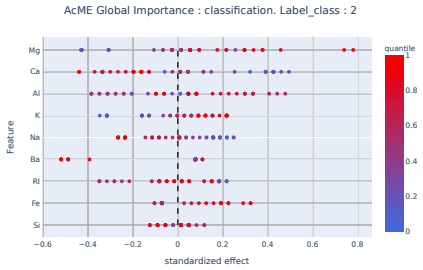
Figure 4.15: AcME local importance scores visualization for Glass classification task.



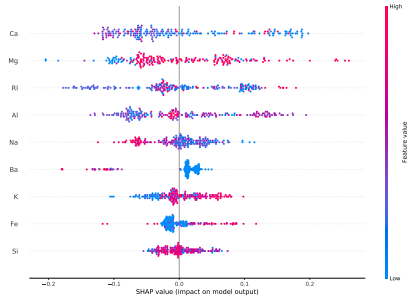
(a) AcME on the Glass dataset: glass type 1



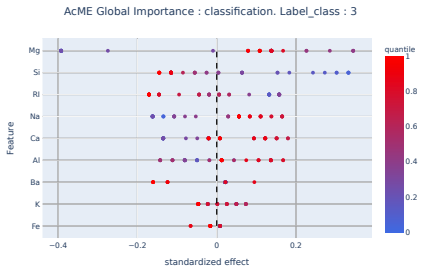
(b) SHAP on the Glass dataset: glass type 1



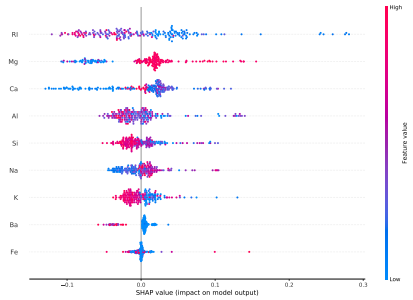
(c) AcME on the Glass dataset: glass type 2



(d) AcME on the Glass dataset: glass type 2

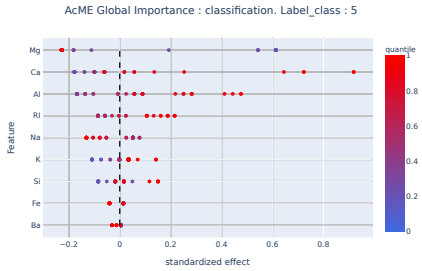


(e) AcME on the Glass dataset: glass type 3

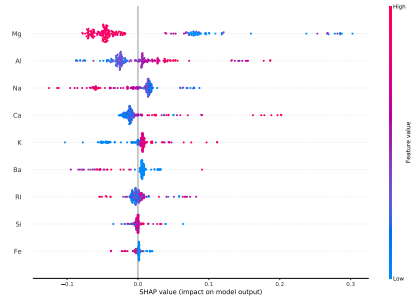


(f) AcME on the Glass dataset: glass type 3

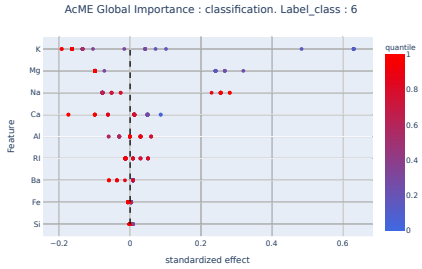
Figure 4.16: [Glass dataset]: Feature effect according to AcME (left) and KernelSHAP (right) for glass type 1, 2 and 3.



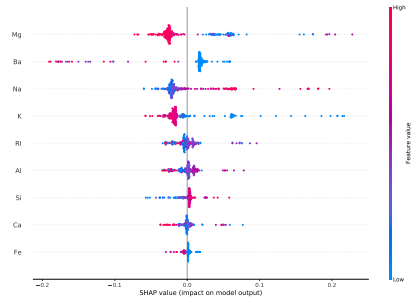
(a) AcME on the Glass dataset: glass type 5



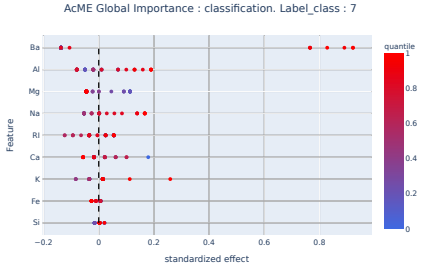
(b) SHAP on the Glass dataset: glass type 5



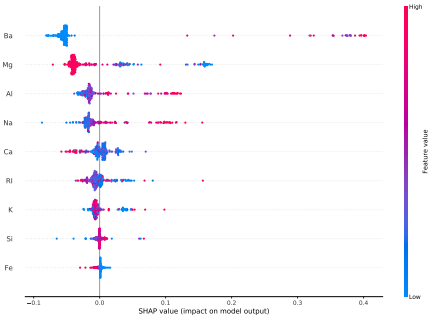
(c) AcME on the Glass dataset: glass type 6



(d) SHAP on the Glass dataset: glass type 6

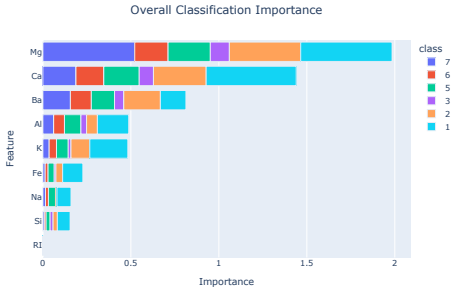


(e) AcME on the Glass dataset: glass type 7

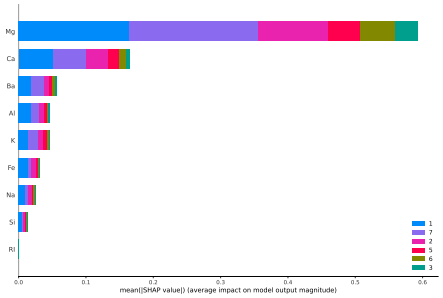


(f) SHAP on the Glass dataset: glass type 7

Figure 4.17: [Glass dataset]: Feature effect according to AcME (left) and KernelSHAP (right) for glass type 5, 6 and 7.



(a) AcME Feature Importance



(b) KernelSHAP Feature Importance

Figure 4.18: [Glass dataset] Comparison of feature importance provided by KernelSHAP and AcME for MLP. Classifier with accuracy: 0.6094.

Chapter 5

Replay for Multilabel Setting

In this chapter, we will consider the research areas of Alarm Forecasting (subfield of Predictive Maintenace), Multi-label Classification, and Continual Learning. The following research is motivated by the industrial context of the packaging industry. Alarm Forecasting aims to predict alarms in industrial equipment as a low-cost alternative to sensor-based predictive maintenance. We formalize the problem as a multi-label classification problem and propose as solution FORMULA [228] . A deep learning method that uses the Transformer and the Weighted Focal Loss to handle imbalanced data and accurately predict rare but essential alarms. The approach is shown to be effective using real-world data from the packaging industry. To make the approach more feasible in a real-world scenario, we extend the work considering a Continual Learning setting, where new data equipment arrives over time [228]. We also propose a novel approach for multi-label classification in continual learning, for which an efficient approach with logarithmic complexity in the number of tasks is proposed.

5.1 Multilabel

5.1.1 Introduction

Classification is one of the most important machine learning topics [1] . The goal of classification is to train a computational model using a set of labeled samples, and obtain a model that can correctly classify new unlabeled samples. Traditional single-label classification is one of the most well-established machine learning paradigms. Binary and multi-class classifications are subcategories of single-label classification that concern learning from a set of samples that are associated with a single label. Unlike traditional classification, multi-label classification (MLC) assigns a set of relevant labels to an instance simultaneously [292]. Many challenging applications, such as image or video annotation, web page categorization, gene function prediction, and language modeling, can benefit from being formulated as multi-label classification tasks with a large set of

labels [172].

5.1.2 Applications

Multi-label classification assigns multiple labels for each instance simultaneously. This problem has relevance in a variety of fields ranging from protein function classification and document classification, to automatic image categorization [172]. During the past decade, multi-label classification has been successfully applied in computer vision, natural language processing, and data mining [172]. For example, in bioinformatics, one gene sequence can be associated with a set of multiple molecular functions [351], the same gene can belong to the functions of Protein Synthesis, Metabolism and Transcription. In text categorization, a new document can cover multiple topics such as News, Finance, and Sport, so the same sample has associated multiple topics [77]. Other possible applications are medical diagnosis, music categorization and emotion recognition [366]. In computer vision, many natural images usually contain multiple objects. Therefore, it is more practical that each image is associated with multiple labels. For example, an image with the landscape of a mountain can contain multiple objects/labels such as clouds, sky, water, etc. Thus developing deep learning techniques that can address MLC problem is a practical and significant problem in real-world scenarios. Another example is the development of a platform for videos or images like Youtube, Instagram, and Facebook. Efficient and effective indexing and searching for the video and images become more and more important for the research and industry community [172]. In the search industry, revenue comes from clicks on ads embedded in the result pages. Advertising selection and placement can be significantly improved if ads are tagged correctly [172]. Eventually, the recommender systems can be naturally regarded as an MLC task, since we usually recommend multiple items simultaneously to the users.

5.1.3 Challenges

When dealing with Multi-label Classification, we need to keep in mind that there are several challenging aspects. The first challenge is the potentially overwhelming size of the output space since the number of label sets grows exponentially as the number of class labels increases [365]. In other words, the possible combination of labels in the output space grows exponentially with the number of labels. For example, in an output space with 100 items to recommend to a user, there are 2^100 possible combinations of how the model can suggest the items to the user. Moreover, the labels are extremely sparse, leading to the long-tail distribution problem. The unequal label distribution, or imbalance among label frequencies, is present in most multi-labeled datasets [52] and makes the prediction challenging.

Some works [171] have shown that methods that explicitly capture label dependency usually achieve better prediction performance. Therefore, modeling the label dependency is one of the major challenges in multi-label classification

problems. Many methods have been motivated to model the dependence, like the classifier chain (CC) model [243]. It captures label dependency by using binary label predictions as extra input attributes for the following classifiers in a chain. Another example is CPLST [59], which uses principal component analysis to capture both the label and the feature dependencies.
Another important challenge is that current offline MLC methods assume that all data are available in advance for learning. In practice, data is collected sequentially, and data collected earlier in this process may expire as time passes. Existing off-line MLC algorithms are impractical for streaming data sets since they require storing all data sets in memory [172]. Additionally, it is non-trivial to adapt offline multi-label methods to the sequential data.

5.1.4 Evaluation metrics

In a multi-label dataset, it is useful to obtain information of the complexity of the dataset. For example, based on the long tail distribution or the correlation among labels, the dataset could be more or less complex to solve. To evaluate the complexity of a multi-label dataset, several indicators can be utilized. We present in the following text some of the most common metrics used in the field:

- **Label Cardinality**. Let's consider the multi-label dataset D. The most natural way to measure the degree of multi-labeledness is *label cardinality*, i.e. the average number of labels per sample:

$$\text{Card}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i| \tag{5.1}$$

  where Y_i are the labels of sample i.

- **Label Density**: it is similar to label cardinality in which the value is normalized:

$$\text{Dens}(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i|}{|L|} \tag{5.2}$$

  where L is the label set.

- **Label Diversity**. Another popular measure is *label diversity*, i.e. the number of distinct label subsets appearing in the dataset:

$$LDiv(D) = |\{y : \exists\, x : (x, y) \in D\}|. \tag{5.3}$$

- **IR per label (IRLbl)**: An important metric which measures the level of imbalance of a certain label is the imbalance ratio per label [51], which is defined as follows:

$$\text{IRLbl}(y) = \frac{\max_{y' \in L}\left(\sum_{i=1}^{|D|} h\left(y', Y_i\right)\right)}{\sum_{i=1}^{|D|} h\left(y, Y_i\right) 16} h\left(y, Y_i\right) = \begin{cases} 1 & y \in Y_i \\ 0 & y \notin Y_i \end{cases} \tag{5.4}$$

It refers to the difference in the number of instances that belong to each label. When the imbalance ratio is high, it means that some labels are much more common in the dataset than others.

- **MeanIR**: It is an aggregate metric of IRLbl, it average the IRLbl value of each label:

$$\text{MeanIR} = \frac{1}{|L|} \sum_{y \in L} \text{IRLbl}(y) \tag{5.5}$$

A value near to 1 means that the dataset is balanced. The higher the value, the higher the level of imbalance across the labels in the dataset.

5.2 Alarm Forecasting

5.2.1 Motivation

Predictive Maintenance technologies are particularly appealing for Industrial Equipment producers, as they pave the way to the selling of high added-value services and customized maintenance plans. However, standard Predictive Maintenance approaches assume the availability of sensor measurements, and the costs associated with adding sensors or remotely accessing sensor readings may discourage the development of such technologies. In this context, Alarm Forecasting can be very useful as it represents a low-cost alternative or helpful support to sensor-based Predictive Maintenance.
In the manufacturing industry, logs of the alarms generated by industrial equipment represent a valuable source of information. Machines raise alarms to communicate the occurrence of particular events, possibly related to issues or anomalies, that can be representative of the health state of the equipment and some of its components [91, 161, 286, 355]. Machines may also send sensor measurements or other variables about the operating conditions, which could enable Predictive Maintenance (PdM) technologies to prevent faults, reduce down-time and optimize service policy based on equipment health status estimation.
However, often implementing PdM based on sensor readings is not completely feasible yet. Indeed, the lack of data integration, the possibly unaffordable transmission of sensor measurements over the cloud, or the presence of legacy machines not equipped for sensory data acquisition may hinder the adoption of PdM methodologies.
Conversely, alarm logs generated by the equipment are a valuable source of information and they are available also for legacy equipment, and the logging mechanism is reliable and meticulously maintained because it feeds standard industrial Supervisory Control And Data Acquisition (SCADA) software [316] and Human–Machine Interfaces (HMIs) used by human operators to interact with machines. Thus, alarm or event-log analysis can be a low-cost alternative or helpful support to sensor-based PdM and can be applied to a wide variety of application domains.

Some alarm analysis techniques such as correlation analysis and chattering reduction can provide insight to operators that need to take action on the equipment. Alarms are also used to perform Anomaly Detection (AD), but sometimes the prediction horizon is too short for operators to perform the required corrective actions. On the other hand, if the incoming alarms can be predicted in advance, the operators can manage potential faults by taking corrective action in time. This task, named Alarm Forecasting (AF). It plays an essential role in the safe management of process operations and enables PdM [287] to maximize machine parts exploitation and to minimize the Total Cost of Ownership (TCO) [41, 114, 354, 91].

5.2.2 Related work

Alarm data analysis aims at distilling information provided by alarm logs into actionable insight for operators. This research field encompasses tasks such as alarm correlation analysis [339], alarm floods monitoring [7], and nuisance alarms suppression [82]. In particular, alarm analysis methods can be divided into diagnostic and prognostic approaches. The former ones aim at detecting the occurrence of anomalies or faults, while the latter ones have the goal of predicting future equipment behaviour to provide timely actionable insight to users.

As for diagnostic tools, one of their most frequent applications encountered in the literature is AD. For the nature of the data at hand, researchers and practitioners have resorted in this area to data-driven AD approaches [81] instead of model-based ones. In [346], for instance, authors propose a system to detect suspicious patterns potentially related to security incidents. In [84], a deep neural network (DNN) model based on Long Short-Term Memory (LSTM) units is proposed to model a system log as a natural language sequence, aiming at automatically recognizing abnormal behaviours and supporting root cause analysis. Another diagnostic tool enabled by the availability of system logs is Fault Detection and Classification (FDC) [92]. In [18], for instance, authors proposed a data-driven prediction model that leverages *ad-hoc* feature engineering and gradient boosting methods to predict Trouble Ticket based on alarm streams.

About prognostic tools, AF, the focus of this work, plays a key role. In [338], for instance, historical alarm sequences are exploited using bayesian estimators. In [369], a probabilistic model based on an N-gram model is proposed to predict the probability of alarm occurrence, given the previous alarms. However, N-gram models fail to take into account long-range dependencies, because their estimation is feasible only for small values of N. Therefore, more advanced models based on neural network architectures were proposed, such as in [44], where alarm log information is embedded using Word2Vec and an LSTM-based deep learning model is designed to predict the next alarm. In [310], the authors propose a system that can predict different types of alarms by leveraging a two-stage forecaster–analyzer approach. It combines LSTM neural networks, to forecast the future measurements of various sensors, with Residual Neural Networks to predict the future occurrence of alarms based on estimated future

sensor measurements.

5.2.3 Challenges

One of challenges in AF is that many business-related critical alarms are rare, while, similarly to natural languages [239], alarm sequences are dominated by few uninformative tokens. Often rare alarms are very relevant, because they can go off when critical events occur that lead to costly machine shutdowns, production losses, or, in some cases, severe accidents [44]. Unfortunately, most standard optimization techniques are ineffective about rare classes and tend to consider them as noise. So, it is necessary to reformulate the problem to focus on those alarms. At the same time, it is necessary to limit the impact of highly frequent, uninformative alarms.

5.3 Proposed Approach for Alarm Forecasting

In this work, we propose a new formulation for the Alarm Forecasting problem, framed as a multi-label classification task. We present a novel deep learning-based approach called FORMULA (alarm FORecasting in MUlti-LAbel setting). FORMULA leverages Transformer, a popular Neural Network architecture in the field of Natural Language Processing. To cope with alarm imbalance, we draw inspiration from Segmentation and Object Detection. Thus, FORMULA is trained by minimizing the Weighted Focal Loss, which turns out to be very effective in predicting rare alarms. These alarms, even if they are difficult to predict by nature, often are business-critical. We assess the proposed approach on a representative real-world problem from the packaging industry. In particular, we show that it outperforms not only classic multilabel techniques but also models based on recurrent neural networks. As regards the latter, the proposed approach also exhibits a lower computational burden, both in terms of training time and model size. To foster research in the field and reproducibility, we also publicly share the alarm logs dataset and the code used to perform the experiments

5.3.1 Motivation and Formalization of Alarm Forecasting as Multi-label classification

Motivation

The goal of AF is to predict which alarms will occur in the future, based on the knowledge about past ones. Since users need time to take corrective actions, AF should predict alarms well in advance.

The industrial use case that inspired this work (the same use case that will be used in Section 5.3.7 for the experimental validation of the proposed approach) comes from a real-world problem in the food packaging industry. In particular, we focused on the machines used for the primary packaging of dairy products. The goal is to develop a solution that can be deployed in the context of a

Decision Support System to provide auxiliary information to human operators in the monitoring of such machines.

These machines are fairly complex since they are made of thousands of components (from 2500 to 3000 depending on the specific configuration) and dozens of processes and sub-processes controlled by approximately 450 variables. Each component can fail causing an alarm: some combinations of those alarms stop the machine causing unexpected downtime. Since the processes into the machine are fast (they run all under the second) and due to the complexity of the machine, the operator receives an average of 4 alarms per minute. This implies that is not useful to predict the next alarm or the next few alarms because of the high alarm raising frequency and redundancy. In this case study, and in similar ones with complex industrial machines raising an high volume of alarms, it is much more interesting for the operators to have a list of distinct alarms that are expected to occur in a future time window, independently from the actual sequence order and the exact timestamp.

In other words, the goal is providing a list of the predicted future alarm occurrences in reasonable advance to the machine operator, to prevent unexpected downtime or component failures, in a business context where downtime of few hours means to throw away the product due to high hygiene requirements; given a future time window whose beginning and duration depend on monitoring needs and can be chosen arbitrarily, the goal is to predict which events will occur in this time window, no matter the sequence order and exact timestamp. The formalization described above is general enough to be interesting also for other scenarios, not only industrial ones.

Formalization

Therefore, we propose a Multi-label Classification approach, where, given a future window of fixed length, for each alarm code we predict if it will occur or not, based on the information about alarms raised by the machine in a past time window. Given the set of all alarm codes $\mathcal{A}$, we define a subset of alarm codes $\mathcal{S}_o \subseteq \mathcal{A}$ to be predicted (we may be interested in predicting just some of the existing alarm codes), such that $\mathcal{S}_o = \{a_1, \ldots, a_M\}$. In our problem formulation, the input is the sequence of past alarms $x = (x_1, \ldots, x_L)$, with $x_j \in \mathcal{A}$ for $j = 1, \ldots, L$, such that x belongs to the set $\mathcal{X}$ of all alarm sequences, possibly with repetitions, whose maximum length is L. The corresponding output is the n-hot encoding $y = (y_1, \ldots, y_M) \in \mathcal{Y}$, such that $y_i = 1$ if the occurrence of alarm a_i is predicted, otherwise $y_i = 0$. To recap, we frame AF as the task of estimating the following map from the input set to the label space:

$$\begin{aligned} h &\colon \mathcal{X} \to \mathcal{Y} \\ (x_1, \ldots, x_L) &\mapsto (y_1, \ldots, y_M). \end{aligned} \tag{5.6}$$

It has to be noted that this formulation does not take into account the elapsed time between alarms. Indeed, we draw inspiration from the techniques used in Natural Language Processing, where sequences are modeled as a list of

Abbreviation	Description
AD	Anomaly Detection
AF	Alarm Forecasting
CE	Binary Cross Entropy
DL	Deep Learning
DNN	Deep Neural Network
FDC	Fault Detection and Classification
FL	Focal Loss
HMI	Human-machine Interface
LSTM	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network
SCADA	Supervisory Control And Data Acquisition
TCO	Total Cost of Ownership Entropy
WFL	Weighted Focal Loss

Table 5.1: Abbreviations

subsequent symbols. This choice was made both for simplicity and also because, in some situations, the elapsed time between alarms may not be informative, or it could require domain-specific preprocessing to be useful. For example, in situations where production has several interruptions, time intervals between machine alarms tend to be very different due to requalification phases after maintenances or stops [285].

5.3.2 Dataset design in the multi-label classification setting

Usually, raw alarm logs are just lists of tuples, each consisting of (i) an alarm code, (ii) a timestamp, and (iii) the identifier of the source machine. Thus, the first step is the pre-processing of data to produce the input and output pairs as defined in Eq. (5.6). As for the input, it is given by the ordered sequence of alarms occurred in a time window $(T_i, T_i + D_i)$ of fixed length D_i. To produce the corresponding output, we consider all alarms belonging to $\mathcal{S}_o$ occurred in a subsequent time window of predefined duration D_o, starting at $T_i + D_i + \delta$, with $\delta \geq 0$. Parameters D_i, D_o and δ can be chosen based on the specific business and monitoring needs. Fig. 5.1 depicts an example of the creation of an input-output sequence pair. In addition, we perform data augmentation by considering partially overlapping input windows through a sliding windowing approach. Notice that this procedure is applied separately to the alarms generated by each piece of equipment. The number of alarms occurring in a predefined amount of time may vary, but ML algorithms based on mini-batch gradient descent usually require sequences of fixed length. Therefore, to deal with this problem, two common procedures are used: padding and cutting (also

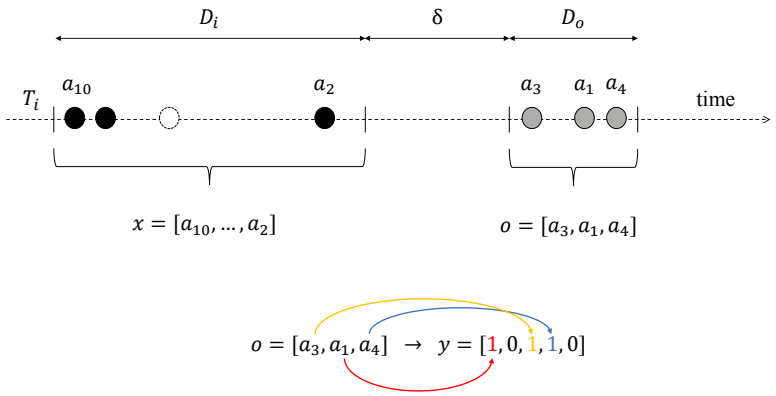


Figure 5.1: Suppose we are interested in predicting five target alarms, so $\mathcal{S}_o = \{a_1, a_2, a_3, a_4, a_5\} \subseteq \mathcal{A}$. We define an input sequence x by considering all alarms occurring in a past time-window with fixed predefined length. Let's consider a future time window of predefined duration. Let o denote the output sequence of alarms occurring in a the future time window. We define our prediction target y as the n-hot encoding of all alarms in o that belong to $\mathcal{S}_o$.

known as 'truncating'). At first, we need to decide a specific target length L which will be the final common size for all samples. A typical choice in the literature is to set L equal to the mean plus 3 times the standard deviation of the original input sequence lengths. When a sequence length is less than L, we apply padding; padding consists in adding 0 values at the end of the sequence to reach the target length. On the opposite, when a sequence longer than L, we need to remove the last symbols of the sequence to cut it down to length L. Eventually, we dealt with uninformative alarms that cause chattering by using the simple technique to prune repeated consecutive alarms.

5.3.3 FORMULA

To the best of our knowledge, addressing AF as a multi-label classification task with past alarm sequences in the input is a new problem statement in the literature. Therefore, in designing the proposed solution, we investigated both classic and advanced ML methods, based on Neural Networks, for multi-label classification. To account for dataset imbalance, which is often the case when dealing with alarms generated by machines in the industrial scenario, we investigated the use of non-standard loss functions. Thus, we assessed the classification performance achieved by different combinations of NN architectures and losses, as detailed in Sections 5.3.4 and 5.3.5, respectively. In light of the experimental results, we propose an approach called FORMULA, i.e. alarms FORcasting in

MUlti-LAbel setting. FORMULA is a new method for multilabel alarm forecasting: in the proposed approach a neural network inspired to Transformer [307] is trained by minimizing Weighted Focal Loss to perform AF.

5.3.4 NN Architectures

To learn the map defined by Eq. (5.6), we consider models based on Neural Networks. It is easy to adapt to the structure of a NN to produce multi-label predictions. Indeed, to address multi-label classification with M different alarm codes, we consider NNs whose output layer is given by M sigmoid activations. Since the sigmoid activations share the hidden layers, whose weights are trained using back-propagation, the network should be able of capturing relations among labels, in contrast to some standard ML approaches based on Binary Relevance problem transformation, described in Sec. 5.3.6.

As stated above, in applying NNs to AF we draw inspiration from approaches that have been successful in NLP [148]. Thus, we focus on recurrent [22] and attention-based architectures [345, 307]. In particular, we consider the following three paradigmatic models:

- REC: a bidirectional recurrent model based on Gated Recurrent Units, depicted in Fig. 5.2(a);
- ATT: a model that combines an attention mechanism with recurrent units, shown in Fig. 5.2(b), along the same lines of [345];
- TRM: a model based on the Transformer block [307], depicted in Fig. 5.2(c). Details about the architecture of the Transformer block are shown in Fig. 5.3.

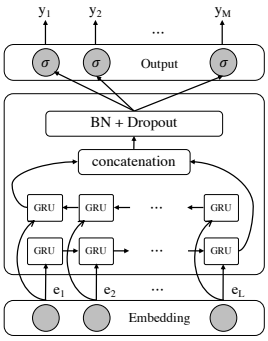
5.3.5 Loss functions for rare alarms forecasting

High label imbalance makes alarm prediction challenging. To overcome this issue, we considered different choices for the loss function used to train the proposed NNs models. In particular, we draw inspiration from the fields of Object Detection and Image Segmentation in computer vision, where foreground-background class imbalance is often very high, whenever there are many candidate locations per image but only a few of them are relevant [167, 258].

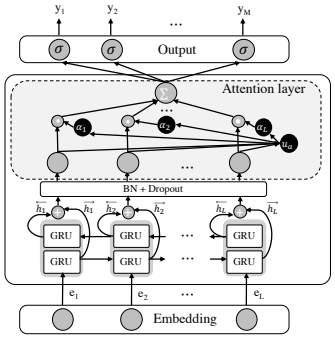
To shorten notations for some of the considered losses, for each alarm, we denote its true label by $y \in \{0, 1\}$. Then, we introduce

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases}, \tag{5.7}$$

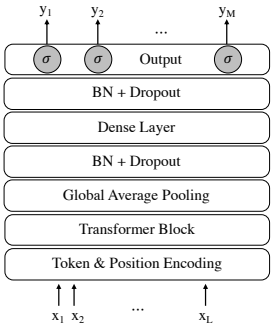
where p is the probability given in output by the network corresponding to a specific alarm. By introducing $\mathbf{p}$, the output array corresponding to the



(a) Bidirectional recurrent model (REC).



(b) Recurrent model with attention (ATT).



(c) Transformer-based model (TRM).

Figure 5.2: Considered NN architectures.

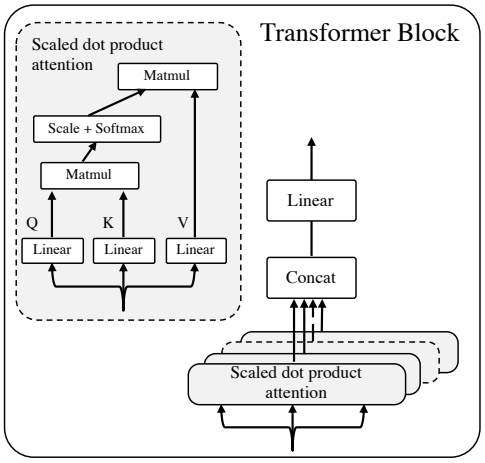


Figure 5.3: Architecture of the Transformer block.

whole label set, we can define $\mathbf{p}_t$ analogously. Along the same line, we define a weighting factor β_t as follows

$$\beta_t = \begin{cases} \beta & \text{if } y = 1 \\ 1 - \beta & \text{otherwise} \end{cases}, \tag{5.8}$$

for $0 < \beta < 1$. Fig. 5.4 depicts an example for each of the considered losses that are detailed below.

Binary Cross Entropy (CE)

Binary Cross Entropy provides a benchmark for our experiments, since it is the standard loss used in binary classification tasks but it is not optimized to deal with class imbalance. By Eq. (5.7), the binary Cross Entropy (CE) can be written as

$$\mathrm{CE}(\mathbf{p}_t) = -log(\mathbf{p}_t). \tag{5.9}$$

Weighted Cross Entropy (WCE)

Weighted cross entropy is a variation of Binary Cross Entropy where a weight parameter is introduced to compensate class imbalance [19]. By (5.7) and (5.8) we can write WCE as:

$$\mathrm{WCE}(\mathbf{p}_t, \beta_t) = -\beta_t log(\mathbf{p}_t), \tag{5.10}$$

where $0 \leq \beta_t \leq 1$.

Focal Loss (FL)

Drawing inspiration from a successful approach to one-stage object detection described in [167], we also consider another variation of Binary Cross Entropy called Focal Loss:

$$\mathrm{FL}(\mathbf{p}_t, \gamma) = -(1 - \mathbf{p}_t)^{\gamma} log(\mathbf{p}_t). \tag{5.11}$$

As suggested by Fig. 5.4, the term $(1-\mathbf{p}_t)^{\gamma}$ acts as an adaptive weighting factor that is used to down-weight the contribution of easy samples in the training, so the model can focus more on hard examples. The focusing parameter $\gamma \geq 1$ defines the rate at which easy examples are down-weighted: the higher γ, the wider the range in which an example receives low loss.

Weighted Focal Loss (FL)

This variation of Focal Loss is proposed in [167]. It includes a fixed weighting factor β_t, as defined in the following equation:

$$\mathrm{WFL}(\mathbf{p}_t, \beta_t, \gamma) = -\beta_t (1 - \mathbf{p}_t)^{\gamma} log(\mathbf{p}_t), \tag{5.12}$$

where γ, β_t and $\mathbf{p}_t$ are defined as above.

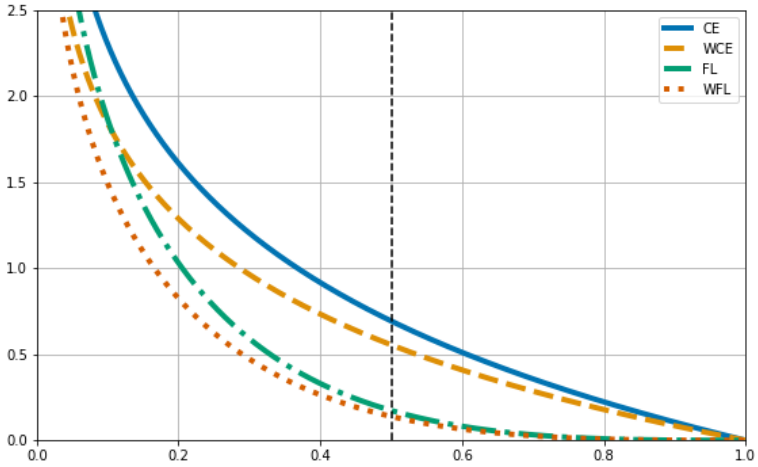


Figure 5.4: Loss functions used to address data imbalance, compared to CE, when $y = 1$. In FL and WFL, $\gamma = 2$. In WFL, $\beta = 0.8$.

5.3.6 Benchmark Approaches

It is non trivial to provide a fair comparison between FORMULA and standard AF methods. In the literature, the alarm forecasting problem is typically based on the prediction of the single next alarm; however, in the industrial scenario motivating the proposed approach, the prediction of the next alarm is not enough to help users achieve a more efficient monitoring and management of the industrial machine. In the considered scenario, alarms are frequent overall, and predicting co-occurrences in a future time slot is more relevant than providing the actual temporal sequence of alarm occurrences. However, by extending standard methods to predict many future alarms at once, we would encounter error propagation issues, because we should use the prediction of the i-th alarm as input for the prediction of the $(i{+}1)$-th one. Thus, the comparison would not be fair with regard to ML methods based on the forecasting of the next alarm. Therefore, we focus on AF framed as Multi-Label Classification, and we consider classic ML methods for Multi-Label Classification as benchmarks.

Drawing inspiration from the analogy between alarm sequences and natural language sentences, we perform feature engineering by mapping input alarm sequences to *tf-idf* representations [148]. To apply standard classification algorithms to a multi-label setting with M labels, we consider problem transformation approaches that convert multi-label problems to single-label problems, either single-class or multi-class, as detailed in [183, 242]:

- Binary Relevance (BR): M classifiers are trained separately, one for each label.
- Classifier Chain (CC): it trains M classifiers ordered in a chain according

to the Bayesian chain rule.

- Label Powerset (LP): it transforms a multi-label problem to a multi-class problem where each label combination found in the training data is considered as a separate class.

5.3.7 Experimental Results

As described in Section 5.3.1, the proposed approach was designed while investigating a real-world AF problem, coming from dairy products packaging industry. One of the novel contributions in this research is the public release of the data used to design and evaluate the proposed approach [297]. This aims to alleviate the limited availability of public data on alarm sequences generated by real industrial processes, to foster research and provide a benchmark. Not only the dataset, but also the code used in the experiments is provided in an open repository [1].

To evaluate the proposed AF approach, we consider logs acquired from February 2019 to June 2020 by 20 packaging machines deployed in different plants around the world. In the logs, there are 154 distinct alarm codes. For confidentiality issues, we cannot provide too many details on the nature of the underlying alarms, however it can be said that such alarms are quite an heterogeneous set: for instance, some of them are related to how the machine is used, while other are associated with internal mechanisms not directly affected by the users and the products.

We focus on two forecasting scenarios:

[S.T] Target alarms scenario. The goal is to predict a subset of 15 alarms that are considered the most valuable based on the business context. This task is the real-world challenge that inspired this work;

[S.R] Rare alarms scenario. Aiming at extending this work to other equipment and industrial contexts, we need to understand to what extent the proposed approach can deal with very infrequent alarms. Indeed, alarms that are very relevant from the business point of view, often are also rare. Thus, we deal with the forecasting of a subset of 10 rare alarms.

For each alarm label a_i, we define its frequency f_{a_i} as

$$f_{a_i} = \frac{|s \in \mathcal{O} \text{ where } a_i \in s|}{|\mathcal{O}|}, \tag{5.13}$$

where $\mathcal{O}$ is the set of output sequences and $|\mathcal{O}|$ its size. In Tab. 5.2, we provide some statistics about the output sequences described above:

- Number of distinct alarms in output sequences;
- Label cardinality, i.e. the average number of occurring alarms in output sequences;

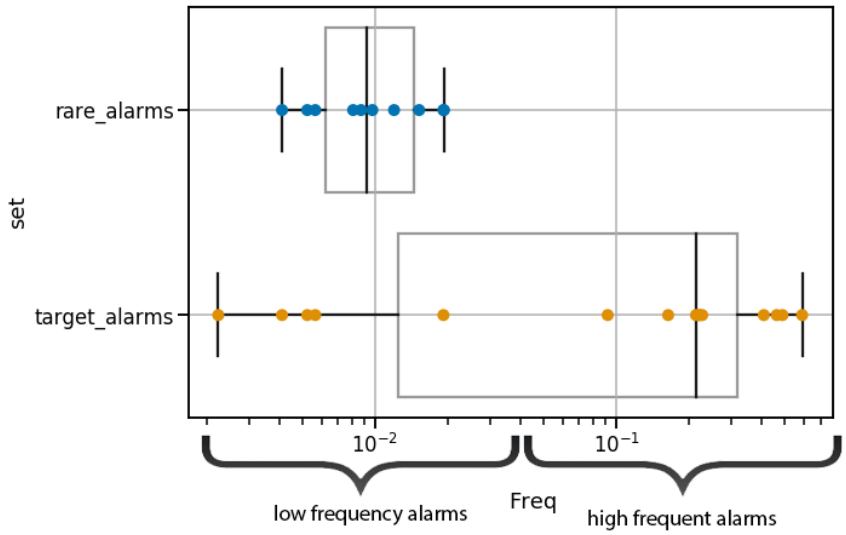


Figure 5.5: Distribution of output labels (i.e. subset of alarm codes) for S.R. and S.T. scenarios. Each alarm is depicted as a dot, corresponding to its frequency. The horizontal axis, i.e. frequency axis, is in logarithmic scale. We can observe that in scenario S.T. there are both frequent and rare alarms, with a frequency range [0.22% - 58.84%]. Instead, frequencies of alarms in scenario S.R. are all rare, within the interval [0.41% - 1.93%].

- Maximum label cardinality in dataset;
- Percentage of output sequences with no alarms;
- Information about alarm frequencies: first quartile, median, third quartile, average, and the ratio between the highest and lowest frequency among alarms (this value can be interpreted as a measure of class imbalance)
- Label diversity, as defined in Sec. 5.3.1.

Alarm frequency distribution is quite imbalanced in scenario [S.T], as shown in Fig. 5.5, too. In particular, there are two orders of magnitude between the highest and lowest alarm frequency in the dataset. In scenario [S.R], instead, the maximum frequency ratio is below 5, indicating a more balanced alarm frequency distribution. It is interesting to notice that, in scenario [S.R], most output sequences have no relevant alarms, resulting in label cardinality being below 1.

Table 5.2: Output labels statistics.

	Target alarms [S.T]	Rare alarms [S.R]
Distinct alarms number	15	10
Label cardinality	6.69 ± 5.83	0.11 ± 0.36
Max label cardinality	10	4
Label diversity	1.68e-2	6.82e-4
Percentage of output sequences with no alarms	1.52%	89.54 %
Q1 frequency	1.24%	0.63%
Median frequency	21.36%	0.92%
Q3 frequency	31.74%	1.44%
Avg frequency	20.8%	1.07%
Min frequency	0.22%	0.41%
Max frequency	58.84%	1.93%
Max frequency ratio	263.52	4.69

Experimental Settings and Evaluation

Taking into account the requirements coming from the motivating industrial case at hand, we considered alarm sequences designed with $D_i = 1720$ minutes, $D_o = 480$ minutes, and $\delta = 0$.

To evaluate the proposed ML models, the set of input and output alarm sequence pairs is split into a training dataset and a subsequent test dataset. The splitting is performed based on time to avoid data leakage. 70% of the sequence pairs is used to fit models, and the remaining 30% to assess their performance.

To measure performance, we consider the macro f1-score. We recall that f1-score for alarm a_i is defined as

$$f_1(a_i) = 2 \cdot \frac{p_i \cdot r_i}{p_i + r_i},$$

where p_i and r_i denote precision and recall for alarm a_i, respectively. Then, macro f_1 score is defined as the average of f_1-scores corresponding to each alarm label, no matter its frequency:

$$\text{macro } f_1 = \frac{1}{M} \sum_{i=1}^{M} f_1(a_i). \tag{5.14}$$

The choice of macro f_1-score is made to have a fair metric about infrequent alarms prediction: since alarm distribution is imbalanced, a model that ignore rare alarms in favor of the most frequent alarms, could have incorrectly been evaluated as a good model with 'standard' f_1-scores.

Table 5.3: Comparison between classic ML algorithms for multi-label classification considering different strategies (BR, CC, LP) and models (RF, NB).

Scenario	Model	Macro f1
[S.T]	BR + RFC	0.404
	CC + RFC	0.406
	LP + RFC	***0.422***
	BR + NB	0.181
	CC + NB	0.1867
	LP + NB	0.178
[S.R]	BR + RFC	0.067
	CC + RFC	***0.084***
	LP + RFC	0.067
	BR + NB	0.00695
	CC + NB	0.00777
	LP + NB	0.00789

Table 5.4: Comparison between classic ML algorithms for multi-label classification and FORMULA, i.e. TRM + WFL.

Scenario	Approach	Model	Macro f1	Improvement w.r.t. Classic ML
[S.T]	Classic ML	BR + RFC	0.404	-
		CC + RFC	0.406	-
		LP + RFC	*0.422*	-
	FORMULA	TRM + WFL	**0.549**	+ 30%
[S.R]	Classic ML	BR + RFC	0.067	-
		CC + RFC	*0.084*	-
		LP + RFC	0.067	-
	FORMULA	TRM + WFL	**0.367**	+ 337%

Results in scenario [S.T] for classic multilabel approaches

We first tested the classic ML-based approaches described in Sec. 5.3.6 to provide a benchmark. To identify the best base classifier, we considered both Multinomial Naive Bayes (NB) and Random Forest Classifier (RFC) as candidates. For each model, hyper-parameter tuning is performed through 5-fold cross validation applied to the training dataset. Predictive performance is then evaluated on the test set.

In Table 5.3, we show the performance obtained by RFC and NB models when trained according to Binary Relevance (BR), Classifier Chain (CC), and Label Powerset (LP) multi-label approaches. In scenario [S.T] the best performing solution is Label Powerset transformation with Random Forest.

Tab. 5.4 also shows the comparison with our approach FORMULA i.e.

Table 5.5: Comparison between Binary Relevance approaches and FORMULA (i.e. TRM + WFL), scenario [S.T].

Scenario	Approach	Model	Macro f1
[S.T]	Binary Relevance	BR + RFC	0.404
		BR + TRM (WFL)	0.475
	FORMULA	TRM + WFL	**0.549**

TRM with WFL. We can observe that our approach performs better in [S.T] scenario with respect to classic multilabel approaches. A problem with Label Powerset (LP) is that it may suffer from underfitting issues, since it disregards combinations of labels not appearing in the training dataset. Moreover, it is not scalable as the set of alarms to predict is extended, due to the high number of classes resulting from alarm combinations. Binary Relevance, on the other hand, is more robust. However, it does not take relations among labels into account, since each classifier is trained separately. To investigate this, we also train a dedicated TRM classifier for each target alarm, in a Binary Relevance approach, and compare the results with standard TRM model with M output units sharing hidden layers, as described in Sec. 5.3.4. As detailed in Tab. 5.5, FORMULA approach, i.e. a TRM model with M outputs trained by minimizing WFL, consistently outperforms the pool of M independent TRM classifiers trained with WFL according to a Binary Relevance transformation of multilabel classification task. Besides the lower performance, another drawback in using NN-based classifiers in a Binary Relevance framework is the computation burden, since it is necessary to train one NN for each output class.

Results in scenario [S.T] for FORMULA

To motivate the effectiveness of FORMULA, we evaluate all the NN architectures introduced in Sec. 5.3.4, combined with the loss functions described in Sec. 5.3.5. To assess performance, for each combination of model and loss function, we consider twenty repeated runs for the training algorithms. Then, we evaluate models on the test set.

Fig. 5.6 allows us to compare the performance achieved by the considered NN architectures when trained using different loss functions. As already anticipated, TRM model trained by minimizing WFL is the best performing combination. In particular, WCE and WFL consistently outperform their non-weighted counterparts for all the considered NN architectures, suggesting that the weighting policy is actually effective in dealing with class imbalance. In addition, it seems that the attention mechanism used in ATT and TRM allows for improving classification performance with regard to REC, for each choice of the loss function used in training the model. As reported in Tab. 5.4, we can conclude that approaches based on NNs and WFL perform consistently better than classic ones.

To shed light on how the choice of the loss function affects the classification

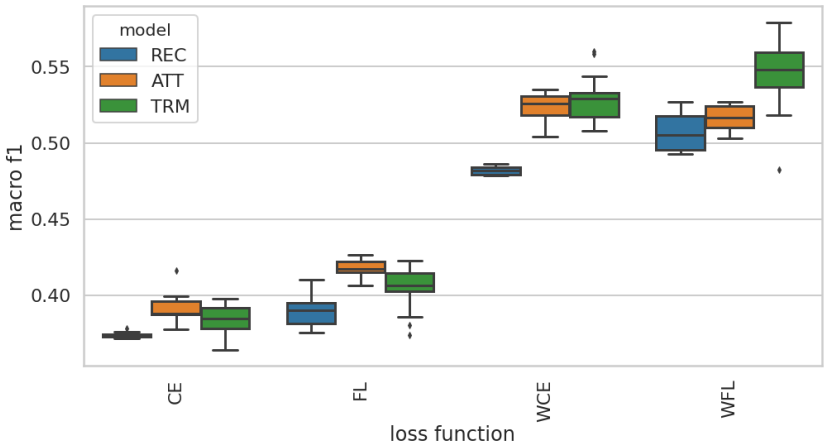


Figure 5.6: Scenario [S.T]: comparison among different models and loss functions tested with optimal hyper-parameters.

performance, it is interesting to analyze results at single alarm code level. The bubble chart in Fig. 5.7 plots f1 score versus alarm frequency. Each bubble is a specific alarm code: its size is proportional to the recall, while warmer colour means higher precision. A circle marker is used to denote the results attained with WFL, while a square is used to depict results obtained through standard CE. From the plot, we can conclude that the increase of macro f1 score obtained with WFL, w.r.t. standard CE, depends on achieving a higher f1 score for less frequent classes, as expected based on the considerations detailed in Sec. 5.3.5. In particular, WFL provides a much higher recall for low-frequency alarms, partly at the expense of precision, which is slightly lower, in general. The precision-recall trade-off is a well-known issue in classification tasks. In principle, in the industrial scenario, higher precision means not wasting human resources time on false positives. On the other hand, higher recall implies recognizing most of the situations that may lead to downtime. In our case study, coming from dairy products packaging industry, where downtime also involves costly raw material losses due to hygienic constraints, obtaining a high recall is preferable.

Not only does TRM achieve higher performance than REC and ATT, but it is also faster to train and to deploy. Indeed, it is possible to leverage parallelism to deal with multiple attention mechanisms at once, while recurrent layers are intrinsically sequential. Moreover, when considering model size after hyper-parameter tuning, the returned TRM model also has the lowest number of parameters. Based on the experimental results, it also seems that models with attention mechanism such as ATT and TRM require fewer epochs to converge when an Early Stopping policy is active during training. More details about computational time, convergence, and model size are provided in Tab. 5.6.

We would like to stress that the proposed approach is designed to be included

Table 5.6: Computational time, average number of epochs required to converge and model size for fine-tuned NN-based models. Models are trained by minimizing WFL.

Model	Avg Time per epoch (s)	Required epochs for convergence (Avg)	Num. of trainable parameters
REC	99.09	17.96	172.734
ATT	115.35	8.04	238.250
TRM	10.59	7.29	26.794

Table 5.7: Performance of NN models trained with different loss functions, scenario [S.T]

Scenario	Loss function	Model	Macro f1	Macro precision	Macro recall
[S.T]	CE	REC	0.374	0.454	0.348
		ATT	0.391	0.498	0.366
		TRM	0.384	0.469	0.361
	FL	REC	0.389	0.509	0.355
		ATT	0.418	**0.569**	0.381
		TRM	0.405	0.502	0.375
	WCE	REC	0.482	0.456	0.614
		ATT	0.523	0.491	0.653
		TRM	0.529	0.478	0.678
	WFL	REC	0.508	0.499	0.630
		ATT	0.516	0.490	0.652
		TRM	**0.546**	0.492	**0.692**

in a Decision Support System. This means that the model predictions are not consumed by an automatic system. Instead, they are used by operators to improve their decision making. Before the introduction of FORMULA-based systems in the motivating industrial case, information about alarms was just visualized and there was no 'intelligent' system helping operators in coping with process monitoring. In this scenario, each piece of additional information coming from the proposed algorithm is valuable since it helps operators to focus on the most relevant issues.

Results in scenario [S.R]

Class imbalance can have a detrimental effect on classification performance for low-frequency classes. However, rare events are often extremely relevant from the point of view of applications. For instance, as regards AF in the manufacturing industry, severe downtime is rare, but being able to forecast related alarms would be extremely useful in taking prompt corrective actions to preserve productivity and prevent energy and raw materials waste. Thus, investigating to

Table 5.8: Performance of NN models trained with different loss functions, scenario [S.R]

Scenario	Loss function	Model	Macro f1	Macro precision	Macro recall
[S.R]	CE	REC	0.165	0.407	0.118
		ATT	0.161	0.376	0.109
		TRM	0.196	0.322	0.151
	FL	REC	0.199	0.422	0.146
		ATT	0.194	**0.449**	0.134
		TRM	0.206	0.337	0.159
	WCE	REC	0.310	0.337	0.307
		ATT	0.307	0.344	0.311
		TRM	0.357	0.321	0.446
	WFL	REC	0.302	0.331	0.305
		ATT	0.327	0.379	0.323
		TRM	**0.367**	0.335	**0.450**

what extent the proposed approach can deal with very infrequent alarms is an important aspect to assess its effectiveness. Therefore, we evaluate the performance achieved by FORMULA in scenario [S.R], where the goal is forecasting the occurrence of 10 alarms rare output alarms. Details about this dataset composition are provided in Tab. 5.2 and Fig. 5.5.

As mentioned above, and detailed in Tab 5.8 and Fig. 5.10, using WFL instead of classic CE improves recall, and translates into higher overall macro f1 score.

In Tab. 5.4, we can compare the proposed approach with classic multilabel techniques, and we can observe that it performs much better when we are trying to predict the rare alarms. In particular, the performance improvement achieved by FORMULA with regard to classic ML approaches is much more relevant in scenario [S.R] than in [S.T], reaching an improvement of 337%.

In Fig. 5.9, we compare the performance achieved by previously introduced NN models and loss functions combinations in the rare alarms scenario [S.R]. TRM model trained by minimizing WFL is the top performer also in this scenario.

In conclusion, results confirm the effectiveness of the proposed approach, in particular when the focus is on the prediction of rare alarms.

5.3.8 Conclusions

In the industrial scenario, forecasting alarms generated by the equipment may allow taking prompt corrective actions to prevent or at least minimize downtime. In this work, we propose a novel problem statement for alarm prediction, i.e. multi-label binary classification with sequences in the input. Motivated by the analogies between alarm forecasting and Natural Language Processing,

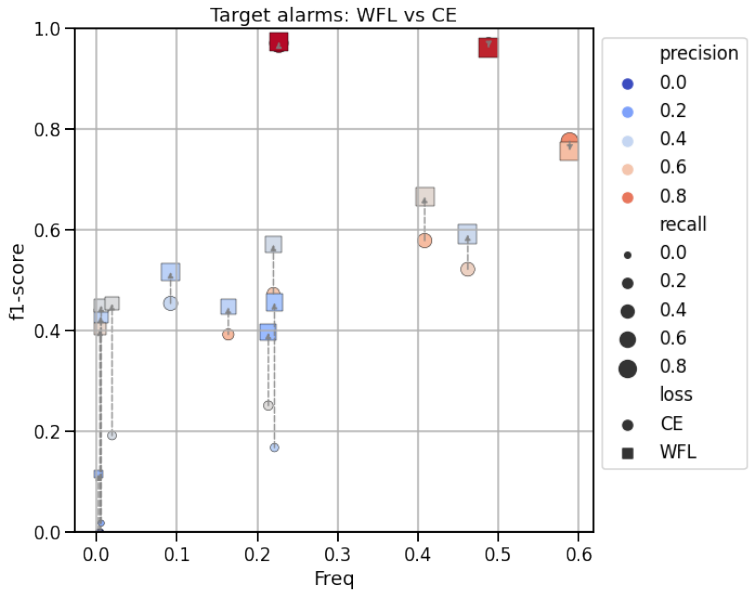


Figure 5.7: Scenario [S.T]. Each bubble is a distinct alarm code plotted with f_1 score on y-axis and alarm frequency on x-axis. Its size is proportional to the recall, while warmer colour means higher precision. A circle marker is used to denote the results attained with WFL, while a square is used to depict results obtained through standard CE. For each alarm code there is an arrow that connect its CE version with the WFL version. In general, WFL results in higher f1-score for low frequency classes, with increased recall at the price of slightly reduced precision.

we assess the performance of state-of-the-art Neural Network architectures. To address alarm frequency imbalance, we consider non-standard loss functions beyond classic Binary Cross Entropy, in the same vein as in Object Detection and Image Segmentation. The resulting approach, named FORMULA (alarm FORecasting in MULti-LAbel setting), aims at predicting future alarms in a multi-label classification setup through a model inspired by the Transformer and trained with Weighted Focal Loss as the optimization target.

Our experiments on a real industrial dataset suggest that FORMULA performs consistently better than classic ML and that it can focus on low-frequency alarms too, which is relevant because many business-critical alarms are rare. Finally, the dataset used in the experiments was publicly shared to foster research. This dataset contains industrial alarm logs generated by packaging equipment. In support of the added value provided provided by the proposed approach, we remark that the industrial partner collaborating in this work is currently rolling out a new service based on FORMULA.

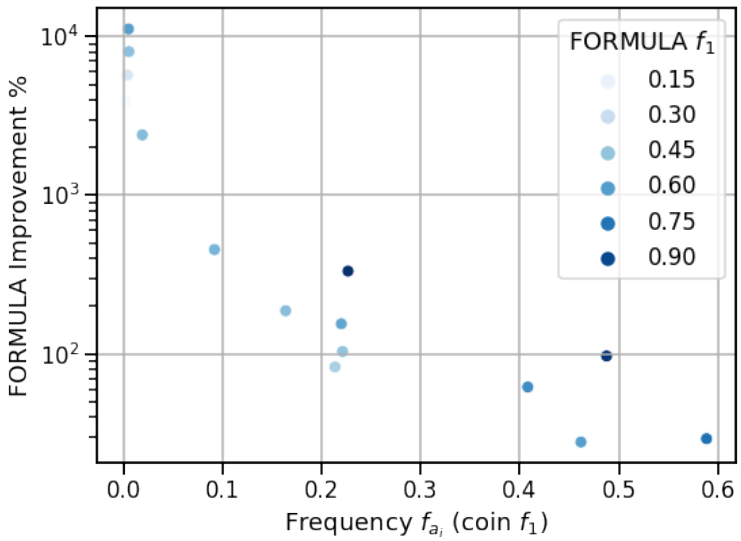


Figure 5.8: Scenario [S.T]: comparison between FORMULA f_1 and coin toss f_1^c scores. Each bubble is a distinct alarm code. Horizontal axis reports alarm frequency, that is equal to f_1 score achieved by coin toss, for each alarm. Vertical axis depicts the f_1 score improvement obtained by FORMULA defined as $100\,(f_1(a_i) - f_1^c(a_i))/(f_1^c(a_i))$.

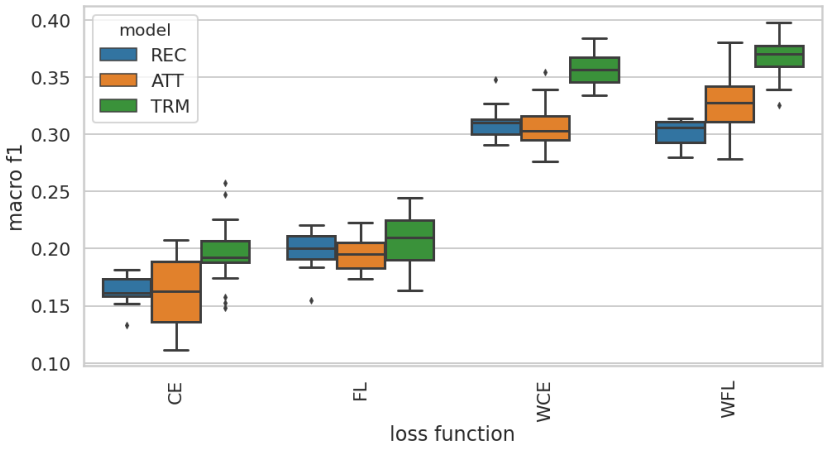


Figure 5.9: Scenario [S.R]: comparison among different models and loss functions tested with optimal hyper-parameters.

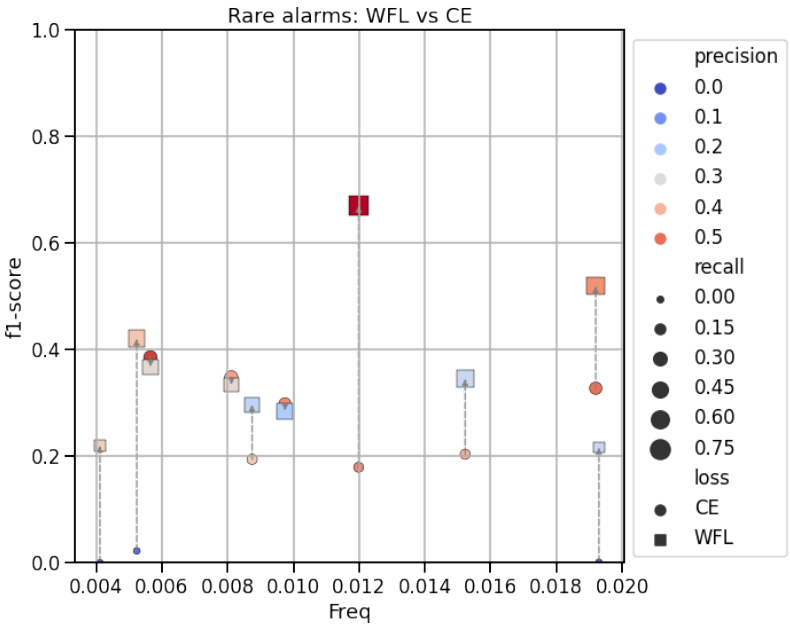


Figure 5.10: Scenario [S.R]. In the bubble chart, each bubble is a distinct alarm code plotted with f1 score on y-axis and alarm frequency on x-axis. Its size is proportional to the recall, while warmer colour means higher precision. A circle marker is used to denote the results attained with WFL, while a square is used to depict results obtained through standard CE. For each alarm code there is an arrow that connect its CE version with the WFL version.

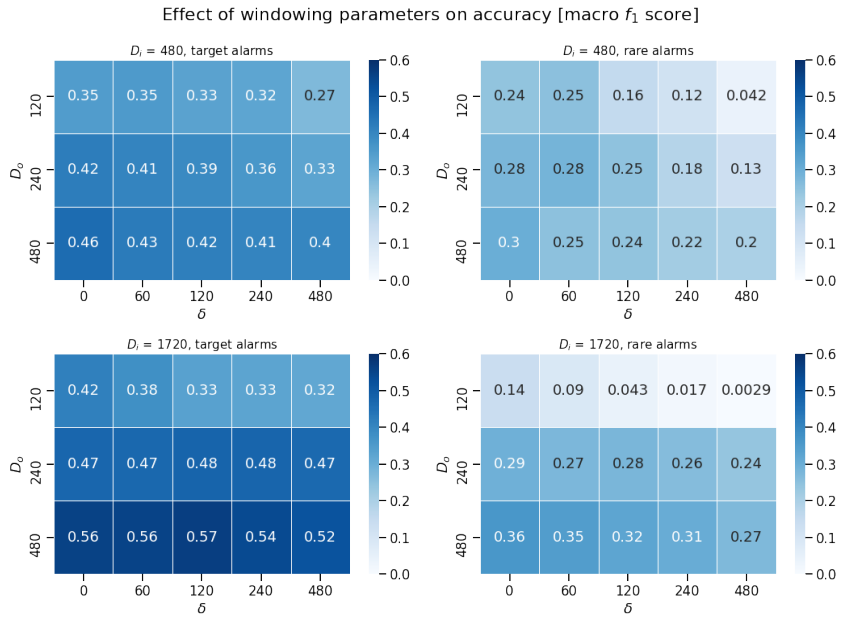


Figure 5.11: Average macro f_1 score over 10 runs for each combination of D_i, D_o and δ in the grid search. Each column represents one of the considered scenarios, either S.T. or S.R.. Different choices of input window length D_i correspond to different rows. Each heat map has δ values on x axis and window output lengths D_o on y axis.

5.4 Replay for Multilabel Setting

5.4.1 Motivation

A significant challenge is that current MLC methods are studied in the classic offline setting, which assumes that all data are available in advance for learning. In practice, data is collected sequentially, and data collected earlier in this process may not be available as time passes. We consider such a problem from the point of view of Industry 4.0. Manufacturing industries are in a continuous effort to improve their production yield, uptime, and throughput to increase production quality and minimize costs. An example is Predictive Maintenance, where machine learning algorithms can be used to analyze data from sensors and other sources to predict when equipment is likely to fail, allowing for preventive maintenance to be scheduled before the failure occurs. This can help reduce downtime and improve the overall reliability of manufacturing systems. In the practice case of Predictive Maintenance, Alarm Forecasting as multi-label classification is studied as a real-world application of continual learning in the multi-label setting. In our problem, we assume data from new machines be-

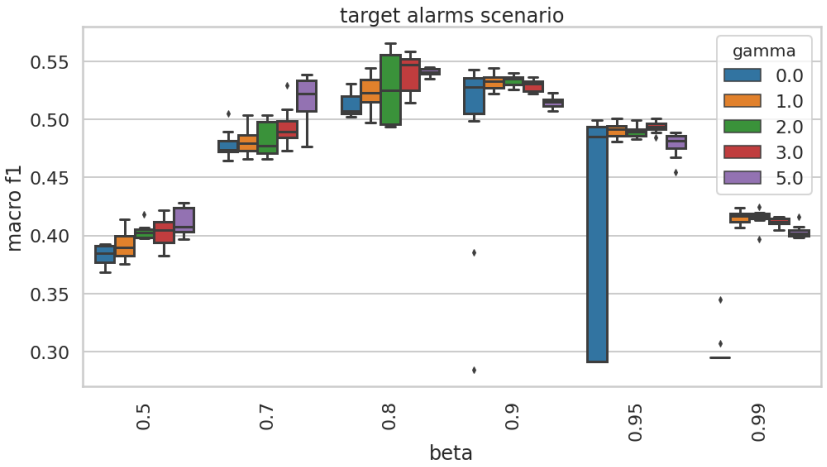


Figure 5.12: Effect of tuning of the parameters β and γ in the rare scenario (S.T.) with TRM model.

comes available over time. We also propose a novel approach for multi-label classification in continual learning, for which an efficient approach with logarithmic complexity in the number of tasks is proposed. In short, this work aims to adopt a CL framework to achieve a production-ready solution for AF, where to be scalable it must learn with new pieces of equipment that arrive over time.

5.4.2 Related work

CL tries to learn from a stream of tasks with non-stationary distribution. New experience is constantly being acquired over time, while old experience is still relevant and it needs to be preserved. In the classic paradigm, model retraining using the new data leads to a sharp decrease in performance on previously learned tasks, a phenomenon known as Catastrophic Forgetting(CF)[106]. The related literature suggests that the Rehearsal (also know as replay-based) approach appears to be a strong solution to CF [223, 42, 190, 140]. To the best of our knowledge, currently, only two main replay-based approaches designed for Multi-Label classification are proposed in the literature:

(i) The first approach is Partitioning Reservoir Sampling (PRS)[140]. The idea is that it is sufficient to allocate portion of the memory to the minority classes to retain a balanced knowledge of present and past experiences.

(ii) The second approach is called Optimizing Class Distribution in Memory (OCDM) [164]. It aims at speeding up the processing time required to select the samples, compared to PRS, obtaining a significant improvement. OCDM is a greedy approach that selects a subset of samples such that the final distribution of the labels in memory is as close as possible to a uniform target distribution.

Both approaches are derived from the scenario of Online CL (OCL), where, in contrast to classic CL, a single batch of the task is shown at each time. Thus, in OCL, multiple epochs on the same batch are not allowed. The PRS and OCDM were proposed for the CIL scenario[304] where, for each new task, new labels may appear.

In our case, we are considering a Domain Incremental Leaning (DIL) scenario[304] instead, where the set of output labels remains the same, but the input distribution changes based on the task. In fact, the set of alarm codes is the same for each piece of equipment, while the alarm frequency is different for the different deployments. In this work, we start from the OCDM to design a task-aware replay-based approach, as detailed in the next section.

5.4.3 Optimizing Class Distribution in Memory (OCDM)

Before introducing the novel approach used to address the Continual multilabel classification problem at hand, we recall some notions about the OCDM approach. OCDM formulates the memory update mechanism as an optimization problem. This greedy algorithm, detailed in Alg. 3, provides a solution with a complexity that is linear in the number of tasks.

The core of the algorithm is the procedure to update the memory when a new batch of data $\mathcal{B}_t$ of size b_t coming from the task $t \in \mathcal{T}$ arrives, as depicted in the Algorithm 1. Given that the memory $\mathcal{M}$ has fixed size M, for each task the goal of OCDM is to select M samples to save in the memory among the $M + b_t$ available ones. This can be represented as an optimization problem:

$$\min_{\Omega} Dist(\mathbf{p}_\Omega, \mathbf{p}) \quad \text{SUBJECT TO} \quad \Omega \subseteq \mathcal{M} \cup \mathcal{B}_t, \quad |\Omega| = M \tag{5.15}$$

where $\mathbf{p}$ represents the target distribution i.e. the ideal optimal solution, while, $\mathbf{p}_\Omega$ represents the distribution of the labels produced from the samples of the set Ω. The $Dist(\cdot, \cdot)$ function used to measure the difference between the two distributions is the Kullback–Leibler (KL) Distance. The target distribution proposed in [164] is defined as follows:

$$p_i = \frac{(n_i)^\rho}{\sum_{j=1}^{C} (n_j)^\rho} \tag{5.16}$$

where n_i is the frequency of a class i and ρ is the allocation power. Using $\rho = 0$ the samples are saved in memory $\mathcal{M}$ in order to have equally distributed classes.

Algorithm 1: Memory Update (MU)

Input: Memory $\mathcal{M}$, b

`/* Given a memory` $\mathcal{M}$`, it will delete` b `elements from the memory */`

for $k \in [1, 2, ..., b]$ **do**
  $i = arg\min_{j \in \mathcal{M}} Dist(\mathbf{p}_{\mathcal{M} \setminus \{j\}}, \mathbf{p})$
  $\mathcal{M} \leftarrow \mathcal{M} \setminus \{i\}$
end
return $\mathcal{M}$

Algorithm 2: OCDM_task_update

Input: dataset $\mathcal{D}_t$ of task $t \in \mathcal{T}$, memory $\mathcal{M}$, total size of memory M

for $B_t \in \mathcal{D}_t$ **do**
  if $|\mathcal{M}| \leq M$ **then**
    diff $\leftarrow |\mathcal{M}| - M$
    $V_t \leftarrow$ select randomly $min(|B_t|, \text{diff})$ samples from B_t
    $B_t \leftarrow B_t \setminus V_t$
    $\mathcal{M} \leftarrow \mathcal{M} \cup V_t$
  end
  if $|B_t| > 0$ **then**
    $\Omega \leftarrow \mathcal{M} \cup B_t$
    $\mathcal{M} \leftarrow Memory\_Update(\Omega, |B_t|)$
  end
end

Algorithm 3: Optimizing Class Distribution in Memory (OCDM)

Input: task stream $\mathcal{T}$, total size of memory M

$\mathcal{M} \leftarrow \{\}$ `/* Initialize the memory */`
for $t \in \mathcal{T}$ **do**
  $\mathcal{D}_t \leftarrow$ Get Dataset $\mathcal{D}_t = \{X_t, Y_t\}$ of task t
  $OCDM\_task\_update(\mathcal{D}_t, \mathcal{M}, M)$
end

5.5 Proposed Approach for Multi-label Replay

5.5.1 Continual Learning classifier design

The industrial scenario is a dynamic environment in which new machines are installed over time. CL provides tools that enable ML solutions to scale efficiently. In this work, we consider new equipment pieces as new tasks, setting our problem in the DIL scenario [304] as previously stated. This means that the set of labels in output $\mathcal{A}_{out}$ remains the same for new deployment of the target machine, while the input and output distributions change according to the current task. Indeed, even if the machines are of the same type, they may differ from the

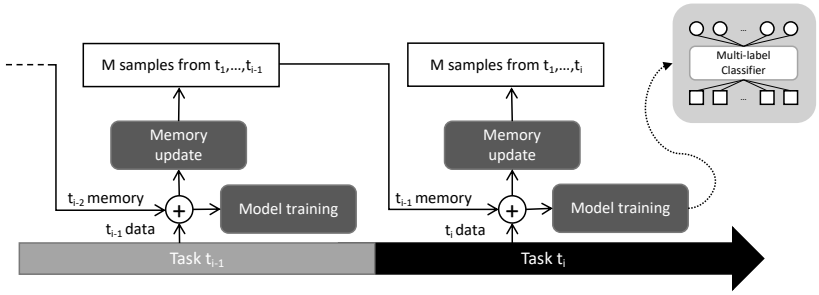


Figure 5.13: Rehearsal-based approach to deal with multi-label classification in a CL fashion.

already deployed ones in terms of settings, recipes, and the behavior of human operators [103]. To make a Multi-Label classifier learn to perform alarm forecasting on new machines without forgetting the old ones, we use replay-based approaches from CL, according to schema depicted in Fig. 5.13.

5.5.2 Balanced Among Tasks OCDM (BAT-OCDM)

The OCDM approach was proposed considering the CIL scenario, where a new set of labels arrives at each new task [304]. As stated above, in our case, we are considering the DIL scenario, where we always have the same labels with potentially new frequencies. Since OCDM focuses only on maintaining balance among labels, it ends up ignoring the differences among tasks.

In particular, OCDM does not guarantee that all tasks seen so far will be kept in memory. As an example, it may happen that when a new task with frequent labels arrives, then the algorithm may decide to never add any sample from this task into the memory because it would worsen the balance.

To address this limitation, we propose to use a separated memory for each task. Through this approach, described in Alg. 4, balance is performed on both labels and tasks, hence the name Balance Among Tasks OCDM (BAT-OCDM). BAT-OCDM requires two steps to handle the memory:

1. **Step 1**: Given the data of a new task, BAT-OCDM selects a subset of samples to be included in the memory, using the same procedure of OCDM to update the memory.

2. **Step 2**: Since the total size of memory has to remain fixed, BAT-OCDM disregards some of the samples belonging to the memories of old tasks through the $Memory\_Update$ (MU) procedure detailed in Alg. 1.

Fig. 5.14 shows a schema of the proposed approach, which uses BAT-OCDM to train the multi-label Classifier in a CL fashion.

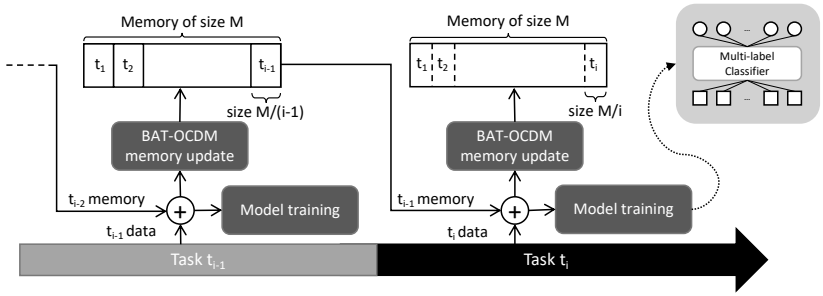


Figure 5.14: CL multi-label classification with BAT-OCDM strategy.

Algorithm 4: BAT-OCDM

Input: task stream $\mathcal{T}$, total memory size M
Constraint: $\sum_{t \in T} |\mathcal{M}_t| = M$

$N \leftarrow 0$
for $t \in \mathcal{T}$ **do**
  `/* We are going to assign a part of the memory to the new task and select the elements to keep in memory. */`
  $\mathcal{D}_t \leftarrow$ Get Dataset $\mathcal{D}_t = \{X_t, Y_t\}$ of task t
  $\mathcal{M}_t \leftarrow \{\}$
  $N \leftarrow N + 1$
  $m \leftarrow \frac{M}{N}$
  $OCDM\_task\_update(\mathcal{D}_t, \mathcal{M}_t, m)$
  `/* We update the memories of old tasks, removing some elements, to give space in memory to the new task. */`
  diff $\leftarrow \frac{M}{N-1} - \frac{M}{N}$
  if $N \geq 2$ **then**
    for $t_{old} \in \mathcal{T}_{1 \cdots N-1}$ **do**
      $\mathcal{M}_{t_{old}} \leftarrow Memory\_Update(\mathcal{M}_{t_{old}}, \text{diff})$
    end
  end
end

Computational complexity

Next we compare BAT-OCDM and OCDM in terms of computational complexity. We also consider Dataset-based OCDM, a version of OCDM where, instead of sequential batches, all the data coming from the task are used for the memory update. This should improve performance, since the original batch-based memory update policy could ignore some samples, and thus it may lead to a less uniform label distribution in the final memory. On the other hand, in this way the update can no longer be performed on-line as in the original OCDM.

Algorithm 5: Dataset-based OCDM

Input: task stream $\mathcal{T}$, total size of memory M

$\mathcal{M} \leftarrow \{\}$ `/* Initialize the memory */`
for $t \in \mathcal{T}$ **do**
  $\mathcal{D}_t \leftarrow$ Get Dataset $\mathcal{D}_t = \{X_t, Y_t\}$ of task t
  $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{D}_t$
  if $|\mathcal{M}| \geq M$ **then**
    $\Omega \leftarrow \mathcal{D}_t \cup \mathcal{M}$
    $diff \leftarrow min(|\Omega| - M, M)$
    $\mathcal{M} \leftarrow Memory\_Update(\mathcal{M}, diff)$
  end
end

Table 5.9: Notation used to show the complexity of the algorithms

Notation	Description
D	size of the dataset associated to a task
T	number of tasks used in the experiment
M	size of the memory used

Tab. 5.11 summarizes the computational complexity of the strategies mentioned above, while more details can be found in the supplementary material [2].

	Task i	Total	Assuming $M = c \cdot D$
OCDM	$O(D \cdot M)$	$O(T \cdot D \cdot M)$	$O(c \cdot T \cdot D^2)$
Dataset-based OCDM	$O(D \cdot M + \frac{D^2}{2})$	$O(T \cdot [D \cdot M + \frac{D^2}{2}])$	O($\frac{2c+1}{2} \cdot T \cdot D^2$)
BAT-OCDM	$O(\frac{D \cdot M}{i} + \frac{M^2}{i-1})$	$O((\ln T + 1) \cdot M \cdot (D + M))$	$O((c + c^2) \cdot (\ln T + 1) \cdot D^2)$

Table 5.10: Computational complexity of the proposed approaches

As for the notation, we denote the set of tasks as $\mathcal{T} = \{t_1, \ldots, t_T\}$. For simplicity, we assume that each task corresponds to D samples and the memory $\mathcal{M}$ has a fixed size M.

The column *Task i* provides the complexity of the $i-$th memory update. For BAT-OCDM the complexity becomes smaller as i increases. In particular, the overall complexity, shown in column *Total*, is logarithmic in the number of tasks T for BAT-OCDM. For OCDM and Dataset-based OCDM instead, the complexity is linear in T.

The last column shows the complexity under the assumption that $M \leq D$, i.e. $M = c \cdot D$ where $c \in [0, 1]$. In this case, the complexity for all methods is quadratic with regard to to the dataset size D. The results on computational

complexity do not hold only for the DIL scenario, but also for the CIL scenario studied in the original paper proposing OCDM, showing the same speed improvement.

5.6 Experimental Setting

5.6.1 Metrics

Following the convention of multi-label classification [317] [101] we are going to use the macro f1 score to evaluate the performance of the model. Let s be the macro f_1 score, i.e. the average of the f_1 scores for each label: $s = \sum_{i=1}^{L} f_1(y_i, \hat{y}_i)$. Let $s_{i,j}$ be the performance of the model on the test set of task j after training the model on task i. To measure performance in the CL setting, we introduce the following metrics:

Average macro f1 The average macro f1 score $S_T \in [0, 1]$ at task T is defined as:

$$S_T = \frac{1}{T} \sum_{j=1}^{T} s_{T,j} \tag{5.17}$$

Average Forgetting $F_T \in [-1, 1]$, the average forgetting measure at task T, is defined as:

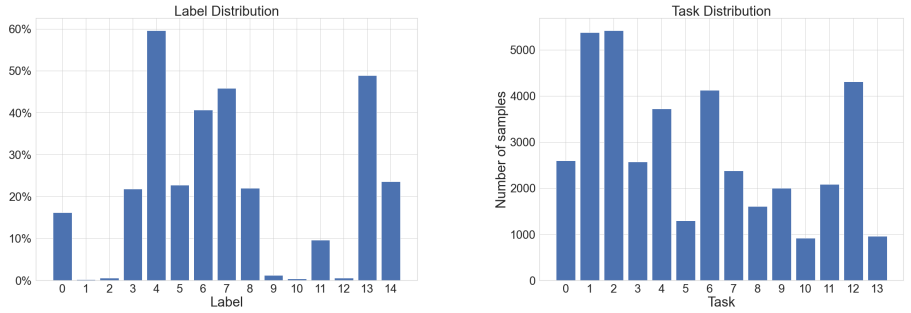
$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} max_{l \in \{1,\cdots,T-1\}} \frac{s_{l,j} - s_{T,j}}{s_{l,j}}. \tag{5.18}$$

With respect to the original definition used in [54], we are scaling respect to the maximum macro f1 score, as done in [140]; this is done to compare the forgetting among labels with very different scores. Notice that the closer the metric F_T is to 1, the higher the forgetting is.

5.6.2 Experiment setup

We test the proposed approach on a publicly available real industrial dataset originating from the monitoring of dairy products packaging equipment [297]. In the experiments, we consider the monitoring of 14 packaging machines deployed in different plants around the world as different tasks to learn in a CL fashion. The splitting of data in train and test is based on time where in the test there will be only samples belonging to the future of the machine.
To obtain the design matrix to train the CL-based multi-label classifier, we draw inspiration from [230], so we consider input windows having a length of 1720 minutes, and output windows of 480 minutes. As described in Sec. 5.3.1, we represent the input windows using normalized alarm counts.
The code for the BAT-OCDM approach is shared publicly[1] as well the dataset used [297].

[1] https://github.com/dallepezze/bat-ocdm



(a) Label frequency

(b) Number of samples per task

Figure 5.15: Experimental dataset: Labels and tasks statistics.

Fig. 5.15 shows in detail the frequencies of the labels and the number of samples for each task. Based on label frequencies, we split them into three groups to better assess the performance:

- high-freq: labels {4,6,7,13};
- medium-freq: labels {0,3,5,8,14};
- low-freq: labels {1,2,9,10,11,12}.

We used grid search to select the hyper-parameters for the rehearsal approaches. The *ratio of replay* is set to 0.5, i.e. at each batch 50% of data will be from the new task and 50% of the data will be from the old tasks. The *memory size* is set to 2000 samples which corresponds also to the 5% of the entire set of samples. Finally, the hyperparameter ρ used in OCDM and BAT-OCDM is set to zero, as suggested in the original paper.

All the experimental settings-related details are available at [2] for reproducibility.

5.6.3 Considered Continual Learning approaches

All the memory management strategies can be schemed according to the framework shown in Fig. 5.13. To assess BAT-OCDM, we compare it not only with OCDM (Alg. 3) and Dataset-based OCDM (Alg. 5), but also with the following approaches:

- Finetune: At each task t_i, the classifier is trained from scratch considering only the data from task t_i. There is no countermeasure against forgetting, so we consider this approach as a lower bound.
- Cumulative: At each task t_i, the classifier is trained from scratch based on all the data seen so far (i.e. from task t_1 to t_i), so there is no limit to memory size.

- Task-based Random: At each task t_i, an equal number of samples is selected at random.
- Reservoir Sampling: This stream-oriented approach from Online Learning selects samples at random at a fixed rate. Thus, it is expected to include more samples for tasks whose dataset size is bigger.

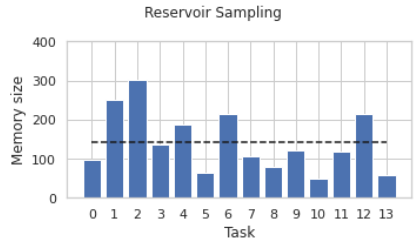
5.6.4 Results

We divide alarm codes into three categories based on their frequency in order to more thoroughly assess how the label frequency influences model performance. As stated, we then calculate the performance measures for low, medium, and high-frequency alerts independently. The computation time (measured in seconds) needed for the memory update for each suggested strategy is also evaluated in addition to the average macro f1 score and the forgetting measure.

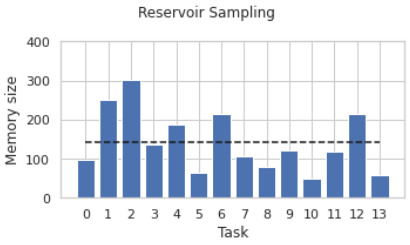
Performance

The experimental results are displayed in Tab. 5.11. Among the two approaches based on the random selection of the samples, there is no significant difference. It is clear from the results that task-based Random and Reservoir Sampling appear to work well on labels with high and medium frequencies. Instead, outcomes are unsatisfactory when low-frequency labels are taken into account, because of the poor representation of these labels in the memory. Unbalanced label distribution is one of the difficulties with multi-label, since it makes it harder for the model to concentrate on the rare labels and results in poor end performance. The best choice of samples to keep in memory also makes the multi-label in continual learning more challenging (with respect to the classic multi-label setting) because it must come up with a good strategy to keep a portion of samples from previous tasks in memory while attempting to have a balanced label distribution that gives a more equal representation to all labels. The OCDM method, in contrast, produces good performance for low frequency labels. These labels have a critical impact on the equipment as evidenced in [230]. However, this result comes with the trade-off of less effective performance on labels with high frequency. Instead, there is no relevant performance difference for the classification of medium-frequency labels.

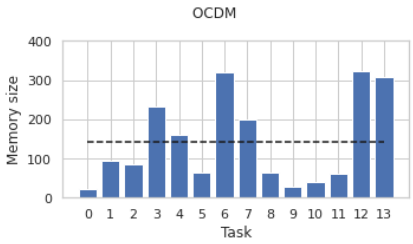
A possible explanation for this, as mentioned above, could be that some tasks are not stored in memory. Indeed, the algorithm focuses only on the balance among labels and not among tasks. This will be explained in more detail in Sec. 5.6.4 and in Fig. 5.15(b) with the task distribution in memory, where it can be seen the very low balance among tasks. For this reason, BAT-OCDM shows superior performance respect to OCDM for low-freq, medium-freq and high-freq labels. In particular, the drop in performance for higher frequency labels observed for OCDM is much lower. As for Dataset-based OCDM, even if it simultaneously uses all data from a task to find a better label distribution in memory, no significant difference appears in the performance. Instead, as



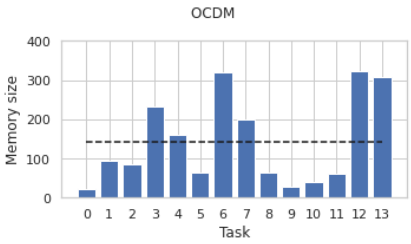
(a) Task distribution for RS



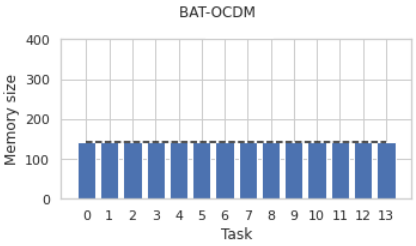
(b) Label distribution for RS



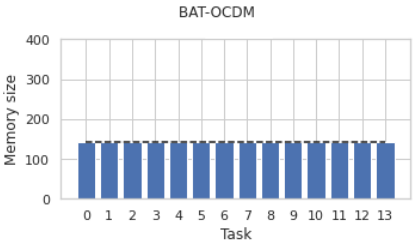
(c) Task distribution for OCDM



(d) Label distribution for OCDM



(e) Task distribution for BAT-OCDM



(f) Label distribution for BAT-OCDM

Figure 5.16: At top RS, at center OCDM and at bottom BAT-OCDM. At left it is shown the final distribution of the number of samples for each task in memory varying the approach.On the x-axis of the plots are represented the task IDs and on the y-axis the number of samples of each machine as absolute value. The dashed line in each plot represents the average number of samples among all the machines.
Instead at right it is shown the final distribution of the labels of the samples in memory varying the approach. In the title for each plot is indicated the name of the approach and KL distance respect to the target uniform distribution: the larger the value, the less balanced the labels.

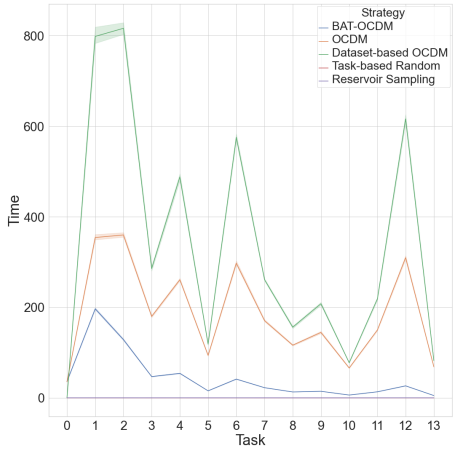


Figure 5.17: Computation time of each technique to handle the selection of the samples to keep in memory and remove from it. It is showed the time required during each task. On the y-axis is represented the time in seconds and on x-axis the current task.

anticipated by the complexity analysis, the time required to update the memory is much higher.

Balance among labels

We can see the label distributions from the RS, OCDM, and BAT-OCDM techniques in Fig. 5.16. Each plot also shows the KL distance to indicate how far apart the final distribution is from the desired distribution defined in Eq (5.16). We can observe that OCDM better approximates the target distribution. As opposed to OCDM, Random Selection displays a KL distance that is roughly ten times larger. This is a result of the heavily unbalanced dataset that was employed. When Fig. 5.15(a) and plot Fig. 5.16(b) are compared, it can be observed that because the selection is made randomly, the memory has the same label distribution as the dataset. So the distribution in memory will be very far from the target distribution with a high KL distance. Regarding BAT-OCDM, it considers samples from all of the available tasks. The resulting distribution shows a KL distance five times higher than OCDM and two times lower than RS.

Balance among tasks

The task distribution in memory achieved by the RS, OCDM, and BAT-OCDM approaches is shown in Fig. 5.16. By design, BAT-OCDM achieves a completely uniform allocation of the tasks (Fig 5.16(e)). Additionally, Task-based Random has the same characteristic because it maintains a separate memory for every task. For the Reservoir Sampling technique, the amount of samples in memory

Table 5.11: The table contains the performance for each approach. Above is the Average macro f1 S_T and below the Average Forgetting F_T. Based on the column, these metrics are calculated on a different set of labels. *Low*, *Medium*, and *High* are label sets grouped by the frequency of the labels, while *Total* consider all the labels together.

Approach	Total	Low	Medium	High	Time (s)
Finetune	0.19 (0.27)	0.04 (0.14)	0.21 (0.39)	0.38 (0.33)	-
Cumulative	0.39 -	0.17 -	0.42 -	0.7 -	-
Task-based Random	0.37 (0.11)	0.14 (0.09)	**0.4** (0.18)	0.69 (0.05)	0.3
Reservoir Sampling (RS)	0.37 (0.1)	0.14 (0.1)	**0.4** (0.16)	**0.69** **(0.04)**	0.3
OCDM	0.32 (0.2)	**0.2** **(0.04)**	0.32 (0.31)	0.49 (0.31)	2607.9
Dataset-based OCDM	0.32 0.2	**0.2** **0.04**	0.33 0.31	0.49 0.31	4702.3
BAT-OCDM	**0.38** **(0.09)**	**0.2** **(0.04)**	**0.4** **(0.15)**	0.64 (0.085)	**624.4**

for each task is proportional to the dataset size of that task as shown by Figs. 5.16(a) and 5.15(b). This is because the random selection occurs at a fixed rate during the stream of samples corresponding to the various tasks. About OCDM and Dataset-based OCDM, they have a similar distribution shown in the center of Fig. 5.16(c). In this case, we can observe that some tasks are almost absent in the memory, with the risk of being forgotten more quickly than the other tasks.

Computation times

In Tab. 5.11, we can examine the computing time to handle the memory for the RS, OCDM, and BAT-OCDM approaches. In comparison to traditional OCDM, BAT-OCDM requires four times less computing time to update memory. This outcome is consistent with the algorithm's anticipated theoretical complexity, which is logarithmic in the number of tasks T, as opposed to the original approach's linear in T. Dataset-based OCDM is slower than OCDM because it takes into account all of the samples from a task at once. In practice, it is twice as slow as OCDM.
Additionally, the computation time (in seconds) for each task for all strategies is shown in Fig. 5.17 For OCDM, we can observe that the time needed for each task is proportional to the size of the task's dataset. In other words, the larger the task's dataset, the longer it will take. When using dataset-based OCDM, we can observe that a task takes twice as long to complete as when

using OCDM. Finally, it appears that the BAT-OCDM computation time gradually reduces over time, which is consistent with the algorithm's logarithmic nature.

5.7 Conclusions and future works

This study is the first to address multi-label classification in the context of domain incremental learning, as far as we know. For multi-label classification, earlier Continual Learning methods have focused on the Class Incremental Scenario, in which additional labels are introduced into the task sequence. Instead, in Domain Incremental Learning, the set of labels remains the same while their distribution changes over time. To overcome this problem, we present BAT-OCDM, an effective replay-based method of managing the memory update. The proposed procedure exhibits higher performance than the simple adaptation of the previous techniques to the Domain Incremental Learning scenario, especially in the presence of class imbalance.

We validate the proposed approach on a real-world industrial Alarm Forecasting task stemming from the monitoring of packaging equipment. Experiments suggest the efficacy of the proposed methodology, especially on low-freq labels. Moreover, the complexity of BAT-OCDM is logarithmic in the number of tasks. The suggested method is therefore more effective than earlier ones that had linear complexity. Given the efficiency of BAT-OCDM, implementation on the Edge is a viable perspective. This would be a step towards the Tiny ML paradigm, which is becoming increasingly popular, also in the scenario of the Industrial Internet of Things [109].

Possible future research directions include validating BAT-OCDM performance in the Continual Incremental Learning scenario. Though the proposed algorithm allows for a significant reduction in the computation costs, the total complexity with regard to the dataset dimension is still quadratic. Therefore, we also envision further investigation of more efficient approaches. Finally, an additional point to research is how to improve the selection of samples from the task. For example, the proposed approach takes into account the relation among labels while training the MLP classifier, but it does not when choosing the samples to keep in memory for the next training stage. This information, if correctly included, may lead to further performance enhancement.

Chapter 6

Anomaly Detection for Continual Learning

Anomaly Detection is a relevant problem that arises in numerous real-world applications, especially when dealing with images. However, there has been little research for this task in the Continual Learning setting. In this work, we introduce a novel approach called SCALE (SCALing is Enough) to perform Compressed Replay in a framework for Anomaly Detection in Continual Learning setting. The proposed technique scales and compresses the original images using a Super Resolution model which, to the best of our knowledge, is studied for the first time in the Continual Learning setting. SCALE can achieve a high level of compression while maintaining a high level of image reconstruction quality. In conjunction with other Anomaly Detection approaches, it can achieve optimal results. To validate the proposed approach, we use a real-world dataset of images with pixel-based anomalies, with the scope to provide a reliable benchmark for Anomaly Detection in the context of Continual Learning, serving as a foundation for further advancements in the field.

6.1 Anomaly Detection

6.1.1 Anomaly Detection

Anomaly Detection (AD) is an important and challenging problem in the field of Machine Learning (ML) and Computer Vision. Anomalies are patterns characterized by a noticeable deviation from the so-called normal data, where normal means compliance with some typical or expected features [330]. To effectively detect anomalies, it is essential to clearly understand what constitutes normal behavior and how to differentiate it from abnormal patterns. Anomaly Detection is a relevant problem that arises in numerous real-world applications, especially when dealing with images. In particular, visual anomaly detection is an important and challenging problem in machine learning and computer vision.

AD differs from the assumption of a static and closed system that most existing machine learning methods are based on. AD tries to research how machine learning models can deal with unknown and uncertain information under the open and dynamic system environment [341]. With the assumption of an open environment, the learning systems developed for anomaly detection are usually expected to leverage the knowledge from the knows (normal data and patterns) to infer the unknowns (abnormal or novel patterns that differ from the normal ones). AD approaches usually extract, characterize and model the patterns with the available normal data and then develop anomaly detectors to discover novel or abnormal patterns in the newly observed data.
AD not only has interesting theoretical aspects connected to an open environment. It also has important applications in real-world scenarios. For example, in the field of manufacturing, defect detection can apply an AD approach to recognize defects in products in a manufacturing process. In the field of medical image analysis, it can be used to detect lesions or other abnormalities in medical images. In the field of cybersecurity, it can be used to detect intrusion and attacks on the network or data.

6.1.2 Benchmark considered

Recently, in [26] is developed a benchmark dataset for evaluating unsupervised image anomaly detection algorithms: MVtec AD. The dataset consists of various texture and object categories, and contains more than 70 different types of anomalies. The authors evaluated many methods based on image reconstruction and feature modeling on the proposed dataset.
We propose to use the MVTec Dataset [26] as a benchmark to study the performance of architectures and CL strategies for Anomaly Detection. The MVTec AD dataset is a recent and extensive industrial dataset that includes 5354 high-resolution images divided into 15 categories: 5 types of textures and 10 types of objects. The training set only contains normal images, whereas the test set includes both defect and defect-free images. The image resolutions range from (700,700,3) to (1024,1024,3) as resolution. In our ADCL framework, we will consider a sequence of 10 tasks. Each task corresponds to one of the 10 classes associated with objects. In this study, the model is presented with a series of tasks, and must be able to identify which pixels in the image are abnormal for each object. The MVTec dataset used in this study is more challenging and complex than the commonly used datasets MNIST and CIFAR-10, which are often used in Continual Learning literature [61, 80, 178]. These datasets have smaller image sizes, (28,28) for MNIST and (32,32) for CIFAR-10, and fewer images, making them not as representative of real-world scenarios as the MVTec dataset.

6.1.3 Strategies for Anomaly Detection

The first important categorization is whether the method uses Deep Learning. In classic approaches, after obtaining the shallow features of images by hand,

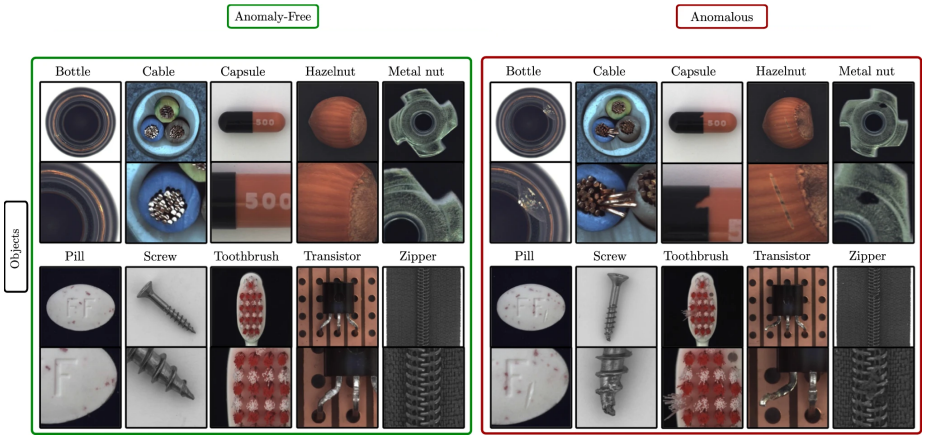


Figure 6.1: Image examples from the MVTec Dataset AD. For each object is shown a normal sample (in green) and an anomalous sample (in red). For each sample, the entire object is displayed, as well as a zoomed-in view of the region containing the defect. The samples illustrate the range of variations and abnormalities that can be found in the dataset.

such as the gray value, SIFT [179], and HOG [68], AD attempts to develop different detection mechanisms based on statistics or traditional machine learning methods, such as density estimation, one-class classification, and image reconstruction. The development of deep learning techniques, especially convolution neural networks, has been successful both in low-level and high-level computer vision tasks [367, 340, 273]. Then relevant research gradually shifted the attention to combine the powerful representation capability of DL with the problem of AD.

Another categorization is based on whether supervised information is available or not, visual anomaly detection can be divided into two research areas: supervised and unsupervised. A review of AD approaches is provided in [341]. However, the unsupervised learning field is more common for most practical application scenarios. This is because in many real-world scenarios, abnormal image samples are very rare and difficult to collect, so there is no effective supervision information to use. Moreover, abnormal patterns are usually variable in shape, color, and size and do not have stable statistical laws. All these will make it difficult for the supervised learning model to capture enough statistical information or salient features about abnormal image patterns [341].

Visual Anomaly Detection can also be organized considering another aspect, the granularity of approaches. Visual Anomaly Detection can be grouped into two categories: image-level and pixel-level. Image-level usually only focuses on whether the whole image is normal or abnormal. Instead, pixel-level requires detecting or locating the abnormal regions in the image.

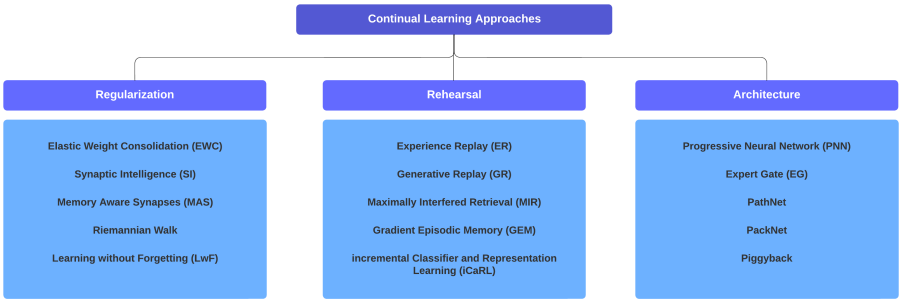


Figure 6.2: Categorization of AD approaches for Unspervised Learning

6.1.4 Image-level Anomaly Detection

Unsupervised image-level anomaly detection methods can be roughly divided into four groups: density estimation, one-class classification, image reconstruction and self-supervised classification.

Density estimation

The density estimation method first estimates the probability distribution model of the normal images or features, then detects and identifies whether the newly observed image is abnormal or normal by testing against the established distribution. Methods based on density estimation usually assume that if the testing image or image feature does not meet the probability distribution model estimated with the normal image samples, it will be classified as an anomaly.
Typical density methods include parameter distribution estimation, such as Gaussian model and Gaussian mixture model, non-parametric estimation methods, such as nearest neighbor and kernel density estimation method [139].
However, estimating a reasonable probability density requires many training samples. And when the feature dimension of the samples is very large (such as image data), this problem of the number of training samples becomes particularly prominent. In addition, the scalability of these classic models is poor.

One-class classification

One-class classification is to classify a single class concretely, which attempts to construct a decision boundary of the target class (normal images) in the feature space. Classic approaches are one-class support vector machines (OCSVM) [262], and support vector data description (SVDD) [293]. One-class methods do not need to estimate the probability of each sample point of the image distribution, so they do not require a large number of training samples. But they still suffer from the problems of dimension disaster and scalability. An example is shown in [225], which proposes a one-class classification method based on transfer learning that fine-tunes the pre-trained convolution network to extract

discriminative image features and then uses the nearest neighbor classification method to construct the classifier.

Image reconstruction

Image reconstruction approaches map the image to a low-dimensional vector representation (latent space), and then try to find an inverse mapping or reconstruction for the original image [342]. The reconstruction-based approaches for anomaly detection assume that the reconstruction errors of the normal images are small, while that of abnormal images are larger. The widely used reconstruction model is autoencoder. Autoencoder is proposed by [117] and the basic idea behind it is redundancy compression and non-redundancy separation. Autoencoder is a neural network that has a narrow middle hidden layer. It tries to compress the input data through the hidden layer and then regenerates the input. Because the hidden layer is very narrow, the network may compress the redundant information in the input data while retaining and distinguish the non-redundant information.
In addition, some methods suggest increasing the difficulty of image reconstruction. Typically, some transformations for the input image are first performed, such as color removal or geometric transformations. Then the model is trained to reconstruct the original input image with the incomplete input image on which the transformation is made. This method can effectively increase the reconstruction difficulties for abnormal images, so their reconstruction errors are usually large. As a result, it can effectively increase the disparities between normal images and abnormal images, thus improving the performance of anomaly detection.

Self-supervised classification

Visual anomaly detection by self-supervised classification mainly owes to the powerful visual feature representation capability of self-supervised learning. Self-supervised learning usually leverages certain auxiliary tasks (pretext) and attempts to mine the available supervision information from large-scale unsupervised data. It takes full use of the self-constructed supervision information to learn the visual representation typically by a deep convolutional neural network. Then the representations can be transferred to many downstream tasks, such as image classification, object detection, and anomaly detection [9].
The idea behind self-supervised classification for anomaly detection is that through self-supervised training, models can learn unique and more significant characteristics and features of normal samples [342]. The representations learned for the target objects not only reflect their color, texture, and other low-level features but also the high-level features such as the location, shape, position, and direction. By learning these features only from normal samples, the model can effectively discern abnormal samples without such characteristics.

6.1.5 Pixel-level Anomaly Detection

In terms of different anomaly detection mechanisms, approaches for unsupervised pixel-level anomaly detection can be roughly divided into two categories: image reconstruction and feature modeling.

Image reconstruction

Reconstruction-based methods include neural network architectures like autoencoders [26, 66, 357], variational autoencoders [173] and generative adversarial networks [8, 261]. The idea is that autoencoders will ignore minor details that don't belong to the training distribution and reconstruct the image without the defects. A typical method of image reconstruction is to compress and reconstruct the input image with the deep convolution autoencoder [27]. It first learns to do the reconstruction of normal images. Then, potential anomalies are detected by evaluating the pixel difference between the input image and the reconstructed image, where pixel-level l2-distance [107] and image structure similarity measure (SSIM) [324] can be used to measure the disparities before and after the reconstruction. Reconstruction-based methods learn to reconstruct normal images during training. However, during the evaluation, images with defects are not expected to be reconstructed well. Consequently, they should exhibit a higher reconstruction error than normal images. In other words, i.e., it cannot reconstruct the abnormal image as well as the normal ones. The deep generative models based on variational autoencoder (VAE) [142] and generative adversarial network (GAN) [105] can also be used as the reconstruction model. In [74] is proposed to detect anomalies by comparing the difference between the test image and its nearest neighbor normal image obtained by optimization. This method needs to perform an iterative search process, they are inefficient for practical applications.
Moreover, some methods propose to increase the difficulty of reconstruction [358]. They remove some information from the input image and then make the autoencoder reconstruct the input image with the incomplete or degraded input. The information degradation process can effectively increase the difficulty of abnormal image reconstruction, increasing the anomaly score between the normal and abnormal samples, thus improving the detection performance.
For example, in [358] an approach based on inpainting is proposed called RIAD. The idea is to hide a portion of the input image and train the model to reconstruct it. In the same spirit, it is possible to also consider Super Resolution to solve Anomaly Detection. In this method, the image is scaled to a lower resolution and then rescaled to the original size, resulting in a loss of information about the original image.
Image reconstruction for pixel-level anomaly detection is very intuitive. But because it detects potential anomalies by evaluating the pixel-wise differences in the pixel or image space, it usually is expected to regenerate high-quality images for comparisons [342]. However, high-quality image generation itself is still

a challenging task. For example, the reconstruction approaches usually struggle to regenerate the sharp edges and complex texture structure for images. As a result, large reconstruction errors often appear in the edge or texture regions, leading to many false abnormal alarms.

Feature modeling

Instead of detecting anomalies in the image space, the feature-based methods detect anomalies in the feature space. In other words, it uses pre-trained neural networks to extract meaningful features from images. These approaches try to construct an effective representation of the local regions of the image by using the handcrafted features, the representation learned by neural networks [266]. Then, many machine learning models such as sparse coding [47], Gaussian mixed model [35], and Kmeans clustering [211] can be used to model the feature distribution of normal images.
For anomaly detection, if the regional feature corresponding to the local region of the test image deviates from the modeled feature distribution, this region will be labeled as abnormal. To improve the detection performance, the multiscale model ensemble strategy [48] is usually adopted, which combines the results of multiple single models derived from different image region sizes. Besides, to locate the abnormal region in the image, feature-based methods usually need to divide the image into many small image patches first and then model and detect the anomalies in the image patch level. Therefore, it is very time-consuming during both training and testing, especially when the deep neural network is required to extract the deep image features. Mover, because each local region of an image is evaluated independently, it may not be able to infer anomalies by leveraging the global context information of the image.
SPADE [65] first performs image feature extraction based on a WideResNet50 [356] trained on ImageNet [149], then retrieves the K-nearest normal images, and finally achieves pixel alignment with deep feature pyramid correspondences. Even if it has virtually no training time, the k-nearest neighbour clustering procedure is slow at test time with highly multidimensional data. Similarly to SPADE, PaDiM [73] relies on ImageNet pre-trained feature extractor with multi-scale pyramid pooling. However, instead of k-nearest neighbor clustering, it computes an anomaly score based on the Mahalanobis distance. To enhance segmentation results, the metric parameters are estimated for each feature vector from the pooled feature maps. Normalizing Flows (NF) [246, 143] are a powerful tool for modeling complex distributions, such as those that arise in Anomaly Detection tasks. By embedding normal data into a high-dimensional standard normal distribution, it becomes possible to use the probability of the data under the distribution as a measure of its normality. This can be useful for identifying and localizing anomalies in data. For example, in the case of image data, normalizing flows can be used to model the distribution of image features, and then use the probability of a given image under the distribution to identify and locate anomalies in the image. FastFlow and CFLOW-AD are two examples of methods that use normalizing flows for anomaly detection, with

FastFlow focusing on improving anomaly localization by using a 2D flow model.

6.2 Compressed Replay

Continual learning provides ML models with the ability to update and expand their knowledge over time, with minimal computation and memory overhead [72]. Thus, an effective CL solution is expected to have low forgetting, require low memory consumption and be computationally efficient.

CL strategies are usually classified into three categories: replay-based [269, 250, 58] methods, regularization-based [233, 240, 110] methods and architecture-based [277, 186] methods. The related literature suggests that the Replay (also known as Rehearsal) approach appears to be the most effective and practical solution to reduce CF [343, 223, 42, 140].

The simplest implementation, known as Experience Replay (ER) [250, 53] consists in storing some samples in the raw format selected randomly from the current task and passing them during the subsequent tasks. Therefore, during training, each batch of data from the current task is combined with a batch of data from the previous tasks (sampled among them with the same probability). However, the memory storage limit is a significant constraint, the smaller the memory size, the further we are from the original training distribution of old tasks.

Methods for reducing the storage cost of old training samples have recently been developed for this problem. We refer to them as **Compressed Replay**, where the goal is to learn a compressed data representation. The most important and vast family of this field is *Latent Replay*. Latent Replay methods propose to store activations from an intermediate layer of a neural network and Replay these representations to prevent forgetting. Typically, in a classification task, these encodings are used as Replay starting from a specific layer, as done in [223], where lower layers are trained at a slower pace to avoid latent representation shift. In [216] authors propose to use pre-trained models to extract features and use them for training a classification model. They study the performance of Replay and Latent Replay using several pre-trained models. In [113], the lower layers are frozen, and only the upper layers are trained. Moreover, to store more samples, they propose to use Product Quantization (PQ) to compress and save the features efficiently, obtaining a compression factor of around 50. In [125] authors propose to apply feature adaptation so that features learned for the old task remain consistent with the new feature space.

The previous works were done for the classification problem, and the compression factor depends on the size of the layer chosen to extract the features to store in memory, so different compression factors can be obtained depending on the chosen layer. However, because we are interested in generative problems, we want to be able to compress the original samples while retaining the ability to reconstruct the original image in the output. For instance, in [20] authors propose EEC, a Generative Replay approach that also memorizes the centroid and covariance of each class to make the training more stable.

In [43] authors introduce the use of Adaptive Quantization Modules(AQM) to Continual Compression on the LiDAR dataset and also allow the selection of the trade-off between the compression factor and quality of reconstructed images.

Many generative models, as described in Section 6.5, can be used for Latent Replay. However, no research on generative models regarding Latent Replay was conducted. Therefore, as an additional contribution, we also provide an accurate comparison of several generative models for Latent Replay, examining their efficacy in terms of compression factor and image quality. Moreover, we propose a novel approach for Compressed Replay called SCALE and described in Section 6.4. This approach shows a high level of compression and at the same time, a high level of quality in the generated images.

6.3 Anomaly Detection for Continual Learning

Despite the fact that it is extremely relevant in practice, only a few works in the literature addressed the AD problem in the CL setting.

In [330], the authors propose using the Variational Autoencoder (VAE) with the Generative Replay approach to learn continually new classes while also performing AD. The datasets used in the evaluation are KDD Cup 1999 and MNIST, which have been artificially adapted to AD by marking one of the classes as anomalous. In [13], using the same dataset KDD Cup '99 and other datasets belonging to Network Intrusion Detection, the authors study the problem of AD formulated as a binary classification supervised learning. The authors of [191] propose a regularization-based approach for continual AD in manufacturing, and evaluate it on a real industrial metal forming dataset.

As for AD for images, the majority of work in this field focuses on predicting whether an image is normal or abnormal. However, in practice, anomaly localization is frequently required. Not only is it necessary to explain the predictions, but it is also necessary to perform root cause analysis. As a result, we investigate a complex image dataset such as MVTec, which contains pixel-based anomalies. We also investigate a wide range of models and assess their performance.

6.3.1 Proposed Framework for ADCL

Anomaly detection is a challenging task, particularly when the data contains distinct objects. In this case, abnormal samples are often much closer to normal ones than to samples from other classes. This is illustrated in Figure 6.3, where different colors represent different object classes, normal samples are depicted as circles, and anomalies are depicted as crosses. The closeness of the anomalies to the normal samples makes it difficult to identify and distinguish them using traditional methods.

The task becomes even more challenging when we want to identify if the single pixel of an image is an anomaly or not in the CL setup where new objects

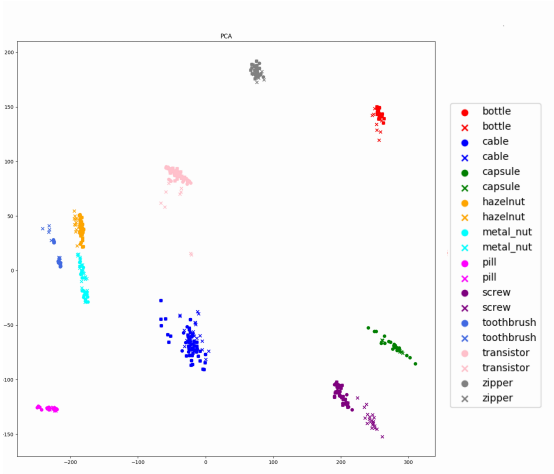


Figure 6.3: We show the PCA on the original images where the first two dimensions capture 73% of the total variance. Each class is depicted in a different colour, with circles representing normal images and crosses representing abnormal images. As can be seen, the separation between normal and abnormal samples coming from the same class is much lower than the distance between classes. The actual scenario is even more challenging since we aim at detecting pixel-level instead of image-level anomalies.

must be learned subsequently.

The goal is to train a neural network model that can detect anomalies while also retaining the knowledge it has learned from previous tasks. The model is trained on a sequence of tasks, each of which corresponds to one or more object classes. During training, the model learns the distribution of normal (i.e., anomaly-free) images for each task. Then, during evaluation, the model is tested on a dataset that contains data from all tasks, both normal and abnormal. This allows the model to detect anomalies in the test data by comparing it to the learned distribution of normal data.

In more formal terms, ADCL is a method for learning a model that can detect anomalies in data. The model is trained on a sequence of tasks with a total of T tasks. Each Anomaly Detection task t corresponds to a dataset D_t. Each dataset D_t consists of pairs of data (X_t, Y_t), where X_t is a set of images or more formally $X_t \subset \mathbb{N}^{H \times W \times 3}$ where H and W denote the spatial dimensions and $t = 1 \cdots T$. Y_t is a set of labels for each pixel of the image, indicating whether the pixel is normal (0) or anomalous (1). The goal of the model is to learn a mapping $f_\theta : X \to R^{H \times W}$ from the space of images X to a probability vector. This allows the model to assign a probability to each pixel in an image, indicating how likely it is to be normal or anomalous.

In the field of Continual Learning, there are three main scenarios that are commonly studied: Task Incremental Learning (TIL), Domain Incremen-

tal Learning (DIL), and Class Incremental Learning (CIL) [309, 184]. In our problem, we are dealing with the DIL scenario because the classes of output remain fixed independently by the change of task (i.e. there are always only two classes, normal and abnormal, for each pixel of the image), while the input distribution changes between tasks i.e. $P(X) \neq P(X')$.

We introduce a general framework for ADCL. We present a general framework for the task of Anomaly Detection in Continual Learning (ADCL). The goal of this framework is to enable incremental learning of new objects while also detecting anomalies in images from previous tasks. The proposed framework, depicted in Fig. 6.5, is broken down into two modules:

1. **Memory Module**: *This component's goal is to retain the memory of previous tasks.* Depending on the type of CL strategy considered, such a module may take various forms. In the case of Replay, for example, such a module consists of a limited memory containing some of the images of previous tasks. In the case of Latent Replay, the module will consist of an autoencoder (AE) and a set of compressed samples memorized through the latent space. In the case of regularization-based strategies, such as EWC [145], the Memory Module will be composed of the importance values associated with each weight of the architecture used in the AD Module.
2. **AD Module**: *It is the architecture used to detect anomalous pixels.* The chosen architecture can be any previously proposed approaches, like AE, VAE, and GANs for reconstruction-based approaches or FastFlow, and CFlow for embedding similarity-based.

It is important to note that the Memory Module and the AD Module are typically described as separate functional entities, but the same model can serve both functions. For example, many reconstruction-based methods can generate images of old tasks for use during the training of a new task, the ability to reconstruct images can also be used for anomaly detection by comparing the reconstructed image to the original image.

The Memory Module is a key component of a Continual Learning system, responsible for storing and updating information about previous tasks. This module is updated using a combination of data from the current task and a batch of old data from the previous task, as depicted in Fig. 6.5. In the case of classic Replay, the Memory Module simply stores images. With Latent Replay and generative models such as CAE or VAE, the Memory Module stores the model of the previous task and samples in the latent space, which allows for more efficient storage of images compared to Replay. It is also possible to use Generative Replay with VAE and GAN models, but it has been shown to result in a high degeneration [159].

6.4 Our Approach: SCALE

Within the above-introduced framework for Visual Anomaly Detection in Continual Learning, we proposed SCALE (SCALing is Enough), a Super Resolution-

based approach, as a Compressed Replay technique. Super Resolution aims to convert a low-resolution image with coarse details into a high-resolution image with improved visual quality and refined details [16]. In practice, as SR model, we consider the Pix2Pix [126], a general-purpose model for image-to-image translation composed of a conditional GAN [202]. The cGAN, unlike the classic GAN [202], is conditioned not only by random noise but also by an input image. The GANs have the flaw of having a high forgetting with Generative Replay (where the images are generated using random noise) [159]. We show how conditioning to an input image plus the noise can result in a method that can still generate images with a low forgetting even though the output still has a random component. Though such architecture is not the state-of-the-art of the SR field, it is sufficient to obtain good results and justify its use in this context, as shown in Section 6.6. Therefore, in our work, the SR model is studied with three final objectives:

1. Since it is the first time that the SR problem is studied in CL, we provide an evaluation in terms of reconstruction of the original images and how this approach behaves in terms of forgetting.

2. We also investigate the use of memory in combination with AD approaches that are by nature not able to perform the function of memory.

3. We consider the efficacy of SR in terms of AD.

The proposed schema for using the SR model in Continual Learning is shown in Fig. 6.4 and can be summarized in the following steps:

a) When a new task is received, the images are resized to a lower resolution and a copy is saved in memory. The objects saved for a task will remain unchanged during training, as opposed to Replay and Latent Replay.

b) The process for handling the images of the new task is depicted in the yellow portion of Fig. 6.4. The other image is resized again (which reduces its quality) and sent to the SR model to learn how to reconstruct the original image.

c) The procedure to obtain images of old tasks to be used in the training of current tasks is depicted in Fig. 1 as the light blue part. To obtain images from old tasks to use in training the current tasks, the scaled image saved in memory of shape (32,32,3) is rescaled to its original size (256,256,3) and labeled as $input_2$ in Fig. 6.4. Then it is passed to the SR model to obtain an image quality similar to the original, such output is labeled as $target_2$ in Fig. 6.4. The SR model of the current task is then trained to reconstruct the rescaled image ($input_2$) using the reconstructed image ($target_2$).

By memorizing the scaled images and increasing the resolution when needed, it is possible to achieve a compression factor. The value of such compression depends on the final quality that is desired, and it will depend on the final scope.

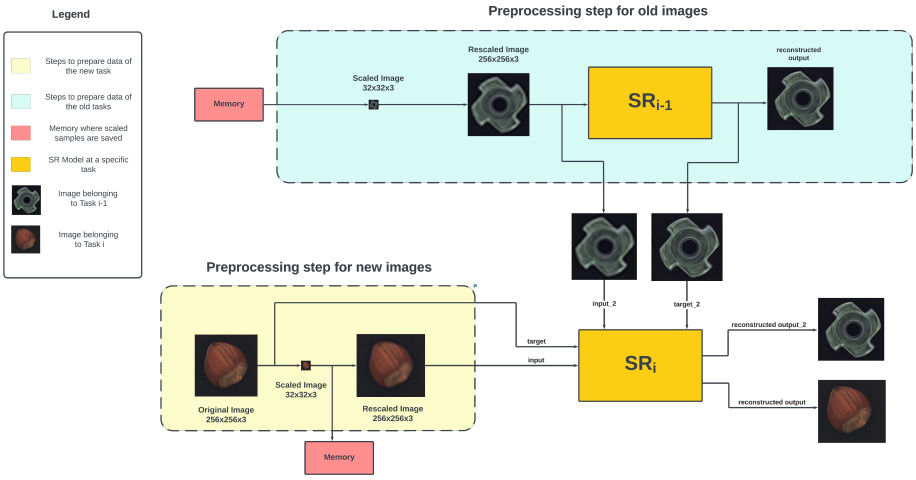


Figure 6.4: Scheme of the proposed approach for compressed Replay called SCALE. To perform Super Resolution (SR), we use a Pix2Pix architecture. First, the current task images are scaled (compressed) and saved in memory (yellow area). Then, they are rescaled (*input*), and the model is trained to reconstruct the original image (*target*). Instead, for old tasks, the images are taken from memory, rescaled, and reconstructed using the SR model from the previous task, in order to retain old knowledge (light blue area). After that, the rescaled blurry image ($input_2$) is fed into the current SR model, with the target corresponding to the image reconstructed by SR in Task $i-1$ (i.e., $target_2$).

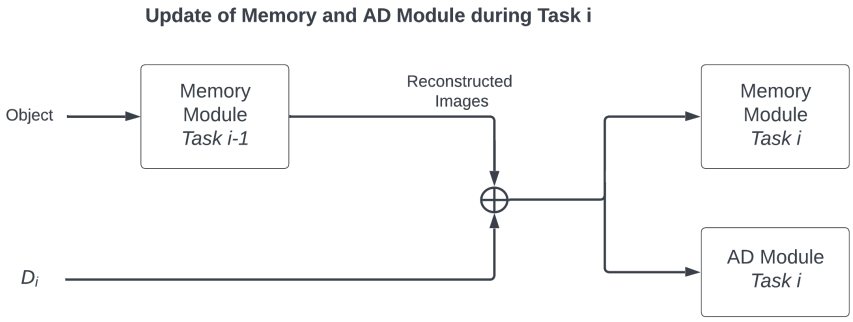


Figure 6.5: The proposed framework is composed of 2 modules: memory and Anomaly Detection. The Memory Module will be updated using the data from the current task as well as a batch of old data retrieved using the Memory Module of previous task.

We are going to present its results in terms of performance for Anomaly Detection in Section 6.6 and in terms of the quality of reconstructed images in the Section 6.6.2. Because this is the first time that SR has been studied in the context of CL, we present additional results for such a model in Section 6.6.3 where we explain why it works compared to classic GANs and its advantages. Among the results, it is shown to be advantageous to keep the memory of a task fixed over time rather than changing continuously as is typical in Latent Replay.

6.5 Experimental Settings

6.5.1 Considered CL Strategies

We are going to evaluate the different architectures in terms of performance for both the Memory Module and the AD Module. Such evaluation will be performed considering different CL strategies that will be applied to the models.

- **Single Model**: As an upper bound, we train a different model for each task.
- **Fine-tuning**: As a lower bound, we consider the fine-tuning approach, in which a model is presented sequentially only with data from the current task.
- **Ideal Replay**: As a benchmark Continual Learning strategy, we consider an approach where we randomly select n images from a task and create a training batch with half data from the current task and half data from previous tasks. The constraint on processing power is respected, but not

the constraint on memory. Indeed, this approach takes into memory all samples seen so far. This implies that the memory can capture the original distribution completely. We refer to this approach as *Ideal Replay*, and we propose to consider it as an additional upper bound for CL approaches.

- **Replay**: We consider a constrained memory size of n images such that: $n << |D|$ where D represents the entire dataset. In the experiments, n is set to 40 images, representing less than 2% of the total dataset. This means that the total memory size is limited to 40 images, i.e., four images for each class at the end of the training.

- **Compressed Replay**: When dealing with Replay, we should keep in mind that the risk of overfitting is high when only a few samples represent a task. Given that memory size can influence final results, the ability to save more samples has the potential to reduce forgetting because the distribution in memory of an old task is more similar to the original one (assuming that the quality of compressed samples is high enough). So in Compressed Replay, the images are compressed. For example, in Latent Replay, we consider their representation in the latent space. Similar to Replay, we consider the constraints on processing power and memory size, but with Compressive Replay, we try to keep more images in memory by utilizing some compression method.
  When considering Latent Replay, because we will be studying architecture such as autoencoders, it is more natural to choose the latent space that corresponds to the Encoder's output and the Decoder's input, i.e. the bottleneck layer, which has the lowest dimensional space.

- **Generative Replay**: In the case of Generative Replay [269], we use models such as GANs to generate images without the need to store any additional information other than the model itself. This can be very useful because it eliminates the need to save any data in memory. However, in previous experiments [159], the Generative Replay had a high level of forgetting and much of the information was quickly lost.

To ensure a fair comparison between Compressed Replay and Replay, we use the same amount of memory in terms of bytes for both approaches. We define the memory size as C, which is the number of bytes needed to store n images using the Replay strategy. For Compressed Replay, a number of compressed samples will be stored, taking up a maximum of C bytes of storage.

In general, we will compare the different methods, with upper bounds included, taking into account that, on the practical side, the only valid CL approaches are *Replay*, Compressed Replay, and Generative Replay, since they are the only ones with a memory size and processing power constraint.

We will be comparing the Continual Learning strategies introduced above in terms of their performance in two key areas: Anomaly Detection and the

quality of the reconstructed images. For the Compressed Replay strategy, we will also be considering the compression factor as a key metric. A higher compression factor means that more samples can be stored, but we must also take into account the quality of the reconstructed images. If the compression factor is high, but the quality of the reconstructed images is poor, then the distribution of the reconstructed images will be significantly different from the original distribution. It is important to find the right balance between these two factors to achieve optimal performance.

6.5.2 Considered Architectures in Memory and Anomaly Detection modules

The following models were considered as both Memory and AD modules:

- **Convolutional Auto-Encoder(CAE)**: It has a latent space of shape (512,4,4) i.e. of dimension 8192, implying a compression factor value equal to 6 (assuming that we are working with 4 bytes for each value). Taking into account an input shape (256,256,3) we can simply calculate the compression factor as $\frac{196608}{8192 \cdot 4} = 6$. We will memorize the activations of the CAE obtained in the latent space in the case of Latent Replay.

- **Variational Auto-Encoder(VAE)**: Using the VAE (Variational Autoencoder) architecture, we were able to achieve good results with a smaller latent space compared to the CAE (Convolutional Autoencoder). Specifically, we obtained a latent space dimension of 256 and a compression factor of 96. It is worth noting that the VAE architecture can be used not only with the Latent Replay strategy but also with Generative Replay, which allows the model to store information without any actual samples. However, as previously mentioned and shown by our results, the quality of the images generated by VAE with Generative Replay degrades quickly.

As for the approaches that can be used as AD Module only, we consider:

- **Inpaint**: We tried to use the same architecture of RIAD [357] in our study, but found that the architecture was very sensitive to the continual learning setting and we were unable to obtain satisfactory results. As an alternative, we adopted the spirit of the RIAD approach (i.e., inpainting) and proposed a different architecture, a Pix2Pix model [126], for the inpainting task. In our version, training is done by masking random areas of the images. During reconstruction, the same image is fed into the model multiple times (with different masks each time), and averaging is used if the same pixel is masked multiple times.

- **FastFlow**: We use FastFlow [352] in combination with the WideResNet50 [356]. This approach, described in Section 6.1.5, learns a Normalizing Flow that models normal images.

It should be noted that this is one of the few studies on Normalizing Flow architectures in CL setting. To the best of our knowledge, only two other studies were done [232, 144].

6.5.3 Evaluation Metrics for Anomaly Detection

There are several metrics commonly used to evaluate the performance of Anomaly Detection models on the MVTec AD dataset. The ROC AUC score at the pixel and image levels is the most widely used, but other metrics such as f1 score and IoU are also taken into account. In the Anomaly Detection literature, it is generally believed that metrics such as the f1 score are more fair when dealing with imbalanced datasets, which is often the case with Anomaly Detection [256]. Therefore, in this study, we use the f1 score to assess a model's ability to detect anomalies. A well-performing model should have a high f1 score and a low FID score.

6.5.4 Evaluation of image reconstruction quality

To evaluate the quality of the reconstruction performed by the models used in the Memory Module, we will use the Fréchet Inception Distance (FID) [115], which is a commonly used metric for evaluating generative models [159]. This type of evaluation is useful for analyzing the performance of all architectures that can function as Memory Modules under various CL strategies. The FID is calculated as:

$$FID = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \tag{6.1}$$

The statistics (μ_r, Σ_r) and (μ_g, Σ_g) are the activations of a specific layer of a discriminative neural network trained on ImageNet for real and generated samples, respectively. A lower FID indicates that real and generated samples are more similar, as measured by the distance between their activation distributions. Beyond the models that can be used in the Memory module, we will evaluate the image reconstruction quality also for Inpaint, even if it lacks the ability to compress them and to act as a Memory Module.

6.5.5 CL Metrics

Following the convention of Continual Learning we are going to considered the average value at the end of the training and the forgetting. Let $s_{i,j}$ be the performance f1 of the model on the test set of task j after training the model on task i. To measure performance in the CL setting, we introduce the following metrics:

Average f1 The average f1 score $S_T \in [0, 1]$ at task T is defined as:

$$S_T = \frac{1}{T}\sum_{j=1}^{T} s_{T,j} \tag{6.2}$$

Average Forgetting $F_T \in [-1, 1]$, the average forgetting measure at task T, is defined as:

$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} max_{l \in \{1,\cdots,T-1\}} \frac{s_{l,j} - s_{T,j}}{s_{l,j}}. \tag{6.3}$$

With respect to the original definition used in [54], we are scaling respect to the maximum f1 score, as done in [140, 228]; this is done to compare the forgetting among tasks with very different scores. Notice that the closer the metric F_T is to 1, the higher the forgetting is. What it was defined above for the metric f1 will be redefined in a very similar way for the metric FID, with the consideration that the goal of FID is to be minimized while with f1 we want to maximize it.

6.6 Results

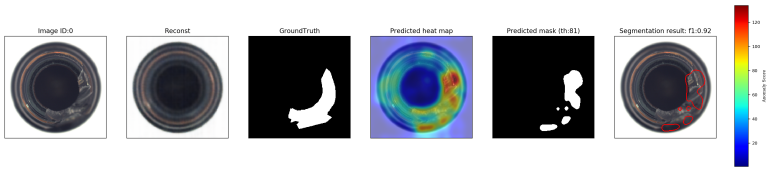
6.6.3. Each test is repeated ten times for each cell in the result tables, and only the average is shown. The plots shown in the Section Results will display the mean and variance of the runs. For the sake of reproducibility, the code used in the experiments is available in a public repository online[1].

6.6.1 Quality of the Anomaly Detection

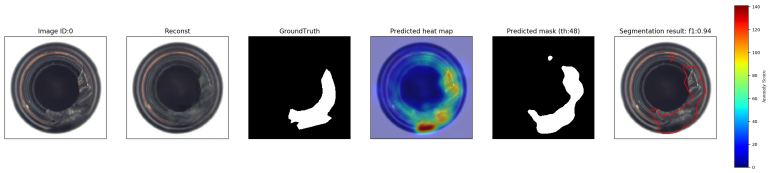
Regarding the AD results, it should be kept in mind that CAE, VAE, SR, and Inpaint are all reconstruction-based methods, whereas FastFlow is an embedding similarity-based method that uses pre-trained models for feature extraction and is considered state-of-the-art in the field. Embedding similarity-based methods, such as FastFlow, are incapable of acting as Memory Module because they produce only the anomaly map as output and not the reconstructed image. As a result, when we use the Compressed Replay strategy on FastFlow and Inpaint, we use the SR architecture as the Memory Module (which turns out to be the best among the generative models as shown in Section 6.6.2) and FastFlow and Inpaint as the AD Module.

In Fig. 6.6 we show the results of AD obtained for a sample with two different architectures VAE and SR. The first image in the row shows the original image, while the second column displays the reconstructed image produced by the model. The third column presents the ground truth for each pixel, which provides a reference for evaluating the accuracy of the reconstructed image. The fourth column displays the anomaly map, which indicates the probability that each pixel is anomalous (i.e., differs significantly from the ground truth). This can help identify areas of the image that may be corrupted. The fifth column shows the predicted class for each pixel, which can provide additional information about the final prediction of the model on what is considered anomalous.

[1] https://github.com/dallepezze/adcl_scale



(a) Reconstruction based on VAE



(b) Reconstruction based on SR

Figure 6.6: The images reported here show examples of the VAE (Variational Autoencoder) architecture in (a) and the SR (Super-Resolution) architecture in (b). The first image in each row is the original image, and the second column is the model's reconstructed image. The third column displays the ground truth for each pixel, which can be used to evaluate the accuracy of the reconstructed image. The fourth column shows the anomaly map, which indicates the probability that each pixel is abnormal (i.e., significantly different from the original pixel). This can help identify corrupted areas of the image. The fifth column displays the predicted class for each pixel, which provides more information about the model's final prediction on what is considered anomalous. The sixth column highlights any potential issues with the image by showing the anomalous parts of the image, making it easier for users to identify and address any problems.

Finally, the sixth column presents the portion of the image that contains anomalous parts, highlighting any potential issues with the image. This can help users quickly identify and address any problems with the image. The final values for AD performance are showed in Table 6.1 varying strategy and architecture. In the Single Model (classic setting), FastFlow performs better than the other models, as expected. It is significantly better than the next best model, with a margin of 0.57. However, in the CL setting, FastFlow performs poorly for both the Replay and Ideal Replay strategies. When using the Compressed Replay strategy, FastFlow performs better than the CAE and VAE models, but still worse than SR and Inpaint. Interestingly, when FastFlow is evaluated using the Compressed Replay strategy, it outperforms Replay in terms of AD performance. We can observe that, among the feasible Replay-based strategies, the

Table 6.1: Summary table for the AD performance under different CL strategies and architectures for the Memory and the AD Module. Since Inpaint and FastFlow cannot achieve Compressed Replay, but only act as AD modules, we consider them in combination with CAE, VAE and SR as Memory Modules. We report the Average f1 value (↑ stands for "the higher the better") and, within round brackets, the forgetting associated to the metric, for each combination. We highlight in bold the best combination. We can observe that, among the viable Replay-based strategies, the best combination is using Inpaint as AD Module with SR as Memory Module adopting the Compressed Replay strategy. The best combination is in **bold** while the best architecture for each strategy is in *italic*.

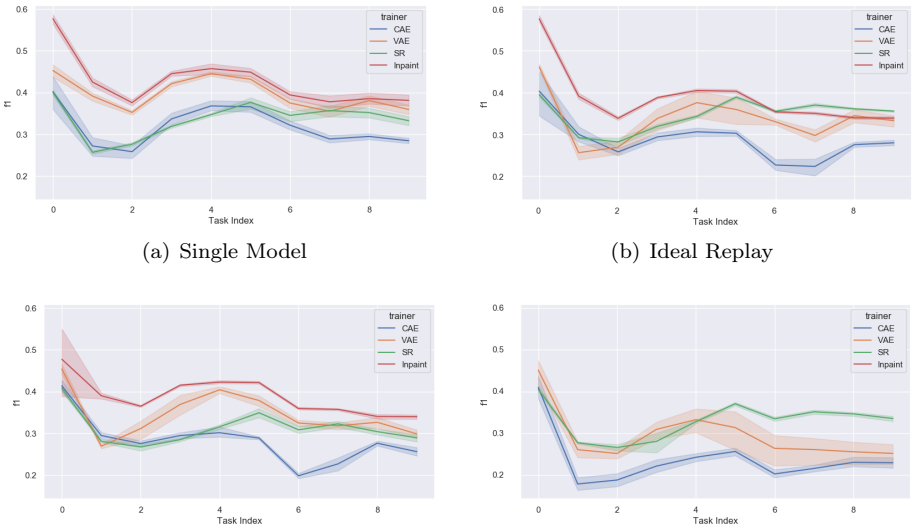
Strategy	**Memory Module**	**AD Module** (Average f1 metric ↑)				
		CAE	VAE	SR	Inpaint	FastFlow
Single Model	-	0.28	0.36	0.33	0.38	*0.57*
FT	-	0.09 (66.35%)	0.11 (68.19%)	0.14 *(59.39%)*	0.11 (72.73%)	*0.19* (69.79%)
Ideal Replay	-	0.28 (13.22%)	0.33 (6.38%)	*0.36 (5.72%)*	0.34 (11.25%)	0.22 (62.53%)
Replay	-	0.26 (19.88%)	0.30 (21.2%)	0.29 (22.23%)	*0.34 (13.27%)*	0.21 (64.57%)
Generative Replay	-	-	0.11 (63.68%)	-	-	-
Compressed Replay	CAE	0.23 (25.61%)	-	-	0.25 (34.00%)	0.17 (53%)
	VAE	-	0.25 (21.87%)	-	0.26 (30.27%)	0.2 (48%)
	SR	-	-	0.34 **(7.99%)**	**0.35** (10.28%)	0.30 (41.27%)

best combination is Inpaint as AD Module with SR as Memory Module trained according to the Compressed Replay Strategy.

In addition, it should be observed from Table 6.1 that the fine-tuning strategy appears to have better values in FastFlow than in other models.

In terms of forgetting, as shown in Table 6.1, FastFlow has a significantly higher forgetting than the other approaches. Instead, in terms of forgetting, SR with Compressed Replay has a very low forgetting.

Fig. 6.7 presents the results in greater detail, displaying the performance progressively as each task is presented in sequence. The Figure shows four different plots, each representing the performance of a specific strategy. The x-axis of each plot indicates the task ID of the training, while the y-axis represents the value of the chosen metric. Each line in the plot represents a different model. The value at the task i include only the performance on the tasks seen so far.



(a) Single Model

(b) Ideal Replay

(c) Replay

(d) Compressed Replay

Figure 6.7: Each plot shows the performance for a strategy comparing different architectures in terms of AD performance i.e. Average f1 score. The Average f1 metric is shown on axis y, and the index of the current training task is shown on axis x.

6.6.2 Quality of the reconstructed images

Table 6.2 shows a summary of the results, which corresponds to the average FID on all tasks evaluated at the end of the training. Table 6.2 also displays the associated percentual forgetting value in round brackets.

Moreover, it is possible to examine average FID on the tasks seen so far in the plots of Fig. 6.8. For each strategy, the performance of each model is shown, where a lower value means a better quality of the reconstructed image.

From Fig. 6.8, we can clearly see that the SR model is the best architecture for each strategy considered. For each of the strategies considered, the gap between SR and other architectures is significant. In other words, regardless of the strategy, the SR model is the best for learning a good representation of the original images. Moreover, its output is conditoned by the scaled image and random noise which means a certain level of randomness in output, while in the classic GANs imply a very quickly forgetting [159], that it doesn't happen in our case. Despite all these aspects, the SR is able to obtain a good representation of the original distribution.

As stated above, Inpaint can't act as Memory Modules like CAE and VAE because it cannot perform Compressed Replay. However, since Inpaint is a generative model, we can still analyze its reconstruction performance in terms

Table 6.2: Summary table about the quality of reconstructed images for different models. For each model and strategy, we report the FID value. Please note that the lower FID, the better the reconstruction is. In round brackets we report also the forgetting value in percentual. Note that for each strategy the best architecture is SR. Moreover, the best strategy results to be Compressed Replay. The best combination is in **bold** while the best architecture for each strategy is in *italic*.

	Average FID ↓			
Strategy	**CAE**	**VAE**	**SR**	**Inpaint**
Single Model	266.52	219.40	*125.15*	204.29
FT	376.64 (73.65%)	399.44 (91.33%)	*244.25* (194.31%)	363.20 (114.41%)
Ideal Replay	216.82 (7.50%)	255.84 (11.93%)	*89.53* (7.74%)	192.12 (5.24%)
Replay	227.88 (14.70%)	245.87 (14.54%)	*111.87* (40.6%)	181.74 (7.90%)
Compressed Replay	248.42 (20.33%)	316.48 (36.17%)	**105.57 (14.66%)**	-
Generative Replay	-	*405.34 (77.84%)*	-	-

of FID in Table 6.2 and Fig. 6.8.

Another interesting fact is that CAE seems to have a more stable trend than VAE in Latent Replay strategy. In fact, it can be observed that while CAE is worse than VAE in Single Model, it obtains better performance for Replay and Compressed Replay.

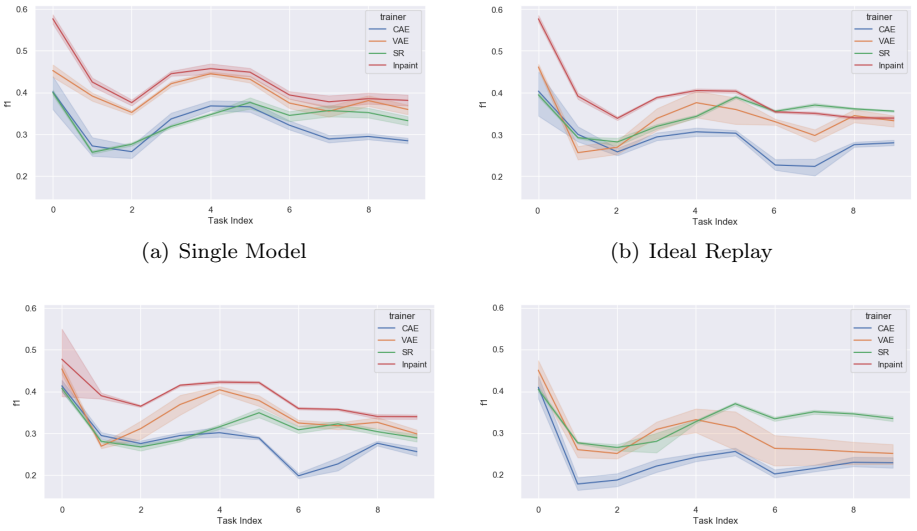
Among the proposed models VAE is the only one to be applied under the Generative Replay strategy. Results are shown in Table 6.2 and it can be seen that, as said before, the forgetting is very high.
According to these graphs, the difference between Compressed Replay and Ideal Replay appears to be greater for models CAE and VAE than for SR and Inpaint.
It should be noted that the compression factors for the CAE, VAE, and SR architectures are 6, 196, and 64, respectively.
The only model that allows Generative Replay is VAE, but as seen in 6.2, the performance degrades very quickly, so it is only shown in the tables and not the plots (the same for fine-tuning).
As a final remark, it can be stated that the SR model allows for a good compression factor while maintaining a very high image quality in memory.

6.6.3 Results of SR Model

As the first study on Super Resolution in the context of Continual Learning, we thoroughly examine how variables such as the number of epochs and the way in which images are saved affect the final performance in terms of the quality of the reconstructed images. This analysis helps us to gain insight into the critical

Strategy	**epochs**	**Average FID ↓**	**FID Increase % wrt Ideal Replay ↓**
Single Model	30	125.15	-
Ideal Replay	30	89.53 (7.74%)	-
Replay	30	111.87 (40.6%)	24.95%
Compressed Degenerative Replay	30	106.49 (13.58%)	18.94%
Compressed Replay	30	**105.57 (14.66%)**	**17.92%**
Single Model	50	82.06	-
Ideal Replay	50	85.27 (11.18%)	-
Replay	50	109.51 (59.41%)	28.43%
Compressed Degenerative Replay	50	97.31 (19.08%)	14.12%
Compressed Replay	50	**91.17 (10.88%)**	**6.92%**

Table 6.3: Comparison of different strategies for training a SR model. The table shows the results for two different values of the number of epochs (30 and 50). The FID score is shown in the third column, with the percentage of forgetting in round brackets. The fourth column shows the increase in performance with respect to the Ideal Replay strategy, in other words more a value is lower and more is close to the optimal. The best results for each number of epochs are highlighted in bold.



(a) Single Model

(b) Ideal Replay

(c) Replay

(d) Compressed Replay

Figure 6.8: The results show the quality of the constructed images for different continual learning strategies and architectures. Each plot represents a different CL strategy, and each line represents a different architecture. The y-axis shows the FID metric, and the x-axis shows the index of the current training task. Each value represents the average FID across all tasks seen so far, with a lower value indicating a higher quality of reconstruction. The plots show that the best architecture for each CL strategy is SR, our approach.

issues facing generative models such as GANs in Continual Learning. Based on this analysis, we conclude that there are at least three critical issues that arise during the training of GANs in CL.

1. The initial quality of the learned distribution is the first critical issue. Based on the number of epochs, we can easily demonstrate that an initial higher quality has a long-term beneficial effect.

2. In classic GANs, we create images only using a random vector for each new task, producing an output image that will be used as input to the model at the new task. However, there will be some perturbation in the final image. These perturbations accumulate in the produced images over time, increasing their distance from the original distribution.

3. The third critical issue is catastrophic forgetting of the model's weights, which is inherent in all models in the CL setting.

To simplify the explanation, we are going to define the following notation:

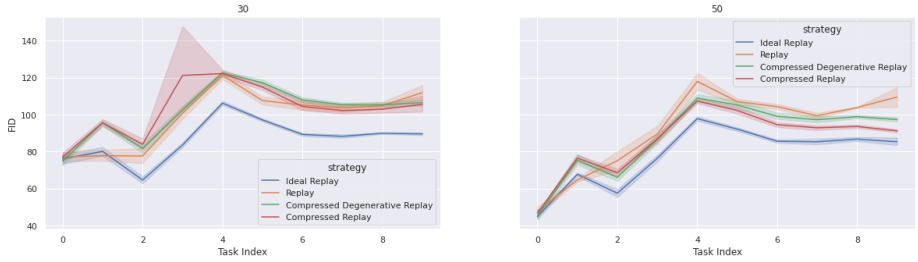


Figure 6.9: The comparison of the SR model using 30 epochs (left) and 50 epochs (right) with different strategies is shown in the figures. The y-axis shows the FID metric, and the x-axis shows the index of the current training task. It can be observed that the Compressed Replay strategy performs closer to the Ideal Replay strategy (i.e. the optimal performance) when there is an increase in the number of epochs from 30 to 50.

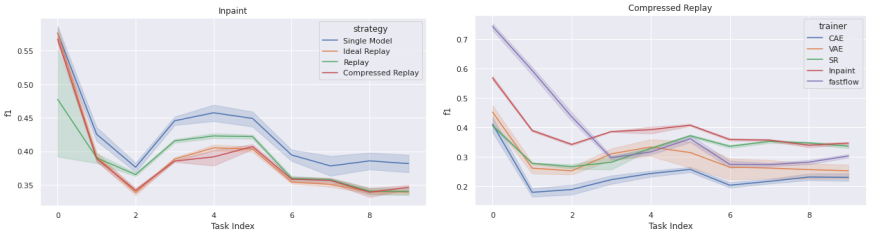


Figure 6.10: The figure on the left shows the best model using different Continual Learning strategies, and the figure on the right shows all models using the Compressed Replay strategy. The y-axis shows the Average f1 metric, and the x-axis shows the index of the current training task.

- X_i: The original data of the i-th Task;
- $O_i(X_i)$: The output produced by SR after training the i-th Task on data X_i;
- S_i: The scaled images of the data X_i;
- M_{ij}: The memory about the $j-$th Task after training the i-th Task.

In Compressed Replay for SR, the scaled image S_i is saved in memory $M_{i,i}$ during the i-th Task i.e. $M_{i,i} = S_i$. Such scaled images will never be changed, they will remain fixed over time or in other terms $M_{i,i} = M_{i+1,i} = \cdots = M_{T,i}$. Now we define a modified version called *Compressed Degenerative Replay* that simulates the critical issue (ii) present in GANs, where the images used for an old task during the training of a new task are different from the images used for a previous task. During the $(i+1)-$th Task, we take the images saved during the previous task $M_{i,i} = S_i$ and produce the new reconstructed images as $O_{i+1}(S_i)$.

Such images are scaled and saved in memory i.e. $M_{i+1,i} = O_{i+1}(S_i)$. This is repeated for all the tasks and then we have that for the i-th Task the memory change over time i.e. $M_{i,i} \neq M_{i+1,i} \neq \cdots \neq M_{T,i}$

The difference in results for each strategy based on the number of epochs (30 or 50) can be seen in Table 6.3 and Fig. 6.9. Generally, the results obtained using a larger number of epochs (i.e., a better initial learned representation) are better, as expected. For example, the Ideal Replay strategy improves from 95.2 to 89.5 in terms of FID, and the Replay strategy improves from 112 to 109.5. However, the most significant improvement is seen with the Compressed Replay strategy, which drops from 105.5 to 91. In terms of percentage increase compared to Replay, Compressed Replay shows a minor decrease at 50 epochs (7%) compared to 30 epochs (18%). This suggests that the proposed solution performs very close to the upper bound of Ideal Replay, particularly when the initially learned representation is very good (i.e., a higher number of epochs).

Finally, when comparing the Compressed Replay strategy vs Compressed Degenerative Replay strategy, we can see that the positive effect of Compressed Replay vs compressed Degenerative Replay increases with the number of epochs. Indeed, with 50 epochs, Compressed Replay has a 7% increase over Ideal Replay, while Compressed Degenerative Replay has a 14% increase. Furthermore, the percentage increase in comparison to the Ideal Replay for Compressed Degenerative Replay is lower for 50 epochs (14% instead of 18%). This means that using more epochs and Compressed Replay instead of Replay or compressed Degenerative Replay allows you to get closer to the Ideal Replay, which uses the original distribution in memory.
However, while Compressed Degenerative Replay is inferior to Compressed Replay, the table shows that Compressed Degenerative Replay with 50 epochs is superior to Compressed Replay with 30 epochs. This means that the main factor is the number of epochs i.e. the initial quality of the learned representation.

6.7 Conclusions

In this research, we propose a framework for performing anomaly detection in a setting where the model is continually learning new tasks. Our approach combines a Memory module, which stores information from previous tasks, and an Anomaly Detection module, which identifies anomalies in new data. By using these modules together, our approach can detect anomalies at the pixel level in new image classes without forgetting how to identify anomalies in previously learned classes. To the best of our knowledge, this is the first time that a method using a Super Resolution module to enable Compressed Replay has been proposed for use in continual learning. We also suggest a real-world benchmark dataset specifically designed for evaluating the performance of anomaly detection approaches in a continual learning setting.

There are several potential future developments. Application and evaluation of other models considered state-of-the-art in Anomaly Detection in the CL setting should be a research direction. Furthermore, some of these models

could be modified to produce better results in the case of Anomaly Detection for Continual Learning. While the MVTec Dataset is an excellent benchmark for Anomaly Detection, other datasets could be used to confirm the models' efficacy. Furthermore, a follow-up study on the SR problem should be carried out, analyzing models closer to the state of the art and using a more complex dataset to validate their efficacy in the Continual Learning setting.

Chapter 7

Conclusions and Future Research

7.1 Conclusions

The aim of this dissertation was to provide novel approaches for Continual Learning and new solutions for Industry 4.0 field. In particular, the first concern was to point out how Industry 4.0 field can benefit from the perspective of Continual Learning setting. In support of this assertion, in Section 3.11.1 was shown many practical applications of Continual Learning in the Manufacturing field. We also point out how observed in [21] that Continual Learning should increase the research in settings different from the classic multi-class classification. In practice, other applications could be more reasonable, like Unsupervised Learning. Indeed, often acquiring labels for a problem can be expensive or unfeasible. From the point of view of Industry 4.0, the goal of CL to allow ML models to be updated, lowering the costs associated with machine learning model maintenance and resource utilization can be very appealing. Indeed, the main focus of Industry 4.0 is to improve efficiency, productivity, and flexibility. Therefore CL seems very aligned with these goals.
To summarize our contributions, we provide an overview of Industry 4.0 in Chapter 2 and of Continual Learning in Chapter 3. After we provide a novel approach for Interpretability in Industry 4.0 Then we consider a real industrial scenario from packaging industry and formalize the problem of Alarm Forecasting as multi-label classification and propose a novel solution for it. Then, we extend the problem in the Continual Learning setting and propose a novel approach for Replay in Multi-label. In Chapter 6 we propose a framework for Anomaly Detection in Continual Learning Setting (ADCL) to evaluate different CL strategies. We study the ADCL using as benchmark a complex dataset designed for AD and evaluating using many AD approaches. Eventually, we propose a novel approach for Compressed Replay.

7.2 Future research

In the first part of this chapter we highlight the research performed and the contributions provided in this manuscript. However, many of problems proposed have potentially many research directions to improve the approaches. For example, though not discussed, studies on Intepretability in the Continual Learning setting could be an interesting direction since few research was done in the field. When considering the problem discussed in Chapter 5 there is definitely potential improvements, in particular in the construction of the memory by selecting the optimal samples for multi-label. Not only the performance in terms of optimal samples should be considered but also the speed of the selection algorithms. When considering the problem Anomaly Detection in the Continual Learning from the results provided in Chapter 6 is clear how is still an open problem and more research should focus on this field.
As said before, the separation in tasks with task id provided in training can be considered a simplification of real scenarios and AD architectures can be a first step in this direction. Though it is not usually considered in the classic CL setting, it is technically possible to use an AD model to detect data distribution shift and avoid the use of task id in training. However, more study should be performed to validate the feasibility of the approach.

Another possible research direction should be performed on generative models for Continual Learning and the mode-collapse problem. Additionally, Compressed Replay seems an interesting approach to reduce the burden of memory in CL approaches.

Bibliography

[1] `https://github.com/dallepezze/formula`. 2021.

[2] `https://github.com/dallepezze/bat-ocdm`. 2021.

[3] Wickliffe C Abraham and Anthony Robins. "Memory retention–the synaptic stability versus plasticity dilemma". In: *Trends in neurosciences* 28.2 (2005), pp. 73–78.

[4] Mounia Achouch et al. "On predictive maintenance in industry 4.0: Overview, models, and challenges". In: *Applied Sciences* 12.16 (2022), p. 8081.

[5] R. Ahmad and S. Kamaruddin. "An overview of time-based and condition-based maintenance in industrial application". In: *Comput. Ind. Eng* 63 (2012), pp. 135–149.

[6] Imran Ahmed, Gwanggil Jeon, and Francesco Piccialli. "From Artificial Intelligence to Explainable Artificial Intelligence in Industry 4.0: A Survey on What, How, and Where". In: *IEEE Transactions on Industrial Informatics* 18.8 (2022), pp. 5031–5042. DOI: `10.1109/TII.2022.3146552`.

[7] Kabir Ahmed et al. "Similarity analysis of industrial alarm flood data". In: *IEEE Transactions on Automation Science and Engineering* 10.2 (2013), pp. 452–457.

[8] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. *GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training*. 2018. arXiv: `1805.06725 [cs.CV]`.

[9] Rabia Ali, Muhammad Umar Karim Khan, and Chong Min Kyung. "Self-supervised representation learning for visual anomaly detection". In: *arXiv preprint arXiv:2006.09654* (2020).

[10] Rahaf Aljundi et al. "Memory aware synapses: Learning what (not) to forget". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 139–154.

[11] Rahaf Aljundi et al. "Online continual learning with maximally interfered retrieval". In: *arXiv preprint arXiv:1908.04742* (2019).

[12] Thomas Altenmuller et al. "Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints". In: *Production Engineering* 14.3 (2020), pp. 319–328.

[13] Suresh Kumar Amalapuram et al. "Continual Learning for Anomaly based Network Intrusion Detection". In: *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE. 2022, pp. 497–505.

[14] Peter Andras et al. "Trusting intelligent machines: Deepening trust within socio-technical systems". In: *IEEE Technology and Society Magazine* 37.4 (2018), pp. 76–83.

[15] Liat Antwarg et al. "Explaining anomalies detected by autoencoders using Shapley Additive Explanations". In: *Expert Systems with Applications* 186 (2021), p. 115736.

[16] Saeed Anwar, Salman Khan, and Nick Barnes. "A deep journey into super-resolution: A survey". In: *ACM Computing Surveys (CSUR)* 53.3 (2020), pp. 1–34.

[17] Hasan Asy'ari Arief et al. "Towards Building a Distributed Virtual Flow Meter via Compressed Continual Learning". In: *Sensors* 22.24 (2022), p. 9878.

[18] Mulugeta Weldezgina Asres et al. "Supporting Telecommunication Alarm Management System with Trouble Ticket Prediction". In: *IEEE Transactions on Industrial Informatics* (2020).

[19] Yuri Sousa Aurelio et al. "Learning from imbalanced data sets with weighted cross-entropy function". In: *Neural Processing Letters* 50.2 (2019), pp. 1937–1949.

[20] Ali Ayub and Alan R Wagner. "Storing encoded episodes as concepts for continual learning". In: *arXiv preprint arXiv:2007.06637* (2020).

[21] Benedikt Bagus, Alexander Gepperth, and Timothee Lesort. "Beyond Supervised Continual Learning: a Review". In: *arXiv preprint arXiv:2208.14307* (2022).

[22] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[23] B Bajic et al. "Machine learning techniques for smart manufacturing: Applications and challenges in industry 4.0". In: *Department of Industrial Engineering and Management Novi Sad, Serbia* 29 (2018).

[24] Adrien Becue, Isabel Praca, and Joao Gama. "Artificial intelligence, cyber-threats and Industry 4.0: Challenges and opportunities". In: *Artificial Intelligence Review* 54.5 (2021), pp. 3849–3886.

[25] A. Bécue, I. Praça, and J. Gama. "Artificial intelligence, cyber-threats and Industry 4.0: Challenges and opportunities". In: *Artif. Intell. Rev* 54 (2021), pp. 3849–3886.

[26] Paul Bergmann et al. "MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9584–9592. DOI: 10.1109/CVPR.2019.00982.

[27] Paul Bergmann et al. "MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9592–9600.

[28] Gerard Biau and Erwan Scornet. "A random forest guided tour". In: *Test* 25.2 (2016), pp. 197–227.

[29] Magdalena Biesialska, Katarzyna Biesialska, and Marta R Costa-Jussa. "Continual lifelong learning in natural language processing: A survey". In: *arXiv preprint arXiv:2012.09823* (2020).

[30] S. Biswal and G.R. Sabareesh. *Design and development of a wind turbine test rig for condition monitoring studies*. May 28, 2015.

[31] Sailendu Biswal and GR Sabareesh. "Design and development of a wind turbine test rig for condition monitoring studies". In: *2015 international conference on industrial instrumentation and control (icic)*. IEEE. 2015, pp. 891–896.

[32] O. Blancke. "Développement D'une Approche de Pronostic Pour les Équipements Complexes Permettant L'application de la Maintenance Prévisionnelle". Ph. D. Thesis, Montreal, QC, Canada, 2020.

[33] CHEN Bojian et al. "Continual learning fault diagnosis: A dual-branch adaptive aggregation residual network for fault diagnosis with machine increments". In: *Chinese Journal of Aeronautics* (2022).

[34] Jorg Bornschein et al. "NEVIS'22: A Stream of 100 Tasks Sampled from 30 Years of Computer Vision Research". In: *arXiv preprint arXiv:2211.11747* (2022).

[35] Tobias Bottger and Markus Ulrich. "Real-time texture error detection on textured surfaces with compressed sensing". In: *Pattern Recognition and Image Analysis* 26.1 (2016), pp. 88–94.

[36] Christos Boutsidis et al. "Randomized dimensionality reduction for k-means clustering". In: *IEEE Transactions on Information Theory* 61.2 (2014), pp. 1045–1062.

[37] L. Breiman. *Random Forests*. Cambridge, UK, 2001.

[38] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[39] Mathias Santos de Brito et al. "Application of the fog computing paradigm to smart factories and cyber-physical systems". In: *transactions on emerging telecommunications technologies* 29.4 (2018), e3184.

[40] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[41] Luca Brunelli et al. "Deep Learning-based Production Forecasting in Manufacturing: a Packaging Equipment Case Study". In: *Procedia Manufacturing* 38 (2019), pp. 248–255.

[42] Pietro Buzzega et al. "Rethinking experience replay: a bag of tricks for continual learning". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 2180–2187.

[43] Lucas Caccia et al. "Online learned continual compression with adaptive quantization modules". In: *International conference on machine learning*. PMLR. 2020, pp. 1240–1250.

[44] Shuang Cai et al. "Process alarm prediction using deep learning and word embedding methods". In: *ISA transactions* 85 (2019), pp. 274–283.

[45] Mikel Canizo et al. "Real-time predictive maintenance for wind turbines using Big Data frameworks". In: *2017 ieee international conference on prognostics and health management (icphm)*. IEEE. 2017, pp. 70–77.

[46] Mattia Carletti, Matteo Terzi, and Gian Antonio Susto. "Interpretable Anomaly Detection with DIFFI: Depth-based Feature Importance for the Isolation Forest". In: *arXiv preprint arXiv:2007.11117* (2020).

[47] Diego Carrera et al. "Defect detection in SEM images of nanofibrous materials". In: *IEEE Transactions on Industrial Informatics* 13.2 (2016), pp. 551–561.

[48] Diego Carrera et al. "Scale-invariant anomaly detection with multiscale group-sparse models". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 3892–3896.

[49] T.P. Carvalho et al. "A systematic literature review of machine learning methods applied to predictive maintenance". In: *Comput. Ind. Eng* 137 (2019), p. 106024.

[50] Thyago P Carvalho et al. "A systematic literature review of machine learning methods applied to predictive maintenance". In: *Computers & Industrial Engineering* 137 (2019), p. 106024.

[51] Francisco Charte and David Charte. "Working with Multilabel Datasets in R: The mldr Package." In: *R J.* 7.2 (2015), p. 149.

[52] Francisco Charte et al. "Dealing with difficult minority labels in imbalanced mutilabel data sets". In: *Neurocomputing* 326 (2019), pp. 39–53.

[53] Arslan Chaudhry et al. "On tiny episodic memories in continual learning". In: *arXiv preprint arXiv:1902.10486* (2019).

[54] Arslan Chaudhry et al. "Riemannian walk for incremental learning: Understanding forgetting and intransigence". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 532–547.

[55] Honglie Chen et al. "Localizing visual sounds the hard way". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16867–16876.

[56] T. Chen and C. Guestrin. *XGBoost: A scalable tree boosting system*. San Francisco, CA, USA, Aug. 13, 2016.

[57] Toly Chen and Yu-Cheng Lin. "Feasibility evaluation and optimization of a smart manufacturing system based on 3D printing: a review". In: *International Journal of Intelligent Systems* 32.4 (2017), pp. 394–413.

[58] Xi Chen et al. "An online continual object detector on VHR remote sensing images with class imbalance". In: *Engineering Applications of Artificial Intelligence* 117 (2023), p. 105549.

[59] Yao-Nan Chen and Hsuan-Tien Lin. "Feature-aware label space dimension reduction for multi-label classification". In: *Advances in neural information processing systems* 25 (2012).

[60] Zhiwei Chen and Aoqian Zhang. "A Survey of Approximate Quantile Computation on Large-Scale Data". In: *IEEE Access* 8 (2020), pp. 34585–34597. DOI: 10.1109/ACCESS.2020.2974919.

[61] Zhiyuan Chen and Bing Liu. "Lifelong machine learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12.3 (2018), pp. 1–207.

[62] KR1442 Chowdhary. "Natural language processing". In: *Fundamentals of artificial intelligence* (2020), pp. 603–649.

[63] Tat-Seng Chua et al. "Nus-wide: a real-world web image database from national university of singapore". In: *Proceedings of the ACM international conference on image and video retrieval*. 2009, pp. 1–9.

[64] Zeki Murat Cinar et al. "Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0". In: *Sustainability* 12.19 (2020), p. 8211.

[65] Niv Cohen and Yedid Hoshen. *Sub-Image Anomaly Detection with Deep Pyramid Correspondences*. 2021. arXiv: 2005.02357 [cs.CV].

[66] Anne-Sophie Collin and Christophe De Vleeschouwer. *Improved anomaly detection by training an autoencoder with skip connections on images corrupted with Stain-shaped noise*. 2020. DOI: 10.48550/ARXIV.2008.12977. URL: https://arxiv.org/abs/2008.12977.

[67] *Creating a Zoo of Atari-Playing Agents to Catalyze the Understanding of Deep Reinforcement Learning*. 2019.

[68] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, pp. 886–893.

[69] Davide Dalle Pezze et al. "FORMULA: A Deep Learning Approach for Rare Alarms Predictions in Industrial Equipment". In: *IEEE Transactions on Automation Science and Engineering* (2021).

[70] David Dandolo et al. "AcME—Accelerated model-agnostic explanations: Fast whitening of the machine-learning black box". In: *Expert Systems with Applications* 214 (2023), p. 119115.

[71] Devam Dave et al. "Explainable AI meets Healthcare: A Study on Heart Disease Dataset". In: *ArXiv* abs/2011.03195 (2020).

[72] Matthias De Lange et al. "A continual learning survey: Defying forgetting in classification tasks". In: *IEEE transactions on pattern analysis and machine intelligence* 44.7 (2021), pp. 3366–3385.

[73] Thomas Defard et al. *PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization*. 2020. arXiv: 2011.08785 [cs.CV].

[74] David Dehaene et al. "Iterative energy-based projection on a normal data manifold for anomaly localization". In: *arXiv preprint arXiv:2002.03734* (2020).

[75] Houtao Deng. "Interpreting tree ensembles with inTrees". In: *International Journal of Data Science and Analytics* 7 (2018), pp. 277–287.

[76] Krantiraditya Dhalmahapatra et al. "Decision support system for safety improvement: An approach using multiple correspondence analysis, t-SNE algorithm and K-means clustering". In: *Computers & Industrial Engineering* 128 (2019), pp. 277–289.

[77] Shweta C Dharmadhikari, Maya Ingle, and Parag Kulkarni. "A novel multi label text classification model using semi supervised learning". In: *International Journal of Data Mining & Knowledge Management Process* 2.4 (2012), p. 11.

[78] Francesco Di Maio et al. "Naive Bayesian classifier for on-line remaining useful life prediction of degrading bearings". In: *MMR2011*. 2011, pp. 1–14.

[79] S. Diamond. *Predictive Maintenance For Dummies®, IBM Limited Edition*. Hoboken, NJ, USA: John Wiley and Sons, Inc, 2013, p. 51.

[80] Natalia Diaz-Rodriguez et al. "Don't forget, there is more than forgetting: new metrics for Continual Learning". In: *arXiv preprint arXiv:1810.13166* (2018).

[81] Rémi Domingues et al. "A comparative evaluation of outlier detection algorithms: Experiments and analyses". In: *Pattern Recognition* 74 (2018), pp. 406–421.

[82] Gyula Dorgo and Janos Abonyi. "Sequence mining based alarm suppression". In: *IEEE Access* 6 (2018), pp. 15365–15379.

[83] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. arXiv: 1702.08608 [stat.ML].

[84] Min Du et al. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1285–1298.

[85] Lian Duan and Li Da Xu. "Business intelligence for enterprise systems: A survey". In: *IEEE Transactions on Industrial Informatics* 8.3 (2012), pp. 679–687.

[86] EV Dudukalov et al. "The use of artificial intelligence and information technology for measurements in mechanical engineering and in process automation systems in Industry 4.0". In: *Journal of Physics: Conference Series*. Vol. 1889. 5. IOP Publishing. 2021, p. 052011.

[87] Gopi Krishna Durbhaka and Barani Selvaraj. "Predictive maintenance for wind turbine diagnostics using vibration signal analysis based on collaborative recommendation approach". In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2016, pp. 1839–1842.

[88] Mathias Eitz, James Hays, and Marc Alexa. "How do humans sketch objects?" In: *ACM Transactions on graphics (TOG)* 31.4 (2012), pp. 1–10.

[89] Samuel Eke et al. "Characterization of the operating periods of a power transformer by clustering the dissolved gas data". In: *2017 IEEE 11th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED)*. IEEE. 2017, pp. 298–303.

[90] Carlos A Escobar and Ruben Morales-Menendez. "Machine learning and pattern recognition techniques for information extraction to improve production control and design decisions". In: *Industrial Conference on Data Mining*. Springer. 2017, pp. 286–300.

[91] Aniekan Essien and Cinzia Giannetti. "A Deep Learning Model for Smart Manufacturing Using Convolutional LSTM Neural Network Autoencoders". In: *IEEE Trans. on Industrial Informatics* 16.9 (2020), pp. 6069–6078.

[92] Shu-Kai S Fan et al. "Data-driven approach for fault detection and diagnostic in semiconductor manufacturing". In: *IEEE Transactions on Automation Science and Engineering* 17.4 (2020), pp. 1925–1936.

[93] Pei Fengque et al. "Research on design of the smart factory for forging enterprise in the industry 4.0 environment". In: *Mechanics* 23.1 (2017), pp. 146–152.

[94] Sofia Fernandes et al. "Forecasting appliances failures: A machine-learning approach to predictive maintenance". In: *Information* 11.4 (2020), p. 208.

[95] Robert M French. "Catastrophic forgetting in connectionist networks". In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[96] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29 (2000), pp. 1189–1232.

[97] Christopher Frye, Colin Rowat, and Ilya Feige. "Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1229–1239.

[98] Joao Gama et al. "A survey on concept drift adaptation". In: *ACM computing surveys (CSUR)* 46.4 (2014), pp. 1–37.

[99] Qiong Gao, Li Da Xu, and Ning Liang. "Dynamic modelling with an integrated ecological knowledge-based system". In: *Knowledge-Based Systems* 14.5-6 (2001), pp. 281–287.

[100] Vangelis Gazis et al. "Components of fog computing in an industrial internet of things context". In: *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops)*. IEEE. 2015, pp. 1–6.

[101] Weifeng Ge, Sibei Yang, and Yizhou Yu. "Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1277–1286.

[102] Jort F Gemmeke et al. "Audio set: An ontology and human-labeled dataset for audio events". In: *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2017, pp. 776–780.

[103] Natalie Gentner et al. "DBAM: Making virtual metrology/soft sensing with time series data scalable through deep learning". In: *Control Engineering Practice* 116 (2021), p. 104914.

[104] Paolo Giudici and Emanuela Raffinetti. "Shapley-Lorenz eXplainable artificial intelligence". In: *Expert Systems with Applications* 167 (2021), p. 114104.

[105] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[106] Ian J Goodfellow et al. "An empirical investigation of catastrophic forgetting in gradient-based neural networks". In: *arXiv preprint arXiv:1312.6211* (2013).

[107] Raia Hadsell, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[108] Michael Haenlein et al. "Artificial intelligence (AI) and management analytics". In: *Journal of Management Analytics* 6.4 (2019), pp. 341–343.

[109] Hui Han and Julien Siebert. "TinyML: A Systematic Review and Synthesis of Existing Research". In: *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 2022, pp. 269–274. DOI: 10.1109/ICAIIC54071.2022.9722636.

[110] Ya-nan Han and Jian-wei Liu. "Online Continual Learning via the Meta-learning update with Multi-scale Knowledge Distillation and Data Augmentation". In: *Engineering Applications of Artificial Intelligence* 113 (2022), p. 104966.

[111] Satoshi Hara and Kohei Hayashi. "Making Tree Ensembles Interpretable: A Bayesian Model Selection Approach". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, Sept. 2018, pp. 77–85. URL: `https://proceedings.mlr.press/v84/hara18a.html`.

[112] Jonathas GD Harb, Régis Ebeling, and Karin Becker. "A framework to analyze the emotional reactions to mass violent events on Twitter and influential factors". In: *Information Processing & Management* 57.6 (2020), p. 102372.

[113] Tyler L Hayes et al. "Remind your neural network to prevent catastrophic forgetting". In: *European Conference on Computer Vision*. Springer. 2020, pp. 466–483.

[114] Yihai He et al. "Integrated predictive maintenance strategy for manufacturing systems by combining quality control and mission reliability analysis". In: *International Journal of Production Research* 55.19 (2017), pp. 5841–5862.

[115] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[116] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* 2.7 (2015).

[117] Geoffrey E Hinton. "Connectionist learning procedures". In: *Machine learning*. Elsevier, 1990, pp. 555–610.

[118] Alex Goh Chee Hong, Tole Sutikno, and Shahreen Kasim. "Predictive Maintenance in Oil and Gas Dataset by using Naive Bayes and Gaussian Elimination Method". In: ().

[119] H.H. Hosamo et al. "A Digital Twin predictive maintenance framework of air handling units based on automatic fault detection and diagnostics". In: *Energy Build* 261 (2022), p. 111988.

[120] Junjie Hu et al. "Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4411–4421.

[121] Boming Huang et al. "Automated trading systems statistical and machine learning methods and hardware implementation: a survey". In: *Enterprise Information Systems* 13.1 (2019), pp. 132–144.

[122] Jing Huang, Qing Chang, and Jorge Arinez. "Deep reinforcement learning based preventive maintenance policy for serial production lines". In: *Expert Systems with Applications* 160 (2020), p. 113701.

[123] Chip Huyen. *Data Distribution Shifts and Monitoring*. 2022. URL: https://huyenchip.com/2022/02/07/data-distribution-shifts-and-monitoring.html.

[124] M Ilchenko, L Uryvsky, and S Osypchuk. "World trends of modern information and telecommunication technologies development". In: *2019 International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo)*. IEEE. 2019, pp. 1–7.

[125] Ahmet Iscen et al. "Memory-efficient incremental learning through feature adaptation". In: *European conference on computer vision*. Springer. 2020, pp. 699–715.

[126] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[127] Zhong Ji et al. "Coordinating Experience Replay: A Harmonious Experience Retention approach for Continual Learning". In: *Knowledge-Based Systems* 234 (2021), p. 107589.

[128] Magdiel Jimenez-Guarneros, Jonas Grande-Barreto, and Jose de Jesus Rangel-Magdaleno. "Multiclass Incremental Learning for Fault Diagnosis in Induction Motors Using Fine-Tuning with a Memory of Exemplars and Nearest Centroid Classifier". In: *Shock and Vibration* 2021 (2021).

[129] Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260.

[130] T. Kaarlela et al. "Common Educational Teleoperation Plat-form for Robotics Utilizing Digital Twins". In: *Machines* 10 (2022), p. 577.

[131] Sachin S Kamble, Angappa Gunasekaran, and Shradha A Gawankar. "Sustainable Industry 4.0 framework: A systematic literature review identifying the current trends and future perspectives". In: *Process safety and environmental protection* 117 (2018), pp. 408–425.

[132] Yue Kang et al. "Natural language processing (NLP) in management research: A literature review". In: *Journal of Management Analytics* 7.2 (2020), pp. 139–172.

[133] Ziqiu Kang, Cagatay Catal, and Bedir Tekinerdogan. "Machine learning applications in production lines: A systematic literature review". In: *Computers & Industrial Engineering* 149 (2020), p. 106773. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2020.106773. URL: http://www.sciencedirect.com/science/article/pii/S036083522030485X.

[134] N. Kashmar, M. Adda, and M. Atieh. *From access control models to access control metamodels: A survey*. San Francisco, CA, USA: Springer, Mar. 14, 2019, pp. 892–911.

[135] Koki Kawabata. *Predictive Maintenance*. Accessed: Month Day, Year. 2021. URL: https://github.com/kokikwbt/predictive-maintenance.

[136] Sarah Keartland and Terence L Van Zyl. "Automating predictive maintenance using oil analysis and machine learning". In: *2020 International SAUPEC/RobMech/PRASA Conference*. IEEE. 2020, pp. 1–6.

[137] A.T. Keleko et al. "Artificial intelligence and real-time predictive maintenance in industry 4.0: A bibliometric analysis". In: *AI Ethics* (2022), pp. 1–25.

[138] Ronald Kemker et al. "Measuring catastrophic forgetting in neural networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[139] Shehroz S Khan and Michael G Madden. "A survey of recent trends in one class classification". In: *Irish conference on artificial intelligence and cognitive science*. Springer. 2009, pp. 188–197.

[140] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. "Imbalanced continual learning with partitioning reservoir sampling". In: *European Conference on Computer Vision*. Springer. 2020, pp. 411–428.

[141] Dong-Hyeon Kim et al. "Smart machining process using machine learning: A review and perspective on machining industry". In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 5.4 (2018), pp. 555–568.

[142] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[143] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in neural information processing systems* 31 (2018).

[144] Polina Kirichenko et al. "Task-agnostic continual learning with hybrid probabilistic models". In: *arXiv preprint arXiv:2106.12772* (2021).

[145] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[146] Nikolaos Kolokas et al. "Forecasting faults of industrial equipment using machine learning classifiers". In: *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE. 2018, pp. 1–6.

[147] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. "Supervised machine learning: A review of classification techniques". In: *Emerging artificial intelligence applications in computer engineering* 160.1 (2007), pp. 3–24.

[148] Kamran Kowsari et al. "Text classification algorithms: A survey". In: *Information* 10.4 (2019), p. 150.

[149] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[150] Kedar Kulkarni et al. "Predictive maintenance for supermarket refrigeration systems using only case temperature data". In: *2018 Annual American Control Conference (ACC)*. IEEE. 2018, pp. 4640–4645.

[151] A. Kumar, R.B. Chinnam, and F. Tseng. "An HMM and polynomial regression based approach for remaining useful life and health state estimation of cutting tools". In: *Comput. Ind. Eng* 128 (2019), pp. 1008–1014.

[152] I Elizabeth Kumar et al. "Problems with Shapley-value-based explanations as feature importance measures". In: *arXiv preprint arXiv:2002.11097* (2020).

[153] S Saravana Kumar et al. "Conceptual Study of Artificial Intelligence in Smart Cities with Industry 4.0". In: *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE. 2021, pp. 575–577.

[154] Andrew Kusiak and Anoop Verma. "Prediction of status patterns of wind turbines: A data-mining approach". In: (2011).

[155] Ju Yeon Lee, Joo Seong Yoon, and Bo-Hyun Kim. "A big data analytics platform for smart factories in small and medium-sized manufacturing enterprises: An empirical case study of a die casting factory". In: *International Journal of Precision Engineering and Manufacturing* 18.10 (2017), pp. 1353–1361.

[156] Subin Lee and Jun-Geol Baek. "Generative pseudorehearsal strategy for fault classification under an incremental learning". In: *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. IEEE. 2019, pp. 138–140.

[157] Timothee Lesort. "Continual learning: Tackling catastrophic forgetting in deep neural networks with replay processes". In: *arXiv preprint arXiv:2007.00487* (2020).

[158] Timothee Lesort, Massimo Caccia, and Irina Rish. "Understanding continual learning settings with data distribution drift analysis". In: *arXiv preprint arXiv:2104.01678* (2021).

[159] Timothée Lesort et al. "Generative models from the perspective of continual learning". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.

[160] Hongfei Li et al. "Improving rail network velocity: A machine learning approach to predictive maintenance". In: *Transportation Research Part C: Emerging Technologies* 45 (2014), pp. 17–26.

[161] Weijun Li et al. "Process fault diagnosis with model-and knowledge-based approaches: Advances and opportunities". In: *Control Engineering Practice* 105 (2020), p. 104637.

[162] Zhizhong Li and Derek Hoiem. "Learning without forgetting". In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947.

[163] Zongmin Li et al. "Social media rumor refutation effectiveness: Evaluation, modelling and enhancement". In: *Information Processing & Management* 58.1 (), p. 102420.

[164] Yan-Shuo Liang and Wu-Jun Li. *Optimizing Class Distribution in Memory for Multi-Label Continual Learning*. 2022. URL: https://openreview.net/forum?id=HavXnq6KyT3.

[165] Chun-Cheng Lin et al. "Key design of driving industry 4.0: Joint energy-efficient deployment and scheduling in group-based industrial wireless sensor networks". In: *IEEE Communications Magazine* 54.10 (2016), pp. 46–52.

[166] Chun-Cheng Lin et al. "Smart manufacturing scheduling with edge computing using multiclass deep Q network". In: *IEEE Transactions on Industrial Informatics* 15.7 (2019), pp. 4276–4284.

[167] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[168] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[169] T. Lindgren and J. Biteus. *IDA2016 - Challenge Data Set*. Last access in 10/2/19. 2016. URL: https://archive.ics.uci.edu/ml/datasets/IDA2016Challenge.

[170] Zachary C Lipton et al. "Combating reinforcement learning's sisyphean curse with intrinsic fear". In: *arXiv preprint arXiv:1611.01211* (2016).

[171] Weiwei Liu, Ivor W Tsang, and Klaus-Robert Muller. "An easy-to-hard learning paradigm for multiple classes and multiple labels". In: *The Journal of Machine Learning Research* 18 (2017).

[172] Weiwei Liu et al. "The emerging trends of multi-label learning". In: *IEEE transactions on pattern analysis and machine intelligence* 44.11 (2021), pp. 7955–7974.

[173] Wenqian Liu et al. *Towards Visually Explaining Variational Autoencoders*. Nov. 2019.

[174] Yongkui Liu and Xun Xu. "Industry 4.0 and cloud manufacturing: A comparative analysis". In: *Journal of Manufacturing Science and Engineering* 139.3 (2017).

[175] Vincenzo Lomonaco. "Continual Learning with Deep Architectures". PhD thesis. alma, Apr. 2019. URL: http://amsdottorato.unibo.it/9073/.

[176] L. S. Lopes and L. M. Camarinha-Mato. *Gearbox Fault Detection Dataset*. Last access in 11/02/2019. 2009. URL: https://c3.nasa.gov/dashlink/resources/997/.

[177] L. S. Lopes and L. M. Camarinha-Mato. *Robot Execution Failures Data Set*. Last access in 11/02/2019. 1999. URL: https://archive.ics.uci.edu/ml/datasets/Robot+Execution+Failures.

[178] David Lopez-Paz and Marc'Aurelio Ranzato. "Gradient episodic memory for continual learning". In: *Advances in neural information processing systems* 30 (2017).

[179] David G Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[180] Scott Lundberg. *SHAP API - Online documentation*. URL: https://shap.readthedocs.io/en/latest/generated/shap.KernelExplainer.html#shap.KernelExplainer.

[181] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. "Consistent individualized feature attribution for tree ensembles". In: *arXiv preprint arXiv:1802.03888* (2018).

[182] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems*. 2017, pp. 4765–4774.

[183] Gjorgji Madjarov et al. "An extensive experimental comparison of methods for multi-label learning". In: *Pattern Recognition* 45.9 (2012). Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011), pp. 3084–3104. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2012.03.004. URL: http://www.sciencedirect.com/science/article/pii/S0031320312001203.

[184] Zheda Mai et al. "Online continual learning in image classification: An empirical survey". In: *Neurocomputing* 469 (2022), pp. 28–51.

[185] J. Maktoubian, M.S. Taskhiri, and P. Turner. "Intelligent Predictive Maintenance (IPdM) in Forestry: A Review of Challenges and Opportunities". In: *Forests* 12 (2021).

[186] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. "Piggyback: Adapting a single network to multiple tasks by learning to mask weights". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 67–82.

[187] Arun Mallya and Svetlana Lazebnik. "Packnet: Adding multiple tasks to a single network by iterative pruning". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 7765–7773.

[188] Daniel J Mankowitz et al. "Unicorn: Continual learning with a universal, off-policy agent". In: *arXiv preprint arXiv:1802.08294* (2018).

[189] Arianna Martinelli, Andrea Mina, and Massimo Moggi. "The enabling technologies of industry 4.0: examining the seeds of the fourth industrial revolution". In: *Industrial and Corporate Change* 30.1 (2021), pp. 161–188.

[190] Wojciech Masarczyk et al. "Logarithmic Continual Learning". In: *arXiv preprint arXiv:2201.06534* (2022).

[191] Benjamin Maschler, Thi Thu Huong Pham, and Michael Weyrich. "Regularization-based Continual Learning for Anomaly Detection in Discrete Manufacturing". In: *Procedia CIRP* 104 (2021), pp. 452–457.

[192] Benjamin Maschler, Sophia Tatiyosyan, and Michael Weyrich. "Regularization-based continual learning for fault prediction in lithium-ion batteries". In: *Procedia CIRP* 112 (2022), pp. 513–518.

[193] Benjamin Maschler et al. "Continual learning of fault prediction for turbofan engines using deep learning with elastic weight consolidation". In: *2020 25th IEEE international conference on emerging technologies and factory automation (ETFA)*. Vol. 1. IEEE. 2020, pp. 959–966.

[194] V. Mathew et al. *Prediction of Remaining Useful Lifetime (RUL) of turbofan engine using machine learning*. Dec. 20, 2017.

[195] Grzegorz Mazurek and Karolina Malagocka. "Perception of privacy and data protection in the context of the development of artificial intelligence". In: *Journal of Management Analytics* 6.4 (2019), pp. 344–364.

[196] Bryan McCann et al. "The natural language decathlon: Multitask learning as question answering". In: *arXiv preprint arXiv:1806.08730* (2018).

[197] James L McClelland. "Complementary learning systems in the brain. A connectionist approach to explicit and implicit cognition and memory." In: *Annals of the New York Academy of Sciences* 843 (1998), pp. 153–169.

[198] A.U. Mentsiev, E.R. Guzueva, and T.R. Magomaev. "Security challenges of the Industry 4.0". In: *J. Phys. Conf. Ser* 032074 (2020).

[199] D Merayo, A Rodriguez-Prieto, and AM Camacho. "Comparative analysis of artificial intelligence techniques for material selection applied to manufacturing in Industry 4.0". In: *Procedia Manufacturing* 41 (2019), pp. 42–49.

[200] Gabriele Merlin et al. "Practical Recommendations for Replay-based Continual Learning Methods". In: *arXiv preprint arXiv:2203.10317* (2022).

[201] George A. Miller. "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information". In: *The Psychological Review* 63.2 (Mar. 1956), pp. 81–97. URL: `http://www.musanim.com/miller1956/`.

[202] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[203] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[204] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. `https://christophm.github.io/interpretable-ml-book/`. 2019.

[205] Laszlo Monostori. "AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing". In: *Engineering applications of artificial intelligence* 16.4 (2003), pp. 277–291.

[206] S. Mousavi. "Simultaneous Control of the Production, Maintenance, and Inspection Strategies for a Failure-Prone Manufacturing System with Quality-Based Financial Penalties/Incentives". Ph. D. Thesis, Montreal, QC, Canada, 2021.

[207] Egon Mueller, Xiao-Li Chen, and Ralph Riedel. "Challenges and requirements for the application of industry 4.0: a special insight with the usage of cyber-physical system". In: *Chinese Journal of Mechanical Engineering* 30.5 (2017), pp. 1050–1057.

[208] Martin Mundt et al. "CLEVA-Compass: A Continual Learning EValuation Assessment Compass to Promote Research Transparency and Comparability". In: *arXiv preprint arXiv:2110.03331* (2021).

[209] S. Munirathinam and B. Ramadoss. *Big data predictive analtyics for proactive semiconductor equipment maintenance*. Washington, DC, USA, Oct. 27, 2014.

[210] W. James Murdoch et al. "Definitions, methods, and applications in interpretable machine learning". In: *Proceedings of the National Academy of Sciences* 116 (2019), pp. 22071–22080.

[211] Paolo Napoletano, Flavio Piccoli, and Raimondo Schettini. "Anomaly detection in nanofibrous materials by CNN-based self-similarity". In: *Sensors* 18.1 (2018), p. 209.

[212] W.P. Neumann et al. "Industry 4.0 and the human factor–A systems frame-work and analysis methodology for successful development". In: *Int. J. Prod. Econ* 233 (2021), p. 107992.

[213] Bojana Nikolic et al. "PREDICTIVE MANUFACTURING SYSTEMS IN INDUSTRY 4.0: TRENDS, BENEFITS AND CHALLENGES." In: *Annals of DAAAM & Proceedings* 28 (2017).

[214] Hussain Nizam et al. "Real-Time Deep Anomaly Detection Framework for Multivariate Time-Series Data in Industrial IoT". In: *IEEE Sensors Journal* 22.23 (2022), pp. 22836–22849.

[215] Mathew Mithra Noel and B Jaganatha Pandian. "Control of a nonlinear liquid level system using a new artificial neural network based reinforcement learning approach". In: *Applied Soft Computing* 23 (2014), pp. 444–451.

[216] Oleksiy Ostapenko et al. "Foundational Models for Continual Learning: An Empirical Study of Latent Replay". In: *arXiv preprint arXiv:2205.00329* (2022).

[217] Volker Paelke. "Augmented reality in the smart factory: Supporting workers in an industry 4.0. environment". In: *Proceedings of the 2014 IEEE emerging technology and factory automation (ETFA)*. IEEE. 2014, pp. 1–4.

[218] Zhaotai Pan et al. "Cognitive acoustic analytics service for Internet of Things". In: *2017 IEEE International Conference on Cognitive Computing (ICCC)*. IEEE. 2017, pp. 96–103.

[219] Marcel Panzer and Benedict Bender. "Deep reinforcement learning in production systems: a systematic literature review". In: *International Journal of Production Research* (2021), pp. 1–26.

[220] Amir Bahador Parsa et al. "Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis". In: *Accident Analysis & Prevention* 136 (2020), p. 105405. ISSN: 0001-4575. DOI: https://doi.org/10.1016/j.aap.2019.105405. URL: http://www.sciencedirect.com/science/article/pii/S0001457519311790.

[221] I. Pater, A. Reijns, and M. Mitici. "Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics". In: *Reliab. Eng. Syst. Saf* 221 (2022), p. 108341.

[222] V. Pedreira, D. Barros, and P. Pinto. "A review of attacks, vulnerabilities, and defenses in industry 4.0 with new chal-lenges on data sovereignty ahead". In: *Sensors* 21 (2021). CrossRef, p. 5189.

[223] Lorenzo Pellegrini et al. "Latent replay for real-time continual learning". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10203–10209.

[224] T. Pereira, L. Barreto, and A. Amaral. "Network and information security challenges within Industry 4.0 paradigm". In: *Procedia Manuf* 13 (2017), pp. 1253–1260.

[225] Pramuditha Perera and Vishal M Patel. "Learning deep features for one-class classification". In: *IEEE Transactions on Image Processing* 28.11 (2019), pp. 5450–5463.

[226] R.S. Peres et al. "IDARTS–Towards intelligent data analysis and real-time supervision for industry 4.0". In: *Comput. Ind* 101 (2018), pp. 138–146.

[227] Ali Pesaranghader, Herna L Viktor, and Eric Paquet. "McDiarmid drift detection methods for evolving data streams". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–9.

[228] Davide Dalle Pezze et al. "A Multi-label Continual Learning Framework to Scale Deep Learning Approaches for Packaging Equipment Monitoring". In: *arXiv preprint arXiv:2208.04227* (2022).

[229] Davide Dalle Pezze et al. "Continual Learning Approaches for Anomaly Detection". In: *arXiv preprint arXiv:2212.11192* (2022).

[230] Davide Dalle Pezze et al. "FORMULA: A Deep Learning Approach for Rare Alarms Predictions in Industrial Equipment". In: *IEEE Transactions on Automation Science and Engineering* (2021), pp. 1–12. DOI: 10.1109/TASE.2021.3127995.

[231] Duc T Pham and Ashraf A Afify. "Machine-learning techniques and their applications in manufacturing". In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 219.5 (2005), pp. 395–412.

[232] Jary Pomponi, Simone Scardapane, and Aurelio Uncini. "Pseudo-Rehearsal for Continual Learning with Normalizing Flows". In: *arXiv preprint arXiv:2007.02443* (2020).

[233] Jary Pomponi et al. "Efficient continual learning in neural networks with embedding regularization". In: *Neurocomputing* 397 (2020), pp. 139–148.

[234] T Praveenkumar et al. "Fault diagnosis of automobile gearbox based on machine learning techniques". In: *Procedia Engineering* 97 (2014), pp. 2092–2098.

[235] Rune Prytz et al. "Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data". In: *Engineering applications of artificial intelligence* 41 (2015), pp. 139–150.

[236] Qinglin Qi and Fei Tao. "A smart manufacturing service system based on edge computing, fog computing, and cloud computing". In: *IEEE Access* 7 (2019), pp. 86769–86777.

[237] W. Qiao et al. "A Survey on Wind Turbine Condition Monitoring and Fault Diagnosis—Part I: Components and Subsystems". In: *IEEE Trans. Ind. Electron* 62 (2015), pp. 6536–6545.

[238] Guo Fang Qiu et al. "A knowledge processing method for intelligent systems based on inclusion degree". In: *Expert systems* 20.4 (2003), pp. 187–195.

[239] Anand Rajaraman and Jeffrey David Ullman. "Data Mining". In: *Mining of Massive Datasets*. Cambridge University Press, 2011, pp. 1–17. DOI: 10.1017/CBO9781139058452.002.

[240] Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. "Lifelong generative modeling". In: *Neurocomputing* 404 (2020), pp. 381–400.

[241] Bruce Ratner. *Statistical and machine-learning data mining:: Techniques for better predictive modeling and analysis of big data*. CRC Press, 2017.

[242] Jesse Read et al. "Classifier chains for multi-label classification". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2009, pp. 254–269.

[243] Jesse Read et al. "Classifier chains for multi-label classification". In: *Machine learning* 85.3 (2011), pp. 333–359.

[244] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. "Learning multiple visual domains with residual adapters". In: *Advances in neural information processing systems* 30 (2017).

[245] A. Reiman et al. "Human factors and ergonomics in manufac-turing in the industry 4.0 context–A scoping review". In: *Technol. Soc* 65 (2021), p. 101572.

[246] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[247] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

[248] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-Precision Model-Agnostic Explanations". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). DOI: 10.1609/aaai.v32i1.11491. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11491.

[249] P.J. Rivera Torres et al. "Modeling preventive maintenance of manufacturing processes with probabilistic Boolean networks with interventions". In: *J. Intell. Manuf* 29 (2018), pp. 1941–1952.

[250] David Rolnick et al. "Experience replay for continual learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[251] Nicky de Rooij et al. "Generalized Class Incremental Learning For Classifying Work Equipment". In: ().

[252] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[253] Matthew Russell and Peng Wang. "Improved Representations for Continual Learning of Novel Motor Health Conditions through Few-Shot Prototypical Networks". In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2022, pp. 1551–1556.

[254] Andrei A Rusu et al. "Progressive neural networks". In: *arXiv preprint arXiv:1606.04671* (2016).

[255] Maria Sahakyan, Zeyar Aung, and Talal Rahwan. "Explainable Artificial Intelligence for Tabular Data: A Survey". In: *IEEE Access* 9 (2021), pp. 135392–135422. DOI: 10.1109/ACCESS.2021.3116481.

[256] Takaya Saito and Marc Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PloS one* 10.3 (2015), e0118432.

[257] Babak Saleh and Ahmed Elgammal. "Large-scale classification of fine-art paintings: Learning the right metric on the right feature". In: *arXiv preprint arXiv:1505.00855* (2015).

[258] Seyed Sadegh Mohseni Salehi, D. Erdoğmuş, and A. Gholipour. "Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks". In: *MLMI@MICCAI*. 2017.

[259] Jania Astrid Saucedo-Martinez et al. "Industry 4.0 framework for management and operations: a review". In: *Journal of ambient intelligence and humanized computing* 9.3 (2018), pp. 789–801.

[260] Abhinav Saxena and Kai Goebel. "Turbofan engine degradation simulation data set". In: *NASA Ames Prognostics Data Repository* (2008), pp. 1551–3203.

[261] Thomas Schlegl et al. "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery". In: *CoRR* abs/1703.05921 (2017). arXiv: 1703.05921. URL: http://arxiv.org/abs/1703.05921.

[262] Bernhard Scholkopf et al. "Estimating the support of a high-dimensional distribution". In: *Neural computation* 13.7 (2001), pp. 1443–1471.

[263] Oscar Serradilla et al. "Interpreting Remaining Useful Life estimations combining Explainable Artificial Intelligence and domain knowledge in industrial machinery". In: *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (2020), pp. 1–8.

[264] E. Sezer et al. *An Industry 4.0-Enabled Low Cost Predictive Maintenance Approach for SMEs*. Stuttgart, Germany, June 17, 2018.

[265] Feng Shan and Li D Xu. "A hybrid knowledge-based system for urban development". In: *Expert Systems with Applications* 10.1 (1996), pp. 157–163.

[266] Yong Shi, Jie Yang, and Zhiquan Qi. "Unsupervised anomaly segmentation via deep feature reconstruction". In: *Neurocomputing* 424 (2021), pp. 9–22.

[267] Dongsub Shim et al. "Online class-incremental continual learning with adversarial shapley value". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 11. 2021, pp. 9630–9638.

[268] Donghee Shin. "The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable AI". In: *International Journal of Human-Computer Studies* 146 (2021), p. 102551. ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2020.102551. URL: https://www.sciencedirect.com/science/article/pii/S1071581920301531.

[269] Hanul Shin et al. "Continual learning with deep generative replay". In: *arXiv preprint arXiv:1705.08690* (2017).

[270] Jong-Ho Shin, Hong-Bae Jun, and Jae-Gon Kim. "Dynamic control of intelligent parking guidance using neural network predictive control". In: *Computers & Industrial Engineering* 120 (2018), pp. 15–30.

[271] Alexander Sigov et al. "Emerging enabling technologies for industry 4.0 and beyond". In: *Information Systems Frontiers* (2022), pp. 1–11.

[272] L. Silvestri et al. "Maintenance transformation through Industry 4.0 technologies: A systematic literature review". In: *Comput. Ind* 123 (2020), p. 103335.

[273] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[274] Henrique Siqueira and Onay Urfalioglu. "Continuous Adaptation to Distribution Drifts Through Continual Learning in Manufacturing". In: *Book of Abstracts*, p. 157.

[275] Abir Smiti. "When machine learning meets medical world: Current status and future challenges". In: *Computer Science Review* 37 (2020), p. 100280. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100280. URL: http://www.sciencedirect.com/science/article/pii/S157401372030126X.

[276] Symone Gomes Soares and Rui Araujo. "An on-line weighted ensemble of regressor models to handle concept drifts". In: *Engineering Applications of Artificial Intelligence* 37 (2015), pp. 392–406.

[277] Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. "Spacenet: Make free space for continual learning". In: *Neurocomputing* 439 (2021), pp. 1–11.

[278] Timothy P Spiller. "Quantum information technology". In: *Materials Today* 6.1 (2003), pp. 30–36.

[279] Carolin Strobl et al. "Conditional variable importance for random forests". In: *BMC bioinformatics* 9.1 (2008), p. 307.

[280] Erik Štrumbelj, Igor Kononenko, and M Robnik Šikonja. "Explaining instance classifications with interactions of subsets of feature values". In: *Data & Knowledge Engineering* 68.10 (2009), pp. 886–904.

[281] Chuan-Jun Su and Shi-Feng Huang. "Real-time big data analytics for hard disk drive predictive maintenance". In: *Computers & Electrical Engineering* 71 (2018), pp. 93–101.

[282] Masashi Sugiyama and Motoaki Kawanabe. *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press, 2012.

[283] G.A. Susto, A. Beghi, and C. Luca. "A predictive maintenance system for epitaxy processes based on filtering and prediction techniques". In: *IEEE Trans. Semicond. Manuf* 25 (2012), pp. 638–649.

[284] G.A. Susto et al. "Machine learning for predictive maintenance: A multiple classifier approach". In: *IEEE Trans. Ind. Inform* 11 (2015), pp. 812–820.

[285] Gian Antonio Susto, Alessandro Beghi, and Cristina De Luca. "A predictive maintenance system for epitaxy processes based on filtering and prediction techniques". In: *IEEE Transactions on Semiconductor Manufacturing* 25.4 (2012), pp. 638–649.

[286] Gian Antonio Susto et al. "A hidden-Gamma model-based filtering and prediction approach for monotonic health factors in manufacturing". In: *Control Engineering Practice* 74 (2018), pp. 84–94.

[287] Gian Antonio Susto et al. "Machine learning for predictive maintenance: A multiple classifier approach". In: *IEEE Transactions on Industrial Informatics* 11.3 (2014), pp. 812–820.

[288] Gian Antonio Susto et al. "Prediction of integral type failures in semiconductor manufacturing through classification methods". In: *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE. 2013, pp. 1–4.

[289] Fei Tao et al. "Cloud manufacturing: a computing and service-oriented manufacturing model". In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 225.10 (2011), pp. 1969–1976.

[290] Fei Tao et al. "Manufacturing service management in cloud manufacturing: overview and future research directions". In: *Journal of Manufacturing Science and Engineering* 137.4 (2015).

[291] D. Tarapore, A. L. Christensen, and J. Timmis. *Generic, Scalable and Decentralized Fault Detection for Robot Swarms*. Last access in 12/02/2019. 2017. URL: https://zenodo.org/record/831471#.WwQIPUgvxPY.

[292] Adane Nega Tarekegn, Mario Giacobini, and Krzysztof Michalak. "A review of methods for imbalanced multi-label classification". In: *Pattern Recognition* 118 (2021), p. 107965.

[293] David MJ Tax and Robert PW Duin. "Support vector domain description". In: *Pattern recognition letters* 20.11-13 (1999), pp. 1191–1199.

[294] Hasan Tercan, Philipp Deibert, and Tobias Meisen. "Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer". In: *Journal of Intelligent Manufacturing* 33.1 (2022), pp. 283–292.

[295] M.C. Testik. "Expert Systems with Applications A review of data mining applications for quality improvement in manufacturing industry". In: *Expert Syst. Appl* 38 (2011), pp. 13448–13467.

[296] Kleanthis Thramboulidis and Foivos Christoulakis. "UML4IoT—A UML-based approach to exploit IoT in cyber-physical manufacturing systems". In: *Computers in Industry* 82 (2016), pp. 259–272.

[297] Diego Tosato et al. *Alarm Logs in Packaging Industry*. http://dx.doi.org/10.17632/4nhx2x67cd.1. 2020.

[298] Diego Tosato et al. *Alarm Logs in Packaging Industry (ALPI)*. 2020. DOI: 10.21227/nfv6-k750. URL: http://dx.doi.org/10.21227/nfv6-k750.

[299] Minh Trinh et al. "Development of a Framework for Continual Learning in Industrial Robotics". In: ().

[300] Cheng-Hao Tu, Cheng-En Wu, and Chu-Song Chen. "Extending conditional convolution structures for enhancing multitasking continual learning". In: *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE. 2020, pp. 1605–1610.

[301] Meghan Reading Turchioe et al. "Systematic review of current natural language processing methods and applications in cardiology". In: *Heart* 108.12 (2022), pp. 909–916.

[302] Wisdom Udo and Yar Muhammad. "Data-driven predictive maintenance of wind turbine based on SCADA data". In: *IEEE Access* 9 (2021), pp. 162370–162388.

[303] Eckart Uhlmann et al. "Cluster identification of sensor data for predictive maintenance in a Selective Laser Melting machine tool". In: *Procedia manufacturing* 24 (2018), pp. 60–65.

[304] Gido M Van de Ven and Andreas S Tolias. "Three scenarios for continual learning". In: *arXiv preprint arXiv:1904.07734* (2019).

[305] Thomas van Klompenburg, Ayalew Kassahun, and Cagatay Catal. "Crop yield prediction using machine learning: A systematic literature review". In: *Computers and Electronics in Agriculture* 177 (2020), p. 105709. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2020.105709. URL: http://www.sciencedirect.com/science/article/pii/S0168169920302301.

[306] Rocio Vargas, Amir Mosavi, and Ramon Ruiz. "Deep learning: a review". In: (2017).

[307] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[308] J.W. Veile et al. "Lessons learned from Industry 4.0 implementation in the German manufacturing industry". In: *J. Manuf. Technol. Manag* 31 (2019), pp. 977–997.

[309] Gido M. van de Ven and Andreas S. Tolias. "Three scenarios for continual learning". In: *CoRR* abs/1904.07734 (2019). arXiv: 1904.07734. URL: http://arxiv.org/abs/1904.07734.

[310] Kevin Villalobos, Johan Suykens, and Arantza Illarramendi. "A flexible alarm prediction system for smart manufacturing scenarios following a forecaster–analyzer approach". In: *Journal of Intelligent Manufacturing* (2020), pp. 1–22.

[311] Athanasios Voulodimos et al. "Deep learning for computer vision: A brief review". In: *Computational intelligence and neuroscience* 2018 (2018).

[312] J. Wan et al. "A Manufacturing Big Data Solution for Active Preventive Maintenance". In: *IEEE Trans. Ind. Inform* 13 (2017), pp. 2039–2047.

[313] Jiafu Wan et al. "Mobile services for customization manufacturing systems: An example of industry 4.0". In: *IEEE Access* 4 (2016), pp. 8977–8986.

[314] Haiya Wang, Zhongqing Yu, and Lu Guo. "Real-time online fault diagnosis of rolling bearings based on KNN algorithm". In: *Journal of Physics: Conference Series*. Vol. 1486. 3. IOP Publishing. 2020, p. 032019.

[315] J. Wang et al. "Big data analytics for intelligent manufacturing systems: A review". In: *J. Manuf. Syst* 62 (2021), pp. 738–752.

[316] Jiandong Wang et al. "An overview of industrial alarm systems: Main causes for alarm overloading, research status, and open problems". In: *IEEE Transactions on Automation Science and Engineering* 13.2 (2015), pp. 1045–1061.

[317] Jiang Wang et al. "Cnn-rnn: A unified framework for multi-label image classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2285–2294.

[318] Jiaxuan Wang, Jenna Wiens, and Scott Lundberg. "Shapley Flow: A Graph-based Approach to Interpreting Model Predictions". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 13–15 Apr 2021, pp. 721–729. URL: `https://proceedings.mlr.press/v130/wang21b.html`.

[319] Jinjiang Wang et al. "Deep learning for smart manufacturing: Methods and applications". In: *Journal of manufacturing systems* 48 (2018), pp. 144–156.

[320] Y. Wang et al. *Advanced Manufacturing and Automation XI*. Singapore: Springer, 2022.

[321] Yingwei Wang. "Cloud-dew architecture". In: *International Journal of Cloud Computing* 4.3 (2015), pp. 199–210.

[322] Yining Wang et al. *A Theoretical Analysis of NDCG Type Ranking Measures*. 2013. arXiv: `1304.6480 [cs.LG]`.

[323] Zhepei Wang et al. "Learning Representations for New Sound Classes With Continual Self-Supervised Learning". In: *arXiv preprint arXiv:2205.07390* (2022).

[324] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[325] Bernd Waschneck et al. "Deep reinforcement learning for semiconductor production scheduling". In: *2018 29th annual SEMI advanced semiconductor manufacturing conference (ASMC)*. IEEE. 2018, pp. 301–306.

[326] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[327] S. Wellsandt et al. "Hybrid-augmented intelligence in predictive maintenance with digital intelligent assistants". In: *Annu. Rev. Control* 53 (2022), pp. 382–390.

[328] *What is industry 4.0?* URL: `https://www.connectwell.com//global/news/blogs/industry-40iiot.aspx`.

[329] *What is PHM?* URL: `https://carleton.ca/phm/what-is-phm/`.

[330] Felix Wiewel and Bin Yang. "Continual Learning for Anomaly Detection with Variational Autoencoder". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 3837–3841. DOI: `10.1109/ICASSP.2019.8682702`.

[331] Maciej Wolczyk et al. "Continual world: A robotic benchmark for continual reinforcement learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28496–28510.

[332] Marilyn Wolf and Dimitrios Serpanos. "Safety and security of cyber-physical and internet of things systems [point of view]". In: *Proceedings of the IEEE* 105.6 (2017), pp. 983–984.

[333] Kok-Seng Wong and Myung Ho Kim. "Privacy protection for data-driven smart manufacturing systems". In: *International Journal of Web Services Research (IJWSR)* 14.3 (2017), pp. 17–32.

[334] Dongsheng Wu et al. "Fault diagnosis of TE process based on incremental learning". In: *2020 39th Chinese Control Conference (CCC)*. IEEE. 2020, pp. 4227–4232.

[335] T. Wuest et al. "Machine learning in manufacturing: Advantages, challenges, and applications". In: *Prod. Manuf. Res* 4 (2016), pp. 23–45.

[336] Thorsten Wuest et al. "Machine learning in manufacturing: advantages, challenges, and applications". In: *Production & Manufacturing Research* 4.1 (2016), pp. 23–45.

[337] Li Da Xu, Eric L Xu, and Ling Li. "Industry 4.0: state of the art and future trends". In: *International journal of production research* 56.8 (2018), pp. 2941–2962.

[338] Yizhou Xu, Jiandong Wang, and Yan Yu. "Alarm Event Prediction From Historical Alarm Flood Sequences Based on Bayesian Estimators". In: *IEEE Transactions on Automation Science and Engineering* 17.2 (2019), pp. 1070–1075.

[339] Fan Yang et al. "Improved correlation analysis and visualization of industrial alarm data". In: *ISA transactions* 51.4 (2012), pp. 499–506.

[340] Jie Yang, Zhiquan Qi, and Yong Shi. "Learning to incorporate structure knowledge for image inpainting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 12605–12612.

[341] Jie Yang et al. "Visual anomaly detection for images: A survey". In: *arXiv preprint arXiv:2109.13157* (2021).

[342] Jie Yang et al. "Visual Anomaly Detection for Images: A Systematic Survey". In: *Procedia Computer Science* 199 (2022), pp. 471–478.

[343] Qihan Yang, Fan Feng, and Rosa Chan. *A Benchmark and Empirical Analysis for Replay Strategies in Continual Learning*. 2022. DOI: 10.48550/ARXIV.2208.02660. URL: https://arxiv.org/abs/2208.02660.

[344] Qihan Yang, Fan Feng, and Rosa HM Chan. "A Benchmark and Empirical Analysis for Replay Strategies in Continual Learning". In: *International Workshop on Continual Semi-Supervised Learning*. Springer. 2022, pp. 75–90.

[345] Zichao Yang et al. "Hierarchical attention networks for document classification". In: *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2016, pp. 1480–1489.

[346] Ting-Fang Yen et al. "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks". In: *Proceedings of the 29th Annual Computer Security Applications Conference*. 2013, pp. 199–208.

[347] *Fault Diagnosis of Control Valve Based on Fusion of Deep Learning and Elastic Weight Consolidation*. Vol. BATH/ASME 2022 Symposium on Fluid Power and Motion Control. Fluid Power Systems Technology. V001T01A023. Sept. 2022. DOI: 10.1115/FPMC2022-89359. eprint: https://asmedigitalcollection.asme.org/FPST/proceedings-pdf/FPMC2022/86335/V001T01A023/6942347/v001t01a023-fpmc2022-89359.pdf. URL: https://doi.org/10.1115/FPMC2022-89359.

[348] Jaemin Yoo and Lee Sael. "EDiT: Interpreting Ensemble Models via Compact Soft Decision Trees". In: *2019 IEEE International Conference on Data Mining (ICDM)* (2019), pp. 1438–1443.

[349] Y. Yoo, S.H. Park, and J.G. Baek. "A Clustering-Based Equipment Condition Model of Chemical Vapor Deposition Process". In: *Int. J. Precis. Eng. Manuf* 20 (2019), pp. 1677–1689.

[350] Jaehong Yoon et al. "Lifelong learning with dynamically expandable networks". In: *arXiv preprint arXiv:1708.01547* (2017).

[351] Guoxian Yu et al. "Transductive multi-label ensemble classification for protein function prediction". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 1077–1085.

[352] Jiawei Yu et al. *FastFlow: Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows*. 2021. arXiv: 2111.07677 [cs.CV].

[353] Tianhe Yu et al. "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning". In: *Conference on robot learning*. PMLR. 2020, pp. 1094–1100.

[354] Wenjin Yu et al. "A global manufacturing big data ecosystem for fault detection in predictive maintenance". In: *IEEE Transactions on Industrial Informatics* 16.1 (2019), pp. 183–192.

[355] Xiaofeng Yuan et al. "A dynamic CNN for nonlinear dynamic feature learning in soft sensor modeling of industrial process data". In: *Control Engineering Practice* 104 (2020), p. 104614.

[356] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[357] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. "Reconstruction by inpainting for visual anomaly detection". In: *Pattern Recognition* 112 (2021), p. 107706. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2020.107706. URL: https://www.sciencedirect.com/science/article/pii/S0031320320305094.

[358] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. "Reconstruction by inpainting for visual anomaly detection". In: *Pattern Recognition* 112 (2021), p. 107706.

[359] J. Zenisek, F. Holzinger, and M. Affenzeller. "Machine learning based concept drift detection for predictive maintenance". In: *Comput. Ind. Eng* 137 (2019), p. 106031.

[360] Friedemann Zenke, Ben Poole, and Surya Ganguli. "Continual learning through synaptic intelligence". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3987–3995.

[361] H. Zhang et al. "Attention-Based LSTM Network for Rotatory Machine Remaining Useful Life Prediction". In: *IEEE Access* 8 (2020), pp. 132188–132199.

[362] Jingxin Zhang, Donghua Zhou, and Maoyin Chen. "Monitoring multimode processes: A modified PCA algorithm with continual learning ability". In: *Journal of Process Control* 103 (2021), pp. 76–86.

[363] Jingxin Zhang et al. "Continual Learning for Multimode Dynamic Process Monitoring with Applications to an Ultra-Supercritical Thermal Power Plant". In: *IEEE transactions on Automation Science and Engineering* (2022).

[364] Lin Zhang, Ying-Chang Liang, and Dusit Niyato. "6G Visions: Mobile ultra-broadband, super internet-of-things, and artificial intelligence". In: *China Communications* 16.8 (2019), pp. 1–14.

[365] Min-Ling Zhang and Zhi-Hua Zhou. "A review on multi-label learning algorithms". In: *IEEE transactions on knowledge and data engineering* 26.8 (2013), pp. 1819–1837.

[366] Yu Zhang et al. "Large-scale multi-label classification using unknown streaming images". In: *Pattern Recognition* 99 (2020), p. 107100.

[367] Yulun Zhang et al. "Residual dense network for image restoration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.7 (2020), pp. 2480–2495.

[368] Junhui Zheng et al. "Bearing Fault Diagnosis via Incremental Learning Based on the Repeated Replay Using Memory Indexing (R-REMIND) Method". In: *Machines* 10.5 (2022), p. 338.

[369] Jianfeng Zhu et al. "Dynamic alarm prediction for critical alarms using a probabilistic model". In: *Chinese Journal of Chemical Engineering* 24.7 (2016), pp. 881–885.

[370] T. Zonta et al. "Predictive maintenance in the Industry 4.0: A systematic literature review". In: *Comput. Ind. Eng* 150 (2020), p. 106889.

[371] Tiago Zonta et al. "Predictive maintenance in the Industry 4.0: A systematic literature review". In: *Computers & Industrial Engineering* 150 (2020), p. 106889.

[372] Bożena Zwolińska and Jakub Wiercioch. "Selection of Maintenance Strategies for Machines in a Series-Parallel System". In: *Sustainability* 14.19 (2022), p. 11953.