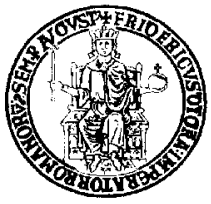
UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II



SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

Department of Industrial Engineering

Doctoral Program in Industrial Engineering

ARTIFICIAL NEURAL NETWORKS AND STATISTICS FOR QUALITY TECHNOLOGY IN RAILWAY INDUSTRY

Gianluca Sposito

Tutors
Prof. Ing. Antonio Lepore
Prof. Ing. Biagio Palumbo

Coordinator
Prof. Ing. Michele Grassi

Year 2024 - XXXVI CYCLE

Summary

In the modern Industry 4.0 and 5.0 framework, advancements in technology have enabled an unprecedented collection of a huge amount of data, which is parallelly becoming more complex and typically demands the implementation of novel statistical methodologies. In particular, this thesis aims to develop methods for Statistical Process Control (SPC) through the integration of artificial neural network, or simply Neural Network (NN), algorithms to tackle modern industrial problems in the digitalization era. The industrial scenario that motivates the research work is the monitoring of the Heating, Ventilation and Air Conditioning (HVAC) systems installed onboard modern passenger trains.

The proposed methodologies are developed into two main parts in this dissertation. In the first part, the simultaneous SPC of the sensor signals coming from each HVAC system of the same train is regarded as produced by a Multiple Stream Process (MSP). In particular, this part presents a NN-based approach designed to enhance the monitoring of a MSP and the identification of the streams responsible for the out-of-control alarm. The second part of this dissertation explores the possibility of using a nonparametric SPC approach based on autoencoders, which are a type of NN used to learn an efficient representation of the input data, and a novel control chart for functional data based on a NN specifically trained to handle data in the form of profiles.

The common ground of all the proposed methodologies is the need to provide tools to industrial practitioners that give clear indications if an anomaly occurs and are easy to implement through the development of open source packages, and the use of operational data in the railway sector to demonstrate their practical applicability.

Contents

Contents ii

List of Figures iv

List of Tables vii

List of Acronyms xii

Introduction 1

1 Neural network based control charting for multiple stream processes with an application to Heating, Ventilation and Air Conditioning (HVAC) systems in passenger railway vehicles 9
- 1.1 Introduction . . . 9
- 1.2 Methodology . . . 13
- 1.3 Simulation study . . . 18
- 1.4 A real-case study . . . 23
- 1.5 Conclusion . . . 30
- 1.6 Supplementary Materials . . . 30

2 An artificial neural network approach for out-of-control stream identification in multiple stream processes 37
- 2.1 Introduction . . . 37
- 2.2 The proposed neural network approach . . . 40
- 2.3 A case study: identification of out-of-control streams from passenger train Heating, Ventilation and Air Conditioning (HVAC) systems . . . 47
- 2.4 Conclusions . . . 52
- 2.5 Supplementary Material . . . 52

3 Neural network for the statistical process control of Heating, Ventilation and Air Conditioning (HVAC) systems in passenger rail vehicles 55
- 3.1 Introduction . . . 55
- 3.2 The proposed approach . . . 56
- 3.3 Real-case study . . . 60
- 3.4 Conclusions . . . 63

4 Functional neural network control chart 65
- 4.1 Introduction . . . 65

4.2 Methodology 67
4.3 Simulation study 73
4.4 Case study: monitoring of Heating, Ventilation and Air Conditioning (HVAC) systems on modern passenger trains 77
4.5 Conclusions 82

5 R and Python Packages **83**
5.1 The `Python` package `NN4MSP` 83
5.2 The `Python` package `NN4OCMSP` 84
5.3 Functional neural network control chart 86

Conclusion and Future Development **91**

Bibliography **97**

List of Figures

1.1 NN architecture 15

1.2 Receiver Operating Characteristic (ROC) curve for a process with $s = 6$, $n = 1$ (a) and $s = 6$ and $n = 5$ (b). 19

1.3 In Control (IC) ΔT stream for each coach of the train 1 over 50 subgroups (10 minutes worth of data). The ΔT signals from each train's coach show similar behavior and all the six onboard Heating, Ventilation and Air Conditioning (HVAC) systems properly work. As we can see, there is clear graphical evidence that the variability over time is much greater than the differences between streams, thus our proposed methodology is a suitable tool to monitor this kind of process. 26

1.4 ΔT stream for each coach of the train 2. The process is Out of Control (OC) and an assignable cause affects the output from one stream. Coach 5 is characterized by higher ΔT values, that is the temperature inside coach 5 is higher than the setpoint temperature set by the European regulations. As we expected, its signal behavior is significantly different from the other coach signals. 26

1.5 Control chart based on the Neural Network (NN) predicted probability. The single stream shifting is properly detected. 27

1.6 ΔT stream for each coach of the train 3. The process is Out of Control (OC) and an assignable cause affects the output from four streams. The ΔT signals of coaches 1,2,4,5 increase together, while the ΔT signals of the remaining two coaches have a typical In Control (IC) behavior. 28

1.7 Control chart based on the Neural Network (NN) predicted probability. The multiple streams shifting off target are properly detected. 28

1.8 Time-series plot of residuals for each coach of the train 2. 29

1.9 Time-series plot of residuals for each coach of the train 3. 29

1.10 A 6-stream simulated process with a shift in one stream at Sample #10. 33

1.11 Time-series plot of residuals of a 6-stream simulated process with a shift in one stream at sample #10. 33

2.1 Data plot of 30 samples of a Multiple Stream Process (MSP) with 4 streams with an upward shift in streams 3 and 4 from sample 15 to 30. 38

2.2 Accuracy (%) averaged over p of the Shewhart Control Chart (SCC) for each stream and the proposed Neural Network (NN) approach for different sample sizes (a) $n = 1$, (b) $n = 3$, (c) $n = 5$ for a Multiple Stream Process (MSP) with $s = 4$ streams and standard deviations $\sigma_A = 4$ and $\sigma_e = 1$, at different numbers $p = 1, 2, \ldots, s$ of Out of Control (OC) streams with mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. 44

2.3 Accuracy (%) averaged over p of the Shewhart Control Chart (SCC) for each stream and the proposed Neural Network (NN) approach for a different number of streams (a) $s = 4$, (b) $s = 6$ and (c) $s = 8$ for a Multiple Stream Process (MSP) with $n = 5$, $\sigma_A = 4$, and $\sigma_e = 1$, at different numbers $p = 1, 2, \ldots, s$ of Out of Control (OC) streams with mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. . 45

2.4 Accuracy (%) averaged over p of the Shewhart Control Chart (SCC) for each stream and the proposed Neural Network (NN) approach for different common variations (a) $\sigma_A = 2$, (b) $\sigma_A = 4$, (c) $\sigma_A = 6$ and (d) $\sigma_A = 8$ for a Multiple Stream Process (MSP) with $s = 6$, $n = 5$, $\sigma_e = 1$, at different numbers $p = 1, 2, \ldots, s$ of Out of Control (OC) streams with mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$.). 46

2.5 In Control (IC) ΔT stream for each coach of train A over 25 samples collected on the 1st of July. The streams show similar behavior and seem to be In Control (IC). There is graphical evidence that the common variability, σ_A^2, is larger than the within-stream variability, σ_e^2. 49

2.6 ΔT stream for each coach of train B over 15 samples, collected on the 8th of August only, from which we know that a fault occurred. The process is Out of Control (OC) and an assignable cause is known to affect the output coach 6 from sample 8. 50

2.7 Range and overall mean control charts for the monitoring of the Heating, Ventilation and Air Conditioning (HVAC) signals collected from each coach of train B. The red horizontal line indicates the Control Limit (CL)s at $\alpha = 0.05$. The range control chart starts signaling from sample 9. 51

2.8 Time-series plot of the residuals from an Out of Control (OC) Multiple Stream Process (MSP) with $s = 6$, $\mu = 0$, $\sigma_e = 1$, $\sigma_A = 4$, $n = 1$ and an upward shift of magnitude $10\sigma_e$ in the 4th stream from sample 15. 53

2.9 Time-series plot of the residuals from an Out of Control (OC) Multiple Stream Process (MSP) with $s = 6$, $\mu = 0$, $\sigma_e = 1$, $\sigma_A = 4$, $n = 1$ and an upward shift of magnitude $5\sigma_e$ in the first four streams from sample 15. 53

3.1 H^2 control chart used for the perspective monitoring of Heating, Ventilation and Air Conditioning (HVAC) data. Each point indicates the monitoring statistic value at each point in time. The bold line indicates the Upper Control Limit (UCL) at $\alpha' = 0.025$, whereas the dashed line indicates the instant at which a fault is known to occur. 63

3.2 Square Prediction Error (SPE) control chart used for the perspective monitoring of Heating, Ventilation and Air Conditioning (HVAC) data. Each point indicates the monitoring statistic value at each point in time. The bold line indicates the Upper Control Limit (UCL) at $\alpha' = 0.025$, whereas the dashed line indicates the instant at which a fault is known to occur. 64

4.1 Outline of the Functional Neural Network Control Chart (FNNCC) approach. . . 73

4.2 Estimated ARL_1 achieved by Functional Neural Network Control Chart (FNNCC), Functional Regression Control Chart (FRCC), and Shewhart Control Chart (SCC) for each simulated scenario, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$. 75

4.3 Estimated ARL_1 achieved by Functional Neural Network Control Chart (FNNCC), Functional Regression Control Chart (FRCC), and Shewhart Control Chart (SCC) for each simulated scenario, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$ when the functional covariate is subject to a mean shift 76
4.4 Estimated ARL_1 achieved by Functional Neural Network Control Chart (FNNCC), RawdataMLPCC, and BsplineMLPCC in Scenario B both in the presence (a) and absence (b) of a mean covariate shift, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$. 78
4.5 Estimated ARL_1 achieved by Functional Neural Network Control Chart (FNNCC), RawdataMLPCC, and BsplineMLPCC in Scenario C both in the presence (a) and absence (b) of a mean covariate shift, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$. 78
4.6 Estimated ARL_1 achieved by Functional Neural Network Control Chart (FNNCC), RawdataMLPCC, and BsplineMLPCC in Scenario D both in the presence (a) and absence (b) of a mean covariate shift, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$. 79
4.7 Estimated ARL_1 achieved by Functional Neural Network Control Chart (FNNCC), RawdataMLPCC, and BsplineMLPCC in Scenario E both in the presence (a) and absence (b) of a mean covariate shift, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$. 79
4.8 100 random samples of the two functional profiles from the Heating, Ventilation and Air Conditioning (HVAC) training set. 81
4.9 Phase II Functional Neural Network Control Chart (FNNCC). Each point corresponds to a voyage and the values of the corresponding residual are reported. Horizontal dashed lines are the Control Limit (CL)s and the point above the Control Limit (CL)s denotes the Out of Control (OC) observation. 81

List of Tables

1.1 Cut-off Value (CV)s of the proposed Neural Network (NN)s corresponding to typical false alarm rate values $\alpha = 0.0027, 0.01, 0.02, 0.05$ for a process with a number of streams $s = 6, 8, 10$ and subgroup sample size $n = 1$ and $n = 5$. 20

1.2 ARL_1 of the R_t, CUmulative SUM (CUSUM)-R_t control charts (Mortell & Runger 1995) and the proposed Neural Network (NN) control charting scheme based on 100000 simulations of a Multiple Stream Process (MSP) with $s = 6$ streams, at different number $l = 1, 2, .., s - 1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 1$. For each Out of Control (OC) scenario, the lowest ARL_1 value is marked in bold. . . 20

1.3 ARL_1 of the R_t, CUmulative SUM (CUSUM)-R_t control charts (Mortell & Runger 1995) and the proposed Neural Network (NN) control charting scheme based on 100000 simulations of a Multiple Stream Process (MSP) with $s = 6$ streams, at different number $l = 1, 2, .., s - 1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 5$. For each Out of Control (OC) scenario, the lowest ARL_1 value is marked in bold. . . 21

1.4 ARL_1 of the R_t, CUmulative SUM (CUSUM)-R_t control charts (Mortell & Runger 1995) and the proposed Neural Network (NN) control charting scheme based on 100000 simulations of a Multiple Stream Process (MSP) with $s = 8$ streams, at different number $l = 1, 2, .., s - 1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 1$. For each Out of Control (OC) scenario, the lowest ARL_1 value is marked in bold. . . 21

1.5 ARL_1 of the R_t, CUmulative SUM (CUSUM)-R_t control charts (Mortell & Runger 1995) and the proposed Neural Network (NN) control charting scheme based on 100000 simulations of a Multiple Stream Process (MSP) with $s = 8$ streams, at different number $l = 1, 2, .., s - 1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 5$. For each Out of Control (OC) scenario, the lowest ARL_1 value is marked in bold. . . 21

1.6 ARL_1 of the R_t, CUmulative SUM (CUSUM)-R_t control charts (Mortell & Runger 1995) and the proposed Neural Network (NN) control charting scheme based on 100000 simulations of a Multiple Stream Process (MSP) with $s = 10$ streams, at different number $l = 1, 2, .., s - 1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 1$. For each Out of Control (OC) scenario, the lowest ARL_1 value is marked in bold. . . 22

1.7 ARL_1 of the R_t, CUmulative SUM (CUSUM)-R_t control charts (Mortell & Runger 1995) and the proposed Neural Network (NN) control charting scheme based on 100000 simulations of a Multiple Stream Process (MSP) with $s = 10$ streams, at different number $l = 1, 2, .., s - 1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 5$. For each Out of Control (OC) scenario, the lowest ARL_1 value is marked in bold. . . 22
1.8 Operational variables acquired for each train and coach considered in the proposed approach. 24
1.9 Comparison of ARL_1 of the Chi-squared control chart (Wludyka & Jacobs 2002*a*) and of the proposed Neural Network (NN) approach based on 100000 simulations of a Multiple Stream Binomial Process (MSBP) with $s = 6$ streams, at different number $l = 1, ..., s$ of streams that shift off the target and mean shift size $\Delta p = -2.0\sigma, -1.0\sigma, -0.5\sigma, 0.5\sigma, 1.0\sigma, 2.0\sigma$. 35
1.10 Comparison of ARL_1 of the group p control chart (Wludyka & Jacobs 2002*b*) and of the proposed Neural Network (NN) approach based on 100000 simulations of a Multiple Stream Binomial Process (MSBP) with $s = 6$ streams, at different number $l = 1, ..., s$ of streams that shift off the target and mean shift size $\Delta p = -2.0\sigma, -1.0\sigma, -0.5\sigma, 0.5\sigma, 1.0\sigma, 2.0\sigma$. 35

2.1 Accuracy (%) of the Shewhart Control Chart (SCC) and the proposed Neural Network (NN) approach at different sample sizes n for a Multiple Stream Process (MSP) with $s = 4$, $\sigma_A = 4$, $\sigma_e = 1$, averaged over different numbers $p = 1, 2, \ldots, s$ of Out of Control (OC) streams and mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. For each Out of Control (OC) scenario, the highest accuracy value is marked in bold. 43
2.2 Accuracy (%) of the Shewhart Control Chart (SCC) and the proposed Neural Network (NN) approach at different numbers of streams s for a Multiple Stream Process (MSP) with $n = 5$, $\sigma_A = 4$, and $\sigma_e = 1$, averaged over different numbers $p = 1, 2, \ldots, s$ of Out of Control (OC) streams with and mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. For each Out of Control (OC) scenario, the highest accuracy value is marked in bold. 44
2.3 Accuracy (%) of the Shewhart Control Chart (SCC) and the proposed Neural Network (NN) approach at different ratios σ_A/σ_e for Multiple Stream Process (MSP) with $s = 6$, $n = 5$, $\sigma_e = 1$, averaged over different numbers $p = 1, 2, \ldots, s$ of Out of Control (OC) streams and mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. For each Out of Control (OC) scenario, the highest accuracy value is marked in bold. 45
2.4 EM (%) of the Shewhart Control Chart (SCC) and the proposed Neural Network (NN) approach for a Multiple Stream Process (MSP) with $s = 6$ streams, sample size $n = 5$, standard deviations $\sigma_A = 2$ and $\sigma_e = 1$, at different number $p = 1, 2, .., s$ of Out of Control (OC) streams with a positive mean shift size $\Delta\mu = \sigma_e, 2\sigma_e, 3\sigma_e, 4\sigma_e, 5\sigma_e$. For each Out of Control (OC) scenario, the largest EM value is marked in bold. 47
2.5 Operational variables measured for each of the $s = 6$ train coaches. 48

3.1 Operational variables measured for each train's coach. 61
3.2 AutoEncoder (AE) hyperparameter values. In bold, the hyperparameters, with their ranges, chosen through grid-search 10-fold cross-validation procedure 62

3.3 Fault Detection Rate (FDR) (%) for a code layer dimension hyperparameter equal to 2 in the four scenarios described in Section 3.2. The best performance is highlighted in bold 62

List of Acronyms

AE AutoEncoder

AI Artificial Intelligence

ARL Average Run Length

AUC Area Under the Curve

CL Control Limit

CNN Convolutional Neural Network

CUSUM CUmulative SUM

CV Cut-off Value

DL Deep Learning

ELU Exponential Linear Unit

EM Exact Match ratio

EWMA Exponentially Weighted Moving Average

FDR Fault Detection Rate

FMLP Functional MultiLayer Perceptron

FNN Functional Neural Network

FNNCC Functional Neural Network Control Chart

FOF Function-On-Function

FPCA Functional Principal Component Analysis

FRCC Functional Regression Control Chart

HVAC Heating, Ventilation and Air Conditioning

IC In Control

LCL Lower Control Limit

MFPC Multivariate Functional Principal Component

ML Machine Learning

MLP MultiLayer Perceptron

MSBP Multiple Stream Binomial Process

MSE Mean Square Error

MSP Multiple Stream Process

NN Neural Network

OC Out of Control

PCA Principal Component Analysis

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

SCC Shewhart Control Chart

SOF Scalar-On-Function

SPC Statistical Process Control

SPE Square Prediction Error

tanh hyperbolic tangent

UCL Upper Control Limit

VAE Variational AutoEncoder

VN Voyage Number

Introduction

The new paradigm of Industry 4.0 enables companies to increase automation, improve communication, and monitor complex processes (Xu et al. 2018). The driving force behind this transformation lies in three key enablers: data, technology, and analytics (Reis & Gins 2017). *Data* is now more abundant than ever before, and it is expected that the data collected from manufacturing processes will grow exponentially, generating so-called *big data* (He & Wang 2018). The term big data has become a mainstream buzzword (Secchi 2018, Dunson 2018) and refers to large, diverse, and complex data that require ad hoc methods to be processed and analyzed. Big data is often distinguished from other types of data by the three V's: volume, velocity, and variety (Megahed & Jones-Farmer 2015). The increasing availability of high-volume and high-velocity data has been possible through advancements in sensing technologies that now are able to collect massive amounts of data from various sources, enabling its storage in centralized databases and its accessibility whenever and wherever needed. *Technology* also provides the computational *muscle* necessary to process vast volumes of data. Indeed, the Internet of Things, high-performance computing and cloud services are providing unprecedented embedded computing capabilities as well as access to previously unimagined potential uses of data and information (He & Wang 2018). Consequently, big data *analytics* aims at transforming the huge, heterogeneous, and persistent amount of data into usable insights (Vicario & Coleman 2020), allowing industries to make informed decisions on time. In today's data-driven world, companies that cannot convert data into actionable insights will be left behind in the race for success.

The massive amount of data collected is often complex and typically demands the implementation of novel statistical methodologies (Weese et al. 2016). Many applied and industrial settings see a significant increase in approaches based on Artificial Intelligence (AI) to improve process efficiency and find new business solutions (Colosimo et al. 2021). AI refers to a broad spectrum of technologies that can perform tasks that typically require human intelligence (Russell & Norvig 2010). The use of AI is gaining growing attention in a wide variety of fields, ranging from healthcare and education to manufacturing and customer service. It is important to stress the distinctions among the terms AI, Machine Learning (ML), and Artificial Neural Networks or simply Neural Network (NN). ML lies at the heart of AI, allowing computers to learn from data without explicit programming (Thomson 1988). ML algorithms can automatically identify patterns in data, make predictions on future data, or perform other kinds of decision-making under uncertainty (Murphy 2012). NNs represent a class of ML algorithms inspired by the structure of the human brain. That is, NNs consist of layers of interconnected nodes (neurons) that process information through weighted connections, mimicking the synaptic strengths in biological systems. NNs can learn from data by adjusting these weights, enabling them to perform complex tasks such as image recognition, natural language processing, and speech recognition.

Statistical Process Control (SPC) (Montgomery 2019) must inevitably follow this path and evolve to address the challenges posed by the triplet, data, technology, and analytics. In many industrial sectors, the use of NNs for SPC is gaining increasing interest (Colosimo et al. 2021). Zorriassatine & Tannock (1998) and Psarakis (2011) provided a comprehensive overview of early research on the application of NNs with control charts, which mainly focus on identifying patterns of observations. They convey the conclusion that, under some conditions, NNs showed superior performance compared to traditional univariate and multivariate control charts, leading to faster and more accurate detection and recognition of the status of the process. More recently, Weese et al. (2016) and Colosimo et al. (2021) reviewed the most promising research directions applying ML methods to SPC. They agree that these methods warrant further investigation and should be regarded as potential competitors to improve the detection of process changes as well as the identification of the root cause for these changes.

Following this direction, this thesis aims to develop statistical methods for SPC through the integration of NN algorithms to tackle modern industrial problems in the Industry 4.0 and 5.0 era. Whereas Industry 4.0 emphasized automation and data exchange in manufacturing technologies, Industry 5.0 prioritizes sustainability by focusing on renewable resources, waste reduction and energy efficient to create environmentally friendly and socially responsible industrial processes. In this context, this research is motivated by the need to monitor train passenger thermal comfort based on the data streams collected from Heating, Ventilation and Air Conditioning (HVAC) systems installed on each train coach, and made available by the rail transport company Hitachi Rail STS (`https://www.hitachi.eu/en/`). Rail transportation in Europe has now emerged as a compelling alternative to other means of transport, giving rise to fierce competition among railway operators to guarantee passenger satisfaction. Thermal comfort for train passengers has assumed critical importance given the intense competition and crowded coaches, particularly for long journeys, and is generally ensured through the regulation of HVAC systems installed on each train coach. The relevance of this industrial scenario is also highlighted by new European regulations (EN 2006) that settle operational standards for controlling air temperature and comfort level within passenger train coaches. To guarantee the passenger thermal comfort throughout the range of the environmental and climatic conditions required by European regulations (EN 2006), the company Hitachi Rail STS is installing onboard sensor systems that automatically collect massive amounts of observational data with the ultimate goal of monitoring and possibly improving the reliability and maintenance programs of HVAC systems (Barbeito et al. 2017). Specifically, a central unit is installed to monitor the HVAC system performance by using temperature sensors to properly regulate heating and cooling modes in response to temperature changes inside (T_{in}) and outside (T_{out}) the train coach. The air temperature provided by the HVAC system is denoted by T_{supply}. The HVAC system is designed to maintain T_{in} as close as possible to the setpoint temperature (T_{set}), which is automatically and independently set as a function of T_{out} to comply with the current regulation on passengers' comfort. The difference $\Delta T = T_{\text{in}} - T_{\text{set}}$ is allowed to be no larger than 2°. It is worth noting that, even though we are in the era of big and complex data, the role of engineering/process knowledge is even more crucial for correctly handling the research problem and properly interpreting the information extracted through data analysis. Without any process knowledge, a completely automatic and data-driven approach for monitoring industrial processes may be dangerous as AI/ML algorithms require a good deal of data pre-processing (Colosimo et al. 2021). The quality and the availability of the right variables can be only ensured through human and domain expert judgments.

The methodologies proposed in this thesis are shown in the next chapters and are

developed into two main parts. In the first part, the simultaneous SPC of the sensor signals coming from each HVAC system installed on each train coach can be regarded as produced by a Multiple Stream Process (MSP) (Boyd 1950). A MSP is a process at a point in time that generates several streams of output with a quality variable of interest and specifications that are identical in all streams (Montgomery 2019). MSPs occur quite frequently in a wide variety of industrial applications. Typical examples are measurements of identical features on a single component (e.g., a machine with several heads, each one producing the same product (Amin et al. 1999)) or simultaneous measurements at different locations that can be modeled as correlated streams (e.g., diameter measurements at different radii or thickness measurements at a different cross-section (Lanning et al. 2002)). Mortell & Runger (1995) modeled the MSP through the following single-factor random-effects model (Montgomery 2017)

$$Y_{tjk} = \mu + A_t + e_{tjk}, \quad t = 1, 2, ..., T, \quad j = 1, 2, ..., s, \quad k = 1, 2, ..., n,$$

where Y_{tjk} denotes the kth measurement from the jth stream at sample time t and μ the overall process mean. The terms A_t and e_{tjk} are assumed as normally distributed random variables with zero mean and variance σ_a^2 and σ^2, respectively, and independent over time t, stream j and observation k. Furthermore, A_t is assumed as independent from e_{tjk}.
SPC aims at monitoring the stability over time of a quality characteristic and detecting when the process is Out of Control (OC), triggering a signal when assignable sources of variations (i.e., special causes) act on it. On the contrary, when normal sources of variation (i.e., common causes) are present, the process is said to be In Control (IC). Traditional SPC schemes for MSPs can detect a shift in the mean of a few (type 1) and all (type 2) streams at the same time (Runger et al. 1996). Specifically, a shift of all process streams at sample time t can be modeled as a shift in the mean of A_t, while a special cause affecting the jth stream at sample time t can be modeled as a shift in the mean of e_{tj}. This model allows us to take into account separately the two types of special causes that typically affect a MSP. These types of departure from the IC state may have different root causes and their distinction may facilitate the failure diagnosis. Despite the high suitability of MSPs in the modeling of many industrial processes, the literature on the SPC of MSPs is not extensive, as also appears from the reviews provided by Jirasettapong & Rojanarowan (2011) and Epprecht (2015). In particular, there are only very few works devoted to the identification of the OC streams in a MSP. To this aim, two NN approaches are presented in the first two chapters of this thesis with the twofold objectives of (i) improving the monitoring of a MSP and the detection of changes in individual streams, and (ii) the identification of the stream or group of streams responsible for the OC alarm. The two proposed approaches are based on the identification and classification of OC patterns in the data in a supervised fashion (Hastie et al. 2009), extending modern control chart pattern-recognition methods (Cheng & Cheng 2011, Psarakis 2011) to the SPC of MSPs. In particular, in Chapter 1, the NN is trained on a simulated reference data set containing IC and OC samples generated from a MSP to classify future observations as IC or OC (Lepore, Palumbo & Sposito 2022*a*). Whereas, in Chapter 2, a reference data set is properly designed to contain all the OC scenarios in which at least one stream shifts off-target to enable a NN to identify the streams responsible for the OC alarm. To show the practical applicability and to highlight the benefits of the proposed approaches, they are applied to a case study from the railway field in the monitoring of HVAC systems installed on a 6-coach passenger train. Specifically, the ΔT signal is chosen as the quality characteristic since it summarises any deviation of the T_{in} from the corresponding T_{set}. When a fault occurs, the HVAC system is no longer able to perform its designed

function and the temperature inside the coach will rapidly increase moving away from the setpoint temperature, and the ΔT will increase as well. In this context, the simultaneous SPC of the ΔT signals coming from each HVAC data acquisition system installed onboard passenger 6-coach modern trains can be regarded as a MSP with six streams. In this setting, the proposed NNs are successfully employed to detect any anomaly in one or more HVAC systems and automatically identify the faulty systems.

Stimulated again by the challenging railway case study, the second part of this dissertation further investigates the advantages of using more advanced and complex NNs in the SPC field. Chapter 3 shows an application of a nonparametric SPC approach based on AutoEncoder (AE)s, which are a particular type of NN used for nonlinear feature extraction and dimensionality reduction (Ambrosino et al. 2021). The proposed approach is able to simultaneously exploit the four temperature measurement information available for each of the six train coaches with the ultimate goal of possibly improving the detection of anomalies that may have occurred in the railway HVAC systems. The use of projection or feature extraction methods applied to SPC has been implemented since the 1980s in chemometrics (Wise et al. 1988, Kresta et al. 1991, Megahed et al. 2011). In these applications, a multivariate control chart, such as Hotelling's T^2 control chart (Hotelling 1947), is used to monitor the projection of the original observations into the feature or latent space typically learned by Principal Component Analysis (PCA) (MacGregor & Kourti 1995, Ferrer 2014). That is, Hotelling's T^2 control chart monitors the variation inside the feature space spanned by the feature extracted by PCA. Then, another control chart is elaborated on the Square Prediction Error (SPE), also known as Q-statistic, to monitor changes along directions orthogonal to the latent space. That is, the latter control chart signals OC conditions that make the observation move away from the feature space defined by the reference model. However, PCA -based control charts tend to perform poorly when monitoring complex nonlinear data because PCA only involves the linear transformation of the original variables. In recent years, AEs have been commonly used in a wide variety of applications, from image compression to speech signal processing, thanks to their ability to capture both linear and non-linear relationships between the variables, making them more flexible than PCA (Baldi & Hornik 1989, Goodfellow et al. 2016). AEs have also been investigated for SPC applications (Yan et al. 2016, Zhang et al. 2018, Yu & Liu 2022). Implementation of the SPC approach based on AEs is similar to those based on PCA. Specifically, two monitoring statistics, which are inspired by the definition of the classical T^2 and SPE statistics, are built in the feature and residual space learned by the AE. The Control Limit (CL)s of the two statistics are usually determined by kernel density estimation (Dehnad 1987).

In Chapter 4, raw temperature measurements are grouped to form temperature profiles for each train voyage, and, then, are re-mapped as a function of the fraction of the total distance traveled by train at each voyage. This choice is motivated by the fact that maintenance operations cannot be performed by railway engineers until the train has finished its voyage or reached the terminal station. The SPC of a process that is better modeled as functions, often referred to as *profiles* or *functional data* (Ramsay & Silverman 2005, Menafoglio & Secchi 2017, Hsing & Eubank 2015, Ferraty 2006), is known as *profile monitoring* (Saghaei et al. 2013, Woodall et al. 2004). Specifically, profile monitoring focuses on testing the stability of the functional relationship between the quality characteristic of interest, also referred to as the *response*, and one or more exploratory variables, referred to as *covariates*. That is, the quality characteristic of interest is monitored conditionally on the covariate levels. To address this issue, Centofanti et al. (2021) developed the Functional Regression Control Chart (FRCC) framework, where the residuals obtained from a function-on-function linear

regression (Morris 2015) of the functional quality characteristic on the functional covariates are monitored by using a profile monitoring approach based on the simultaneous application of the Hotelling's T^2 and SPE control charts (Woodall et al. 2004, Pini et al. 2018). Along this direction, Capezza et al. (2020) focused on the monitoring of a *scalar* quality characteristic adjusted for the effects of the functional covariates through a *linear* scalar-on-function regression model (Reiss et al. 2017). However, Capezza et al. (2020) force the relation between the scalar response and the functional covariates to be necessarily linear. To better model this relationship in a nonlinear way, many researchers have started to investigate new NN architectures to properly handle functional data. Rossi et al. (2002, 2005) proposed a reparametrization of the NN weight matrices and the integration of functional pre-processing techniques, e.g., Functional Principal Component Analysis (FPCA) (Shang 2014) and basis expansion (Wang et al. 2016), into NNs to handle the infinite dimension of the functional covariates. Recently, Thind et al. (2023) introduced the Functional Neural Network (FNN), which is able to learn a possibly nonlinear relationship between a scalar response and multiple scalar and functional covariates through the introduction of smooth weight functions that can be visualized during the training process and be easier to interpret than the traditional NN weight matrix, leading to greater model interpretability (Stevens & De Smedt 2023, Molnar 2020). Through a simulation study, FNN is demonstrated to outperform functional regression models (Ramsay & Silverman 2005) and multivariate methods (James et al. 2013) in terms of the mean squared prediction error (Montgomery et al. 2021) when the true relationship is nonlinear. In this chapter, a new profile monitoring strategy, named Functional Neural Network Control Chart (FNNCC), is elaborated on the residuals obtained from the FNN (Kulahci et al. 2023) and can be regarded as an implementation of the FRCC framework where the influence of functional or scalar covariates on a scalar response does not need to be necessarily linear as in Capezza et al. (2020, 2023*b*). Also in this case, the application is in the railway industry, where the FNNCC is used to monitor the root mean square of the difference between T_{in} and T_{set} throughout each train voyage at given values of two functional covariates, the T_{out} and the T_{set} derivative (with respect to the fraction of the total distance traveled) profiles.

The last chapter presents the `R` and `Python` software implementation of the methodologies discussed in Chapters 1, 2 and 4.

The dissertation is developed in five chapters, whose structure is briefly summarized hereinafter.

> **Chapter 1**: *Neural network based control charting for multiple stream processes with an application to HVAC systems in passenger railway vehicles*
> The present chapter shows a new control charting procedure based on NNs, whose performance is prone to be measured in terms of the IC and OC Average Run Length (ARL) (Montgomery 2019), to enhance the monitoring of a MSP and the detection of changes in individual streams. In particular, the potential of the proposed method is demonstrated by applying it to the monitoring of HVAC systems installed onboard modern trains.
>
> The results here presented are illustrated in
>
> - Lepore, A., Palumbo, B., & Sposito, G. (2022). Neural network based control charting for multiple stream processes with an application to HVAC systems in passenger railway vehicles. *Applied Stochastic Models in Business and Industry*, *38*(5), 862-883.

Chapter 2: *An artificial neural network approach for out-of-control stream identification in multiple stream processes*
In this chapter, we present a NN specifically trained for the identification of the stream or group of streams responsible for the OC alarm. This research can be regarded as the first comparative study for MSP OC stream identification and is motivated by the need for the correct and automatic identification of the faulty HVAC systems installed on each train coach, given an OC alarm, to optimize maintenance and reduce the relative costs to ensure passenger thermal comfort.

The results here presented are illustrated in

- Lepore, A., Palumbo, B. & Sposito, G. (2024+). An artificial neural network approach for out-of-control stream identification in multiple stream process control. Submitted to *Quality and Reliability Engineering International*.

Chapter 3: *Neural network for the statistical process control of HVAC systems in passenger rail vehicles*
An AE, which is a particular type of NN developed to automatically extract significant features, is trained to simultaneously monitor and detect anomalies that may have occurred in the data streams acquired from each HVAC system installed on passenger rail vehicles. In particular, two control charts based on statistics H^2 and SPE are built in the feature space and the residual space spanned by the feature extracted by the AE and the corresponding reconstruction error, respectively.

The results of this chapter are part of a collaboration with Fiorenzo Ambrosino (ENEA, Italian National Agency for New Technologies Energy and Sustainable Economic Development, Portici, Italy), Giuseppe Giannini (Head of Operation Service and Maintenance Product Evolution, Hitachi Rail STS, Naples, Italy), Antonio Lepore (University of Naples Federico II, Naples, Italy) and Biagio Palumbo (University of Naples Federico II, Naples, Italy), and are illustrated in

- Ambrosino, F., Giannini, G., Lepore, A., Palumbo, B., & Sposito, G. (2023). Neural Network for the Statistical Process Control of HVAC Systems in Passenger Rail Vehicles. *Studies in Theoretical and Applied Statistics*, 393-406.

Chapter 4: *Functional neural network control chart*
In this chapter, a new profile monitoring strategy based on a FNN (Thind et al. 2023) is presented to adjust a scalar quality characteristic for any influence by one or more covariates in the form of functional data. The proposal can be regarded as an implementation of the FRCC framework (Centofanti et al. 2021) where the influence of functional or scalar covariates on a scalar response does not need to be necessarily linear as in Capezza et al. (2020, 2023*b*), which is based on a linear scalar-on-function regression model (Reiss et al. 2017).

The results of this chapter are part of ongoing research in collaboration with Murat Kulahci (DTU Compute, Technical University of Denmark, Kgs. Lyngby, Denmark), Biagio Palumbo (University of Naples, Naples, Italy), and Antonio Lepore (University of Naples, Naples, Italy), and are illustrated in

- Kulahci, M., Lepore, A., Palumbo, B. & Sposito, G.(2023). Functional neural network control chart. *arXiv preprint arXiv:2311.11050.*

Chapter 5: *R and Python packages*

This chapter introduces the R and Python packages that provide the software for the methodologies proposed in this thesis.

In particular, Chapters 1 and 3 are based on papers that have been developed in the framework of the R&D project of the multiregional investment programme "REINForce: REsearch to INspire the Future" (CDS000609) with Hitachi Rail STS, supported by the Italian Ministry for Economic Development (MISE) through the Invitalia agency.
Works from Chapters 2 and 4 were supported by the MOST – Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU.

Chapter 1

Neural network based control charting for multiple stream processes with an application to HVAC systems in passenger railway vehicles

A MSP is a process at a point in time that generates several streams of output with a quality variable of interest and specifications that are identical in all streams. In this chapter, a new control charting framework based on NNs, whose performance is prone to be measured in terms of ARL_0 and ARL_1, is proposed to improve the monitoring of a MSP and the detection of changes in individual streams. To the best of our knowledge, this is the first time that a NN has been applied to the monitoring of a MSP. The performance of the proposed control charting is evaluated through a wide Monte Carlo simulation and is compared with the traditional Mortell and Runger's MSP control charts based on the range statistic. The proposed method's potential is demonstrated through a real-case study in the monitoring of HVAC systems installed onboard modern trains. The `NN4MSP` package that implements the proposed monitoring scheme through the software environment `Python`, and the `HVAC` data set are made openly available online at `https://github.com/unina-sfere/NN4MSP` and on PyPI, together with a tutorial that shows how to practically implement the proposed methodology to the real-case study.

1.1 Introduction

In recent years, railway transportation in Europe has been regarded as a viable alternative to other means of transport, which naturally leads to fierce competition between operators in terms of passenger satisfaction. One of the challenging aspects in this regard is the comfort of the thermal environment of passenger rail coaches, especially for long journeys. Over the past few years, new European standards (EN 2006) have been developed indeed, as a function of the different operating requirements of rail vehicles for controlling the air temperature, relative humidity, air speed, and thus, comfort level, and air quality of passenger rail coaches. Urged by these regulations and by the increased thermal comfort demand, railway companies are installing sensors to collect and store data from onboard HVAC systems, which are responsible for passenger thermal comfort, to better monitor and improve their reliability

and maintenance programs. Typically each train is composed of more than one coach and each coach is equipped with a dedicated HVAC system, whose sensor signals are streamed to maintenance engineers. The simultaneous SPC of the sensor signals coming from each HVAC system can be regarded as a MSP, firstly introduced by Boyd (1950).

A MSP is a process at a point in time that generates several streams of output with quality variables of interest and specifications that are identical in all streams. When a MSP process is IC, the sources or streams are assumed to be weakly stationary (Montgomery et al. 2015). The output from each stream may have a different average and/or standard deviation. A MSP differs from a traditional multivariate process, which instead concerns the monitoring of two or more related quality characteristics or process variables with their units and specifications. MSP occurs in a large variety of industrial applications, such as in the production of food, plastics, cosmetics, and pharmaceuticals. Typical examples are multi-head machines that produce identical units of product and thus, measurements of identical features on the same mechanical component, or a process whose quality variable is measured at several points at the same time (e.g., diameter measurements at different radii or heights). MSPs are also frequent in service activities, such as linked cashiers at a supermarket. Although MSPs are very frequent in industry, the literature on SPC schemes for such kinds of processes is not extensive and only a few monitoring methods for detecting OC states have been developed. In this chapter, the ARL is used as a performance measure to compare different control charting procedures. Essentially, the ARL is the average number of points that must be plotted before a point indicates an OC condition (Montgomery 2019). When a process is IC, we want the IC ARL, called ARL_0, to be large. When a shift has occurred, on the other hand, we want to identify the shift as quickly as possible, so the OC ARL, called ARL_1, should be small. MSPs pose new problems (Bajaria & Skog 1994) for the traditional SPC. For instance, trends and shifts on individual streams would be ignored by the traditional SPC charting procedures. Therefore, MSP monitoring procedures usually have also the objective of detecting the OC type (i) when the output of all streams shifts off-target or (ii) when this occurs only on one or few streams. This distinction is very important since it is helpful in the identification of the problem and in eliminating the assignable causes because these two types of OC state generally require different corrective actions. The basic control charting strategies that one can envisage to detect and discriminate both OC types are based on one control chart for (a) one randomly selected stream, (b) all the streams together, or (c) each stream. When the output streams are nearly perfectly correlated, strategy (a) can be appropriate based on the conjecture that when one stream is OC, the others tend to be off-target. On the opposite side, when the output streams are not highly correlated, strategy (a) is no longer effective and it is needed e.g. strategy (c). However, this may result in a very large number of control charts and is often practically unfeasible, especially when the number of streams is particularly high. Additionally, to control the familywise error rate at a given significance level α, the length of each chart control interval dramatically increases and so does the Type-II error rate. The first partial solution to this issue is the group control chart (Boyd 1950). The sampling is performed as if separate control charts were to be set up on each stream (Montgomery 2019). Specifically, at each sample time t, a subgroup size of n is sampled from every stream j. The samples are drawn from a non-overlapping time window of the same length. Then, the corresponding sample mean and range are calculated. Only the largest and smallest sample means over all streams are simultaneously plotted for each t on the same $\bar{X}$-chart against proper CLs, which are calculated in the same way as it is typically done for the conventional Shewhart Control Chart (SCC)s (Montgomery 2019). If both the largest and smallest sample means

plot outside the CLs, all the streams are judged OC of type (i); else, if only one plots outside CLs, then one can conclude that at least one stream and thus the process is OC of type (ii); otherwise, the process is judged IC. Additionally, the author suggested the use of a range control chart to monitor the largest range among the streams. In general, each plotted point is identified on the chart by the stream number, which also facilitates the identification of the stream(s) responsible for the OC condition. The group control chart is then able to detect both OC of type (i) and (ii), earlier mentioned, separately. To improve the sensitivity of the group control chart against causes that affect only one stream (OC of type (ii)), the following additional run tests, also referred to as sensitizing rules (Company 1958), which are not sensitive to OC of type (i), can be implemented based on the assumption that, if a particular stream consistently gives the highest (or lowest) value for a certain number, say r, of consecutive samples, then the stream is different from the other. Let the MSP have s streams with identical (not necessarily normal) distribution, and let r denote the number of consecutive times that a particular stream generates the largest (or smallest) observations in the sample. Then, the expected number of samples, given that the considered MSP is IC state until r consecutive largest or r consecutive smallest means come from the same stream is usually denoted by $ARL(1)_0$ and calculated as follows (Nelson 1986)

$$ARL(1)_0 = \frac{s^r - 1}{s - 1}, \tag{1.1}$$

and referred to as one-sided ARL_0. The critical value of r is selected by using Equation (1.1) to find an $ARL(1)_0$ that is larger than or equal to the wanted ARL_0. However, the discreteness of r limits the choice of $ARL(1)_0$, and for some numbers of streams s in the process, no r value corresponds to a good trade-off between the Type-I and Type-II errors, that is, the $ARL(1)_0$ so obtained may be much smaller or larger than the wanted ARL_0. Alternatively, one may want to account for *either* r consecutive largest *or* r consecutive smallest means coming from the same stream. In this case, $ARL(1)_0$ in Equation (1.1) should be replaced by its two-sided version $ARL(2)_0$ introduced by Mortell & Runger (1995). Trivially, if we have only two streams, then $ARL(1)_0$ coincides with $ARL(2)_0$. Over the years some authors have pointed out the disadvantages of the group control chart and tried to overcome them. Grimshaw et al. (1999) were the first authors to point out that the ARL_0 of the group control chart decreases when the number of streams increases, as

$$ARL_0 = \frac{1}{1 - (1 - \alpha)^s}. \tag{1.2}$$

Their formulation corresponds to the Dunn-Sidak (Dunn 1958, Šidák 1967) correction. Another possibility may be the Bonferroni correction (Johnson et al. 2002). They also suggested adjusting the CLs of the group control chart with Equation (1.2) to maintain the ARL_0 equal to 370, which is the typical value for a SCC in the 6-sigma quality approach (Montgomery 2019). Mortell & Runger (1995) pointed out that Nelson's run tests based on Equation (1.1) may fail if more than one stream shifts at the same time and with a similar magnitude. The shifted streams will most likely alternate having the maximum (minimum) value and the group control chart may not be capable of properly signaling the OC state. Mortell & Runger (1995) presented several alternative methods to solve some of the disadvantages of the group control chart. They proposed the use of the statistics $R_t = \bar{X}_{max} - \bar{X}_{min}$, that is the range of the subgroup means at the time t (among all the streams), to detect OC state of type (ii); and the use of a $\overline{\overline{X}}$ control chart, the grand

average of all the streams at the time t to detect OC state of type (i). Additionally, they compared the performance of the Shewhart, CUmulative SUM (CUSUM), and Exponentially Weighted Moving Average (EWMA) schemes using R_t as the control variable with that of the group control chart with run tests and showed that in most cases the CUSUM control chart performs better than the alternatives, depending on the number of streams and the size of the shift. In particular, the EWMA control chart never exceeds the performance of the CUSUM control chart. Other control charting methods for MSP that are worth mentioning are those developed by Meneces et al. (2008) who proposed the strategy (c), which is useful when the mean of each stream is significantly different from the others and the between-stream correlation among the streams is low. They recommended the use of a single control chart for each stream to identify streams possibly affected by special causes. However, the Meneces et al. (2008) strategy requires widening the control interval of each SCC, as a function of the correlation between streams, to correct and keep the overall ARL_0 within the desired values, thus, reducing the sensitivity to detect OC of type (ii). Jirasettapong & Rojanarowan (2011) presented a guideline to select control charts for a MSP by taking into account the stream characteristics, the correlation among streams, the number of streams, the differences among stream average and the magnitude of stream mean shift to be detected with a probability larger than a fixed threshold, usually denoted by $1 - \beta$ (Montgomery 2019). An extensive review of the techniques for the statistical control of industrial MSPs is provided by Epprecht (2015).

In the face of this classical SPC literature, due to today's growth in computing performance, many researchers have investigated the integration of NNs into classical control charting procedures, which are proved to be capable of automatically providing additional useful information and interpretability, such as the quantification of the shift magnitude and identification of the variable(s) that are responsible for the OC signals. See Zorriassatine & Tannock (1998) and Psarakis (2011) for a review of the application of NNs to SPC. Most of the applications of NN to SPC have focused primarily on univariate control charts. For example, Pugh (1989), Allen Pugh (1991) made a comparison between the performance of SCCs with that of NN for detecting univariate process mean shift. He demonstrated that NNs can be designed to outperform these charts in terms of Type-II error. In recent years, a large effort has been made in the multivariate quality statistical control framework. Traditional multivariate control charts, such as *Hotelling's* T^2 control chart (Hotelling 1947), under some conditions, do not provide sufficient capability for monitoring more than one quality characteristic simultaneously, especially in the presence of large correlation among multiple variables (Psarakis 2011). For this reason, solutions based on NNs were proposed to solve the problem of detecting shifts in multivariate processes and identifying the variable or group of variables that produced the shift. The results showed that NNs applied to large and complex data sets usually outperform multivariate control charts and provide better performance. For example, Niaki & Abbasi (2008) designed a NN to detect and diagnose shifts in multi-attribute processes. Aparisi et al. (2006) used a NN based method to interpret the OC signals of a multivariate T^2 control chart, that in most of the cases outperforms the Tracy, Mason and Young *Hotelling's* T^2 decomposition (Mason et al. 1995, 1997) in the identification of aberrant variables. To properly use NN in SPC, the probabilities of Type-I and Type-II errors must be assessed to determine the NN performance. Similar to the conventional SPC measures, ARL can be a useful performance measure. Given a prespecified ARL_0, the NN needs to be tuned to a corresponding level of discriminating power (Hwarng 2014). The way this can be achieved will be discussed in detail in the following sections.

Motivated by the industrial context and following these very promising applications of the

NN to SPC, the present chapter explores the possibility of exploiting the classification and pattern recognition properties of NNs by proposing a new NN approach for the monitoring of a MSP. In particular, the proposed framework pursues the objective of detecting changes in individual stream means. Alongside, the job of detecting the overall process mean and variance shifts is classically tackled by $\overline{\overline{X}}$ and s control charts (Mortell & Runger 1995, Liu et al. 2008). A wide Monte Carlo simulation is performed to assess the performance of the proposed method to the MSP SCC and CUSUM control charts based on the range statistic (Mortell & Runger 1995), hereinafter referred to R_t and CUSUM-R_t control charts, respectively.

The remainder of the chapter is as follows. In Section 1.2, the MSP model and NNs are introduced. Then, a possible use of NN for the monitoring of a MSP is presented. Section 1.3 contains the simulation study and the comparison with R_t and CUSUM-R_t control charts for the monitoring of MSPs. In Section 1.4 the proposed framework is applied to a real-case study concerning the monitoring of the HVAC systems installed onboard passenger railway vehicles and has motivated this research. Real data were acquired and made available by the rail transport company *Hitachi Rail STS* based in Italy. In Section 1.5, we conclude and envisage recommendations for possible future paths in this direction. With this chapter, we want to overcome some criticisms (Weese et al. 2016) about the application of NNs to real industrial processes and the definition of a method to properly define the baseline sample and choose among the different NN architectures. To allow the reader to possibly investigate other approaches with this data set and to encourage the fruitful spread of the application of NNs to the SPC among practitioners in the industry, the `NN4MSP` package that implements the proposed monitoring scheme through the software environment `Python` (Van Rossum & Drake Jr 1995), and the `HVAC` data set are made openly available online (Lepore et al. 2021) and on PyPI (Lepore, Palumbo & Sposito 2022*b*), together with a tutorial that shows how to practically implement the proposed methodology to the real-case study.

1.2 Methodology

In this section, a NN based control charting scheme is presented with the ultimate goal of detecting a shift in the mean of an individual stream of a MSP. Moreover, the traditional MSP model introduced by Mortell & Runger (1995) is discussed.

Multiple stream process model

Mortell & Runger (1995) proposed the following model for MSPs that is as a single-factor random-effects model (Montgomery 2017)

$$Y_{tjk} = \mu + A_t + e_{tjk}, \quad t = 1, 2, ..., T, \quad j = 1, 2, ..., s, \quad k = 1, 2, ..., n, \tag{1.3}$$

where Y_{tjk} denotes the kth measurement from the jth stream at sample time t, μ the overall process mean and A_t the difference between the process mean across the s streams and n measurements, which will be hereinafter denoted by Y_t, and μ. The error term e_{tjk} represents the deviation for the kth measurement of the jth stream at sample time t from the mean Y_t, that is the difference between Y_{tjk} and Y_t. For the sake of simplicity, when we omit one or more subscripts from a variable symbol, we implicitly mean that variable as averaged over the omitted subscript. For example, Y_{tj} denotes the subgroup mean from stream j at sample time t across the n measurements. The terms A_t and e_{tjk} are assumed as normally distributed

random variables with zero mean and variance σ_a^2 and σ^2, respectively. Furthermore, A_t is assumed as independent from e_{tjk}. Explicitly note that, according to Equation (1.3), Y_{tjk} is modeled as the sum of the overall mean μ, the common component A_t, which interprets variations that may affect all streams in the same way, and a stream-wise error e_{tjk}. A shift of all process streams at sample time t can be modeled as a shift in the mean of A_t, while a special cause affecting the jth stream at sample time t can be modeled as a shift in the mean of e_{tj}. This model allows us to take into account separately the two types of special causes that typically affect a MSP. The model in Equation (1.3) also takes into account the correlation between streams at sample time t, through the presence of the common term A_t, which however may not be independent over time. Having explicitly stated the assumptions made by Mortell and Runger, the expected value $\mathrm{E}(\cdot)$ and variance $\mathrm{Var}(\cdot)$ of Y_{tjk} are equal to μ and $\sigma_a^2 + \sigma^2$, respectively. Then, by writing the subgroup mean Y_{tj} from the jth stream at sample time t as

$$Y_{tj} = \mu + A_t + e_{tj}, \tag{1.4}$$

it is easy to calculate $\mathrm{Var}(Y_{tj}) = \sigma_a^2 + \frac{\sigma^2}{n}$. From Equation (1.4), it follows that Y_{tj} is normally distributed because it is a sum of the two normally and independently distributed random variables A_t and e_{tj}. Moreover, it can be proved (see Section 1.6) that the covariance $\mathrm{Cov}(\cdot,\cdot)$ of any two subgroup means at a given stream or sample time respectively, result as follows

$$\begin{aligned} \mathrm{Cov}(Y_{tj}, Y_{t'j}) &= 0 \quad t \neq t', \quad t = 1,2,...,T, \quad t' = 1,2,...,T, \\ \mathrm{Cov}(Y_{tj}, Y_{tj'}) &= \sigma_a^2 \quad j \neq j', \quad j = 1,2,...,s, \quad j' = 1,2,...,s. \end{aligned} \tag{1.5}$$

Hence, the correlation coefficient between Y_{tj} and $Y_{tj'}$ corresponding to two different streams j and j' at a given sample time t is

$$\rho_{jj'} = \frac{\sigma_a^2}{\sigma_a^2 + \frac{\sigma^2}{n}}, \quad j \neq j', \quad j = 1,2,...,s, \quad j' = 1,2,...,s. \tag{1.6}$$

To eliminate the component of process variability common to all streams, Mortell & Runger (1995) subtracts Y_t from Y_{tj} and define the following residuals (Ott & Snee 1973)

$$X_{tj} = Y_{tj} - Y_t, \quad t = 1,2,...,T, \quad j = 1,2,...,s, \tag{1.7}$$

that, given the model in Equation (1.3), does not contain the term A_t and thus, its variance does not depend on σ_a^2. A control charting procedure based on X_{tj} in place of Y_{tj} has then the advantage of being sensitive to assignable causes that shift one or a few streams, and effectively monitors the individual components. However, while e_{tj}'s are independent and identically distributed, it can be readily shown (see Section 1.6) that the correlation between the residuals X_{tj} and $X_{tj'}$ of the jth and j'th streams, at a given sample time t, is

$$\rho_{jj'} = -\frac{1}{s-1}, \quad j \neq j', \quad j = 1,2,...,s, \quad j' = 1,2,...,s. \tag{1.8}$$

Whereas, the variance of X_{tj}, denoted by σ_x^2, is (see Section 1.6)

$$\sigma_x^2 = \mathrm{Var}(X_{tj}) = \frac{s-1}{s}\frac{\sigma^2}{n}. \tag{1.9}$$

From Equation (1.8), it is clear that when the number of streams is $s = 2$, the residuals of the two streams are perfectly negatively correlated. However, for $s \geqslant 4$ (Epprecht et al.

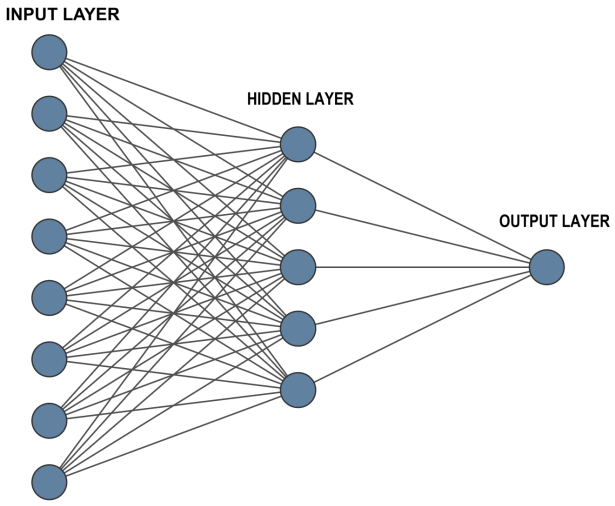


Figure 1.1: NN architecture

2011) correlation can be neglected and residuals can be roughly assumed as independent from each other. Note that in the MSP setting, variability over time (σ_a^2) is typically much larger than the variability within stream (σ^2), as in the real-case study presented in Section 1.4. In this case, traditional control charts (Montgomery 2019) based on the variable Y_{tjk} need a longer time to detect a mean shift in one or few streams only, e.g, a shift in the mean of e_{tjk}, because CL width is inflated by the presence of a large variability component due to σ_a^2.

Artificial neural networks

NNs allow one to model nonlinear processes and have become a very popular tool to solve learning and visualization problems such as classification, clustering, regression, function approximation, and computer vision. A NN is a collection of interconnected processing elements, called neurons or nodes, which are organized in layers and loosely mimic the biological NN. Layers are divided into three layers, viz., input layer, output layer, and hidden layers. A simple NN architecture is shown in Figure 1.1. The input layer is the first layer of the NN and aims to merely acquire data from the network and transfer them to the hidden layers. The hidden layers perform computations and transfer information from the input layer to the last layer, which is the output layer, and produce the final result. The nodes in the input layer are called input neurons and are as many as the size of the input vector. Their definition has a great impact on the learning process and generalization of results. The neurons referred to in the output layer are called output neurons and are as many as the number of output quantities of interest. For example, a simple regression problem needs a single output neuron, whereas a multi-class classification needs one neuron for each class. In general, we can design a NN through the definition of (I) the input and the output layers, (II) the activation functions, (III) the network architecture (e.g., hidden layers) and (IV) the cost function. All these choices significantly influence its performance. In this chapter, we consider a feedforward (Goodfellow et al. 2016) NN, in which each neuron in one layer is connected to every neuron in the next layer, and the information flows in only one direction,

from the input nodes to the output nodes, without any cycle through the hidden nodes. Each connection has a weight that represents the strength of the connection between neurons and it is used to calculate the output of that neuron to be given as input to a neuron of the next layer, which is connected to. In particular, consistently with the notation used by Nielsen (2015), w^l_{jk} denotes the weight for the connection from the kth neuron in the $(l-1)$th layer to the jth neuron in the lth layer. Neurons in each layer perform a weighted sum of their inputs. Then, this sum is passed to an activation function to produce the output. The output of the jth neuron in the lth layer is denoted by a^l_j and then calculated as

$$a^l_j = g\left(\sum_k w^l_{jk} a^{l-1}_k\right), \tag{1.10}$$

where k indexes all the neurons in the $(l-1)$th layer and $g(\cdot)$ denotes the activation function. The most common activation functions are the sigmoid function $g(y) = \frac{1}{1+e^{-y}}$ with output value ranging from zero to one; the hyperbolic tangent (tanh) $g(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}$ with output values in the interval $[-1, 1]$; the Rectified Linear Unit (ReLU), that returns the input itself if it is positive, and zero otherwise. By defining a weight matrix $\mathbf{W}^l$ for each layer l, whose jth row and kth column is w^l_{jk}, and organising the a^l_j in a column vector $\mathbf{a}^l$, Equation (1.10) can be compactly rewritten in matrix form as follows

$$\mathbf{a}^l = g(\mathbf{W}^l \mathbf{a}^{l-1}). \tag{1.11}$$

The weight $\mathbf{W}^l$ is a $N_l \times N_{l-1}$ matrix, being N_{l-1} and N_l the number of neurons in the $(l-1)$th and lth layer, respectively. Equation (1.10) or (1.11) highlight how each neuron in one layer is related to the others in the previous layer. Backpropagation (Goodfellow et al. 2016) is the most used algorithm for training a feedforward NN. The algorithm works iteratively through a certain number of epochs, in which the NN scans all the training samples, calculates a cost function C to be minimized and constructs an error surface by comparing the actual predicted output with the corresponding real class or value. The usual functions C are based on the Mean Square Error (MSE) (e.g., for regression problems) or on the cross-entropy (e.g., for classification tasks) (Goodfellow et al. 2016). In this chapter, we chose to use the cross-entropy cost function written in the following form

$$C = -\frac{1}{M}\sum_{i=1}^{M}\Big[\mathbf{y}(\mathbf{x}_i)\log(\mathbf{a}^L(\mathbf{x}_i)) + (1-\mathbf{y}(\mathbf{x}_i))\log(1-\mathbf{a}^L(\mathbf{x}_i))\Big] \tag{1.12}$$

where $\mathbf{x}_i$ is the ith input vector (i.e., one of the ith training sample), $\mathbf{y}(\mathbf{x}_i)$ is the output/label corresponding to $\mathbf{x}_i$, and M is the total number of training samples. The sum is over all the M training samples $\mathbf{x}_i$ and $\mathbf{a}^L(\mathbf{x}_i)$ returns the output corresponding to the input $\mathbf{x}_i$ with the superscript L referring to the last layer. The crux of the matter in the backpropagation algorithm is to efficiently evaluate the gradient of the cost function $\nabla \mathbf{C}_w$ with respect to the weights w^l_{jk} for each layer $l = 1, 2, \ldots, L$. This efficient computation makes it feasible to use gradient methods(Goodfellow et al. 2016) for training NNs and updating weights to find a (local) minimum of the function C, by exploring the gradient vector in the direction of the error surface. The gradient can be then used to adjust the NN weights w^l_{jk} to minimize the cost function, by adding the following quantity

$$\Delta w^l_{jk} = -\eta \frac{\partial C}{\partial w^l_{jk}}, \tag{1.13}$$

where $\eta > 0$ is usually referred to as learning rate. The minus sign in Equation (1.13) guarantees that w_{jk}^l is updated in a way that always decreases C, because the partial derivatives $\partial C/\partial w_{jk}^l$ are the components of $\nabla \mathbf{C}_w$, which represents the direction of steepest descent at each point. Interested readers can find more comprehensive discussions on these aspects in the textbooks of Bishop (1995) and Goodfellow et al. (2016).

Another general crucial task is the proper definition of the training, validation and test sets (James et al. 2013). The training set is intended as the set of data used by the NN to estimate the weights or model parameters w_{jk}^l, usually in a supervised fashion in the so-called training phase. The number of neurons, hidden layers, the type of the activation function and the learning rate, which are usually called hyperparameters, in contrast to the model parameters, cannot be directly estimated from the data, but are tentatively set before the training phase of the model and optimized by trial and error procedure based on the model performance on a separate set, which is called validation, development or dev set (James et al. 2013) because it leads the development of the model. Lastly, the test set is a set of samples, which are not included in the training nor in the validation set, used to compare the prediction error of the fully specified NN with any competing models.

The proposed neural network based control charting procedure

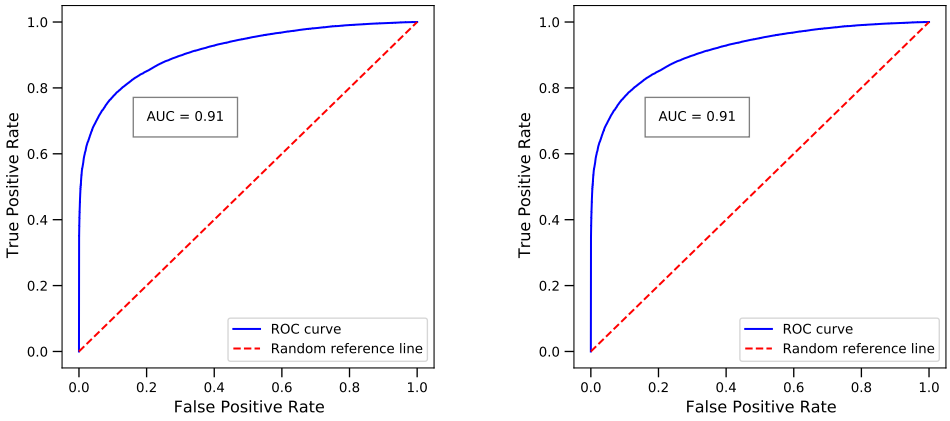
A MultiLayer Perceptron (MLP) with error back-propagation (Goodfellow et al. 2016) is one of the most commonly used NN to solve non-linear problems and is known to generally achieve better performance than traditional univariate and multivariate control charts (Zorriassatine & Tannock 1998). It is a feedforward NN under the umbrella of supervised learning techniques (James et al. 2013) and consists of three layers, viz., input, hidden and output layer. In this chapter, we aim to design a MLP to be used for the SPC of a MSP, with the ultimate goal of detecting whether the current input vector is likely to be drawn from an IC process or not, which can be regarded as a binary classification problem. The MLP is trained such that, when the process is IC, the target value of the output neuron is coded as zero, otherwise (i.e., the process is OC) it is coded as one. From a classification point of view, samples from IC process are called negative samples and labeled as 0, otherwise, they are called positive samples and labeled with 1. The training set should be representative of the shifts of interest, collected as input data and fed in random order to the NN. The magnitude of the shift in the process mean is measured in units of standard deviation. Positive samples are generated from all possible scenarios in which $l = 1, ..., s-1$ streams shift off-target at the same time by as much as $\Delta\mu = \sigma_x, 2\sigma_x, 3\sigma_x$, with σ_x the standard deviation of the residuals X_{tj} defined in Equation (1.9). This means that, by excluding the case in which streams are all simultaneously IC or OC, the number of possible OC scenarios is $3\sum_{i=1}^{s-1}\binom{s}{i}$. Through a Monte Carlo simulation 300 (positive) samples are drawn from each OC scenario, together with an equal number of negative samples. Generally, 2/3 of the latter forms the training set and the rest the validation set. After generating the training samples and their corresponding target class, the MLP is trained via a backpropagation algorithm. The input vectors and the desired classes are needed to allow the model to approximate the function that maps the input to the target value. The input layer has $s + 2$ neurons that represent the $s + 2$ monitored statistics, viz., the s sample mean X_{tj} at sample time t for each stream j, the average X_t over all the streams and the range R_t of the subgroup means of the residuals X_{tj} at sample time t. These inputs have been selected to take advantage of different information from the sample. A NN with one hidden layer can approximate any continuous function from the input patterns to the output patterns, to an arbitrary degree of accuracy (Goodfellow

et al. 2016), provided that there are enough neurons in the hidden layer. Anyway, the selection of a proper number of neurons in the hidden layer is a delicate task that is included in the hyperparameter optimization or tuning phase. Since the output of a sigmoid activation function is always between 0 and 1, the NN predicted output can be interpreted as the probability of a sample being drawn from an OC process, i.e, the probability of a sample $\mathbf{x}_i$ belonging to class 1 is equal to $g(\mathbf{a}^L(\mathbf{x}_i)) = 1/(1 + exp(-g(\mathbf{a}^L(\mathbf{x}_i))))$. Then we can judge the process to be OC if the probability exceeds a given threshold, which is the Cut-off Value (CV) of the neuron in the output layer. Low CVs values increase the probability of Type-I error, i.e. the process is judged as OC when it is IC, but reduce the probability of Type-II error, i.e. the process is judged as IC when it is OC. On the other hand, high CVs increase the probability of Type-II error and reduce the probability of Type-I error. To compare different NN configurations (i.e. NN hyperparameter choices), we use the Area Under the Curve (AUC) (Fawcett 2006), as a performance measure. A Receiver Operating Characteristic (ROC) curve is a graphical plot showing the performance of a binary classifier as a function of the threshold. It is a plane where the y and the x axes are the True Positive Rate (TPR $= 1 - \beta$) and the False Positive Rate (FPR $= \alpha$), respectively, i.e, the probability of correct OC state detection and the probability of false alarm. For each threshold, we can compute a couple of values (FPR, TPR) on the validation set and thus, we can draw the ROC curve. In this way, we can evaluate the overall performance of the classifier under different decision thresholds. We choose the proper number of nodes, the type of the activation function in the hidden layer, the learning rate η and the number of epochs that achieve the highest AUC value on the validation set. AUC measures the entire two-dimensional area underneath the entire ROC curve and its value is always between 0 and 1. The greater is the area, the better will be the average performance of the classifier. AUC is also proved to be equal to the probability that a classifier will predict a higher probability for a randomly chosen positive sample than a randomly chosen negative one (Fawcett 2006).

1.3 Simulation study

Simulation analysis is used to evaluate and compare the IC and OC performance of the proposed scheme with that of the traditional competitors (Mortell & Runger 1995), for detecting shifts in one (or a few) stream(s) of a MSP. We assume that the IC MSP follows the model in Equation (1.3), the number of streams is 6 (inspired by the real-case study), 8 and 10 and the subgroup size $n = 1, 5$. As stated in Section 1.2, by considering the residuals X_{tj}, we theoretically eliminate the term A_t, and thus, the variability of the process common to all streams. From this point on, the common effect A_t is then assumed to be equal to zero. Therefore, without loss of generality, we can generate independent observations from X_{tjk}. Each subgroup is thus independent of the other, normally distributed and standardized to have zero mean and unit variance. In a real-case scenario, the standard deviation of the residuals X_{tjk} should be estimated from Phase I data according to some estimator (Mahmoud et al. 2010). In this way, the proposed framework can be applied to any MSP, since the standardized sample means measure the deviation from the target mean vector in standard deviation units. This is convenient in practice, as the user shall only standardize sample means before giving them as input to the NN.

In order to design, and train the NN, as well as simulate a proper data set, we draw 300 samples for each OC scenario. The number of simulated IC conditions is equal to the total number of data simulated for all OC scenarios. Note that, even if in this chapter we do not explicitly train the considered NN to signal which stream is OC, it is still crucial



(a) (b)

Figure 1.2: ROC curve for a process with $s = 6$, $n = 1$ (a) and $s = 6$ and $n = 5$ (b).

that the training set is balanced with positive samples from all possible OC scenarios to allow the MLP to classify with the same probability as OC a MSP with l out of s streams shifted off-target, regardless of their order. Different NN architectures have been explored in terms of the number of hidden neurons, activation functions, learning rate η and number of epochs, as there is no exact procedure to optimize the performance of such NN. To do that, we consider a validation set obtained by retaining the 30% of generated samples. Then, a MLP with one hidden layer with five neurons, a ReLU and a sigmoid activation function for the hidden and output layer, respectively, is shown to have the best performance in terms of AUC, for all considered simulated processes with $s = 6, 8, 10$ and both subgroup sizes $n = 1, 5$. As an example, the ROC curve and the corresponding AUC value are shown in Figure 1.2 for the process with $s = 6$ and $n = 1, 5$. As we can see, a ROC curve always starts at the origin of the axes, where the CV is set to 1, and all samples are classified as IC. Then, the TPR and the FPR increase with CV. CV equal to zero is represented by the point at (1,1) on the ROC plane, that is all the samples are classified as OC. The dashed lines in Figure 1.2 represent a random classifier, which, by definition, correctly predicts both classes half the time, while the perfect classifier is represented by the point $(0, 1)$. A ROC curve close to the dashed line represents a classifier with a performance as poor as a random guess. The further the shape of the ROC curve from the diagonal, the better the performance of the classifier. Thus, from an intuitive point of view, AUC describes how much the curve is stretched towards this point from the diagonal.

After training the NN, for detection purposes, the CV value must be set, and this can be done here, thanks to the connection made by the SPC. As the main aim of this chapter is to demonstrate the possible advantages deriving from the use of NN for the SPC of MSPs, it is crucial to allow a fair comparison with the classical SPC control charting procedures. In this regard, the idea is to properly fix the CV of the neuron in the output layer, which may be regarded as the key threshold to set the Type-I and Type-II errors. To determine this

Table 1.1: CVs of the proposed NNs corresponding to typical false alarm rate values $\alpha = 0.0027, 0.01, 0.02, 0.05$ for a process with a number of streams $s = 6, 8, 10$ and subgroup sample size $n = 1$ and $n = 5$.

	$s = 6$		$s = 8$		$s = 10$	
α	$n = 1$	$n = 5$	$n = 1$	$n = 5$	$n = 1$	$n = 5$
0.0027	0.957	0.940	0.961	0.924	0.963	0.892
0.01	0.906	0.814	0.895	0.729	0.888	0.653
0.02	0.846	0.664	0.817	0.537	0.805	0.450
0.05	0.768	0.390	0.644	0.267	0.629	0.188

Table 1.2: ARL_1 of the R_t, CUSUM-R_t control charts (Mortell & Runger 1995) and the proposed NN control charting scheme based on 100000 simulations of a MSP with $s = 6$ streams, at different number $l = 1, 2, .., s-1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 1$. For each OC scenario, the lowest ARL_1 value is marked in bold.

	Range control chart $\Delta\mu$					NN based control charting procedure $\Delta\mu$					CUSUM-R_t control chart $\Delta\mu$				
	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$
l															
1	138.31	53.53	23.42	10.82	5.56	**95.51**	45.51	**19.50**	**9.56**	**5.17**	171.21	**40.58**	**19.50**	12.37	9.01
2	95.69	38.96	15.22	7.06	3.83	**33.63**	**11.34**	**4.67**	**2.41**	**1.55**	75.12	24.29	13.68	9.49	7.38
3	85.47	32.74	13.80	6.53	3.51	**14.59**	**4.41**	**1.99**	**1.28**	**1.06**	62.96	21.88	12.69	9.01	7.09
4	99.80	37.37	15.53	7.21	3.18	**8.72**	**2.64**	**1.37**	**1.06**	**1.00**	75.19	24.29	13.72	9.53	7.38
5	133.05	54.81	23.01	10.64	5.52	**5.16**	**1.77**	**1.12**	**1.01**	**1.00**	170.13	40.27	19.53	12.42	9.06

threshold, we generate random samples from an IC process and evaluate the ARL_0 as $\frac{1}{\alpha}$, where α is the proportion of misclassified samples (i.e., wrongly classified as drawn from an OC process). Through 100000 Monte Carlo simulations from an IC process, CV is obtained as the value such that the misclassification error α (on this generated data set) achieves an ARL_0 as close as possible to 370.

Regarding the tuning of the NN and the selection of hyperparameters, the ROC curve was chosen among the available cross-validation methods (James et al. 2013) to help practitioners visually analyze the performance of the proposed approach and balance the trade-off between Type-I and Type-II error rates. The AUC has been preferred to measure the performance of the NN, because (differently from e.g., the accuracy (James et al. 2013)) it is a threshold-independent metric. This property, independently from the threshold values to be set to achieve the desired ARL_0, allows AUC to measure the average performance of a classifier over all possible CVs (Fawcett 2006) and to optimize NN hyperparameters accordingly. Thus, once these hyperparameters have been optimized, practitioners are allowed to modify the ARL_0 through the corresponding CV alone, without the need to repeat the hyperparameter search and the training of the NN. Table 1.1 shows the CVs corresponding to the typical false alarm rate values α used in the SPC application for all the considered processes. As expected, Table 1.1 shows that the Type-I error rate increases as CV decreases, given the same number of streams s and subgroup size n.

Table 1.3: ARL_1 of the R_t, CUSUM-R_t control charts (Mortell & Runger 1995) and the proposed NN control charting scheme based on 100000 simulations of a MSP with $s = 6$ streams, at different number $l = 1, 2, .., s-1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 5$. For each OC scenario, the lowest ARL_1 value is marked in bold.

	Range control chart $\Delta\mu$					NN based control charting procedure $\Delta\mu$					CUSUM-R_t control chart $\Delta\mu$				
	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$
l															
1	15.07	3.63	1.60	1.12	1.01	11.22	3.03	**1.46**	**1.09**	**1.01**	**9.85**	**2.83**	1.61	1.18	1.03
2	9.98	2.59	1.29	1.03	1.00	**3.07**	**1.23**	**1.01**	**1.00**	**1.01**	5.78	2.16	1.37	1.07	1.00
3	9.05	2.39	1.23	1.02	1.00	**1.59**	**1.02**	**1.00**	**1.00**	**1.00**	5.16	2.05	1.33	1.05	1.00
4	10.08	2.57	1.29	1.03	1.00	**1.16**	**1.00**	**1.00**	**1.00**	**1.00**	5.76	2.18	1.37	1.07	1.01
5	15.04	3.65	1.59	1.11	1.01	**1.05**	**1.00**	**1.00**	**1.00**	**1.00**	9.6	2.86	1.62	1.19	1.03

Table 1.4: ARL_1 of the R_t, CUSUM-R_t control charts (Mortell & Runger 1995) and the proposed NN control charting scheme based on 100000 simulations of a MSP with $s = 8$ streams, at different number $l = 1, 2, .., s-1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 1$. For each OC scenario, the lowest ARL_1 value is marked in bold.

	Range control chart $\Delta\mu$					NN based control charting procedure $\Delta\mu$					CUSUM-R_t control chart $\Delta\mu$				
	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$
l															
1	157.98	60.39	25.72	**11.83**	**5.91**	**118.62**	53.76	25.18	12.57	6.75	254.02	**52.71**	**24.04**	14.46	10.02
2	100.20	41.24	15.72	7.25	3.18	**42.07**	**14.74**	**6.04**	**3.04**	**1.86**	93.25	28.82	15.62	10.36	7.69
3	88.42	32.08	13.22	6.05	3.23	**19.82**	**6.01**	**2.57**	**1.51**	**1.15**	66.65	23.65	13.48	9.27	7.07
4	88.11	32.05	12.54	5.77	3.07	**10.77**	**3.20**	**1.56**	**1.13**	**1.02**	60.95	22.38	12.94	9.00	6.91
5	89.05	32.71	13.16	5.98	3.21	**6.42**	**2.0**	**1.20**	**1.02**	**1.01**	67.12	23.61	13.49	9.28	7.08
6	107.18	38.80	16.15	7.24	3.80	**4.26**	**1.53**	**1.07**	**1.01**	**1.00**	94.57	29.01	15.65	10.37	7.70
7	157.48	62.11	26.01	11.88	5.89	**3.06**	**1.28**	**1.02**	**1.00**	**1.00**	252.66	52.91	24.19	14.53	10.07

Table 1.5: ARL_1 of the R_t, CUSUM-R_t control charts (Mortell & Runger 1995) and the proposed NN control charting scheme based on 100000 simulations of a MSP with $s = 8$ streams, at different number $l = 1, 2, .., s-1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 5$. For each OC scenario, the lowest ARL_1 value is marked in bold.

	Range control chart $\Delta\mu$					NN based control charting procedure $\Delta\mu$					CUSUM-R_t control chart $\Delta\mu$				
	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$
l															
1	18.22	4.02	**1.66**	**1.12**	**1.01**	16.75	4.03	1.69	1.14	1.02	**10.54**	**2.98**	1.67	1.20	1.03
2	11.32	2.70	1.29	1.03	1.00	**4.13**	**1.40**	**1.03**	**1.00**	**1.00**	5.88	2.20	1.39	1.07	1.00
3	9.34	2.32	1.19	1.01	1.00	**1.93**	**1.05**	**1.00**	**1.00**	**1.00**	4.82	2.01	1.31	1.04	1.00
4	8.91	2.21	1.18	1.01	1.00	**1.31**	**1.00**	**1.00**	**1.00**	**1.00**	4.58	1.96	1.29	1.04	1.00
5	9.20	2.31	1.20	1.01	1.00	**1.09**	**1.00**	**1.00**	**1.00**	**1.00**	4.85	2.02	1.31	1.04	1.00
6	11.38	2.71	1.29	1.03	1.00	**1.03**	**1.00**	**1.00**	**1.00**	**1.00**	5.93	2.23	1.40	1.08	1.01
7	18.27	4.03	1.66	1.12	**1.01**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	10.62	3.01	1.68	1.21	1.04

Table 1.6: ARL_1 of the R_t, CUSUM-R_t control charts (Mortell & Runger 1995) and the proposed NN control charting scheme based on 100000 simulations of a MSP with $s = 10$ streams, at different number $l = 1, 2, .., s-1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 1$. For each OC scenario, the lowest ARL_1 value is marked in bold.

	Range control chart $\Delta\mu$					NN based control charting procedure $\Delta\mu$					CUSUM-R_t control chart $\Delta\mu$				
	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$
l															
1	167.22	67.39	28.19	**12.74**	**6.26**	**128.86**	63.57	31.03	15.17	8.14	293.40	**60.00**	**26.86**	15.89	10.86
2	113.77	42.37	16.87	7.55	3.91	**55.16**	**19.82**	**8.13**	**3.96**	**2.30**	102.56	31.71	16.89	11.07	8.16
3	96.15	33.40	13.35	6.02	3.17	**25.39**	**7.95**	**3.28**	**1.80**	**1.27**	68.58	24.83	14.12	9.68	7.35
4	87.26	31.40	12.05	5.42	2.88	**13.82**	**4.07**	**1.85**	**1.23**	**1.05**	57.76	22.25	13.03	9.12	7.02
5	82.44	29.42	11.52	5.20	2.80	**8.28**	**2.52**	**1.34**	**1.05**	**1.01**	55.08	21.53	12.73	8.96	6.92
6	85.25	30.30	11.74	5.39	2.88	**5.35**	**1.78**	**1.13**	**1.01**	**1.00**	58.00	22.26	13.00	9.10	7.00
7	99.40	34.82	13.64	6.00	3.17	**3.82**	**1.42**	**1.04**	**1.00**	**1.00**	68.72	24.90	14.12	9.66	7.34
8	119.19	42.27	16.64	7.58	3.87	**2.78**	**1.21**	**1.01**	**1.00**	**1.00**	102.59	31.82	16.88	11.05	8.14
9	165.84	68.87	28.31	12.36	6.18	**2.20**	**1.11**	**1.00**	**1.00**	**1.00**	297.72	60.38	26.91	15.92	10.87

Table 1.7: ARL_1 of the R_t, CUSUM-R_t control charts (Mortell & Runger 1995) and the proposed NN control charting scheme based on 100000 simulations of a MSP with $s = 10$ streams, at different number $l = 1, 2, .., s-1$ of streams that shift off-target and mean shift size $\Delta\mu = 1.0\sigma_x, 1.5\sigma_x, 2.0\sigma_x, 2.5\sigma_x, 3.0\sigma_x$ with subgroup sample size $n = 5$. For each OC scenario, the lowest ARL_1 value is marked in bold.

	Range control chart $\Delta\mu$					NN based control charting procedure $\Delta\mu$					CUSUM-R_t control chart $\Delta\mu$				
	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$	$1.0\sigma_x$	$1.5\sigma_x$	$2.0\sigma_x$	$2.5\sigma_x$	$3.0\sigma_x$
l															
1	17.99	3.94	1.63	**1.12**	**1.01**	18.09	4.44	1.84	1.19	1.03	**11.20**	**3.06**	**1.70**	1.21	1.04
2	10.88	2.58	1.26	1.02	1.00	**4.61**	**1.50**	**1.04**	**1.00**	**1.00**	5.92	2.23	1.4	1.07	1.00
3	8.63	2.16	1.15	1.01	1.00	**2.19**	**1.08**	**1.00**	**1.00**	**1.00**	4.72	2.00	1.30	1.04	1.00
4	7.66	1.98	1.12	1.01	1.00	**1.43**	**1.00**	**1.00**	**1.00**	**1.00**	4.26	1.91	1.27	1.03	1.00
5	7.26	1.94	1.11	1.00	1.00	**1.15**	**1.00**	**1.00**	**1.00**	**1.00**	4.16	1.88	1.25	1.02	1.00
6	7.64	2.00	1.12	1.01	1.00	**1.04**	**1.00**	**1.00**	**1.00**	**1.00**	4.26	1.91	1.27	1.03	1.00
7	8.67	2.15	1.16	1.01	1.00	**1.01**	**1.00**	**1.00**	**1.00**	**1.00**	4.76	2.01	1.31	1.04	1.00
8	10.76	2.58	1.26	1.02	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	6.05	2.26	1.41	1.08	1.01
9	18.05	3.95	1.63	1.11	1.01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	11.31	3.09	1.71	1.22	1.04

At a given α, we can then properly evaluate the proposed network performance in terms of $ARL_1 = 1/(1-\beta)$. By simulating 100000 data sets from each of the OC scenario in which different number $l = 1, .., s-1$ of streams shift off-target ($\mu = 0$) at different mean shift size $\Delta\mu = \sigma_x, 1.5\sigma_x, 2\sigma_x, 2.5\sigma_x, 3\sigma_x$, number of streams $s = 6, 8, 10$ and subgroup size $n = 1, 5$. In particular, β can be evaluated as the proportion of negative samples that the NN incorrectly classifies as drawn from an IC process. Based on ARL_1, the proposed control charting procedure, the R_t and CUSUM-R_t control charts can be thus compared when at least one stream shifts off-target. Unfortunately, Mortell and Runger provide ARL_1 values at different ARL_0 and the number of streams from those considered here. The CUSUM-R_t is calculated as $S_t = max(0, S_{t-1} + R_t - k)$, where $S_0 = 0$ and k is a constant found by numerical investigation as the expected value of R_t when the mean of one stream shifts off-target by σ_x. For comparison purposes, we carried out further 100000 simulations to get also a fair Upper Control Limit (UCL) for the R_t and CUSUM-R_t control charts corresponding to an ARL_0 as close as possible to 370. The ARL_1 values are then calculated by using additional 1000000 simulations at different number $l = 1, .., s-1$ of streams that shift off-target ($\mu = 0$), at different mean shift size $\Delta\mu = \sigma_x, 1.5\sigma_x, 2\sigma_x, 2.5\sigma_x, 3\sigma_x$, number of streams $s = 6, 8, 10$ and subgroup size $n = 1, 5$. Table 1.2 and Table 1.3 show the corresponding ARL_1 values

achieved by the R_t control chart and the proposed one for a process with $s = 6$, whereas Table 1.4 and Table 1.5 show the same comparison for a process with $s = 8$, and Table 1.6 and Table 1.7 with $s = 10$. When the output of more than one stream shifts off-target, for the sake of conciseness, we consider the case where the output of the l streams shift simultaneously in the same direction and by the same magnitude $\Delta\mu$. It is clear from Tables 2-7 that the proposed NN method significantly outperforms the R_t control chart in every angle of comparison and the CUSUM-R_t control chart in the majority of the considered scenarios. However, the ARL_1 differences become smaller as the subgroup sample size n or the shift in the process mean $\Delta\mu$ increases. As an example, for $n = 5$, $l = 4$, $s = 6$ and $\Delta\mu = \sigma_x$ the ARL_1 is equal to 1.31 for the proposed method, 8.91 for the R_t control chart and 4.58 for the CUSUM-R_t control chart. Whereas, for $n = 5$, $l = 4$, $s = 6$ and $\Delta\mu = 2\sigma_x$ the ARL_1 is equal to 1.00 for the proposed method, 1.18 for the R_t control chart and 1.29 for the CUSUM-R_t control chart The ARL_1 of the three competing methods generally increases with the number of stream s at given l and n. When $n = 5$, as the shift in the process mean $\Delta\mu$ increases, the detection power (1 - β) of the competing methods approaches one in many scenarios. R_t and CUSUM-R_t control chart performance are highly affected by the number of OC streams. For example, when half of the streams shift off-target, the ARL_1 is minimum, while, it rapidly increases with the number of OC streams. The advantages of the proposed control charting are increasing with the number of OC streams. Indeed, as the number of OC streams increases, the proposed NN control charting performance improves faster than the two control charts based on the R_t statistic initially. Furthermore, the proposed scheme demonstrates to signal OC conditions more promptly, even for combinations of large s and l. It is worth considering the cases in which the mean shift is small, say $\Delta\mu \leqslant 2\sigma_x$. As expected, the SCC based on R_t is not recommended in these cases, as it is outperformed both by the proposed approach, especially when $n = 1$, and the CUSUM-R_t control chart, coherently with the conclusions of Mortell & Runger (1995). In particular, the CUSUM-R_t control chart outperforms the proposed approach only in the case that one stream shifts off-target and the mean shift is small. Therefore, even a small change in one or a few streams may be well and better detected by the proposed control charting scheme.

The entire simulation and training of the NN was performed through the software package `Python` (Van Rossum & Drake Jr 1995) with the aid of Keras (Chollet 2015) and TensorFlow (Abadi et al. 2015) libraries.

1.4 A real-case study

HVAC and data description

Operational data from modern train HVAC systems were automatically acquired during about two months (from the 1st of July to the 19th of August) and stored for analysis in the current railway industry scenario. Train names and the voyage year are intentionally omitted for confidentiality reasons. HVAC is the technology of interior and vehicular environmental comfort, whose goal is to maintain internal air quality and regulate interior temperatures and humidity through ventilation, heating and cooling operations. Ventilation is the process of replacing or exchanging air to provide better interior air quality, removing smoke, odors, dust and bacteria. The heating and conditioning process provides heat and cooling, respectively. A typical HVAC system consists of a condenser section and an evaporator section. A compressor forces the refrigerant substance into the condenser section, changing it from a gas to a liquid, and releasing heat energy into the air. The liquid is then forced through the evaporator

Table 1.8: Operational variables acquired for each train and coach considered in the proposed approach.

	Variable
1	Interior temperature, T_{in}
2	Outside temperature, T_{out}
3	Theoretical temperature to be achieved, T_{set}
4	Air temperature provided by the HVAC, T_{supply}
5	Difference between T_{in} and T_{out}, ΔT
6	Date and time of the signal
7	Vehicle name

section, where the refrigerant evaporates into a cold gas, being able to remove any heat from the surrounding air. As the heat is removed from the air, the air is cooled and blown back into the coach. Then, the condenser turns the cold gas back into a liquid. This process continues again and again until the interior temperature reaches the desired temperature.

A microprocessor temperature control monitors the HVAC system performances by using temperature sensors to properly regulate heating and cooling modes in response to temperature changes inside and outside the coach and to maintain comfortable internal temperature and humidity levels in compliance with European regulation (EN 2006). In particular, sensors gather signals from the *outside temperature (T_{out})* and *interior temperature (T_{in})*. The *setpoint temperature (T_{set})* represents the target value at which the HVAC systems aim and is a function of T_{out}. The HVAC system is designed adequately to maintain T_{in} as close as possible to the T_{set}. The difference $\Delta T = T_{in} - T_{set}$ must be close to zero and can admit a tolerance of $2°C$, at most, as established by the European standards (EN 2006). A process capability analysis is beyond the scope of this work, and nevertheless, it is not well clarified the subgroup size on which apply the $2°C$ tolerance on ΔT. On the other hand, it is undoubtedly that is of great interest to monitor ΔT to detect any changes in HVAC system performance. Table 1.8 summarises the available variables used to describe the HVAC operating conditions and to build the proposed model.

In this real-case study, data were collected from three different trains, train 1, train 2 and train 3 with a sampling period of 2 minutes. In particular, data from train 1 will be used as a reference sample for mean and variance estimation of the residuals X_{tj}, as the six onboard HVAC systems have worked for the entire considered period without any failure. Data from train 2 and train 3 will be considered to show a real-case scenario in which one and more than one (four) stream shifts off-target, respectively.

Results

To look for changes in the overall process mean and variability (OC state of the type (i)), a Mortell and Runger's $\overline{\overline{X}}$ and a Liu, Xuyuan and MacKay's s chart for the overall subgroup may be implemented. As said, note that these control charts are not sensitive to the within-stream variation and are out of the specific scope of this work, which instead focuses on the monitoring of the individual stream components. In the real-case study at hand, this condition corresponds to a very rare event, in which all the HVAC systems shift off-target at the same time. The root cause of this event is to be found in the breakdown of the power

supply or, in general, in a problem independent of the system under investigation. For this reason, we mainly focus on the monitoring of the within-stream variability.

To look for changes in one or few streams (OC state of the type (ii)) through the proposed framework, the ΔT signal is considered as the monitoring variable since it summarises the deviation between the interior temperature T_{in} and the setpoint temperature T_{set} set by the European regulations. When a fault occurs, the HVAC system is no longer able to keep the interior temperature within the tolerance limits and ensure passenger thermal comfort inside the coach. Therefore, the ΔT signal coming from the coach of the faulty HVAC system will increase and will be significantly different from the ΔT signals of other train coaches. Typical causes of malfunctioning of the HVAC systems are compressor failure or the lack of refrigerant gas. The simultaneous SPC of the ΔT signals coming from each train coach can be regarded as a MSP. In this context, Y_{tjk} is the kth measurement of the ΔT signal from the jth coach at sample time t.

In this section, we describe in detail how to apply the proposed NN based control charting scheme for the monitoring of onboard HVAC systems for railway applications. Firstly, data are cleaned to remove unsteady working conditions and sensor measurement errors and validated by domain experts. To get the residuals X_{tj}, to give as input to the NN, we subtract the average value of the ΔT of all streams (coaches) Y_t from the subgroup mean Y_{tj} from each individual stream j at every sample time t. The number of streams is $s = 6$, corresponding to the six train coaches. Data are collected every 2 minutes, but they are downloaded on company servers and made available for analysis every 10 minutes by the train data acquisition system. Thus, although a different sample size can be theoretically chosen, in this application we set $n = 5$, as it corresponds to 10-minute-worth-of-data and represents the natural subgroup size. Recall that in the simulation study of Section 1.3, as n increases, the proposed scheme achieves larger power in detecting a specified mean shift $\Delta\mu$ and it makes it easier to detect small process mean shift. Before feeding the residuals to the NN to judge whether the process is IC or OC, X_{tj} is standardized to have zero mean and unit variance. To do this, we need to consider an IC reference sample to evaluate the mean and the variance of X_{tj}. We then collected 50 subgroups of size $n = 5$ from each coach of train 1. After subtracting Y_t from Y_{tj} at every sample time t, the sample mean and variance of the residuals X_{tj} are calculated. As an example, Figure 1.3 shows ΔT streams for each coach of train 1 over 50 subgroups, each representing 10 minute-worth of data. The ΔT signals from each train's coach show similar behaviors and all the six onboard HVAC systems properly work. As we can see, there is clear graphical evidence that the variability over time is much greater than the within-stream variability, i.e., when a stream behaves differently from the others. As already pointed out, a shift in a single stream, that is, a shift in the mean of the individual component of the stream is quite difficult to detect for a traditional control chart when σ_a^2 is larger than σ^2. Nevertheless, the proposed methodology is demonstrated to be a suitable tool because of its ability to promptly detect as soon as possible even a small shift in the mean of Y_{tj}. Let us consider fault data in Phase II for train 2 and train 3, in which one and four streams shift off-target at the same time, respectively. Figure 1.4 shows MSP data in which the process is OC and assignable causes plausibly affect the output from one stream. From Figure 1.4, we conclude that the temperature inside coach 5 is higher than the setpoint temperature, and, as expected, ΔT behavior is significantly different from the other coach signals. After computing and standardizing residuals from each coach, the range of the subgroup means of the residuals X_{tj} and the overall mean X_t are calculated at each sample time. Then the range, the overall mean and the six residuals X_{tj} for each coach are given as input to the NN. If the predicted probability of being drawn from an OC process is

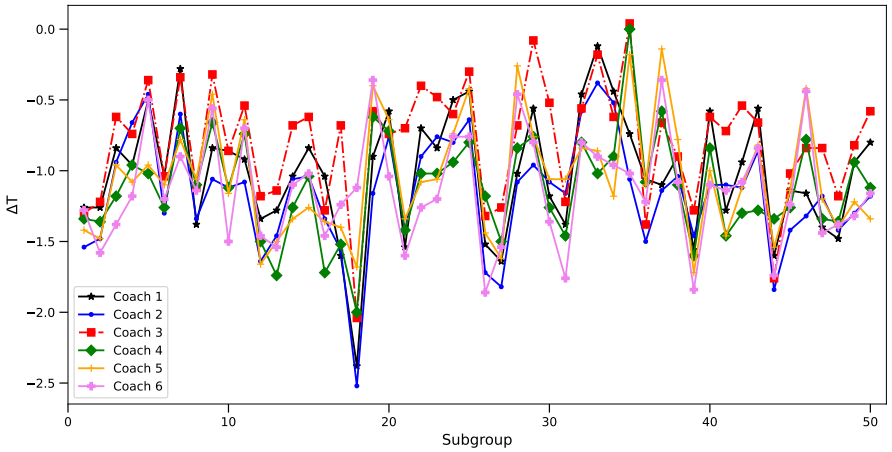


Figure 1.3: IC ΔT stream for each coach of the train 1 over 50 subgroups (10 minutes worth of data). The ΔT signals from each train's coach show similar behavior and all the six onboard HVAC systems properly work. As we can see, there is clear graphical evidence that the variability over time is much greater than the differences between streams, thus our proposed methodology is a suitable tool to monitor this kind of process.

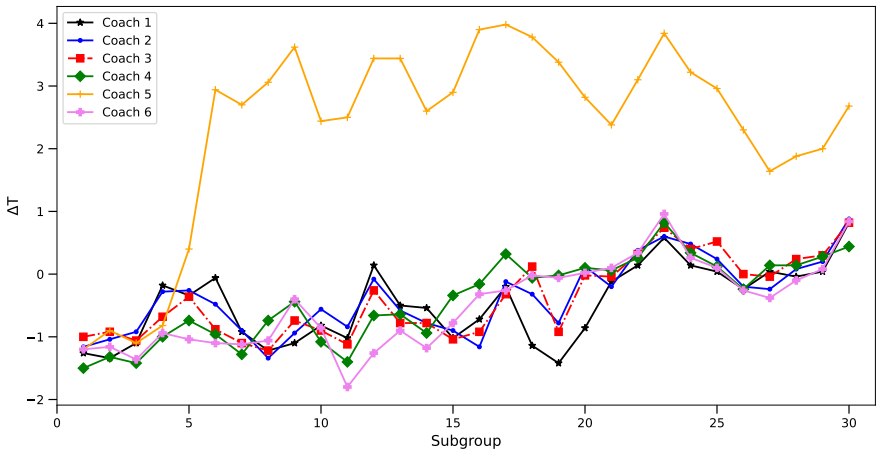


Figure 1.4: ΔT stream for each coach of the train 2. The process is OC and an assignable cause affects the output from one stream. Coach 5 is characterized by higher ΔT values, that is the temperature inside coach 5 is higher than the setpoint temperature set by the European regulations. As we expected, its signal behavior is significantly different from the other coach signals.

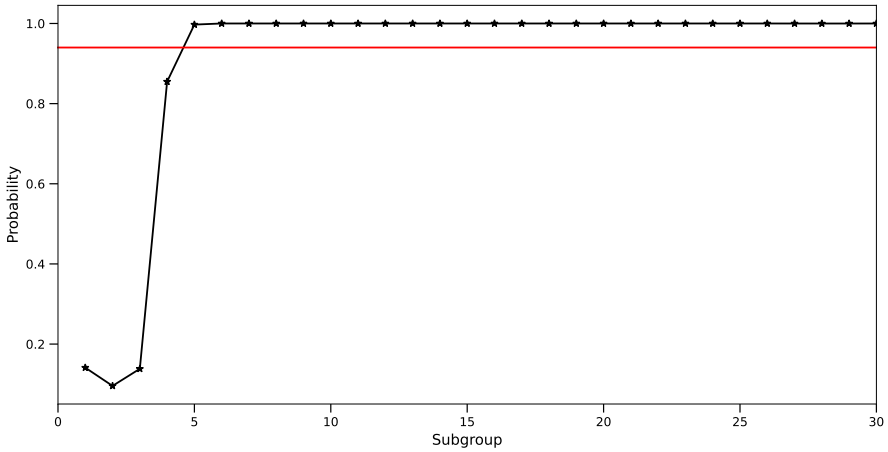


Figure 1.5: Control chart based on the NN predicted probability. The single stream shifting is properly detected.

larger than the CV, found in the simulation study, the sample is judged to be drawn from an OC process, otherwise, the process is judged as IC. Therefore, a control chart based on the NN predicted probability can be designed. This probability is plotted on the control chart and the UCL is set equal to the CV. If all the plotted points fall inside the UCL, the process is considered to be IC and no action is required. In contrast, if a point plots outside the UCL, there is evidence that the process is OC. The proposed control chart is shown in Figure 1.5. The separation among streams is easily detected by the proposed monitoring strategy. The NN starts in fact to promptly signal the process as OC from the first sample in which the temperature in coach 5 becomes slightly larger than the others. The advantage of designing such a control chart is to automate visual interpretation and identification of patterns in the plotted points by maintenance operators. Figure 1.6 shows a MSP data in which the process is OC and that assignable causes affect the output from four streams. The ΔT signals of coaches 1,2,4,5 increase together, while the ΔT signals of the remaining two coaches show a typical IC behavior. The control chart is shown in Figure 1.7. Even though the number of streams that shift off-target is more than one (four), the proposed methodology demonstrates to be capable of detecting these shifts. In particular, the control chart produces an OC signal starting from sample #21, i.e., as soon as the streams show little different behavior. From sample #24, the mean shift of the four streams becomes more evident.

As in the multivariate SPC, it would be of practical interest, given an OC signal, to identify how many and which streams have shifted from the IC state. However, this is not a standard task tackled by MSP control charts. For example, Runger et al. (1996) suggest observing raw data at the time of the OC signal to identify the involved streams. Even for the more recent MSP literature, the OC stream identification is still an open issue (Epprecht 2015) and falls beyond the scope of this chapter. Keeping confined to the railway passenger transportation context, where the number of streams (i.e., the number of coaches of a passenger train) is small, we suggest performing the OC stream identification, at a

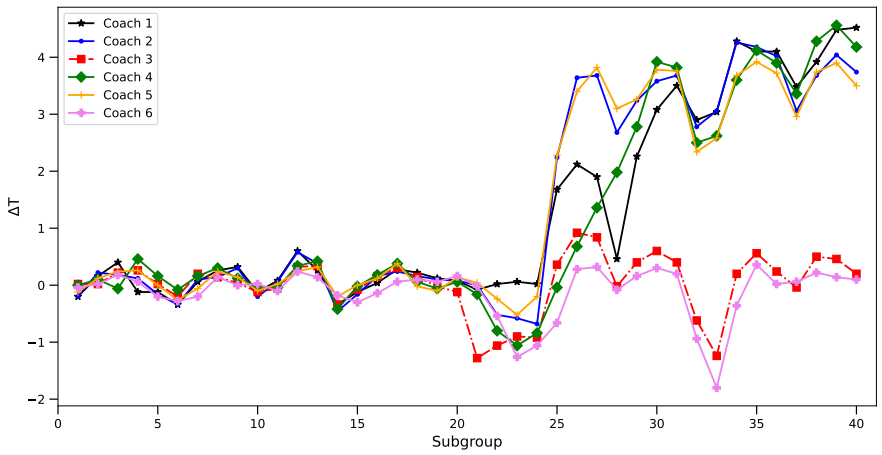


Figure 1.6: ΔT stream for each coach of the train 3. The process is OC and an assignable cause affects the output from four streams. The ΔT signals of coaches 1,2,4,5 increase together, while the ΔT signals of the remaining two coaches have a typical IC behavior.

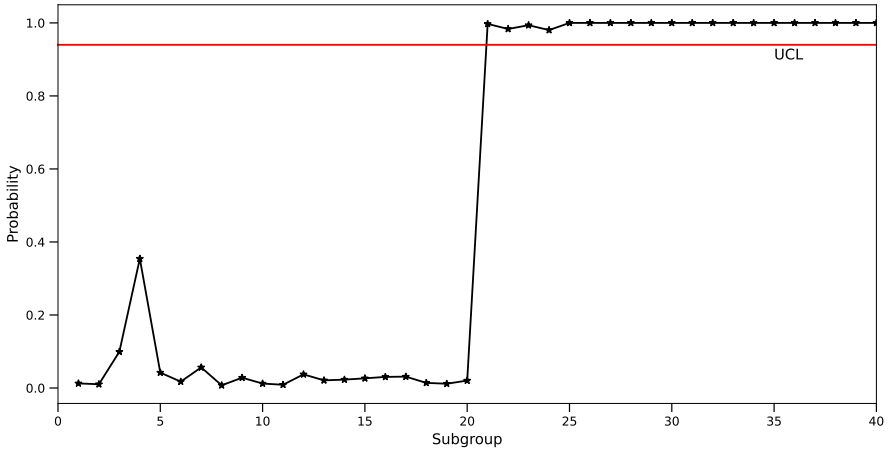


Figure 1.7: Control chart based on the NN predicted probability. The multiple streams shifting off target are properly detected.

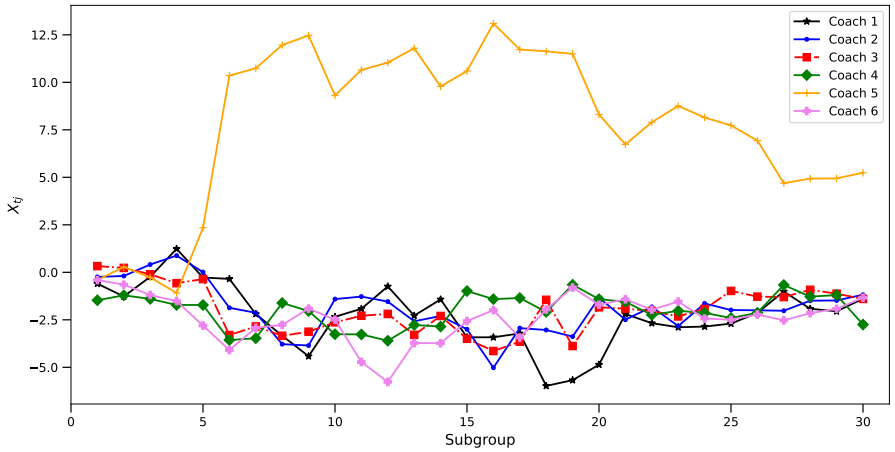


Figure 1.8: Time-series plot of residuals for each coach of the train 2.

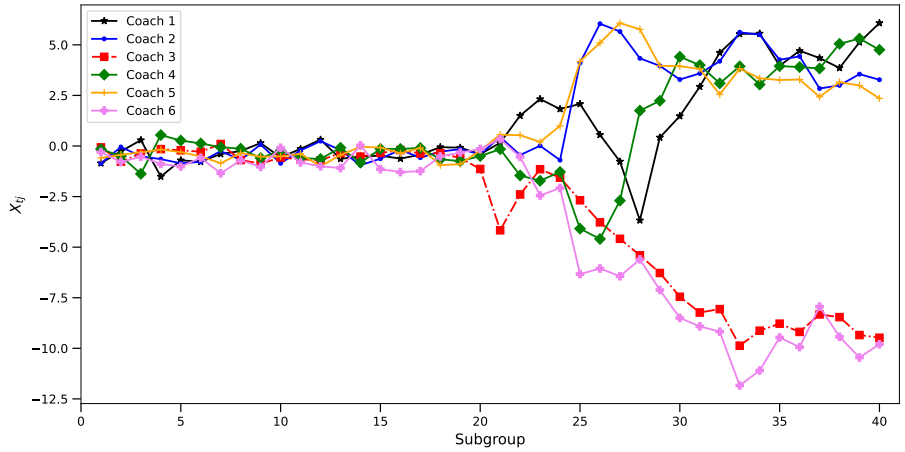


Figure 1.9: Time-series plot of residuals for each coach of the train 3.

given OC signal, by visually checking the time-series residual plot for each stream, because differently from the raw data streams of quality characteristics (i.e., ΔT in this real-case), it does not depend on the process variability common to all streams and is sensitive to stream-to-stream differences alone. Residuals for each coach of train 2 and train 3 are plotted in Figure 1.8 and 1.9, respectively. Figure 1.8 shows that coach 5 residuals (and then the relative ΔT) are larger than those corresponding to other coaches. Analogously, Figure 1.9 shows that coaches 1,2,4,5 of train 3 underperform the other two coaches. Note that residuals are constrained to have zero mean at each point in time. Therefore, when an OC occurs, their behaviors tend to separate themselves and highlight IC and OC streams by

small and large residual time series, respectively. The advantage of using the time-series plot of the residuals for this scope is also underlined in Section 1.6, which appositely presents a simulated scenario where an upward shift of magnitude 2σ is applied to the mean of only one stream out of the $s = 6$ streams with $\sigma_a = 5\sigma$. The residual plot is thus proven to be a very insightful graphical tool to help practitioners obtain a correct interpretation of OC situations. The reader may find additional offline analysis methods (Ott & Snee 1973) to identify specific stream(s) with abnormal performance.

1.5 Conclusion

Processes with multiple streams can be monitored using different traditional control chart methods. However, to the best of the authors' knowledge, this research represents the first attempt to embed a NN approach into a control charting scheme to get a more powerful but still familiar tool for the monitoring of MSPs. A MLP was designed to classify each input sample drawn from an IC or OC process and guidelines were given to properly train the MLP through a back-propagation algorithm.

The performance of the proposed method in identifying changes in multiple streams has been investigated by means of an extensive Monte Carlo simulation study, where the proposed monitoring scheme has been compared with the traditional CUSUM-R_t and R_t control charts proposed by Mortell & Runger (1995) in terms of the OC ARL (ARL_1) at a different number of streams and sample sizes. To allow for a fair comparison and keep the risk of false alarms within usual values, a detailed procedure is described to properly design the NN. In particular, the CV of the neuron in the output layer must be properly set as it is uniquely related to the Type-I error rate. The results showed that the proposed scheme performs far better than the competitors when the number of OC streams is large.

The practical applicability of the proposed scheme is illustrated by means of real operational data in the monitoring of HVAC systems in railway passenger coaches, even though the validity of the proposed approach is not limited to the case study here presented and, in general, can be extended to any MSP. Encouraged by these results, the proposed approach could be developed for the monitoring of a MSP where the quality characteristics are in the form of attribute data. Furthermore, future research paths should be urgently addressed to develop methods, possibly based on NNs, for the automatic identification of stream(s) involved in the OC signals issued by a MSP monitoring scheme.

1.6 Supplementary Materials

Calculation of quality variable covariance

In this section, we derive the covariance of any two subgroup means at a given stream j or sample times t reported in Equation (1.5). According to Equation (1.4), $Y_{tj} = \mu + A_t + e_{tj}$ is normally distributed with mean μ and variance $\sigma_a^2 + \sigma^2/n$. This follows from the model assumptions in Equation (1.3), that is, the terms A_t and e_{tjk} are assumed as normally distributed random variables with zero mean and variance σ_a^2 and σ^2, respectively. Furthermore, A_t is assumed as independent from e_{tjk}. The covariance $\mathrm{Cov}(\cdot,\cdot)$ between Y_{tj} and $Y_{tj'}$ of the jth and j'th streams at a given sample time t can be expressed as the expected value of their product minus the product of their expected values, i.e., $\mathrm{Cov}(Y_{tj}, Y_{tj'}) = \mathrm{E}(Y_{tj}Y_{tj'}) + \mathrm{E}(Y_{tj})\mathrm{E}(Y_{tj'})$. The product of their expected values is simply μ^2. To compute

$\mathrm{E}(Y_{tj}Y_{tj'}) = \mathrm{E}((\mu + A_t + e_{tj})(\mu + A_t + e_{tj'}))$, we can expand the product

$$(\mu + A_t + e_{tj})(\mu + A_t + e_{tj'}) = \mu^2 + \mu A_t + \mu e_{tj'} + A_t\mu + A_t^2 + A_t e_{tj'} + e_{tj}\mu + e_{tj}A_t + e_{tj}e_{tj'}$$

and hence we have

$$\mathrm{E}((\mu + A_t + e_{tj})(\mu + A_t + e_{tj'})) = \mu^2 + \mathrm{E}(A_t^2).$$

Note that all the expected values of the products involving A_t or e_{tj} are zero due to their zero mean. $\mathrm{E}(A_t^2) = \mathrm{Var}(A_t) + \mathrm{E}(A_t)^2 = \sigma_a^2$ and $\mathrm{E}(A_t e_{tj}) = \mathrm{E}(A_t)\mathrm{E}(e_{tj}) = 0$, thus it follows that $\mathrm{Cov}(Y_{tj}, Y_{tj'}) = \mu^2 + \sigma_A^2 - \mu^2 = \sigma_A^2$. This proves the second result in Equation (1.5). The first result in Equation (1.5) underlines that the covariance of any two subgroup means at a given stream j and at different sample time t and t' is zero. Similar consideration can be made and it will be simple to prove that $\mathrm{Cov}(Y_{tj},Y_{t'j})$ is zero because the expected values of the products involving the e_{tj}'s and A_t's at different sample times are zero because samples are drawn independently from the distribution.

Calculation of residual covariance

In this section, we derive the variance and the covariance of residuals X_{tj} reported in Equation (1.9) and (1.8), respectively. From Equation (1.3), it follows $Y_t = \mu + A_t + e_t$ is normally distributed with mean μ and variance $\sigma_a^2 + \sigma^2/(ns)$. Moreover,

$$e_{tj} = \frac{1}{n}\sum_{k=1}^{n} e_{tjk} \quad t = 1, 2, ..., T, j = 1, 2, ..., s,$$

is normally distributed with mean zero and variance σ^2/n and

$$e_t = \frac{1}{s}\sum_{j=1}^{s} e_{tj} \quad t = 1, 2, ..., T,$$

is normally distributed with mean zero and variance $\sigma^2/(ns)$. In Equation (1.7), the residuals X_{tj} are defined as $Y_{tj} - Y_t = \mu + A_t + e_{tj} - \mu - A_t - e_t = e_{tj} - e_t$. Explicitly note that $e_{tj} - e_t$ are not independent random variables. The variance of X_{tj} can be expressed as the mean of the square of X_{tj} minus the square of the mean of X_{tj}, i.e., $\mathrm{Var}(X_{tj}) = \mathrm{E}(X_{tj}^2) - \mathrm{E}(X_{tj})^2$. The mean of X_{tj} is $\mathrm{E}(X_{tj}) = \mathrm{E}(e_{tj} - e_t) = \mathrm{E}(e_{tj}) - \mathrm{E}(e_t) = 0$. It follows that $\mathrm{E}(X_{tj})^2 = 0$. To compute $\mathrm{E}(X_{tj}^2) = \mathrm{E}((e_{tj} - e_t)^2)$, we can expand the square, making the following considerations

$$e_{tj} - e_t = e_{tj} - \frac{1}{s}\sum_{j=1}^{s} e_{tj} = \left(1 - \frac{1}{s}\right) e_{tj*} - \frac{1}{s}\sum_{\substack{j=1 \\ j\neq j^*}}^{s} e_{tj}, \tag{1.14}$$

$$(e_{tj} - e_t)^2 = \left[\left(1 - \frac{1}{s}\right) e_{tj*} - \frac{1}{s}\sum_{\substack{j=1 \\ j\neq j^*}}^{s} e_{tj}\right]^2 \tag{1.15}$$

$$= \left(1 - \frac{1}{s}\right)^2 e_{tj*}^2 + \frac{1}{s^2}\sum_{\substack{j=1 \\ j\neq j^*}}^{s} e_{tj}^2 - 2\left(1 - \frac{1}{s}\right)\frac{1}{s} e_{tj*}\sum_{\substack{j=1 \\ j\neq j^*}}^{s} e_{tj}, \tag{1.16}$$

where j^* denotes the stream with respect to which we are defining the residual. The expected value of the cross-products in Equation (1.16) are all zero since the e_{tj}'s are independent and with zero mean. Thus, it follows that

$$\mathrm{E}(X_{tj}^2) = \mathrm{E}\left(\left(1-\frac{1}{s}\right)^2 e_{tj*}^2 + \frac{1}{s^2}\sum_{\substack{j=1\\ j\neq j*}}^{s} e_{tj}^2\right) = \left(1-\frac{1}{s}\right)^2 \mathrm{E}(e_{tj*}^2) + \frac{1}{s^2}\sum_{\substack{j=1\\ j\neq j*}}^{s} \mathrm{E}(e_{tj}^2)$$
$$= \left(1-\frac{1}{s}\right)^2 \frac{\sigma^2}{n} + \frac{1}{s^2}(s-1)\frac{\sigma^2}{n} = \frac{s-1}{s}\frac{\sigma^2}{n},$$
$$\text{where } \mathrm{E}(e_{tj}^2) = \mathrm{Var}(e_{tj}) + \mathrm{E}(e_{tj})^2 = \sigma^2/n.$$

This proves the result in Equation (1.9).

The covariance between the residuals X_{tj} and $X_{tj'}$ of the jth and j'th streams can be expressed as the expected value of their product minus the product of their expected values, i.e., $\mathrm{Cov}(X_{tj}, X_{tj'}) = \mathrm{E}(X_{tj}X_{tj'}) + \mathrm{E}(X_{tj})\mathrm{E}(X_{tj'})$. The expected value of X_{tj} is zero, thus $\mathrm{E}(X_{tj})\mathrm{E}(X_{tj'}) = 0$. $\mathrm{E}(X_{tj}X_{tj'})$ can be written as $\mathrm{E}((e_{tj} - e_t)(e_{tj'} - e_t)) = \mathrm{E}(e_{tj}e_{tj'} + e_t^2 - e_{tj}e_t - e_{tj'}e_t)$. Explicitly note that $\mathrm{E}(e_{tj}e_{tj'}) = 0$ since the $e'_{tj}s$ are independent and $\mathrm{E}(e_t^2) = \mathrm{Var}(e_t) + \mathrm{E}(e_t)^2 = \sigma^2/(ns)$. Then, we can expand the product $e_{tj}e_t$ as

$$e_{tj}e_t = e_{tj}\frac{1}{s}\sum_{j=1}^{s} e_{tj} = \frac{1}{s}e_{tj*}^2 + \frac{1}{s}e_{tj*}\sum_{\substack{j=1\\ j\neq j*}}^{s} e_{tj}. \tag{1.17}$$

The expected value of the cross-products in Equation (1.17) will be zero, thus it follows that

$$\mathrm{Cov}(X_{tj}, X_{tj'}) = \mathrm{E}(e_t^2) - \mathrm{E}(e_{tj}e_t) - \mathrm{E}(e_{tj'}e_t)$$
$$= \mathrm{E}(e_t^2) - \frac{1}{s}\mathrm{E}(e_{tj}^2) - \frac{1}{s}\mathrm{E}(e_{tj'}^2) = \frac{\sigma^2}{ns} - \frac{1}{s}\frac{\sigma^2}{n} - \frac{1}{s}\frac{\sigma^2}{n} = -\frac{1}{s}\frac{\sigma^2}{n}$$

Equation (1.8) can be easily derived by dividing the covariance just calculated by the variance of the residuals. Explicitly note that Equation (1.14) expresses the residuals X_{tj} as a linear combination of independent normally distributed random variables, thus X_{tj} is demonstrated to be normally distributed.

Out-of-control stream identification

Figure 1.10 shows the time-series plot of simulated data drawn from the process described by Equation (1.3) with $s = 6$ streams, $\sigma_a = 5\sigma$ and one observation per stream ($n = 1$). An upward shift of magnitude 2σ was applied to the mean of stream 2 at sample #10. As already pointed out in Section 1.6, Figure 1.10 remarks that when σ_a^2 is considerably larger than σ^2, traditional control charts may be not able to identify the OC streams. The corresponding time-series residual plot is shown in Figure 1.11 of each stream $j = 1, \ldots, s$. From this figure, it is clear that the visual inspection of such a plot supports the OC stream identification better than observing raw data at the time of OC signals, as recommended by Runger et al. (1996).

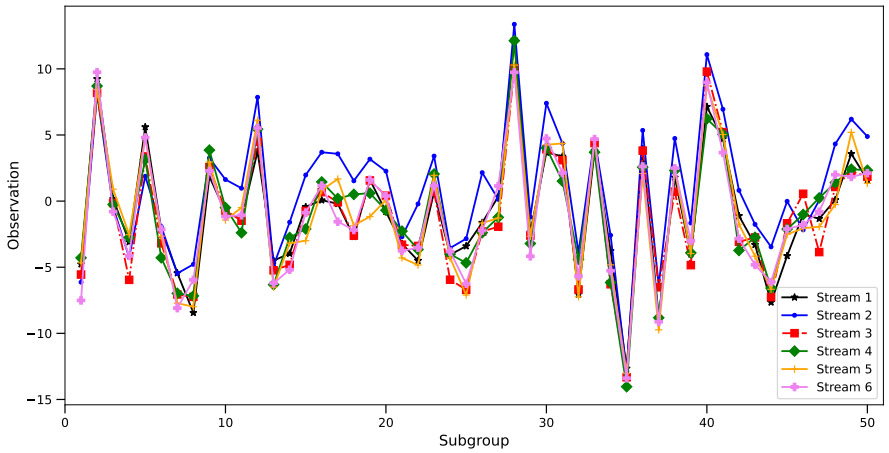


Figure 1.10: A 6-stream simulated process with a shift in one stream at Sample #10.

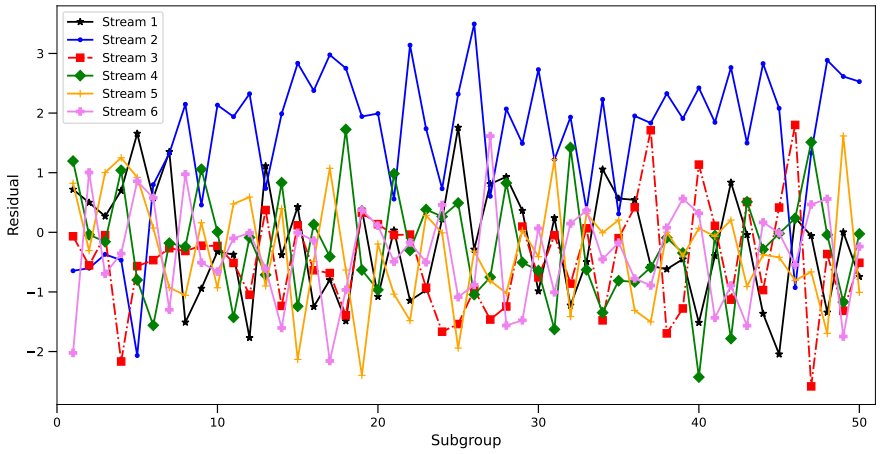


Figure 1.11: Time-series plot of residuals of a 6-stream simulated process with a shift in one stream at sample #10.

Neural network for statistical process control of a multiple stream binomial process with an application to HVAC systems in passenger rail vehicles

A Multiple Stream Binomial Process (MSBP) is a process at a point in time that generates several output streams that can be modeled as binomial processes. From the SPC point of view, the quality of interest and its specifications are identical in all streams. Motivated by the industrial context and the latest attempts at the integration of NNs into classical control charting procedures (Zorriassatine & Tannock 1998, Psarakis 2011, Hwarng 2014), this section aims to develop a new approach based on NNs in the field of SPC of MSBPs with the twofold objective of detecting changes both in the individual stream means and the overall process mean.

Let us suppose that over a subgroup of N consecutive points at time t, the quality of interest is a variable counting the number of times a particular event occurs. Let Y_{tj} denote this counting variable for each stream $j = 1, ..., s$ and time index $t = 1, ..., T$. A MSBP assumes Y_{tj} distributed as independent binomial random variables $Y_{tj} \sim \text{Bin}(N, p_j)$ over time and streams, where p_j denotes the probability that the event of interest occurs for the jth stream at each subgroup point. For an IC homogeneous MSBP p_j is assumed, for each j, equal to the same constant $p \in (0, 1)$, which therefore represents the overall process mean. The process is said to be OC if an assignable cause affects (i) all of the streams uniformly (the overall p changes) or (ii) one or more streams differently (p_j changes for one or more streams). Inspired by the real-case study, for each stream, we generate pseudorandom observations of Y_{tj} from a binomial random variable with $s = 6$, $N = 30$, and $p = 0.52$. To the best of our knowledge, Wludyka & Jacobs (2002*a*,*b*) are the only authors to have investigated the SPC of a MSBP.

In this section, a MLP (Goodfellow et al. 2016), which is one of the most widely used NN, is designed to be used for the SPC of a MSBP, that is, to classify a sample as drawn from an IC or OC process. Specifically, the MLP is trained such that, when a sample (called *negative sample*) is drawn from an IC process, the target value of the output neuron is coded as zero, otherwise, when a sample (called *positive sample*) is drawn from an OC process, it is coded as one. The MLP has s neurons in the input layer to represent the s monitored statistics Y_{tj} at time t, for each stream $j = 1, 2, ..., s$. In order to train and properly design the MLP, 300 positive samples are generated through a Monte Carlo simulation from all possible $\sum_{l=1}^{s} \binom{s}{l}$ OC scenarios in which $l = 1, \ldots, s$ streams shift off the target p at the same time by as much as $\Delta p = -1.0\sigma, -2.0\sigma, -3.0\sigma, 1.0\sigma, 2.0\sigma, 3.0\sigma$, where $\sigma = \sqrt{p(1-p)/N}$ is the standard deviation of p. To get a balanced data set, the total number of negative and positive samples must be equal. The MLP is trained through the back-propagation algorithm (Goodfellow et al. 2016). To compare different NN architectures, AUC (Fawcett 2006) is used as a performance measure and is calculated on the validation set, obtained by retaining the 30% of the generated samples. Then, a MLP with one hidden layer with ten neurons, a ReLU and a sigmoid activation function (Goodfellow et al. 2016) for the hidden and output layer, respectively, are shown to have the best performance in terms of AUC. Furthermore, to allow a fair comparison with the traditional MSBP control charts, the NN must be tuned to a corresponding level of discriminating power (Hwarng 2014), that is, the CV of the output neuron must be properly fixed to set the Type-I and Type-II error rates of the resulting control charting scheme denoted by α and β, respectively. Or, equivalently, one may set the IC or OC ARLs (Montgomery 2019), respectively, defined as $ARL_0 = \frac{1}{\alpha}$ and $ARL_1 = \frac{1}{1-\beta}$. Additional 100000 Monte Carlo simulations from an IC process are generated

Table 1.9: Comparison of ARL_1 of the Chi-squared control chart (Wludyka & Jacobs 2002*a*) and of the proposed NN approach based on 100000 simulations of a MSBP with $s = 6$ streams, at different number $l = 1, ..., s$ of streams that shift off the target and mean shift size $\Delta p = -2.0\sigma, -1.0\sigma, -0.5\sigma, 0.5\sigma, 1.0\sigma, 2.0\sigma$.

	NN based control charting procedure Δp						Chi-squared control chart Δp					
	-2.0σ	-1.0σ	-0.5σ	0.5σ	1.0σ	2.0σ	-2.0σ	-1.0σ	-0.5σ	0.5σ	1.0σ	2.0σ
l												
1	36.21	143.88	276.24	249.38	123.46	26.08	22.68	138.5	290.7	288.18	146.84	25.07
2	6.2	52.97	186.57	164.20	43.16	4.8	5.91	63.41	190.48	238.1	65.4	6.29
3	2.52	23.66	113.38	99.60	21.43	2.29	2.78	33.59	164.2	176.06	37.79	2.85
4	1.47	11.72	73.86	66.36	10.82	1.4	1.78	20.43	129.07	149.03	23	1.79
5	1.15	6.44	45.56	45.13	6.2	1.23	1.37	13.75	101.83	115.47	15.27	1.38
6	1.05	4.14	33.02	32.77	4.18	1.04	1.17	9.81	82.51	101.21	10.99	1.18

Table 1.10: Comparison of ARL_1 of the group p control chart (Wludyka & Jacobs 2002*b*) and of the proposed NN approach based on 100000 simulations of a MSBP with $s = 6$ streams, at different number $l = 1, ..., s$ of streams that shift off the target and mean shift size $\Delta p = -2.0\sigma, -1.0\sigma, -0.5\sigma, 0.5\sigma, 1.0\sigma, 2.0\sigma$.

	NN based control charting procedure Δp						Group p control chart Δp					
	-2.0σ	-1.0σ	-0.5σ	0.5σ	1.0σ	2.0σ	-2.0σ	-1.0σ	-0.5σ	0.5σ	1.0σ	2.0σ
l												
1	36.21	143.88	276.24	249.38	123.46	26.08	12.63	107.76	263.85	374.53	230.95	29.94
2	6.2	52.97	186.57	164.20	43.16	4.8	6.68	59.38	200	389.11	167.5	15.55
3	2.52	23.66	113.38	99.60	21.43	2.29	4.64	41.95	152.44	381.68	131.75	10.4
4	1.47	11.72	73.86	66.36	10.82	1.4	3.63	31.21	124.38	343.64	104.93	7.95
5	1.15	6.44	45.56	45.13	6.2	1.23	3.03	25.63	104.17	343.64	83.96	6.59
6	1.05	4.14	33.02	32.77	4.18	1.04	2.64	22.68	93.02	377.36	77.7	5.6

to set the CV to achieve an ARL_0 as close as possible to the desired one. By simulation, α is the proportion of the wrongly classified samples as drawn from an OC process. At a given ARL_0, we can evaluate the performance of the proposed approach in terms of ARL_1 by simulating 100000 data sets from each of the OC scenarios in which at least one stream of the MSBP shifts off-target at different mean shift sizes. By simulation, β is the proportion of negative samples that the MLP incorrectly classifies as drawn from an IC. The proposed NN based control charting procedure is compared with the competitor Wludyka and Jacobs' Chi-squared (Wludyka & Jacobs 2002*a*) and group p (Wludyka & Jacobs 2002*b*) MSBP control charts by means of the ARL_1, at fixed $ARL_0 = 358$, that is $\alpha = 0.0028$.

Tables 1.9 and 1.10 show the ARL_1 performance at fixed $ARL_0 = 358$ of the Wludyka and Jacobs' Chi-squared and group p MSBP control charts, as well as that of the proposed one at different number $l = 1, ..., s$ of streams that shift off-target by as much as $\Delta p = -2.0\sigma, -1.0\sigma, -0.5\sigma, 0.5\sigma, 1.0\sigma, 2.0\sigma$. In particular, the proposed monitoring strategy outperforms the competitor control charts when at least one stream shifts off-target for all severity levels of mean shift.

In addition, data mentioned in Section 1.4, can be employed to demonstrate the effectiveness of the proposed method for a MSBP. The quality variable of interest Y_{tj} is the cooling mode (C_m) status, which is coded as 1 if the HVAC system is cooling at a given time instant, and 0 otherwise. In particular, for each of the $s = 6$ coach HVAC streams, we monitor

over $N = 30$ samples (i.e., 1 hour) the number of times C_m is 1. The parameter $p = 0.52$ is estimated from an IC sample. The idea is to monitor whether or not an HVAC system is cooling properly to ensure the passenger's thermal comfort. When a stream j returns a p_j estimate that is significantly different from that of the other streams, then an alarm should be provided. The proposed method was demonstrated to be capable of signaling OC conditions in real-time, even when more than one streams (from different coaches) are OC.

Chapter 2

An artificial neural network approach for out-of-control stream identification in multiple stream processes

A MSP is a process at a point in time that generates several streams of output with quality variables and specifications that are identical in all streams. Control charts for MSPs are designed to issue an alarm when a possible change occurs from the IC state of the process. However, in the SPC literature, there is a lack of contributions in the identification of the stream or group of streams responsible for the OC alarm.
In this chapter, we propose a NN specifically trained for this purpose, and, through a Monte Carlo simulation, we show its superiority over competing methods in terms of the correct identification of the OC streams. The research is motivated by the need for OC stream identification in the SPC of multiple streams generated by HVAC systems installed onboard modern train coaches. A case study in this area based on the `HVAC` data (openly available online at `https://github.com/unina-sfere/NN4OCMSP`) illustrates the practical applicability of the proposed method, which is implemented in the `Python` package `NN4OCMSP` (Lepore, Palumbo & Sposito 2022*c*) and published on the PyPI software repository.

2.1 Introduction

A MSP is a process at a point in time that generates several streams of output with quality variables and specifications that are identical in all streams (Montgomery 2019), and occurs quite frequently in a wide variety of real processes. Typical examples of MSP data are measurements of identical features on a single component (e.g., a machine with several heads, each one producing the same product (Amin et al. 1999)) or simultaneous measurements at different locations that can be modeled as correlated streams (e.g., diameter measurements at different radii or thickness measurements at different cross-section (Lanning et al. 2002)). More specifically, throughout the chapter, we will refer to the MSP data generated by HVAC systems installed onboard modern train coaches and the case study thereof discussed in Section 2.3 in the regulation of thermal comfort onboard modern passenger trains. SPC schemes for MSPs aim at detecting the OC state of the process, which is any deviation from the normal operating condition referred to as the IC state. As in any multivariate process, when a given scheme issues an OC alarm, a crucial problem is the identification of the stream

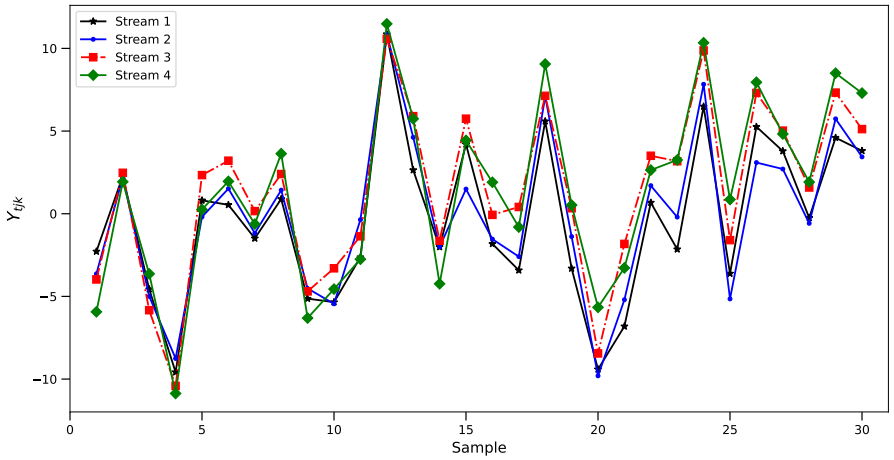


Figure 2.1: Data plot of 30 samples of a MSP with 4 streams with an upward shift in streams 3 and 4 from sample 15 to 30.

or group of streams that most contribute to it, which, in the multivariate SPC literature, is more generally referred to as OC signal interpretation. The literature on SPC for MSPs is not extensive, as also emerges from reviews provided by Jirasettapong & Rojanarowan (2011) and Epprecht (2015), and the literature devoted to the identification of OC streams is even more succinct.

In this regard, a trivial solution was offered by Runger et al. (1996) and Ott & Snee (1973) who suggest the direct observation of the raw data plot around the OC alarm. However, one can be easily convinced that, even when the number of streams is small, this solution is not as straightforward as it appears. As an example, Figure 2.1 displays 30 consecutive samples from a MSP with 4 streams and an upward shift in the last 15 samples of streams 3 and 4.

With the twofold task of monitoring a MSP and identifying the OC streams, Boyd (1950) proposed the *group* control chart, which is based on the monitoring of the largest and smallest sample mean over a sample of size n taken from each of the s streams at each time t. However, Nelson (1986) proposed to improve the identification of the OC stream in the group control chart using a one-sided run test and identify the OC stream as the one that generates the largest (or the smallest) sample mean for a given number, say r, of consecutive samples. An alternative two-sided version of this runs test was proposed by Mortell & Runger (1995) Unfortunately, while, the group control chart with the runs tests is suitable for OC stream identification under the assumption that no more than one stream shifts off-target, the OC stream identification ability of the group control chart with or without either one or two-sided runs tests dramatically fails when more than one stream shifts off-target. By considering only the largest or smallest stream, the group control chart without runs tests is, in fact, not able to store any information about nonextreme streams. The addition of one-sided or two-sided runs tests does not help in the presence of more than one OC stream, because the largest (or the smallest) sample mean randomly alternates between OC streams. The group control chart can be, moreover, criticized as it is based on the assumption that the streams are independent of each other, even though this assumption is rarely met in

practice by a MSP.

As proposed by Mortell & Runger (1995), a suitable model to account for the correlation between the streams is the following single-factor linear model with two sources of variation.

$$Y_{tjk} = \mu + A_t + e_{tjk} \quad t = 1, 2, \ldots, T, \quad j = 1, 2, \ldots, s, \quad k = 1, 2, \ldots, n, \tag{2.1}$$

where Y_{tjk} is the kth observation of the quality variable from the jth stream at time t and μ is the overall process mean. The error term e_{tjk} is assumed to be normally distributed with zero mean and equal standard deviation σ_e and is independent over time t, stream j and observation k. The term A_t is assumed to be normally distributed with zero mean and standard deviation σ_A. Through Equation (2.1), Y_{tjk} is then decomposed into three additive terms: the process mean μ, the common stream component A_t, which depends on time t only, and the individual component e_{tjk}. A shift of all process streams at a given time t^* can be modeled as a change in the mean of A_t for $t \geqslant t^*$, whereas any assignable cause that may affect a stream j^* at sample time t^* is modeled as a change in the mean of e_{tj^*k} for $t \geqslant t^*$. For the sake of simplicity, an omitted variable subscript denotes the variable average over that subscript. For instance, the average over k and j of Y_{tjk} at a given t is denoted by Y_t, $t = 1, 2, \ldots, T$. Under these assumptions, the standard deviation of Y_{tjk} is equal to $\sqrt{\sigma_A^2 + \sigma_e^2}$, and the covariance $\mathrm{Cov}(Y_{tjk}, Y_{tj'k})$, between any pair of streams j and j', at a given k and t, is σ_A^2, which is the variance of A_t.

The average over k of the jth stream at time t is

$$Y_{tj} = \mu + A_t + e_{tj} \quad t = 1, 2, \ldots, T, \quad j = 1, 2, \ldots, s, \tag{2.2}$$

and, consequently, the standard deviation of Y_{tj} and the covariance $\mathrm{Cov}(Y_{tj}, Y_{tj'})$ between any pair of streams j and j' are $\sqrt{\sigma_A^2 + \sigma_e^2/n}$ and σ_A^2, respectively. Under the model in Equation (2.1) and Equation (2.2), Epprecht et al. (2011) proposed an alternative version of the group control chart, which is based on residuals (Ott & Snee 1973) $Y_{tj} - Y_t$ and called it *residual* group control chart. However, it still suffers from the same problems as the original group control chart. Additionally, as also pointed out in Section 2.5, the residuals may lose their capability of directly identifying OC streams for large shift magnitude as they are constrained to have zero mean at each point in time. In spite of that, the use of a separate univariate SCC for the mean of each stream, is the solution proposed by Meneces et al. (2008) to identify more than one stream that shifts off-target at the same time. Furthermore, Ahmadi-Javid & Ebadi (2020) proposed an SPC framework for MSP, where SCC is recommended for the post hoc identification of OC streams.

Today, with the rapid development of modern computer technology and data acquisition systems, many researchers in different scientific areas rely on NNs, which are computational learning systems inspired by human biological neurons and designed to solve a large variety of complex problems. NNs are suitable for modeling multivariate non-linear relationships and, recently, have successfully been employed in classification, speech recognition, computer vision, clustering, and function approximation by relaxing the traditional linearity assumption of parametric statistics. Their wide and successful application has led researchers to investigate their application to SPC (Hwarng 2014) and in the identification of quality variables that are responsible for an OC alarm in multivariate processes. Due to their ability to recognize patterns in the data, NNs proved to improve the performance of traditional control charts in detecting changes in the mean of the process. Zorriassatine & Tannock (1998) and Psarakis (2011) offer a detailed review of the use of NNs with control charts. Whereas, Bersimis et al. (2017) compared OC signal interpretation in multivariate processes, including NNs, and

recommended the latter as the most promising technique. Aparisi et al. (2006) developed a NN to identify a quality variable or group of quality variables responsible for the Hotelling's T^2 control chart (Qiu 2013) OC signal, and demonstrated that, in most cases, their approach outperforms the Mason-Tracy-Young (MTY) decomposition method (Mason et al. 1995, 1997). Specifically, the NN developed by Aparisi et al. (2006) takes as input the value of Hotelling's T^2 statistic and the multivariate observation at the time of the OC alarm. A further application comes from Bersimis et al. (2022) who propose a metamethod to combine the results of several well-known techniques, such as MTY decomposition, to identify OC quality variables.

Along this line, by extending the idea of Aparisi et al. (2006), we propose a novel identification of the OC stream for MSPs that is not related to the specific adopted SPC scheme and does not need human intervention to interpret the raw data stream plots around the OC alarm. As an example, in place of Hotelling's T^2, which turns out not to be the most suitable statistic to monitor a MSP, we apply the proposed NN approach for OC stream identification on top of the range R_t and the mean Y_t statistics, introduced by Mortell & Runger (1995). The performance of the proposed NN approach is compared, in terms of the correct identification of the OC streams, with that of the SCC (Meneces et al. 2008, Ahmadi-Javid & Ebadi 2020), which, as stated before, remains the only competitor capable of identifying multiple OC streams.

The remainder of the chapter is organized as follows. In Section 2.2, the proposed NN approach is presented, and, by means of a Monte Carlo simulation, its performance is compared with that of the competing method. Section 2.3 illustrates its practical applicability through a case study based on data acquired by the transportation company Hitachi Rail STS (`https://www.hitachirail.com/`). These data are referred to as `HVAC` data and are openly available online at `https://github.com/unina-sfere/NN4OCMSP`, together with the analysis code to reproduce the figures and results shown in this section. Section 2.5 provides a discussion on the misleading use of residuals for OC stream identification.
The proposed approach is implemented in the `Python` (Van Rossum & Drake Jr 1995) package `NN4OCMSP` (Lepore, Palumbo & Sposito 2022*c*) that is available on the PyPI software repository.

2.2 The proposed neural network approach

Artificial neural networks for multi-label classification

The OC stream identification can be regarded as a multi-label classification problem (Goodfellow et al. 2016), in which each input vector of predictors $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_j^{(i)}, \ldots, x_d^{(i)}] \in \mathbb{R}^d$, $i = 1, \ldots, N$, $j = 1, 2, \ldots, d$, is associated with a target vector of labels $\mathbf{y}^{(i)} = [y_1^{(i)}, y_2^{(i)}, \ldots, y_r^{(i)}, \ldots, y_s^{(i)}] \in \mathbb{R}^s$, $r = 1, 2, \ldots, s$, of binary components such that if a subset $p \subseteq \{1, 2, \ldots, s\}$ of streams shifts off-target, the corresponding components of the target vector $\mathbf{y}$ are equal to 1. As an example, in a MSP with $s = 2$ streams, there are three possible OC scenarios: only the first stream shift off-target $\mathbf{y} = [1, 0]$; only the second stream shifts off-target $\mathbf{y} = [0, 1]$; both streams shift off-target $\mathbf{y} = [1, 1]$.

NNs consist of a large number of interconnected nodes, called neurons, organized into an input layer, one or more hidden layers, and an output layer. The input layer imports the input vectors $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ to be processed and passes them to the subsequent layers without performing any computations. The neurons in the *input* layer are called input

neurons and are as many as the number of predictors in the input data, denoted by d. The *hidden* layers, which are placed between the input and the *output* layer, are used for the calculation, whereas the output layer produces the final output. Neurons in the output layer are called output neurons and are as many as the number of output labels to be predicted, denoted by s. More specifically, a neuron is a non-linear function f, also called *activation function*, that, for each input $\mathbf{x}^{(i)}$, computes a weighted sum and produces the output $o^{(i)}$ as

$$o^{(i)} = g\left(\sum_{j=1}^{d} w_j x_j^{(i)} + b\right) \quad i = 1, 2, \ldots, N, \tag{2.3}$$

where the constant b is called the *bias* of the neuron and w_j is the weight corresponding to $x_j^{(i)}$. The most used activation functions are the sigmoid, the tanh, and the ReLU. The sigmoid function $g(z) = (1 + e^{-z})^{-1}$ normalizes the output of each neuron between 0 and 1, which is suitable when the predicted output is, e.g., a probability measure. The tanh function $g(z) = (e^z - e^{-z})/(e^z + e^{-z})$ is a shifted zero-centered version of the sigmoid function which also ranges between -1 to 1. The ReLU is a piecewise linear function defined as $g(z) = \max(0, z)$.

NNs can be fully collected or partially connected according to the structure of the neuron connection in different layers. In the first case, each neuron in one layer is connected to every neuron in the next layer; otherwise, it is partially connected. NNs can be further grouped into *feedforward* and *recurrent* NNs. In a feedforward NN, the connection between the neurons does not form any cycle or loop in the network; thus the information moves only from the input layer to the output layer through the hidden layers. In a recurrent NN, the output of a neuron can, instead, affect subsequent input to the same neuron.

The proposed neural network architecture

In this chapter, we consider a MLP, which is a particular type of fully connected and feedforward NN. It has been demonstrated (Cybenko 1989, Funahashi 1989, Hornik et al. 1989) that a MLP, with at least one hidden layer, can approximate any continuous function with arbitrary accuracy. An MLP architecture consists of an input, an output, and one or more hidden layers. Specifically, the input layer of the proposed NN approach consists of the s sample means Y_{tj} at sample time t for each stream j, the range statistic R_t, and the mean Y_t(Mortell & Runger 1995). The output layer consists of s neurons, corresponding to the s possible streams responsible for the OC alarm.

Without loss of generality, let us consider a single hidden-layer MLP. From Equation (2.3), the output of the mth neuron in the hidden layer associated with $\mathbf{x}^{(i)}$ can be computed as

$$h_m^{(i)} = f\left(\sum_{j=1}^{d} w'_{jm} x_j^{(i)} + b'_m\right) \quad m = 1, 2, \ldots, M, \quad i = 1, 2, \ldots, N, \tag{2.4}$$

where w'_{jm} is the weight for the connection from the mth neuron in the hidden layer and the jth neuron in the input layer, b'_m is the bias of the mth neuron and M is the number of neurons in the hidden layer. Then, the output $o_u^{(i)}$ of the uth neuron, $u = 1, \ldots, s$, in the output layer associated with $\mathbf{x}^{(i)}$ is computed as

$$o_u^{(i)} = f\left(\sum_{m=1}^{M} w_{mu} h_m^{(i)} + b_u\right) \quad u = 1, 2, \ldots, s, \quad i = 1, 2, \ldots, N, \tag{2.5}$$

where w_{mu} are the weights for the connection from the mth output of the neuron in the hidden layer and the uth neuron in the output layer and b_u is the bias of the uth neuron in the output layer. We use a sigmoid activation function for the u th output neuron to predict the probability of the OC uth stream. Thus, the MLP predicts a vector $\mathbf{o}^{(i)} = [o_1^{(i)}, o_2^{(i)}, \ldots, o_s^{(i)}] \in \mathbb{R}^u$, $i = 1, 2, \ldots N$, whose components take a value between 0 and 1. A value larger than 0.5 indicates the corresponding stream has shifted off-target.

The training of an MLP aims to optimize the weights and biases, which represent the NN parameters, hereinafter denoted by θ. The optimum θ^* is chosen as a minimizer of a function C, called cost or loss function, that measures the error between the values predicted by the NN and the true values. In this chapter, we choose as a cost function C the cross-entropy (Goodfellow et al. 2016). The minimization problem is solved with the gradient descent (Goodfellow et al. 2016) and backpropagation (Rumelhart et al. 1986) algorithms that simultaneously update the parameter vector θ^t at each iteration t by using the following equation

$$\theta^t = \theta^{t-1} - \eta \frac{\partial C}{\partial \theta^{t-1}}, \tag{2.6}$$

where $\eta > 0$ is the learning rate (Goodfellow et al. 2016) and $\frac{\partial C}{\partial \theta^{t-1}}$ the partial derivative of the cost function C with respect to NN parameters at iteration $t-1$. All the extra parameters that are not included in θ, namely the numbers of hidden layers and neurons, activation functions, learning rate η, and the number of iterations, are called hyperparameters.

To properly train and design the NN, three non-overlapping data sets are customarily defined: the *training set*, which is the set of data used to estimate the NN parameters θ, i.e. weights and biases, and the relationship between inputs and outputs to be used for predicting the outcome for new unseen inputs; the *validation set*, which is used to tune the hyperparameters; the *test set*, which is used for the final assessment of the fully specified NN. Interested readers are referred to the textbooks of Goodfellow et al. (2016) and Bishop (1995) for a broader discussion.

Performance assessment and comparison

The proposed method is compared with the SCC on each stream $j = 1, \ldots, s$ that is recommended by Meneces et al. (2008) and Ahmadi-Javid & Ebadi (2020) and, for the reasons given in the Introduction, is the only competing method able, as the proposed one, simultaneously identify multiple OC streams. On the jth SCC, Y_{tj} is plotted against the CLs that are computed by simulating 100000 IC MSP samples, according to Equation (2.1), to get a probability of false alarm (Montgomery 2019) $\alpha = 0.05$ (Meneces et al. 2008). In this way, the SCC aims at identifying multiple OC streams by testing separately whether or not the mean of each stream is significantly different from the IC mean.

To train and design the NN, a proper data set is generated to mimic a mean shift of the quality variable Y_{tjk} in all the $\sum_{p=1}^{s} \binom{s}{p}$ OC scenarios arising by $p = 1, 2, \ldots, s$ streams that may shift off-target at given t. The IC streams are simulated as normally distributed with mean μ and standard deviation $\sqrt{\sigma_A^2 + \sigma_e^2}$, according to the MSP model in Equation (2.1). Whereas, the OC streams are simulated as normally distributed with a mean shift size $\Delta\mu = K\sigma_e$, where $K = \{-3, -2, -1, 1, 2, 3\}$. Specifically, we simulate 10000 samples per stream with the corresponding target vector for each OC scenario. Data are split into 70% training and 30% validations sets, and, then, normalized to have zero mean and unit standard deviation. All hyperparameters are selected based on the optimization of the metric defined in Equation (2.7) calculated on the validation set. In particular, through a manual

Table 2.1: Accuracy (%) of the SCC and the proposed NN approach at different sample sizes n for a MSP with $s = 4$, $\sigma_A = 4$, $\sigma_e = 1$, averaged over different numbers $p = 1, 2, \ldots, s$ of OC streams and mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. For each OC scenario, the highest accuracy value is marked in bold.

Sample size n	**NN**	**SCC**
1	**61.34**	8.06
3	**71.24**	21.11
5	**77.15**	32.90

hyperparameter optimization (Goodfellow et al. 2016), we use a MLP with $s + 2$ neurons in the input layer, two hidden layers of 20 and 10 neurons respectively, s neurons in the output layer, and ReLU activation functions for the hidden layers.

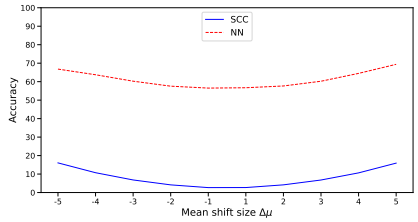
To compare the performance of the proposed approach and the competing method, 100000 samples are generated per stream for each simulated OC scenario. However, unlike training data, the OC streams are shifted with a mean shift size $\Delta\mu = D\sigma_e$, where $D = -5, -4, \ldots, 5$. D is chosen to be larger than K used in the training phase to test performance in a wider range of shifts. The performance is measured in terms of the proportion of correctly predicted labels over the total number of labels (Sorower 2010), which is referred to as *Hamming score* (Godbole & Sarawagi 2004) and is defined as

$$\frac{1}{N_{OC}} \sum_{i=1}^{N_{OC}} \frac{1}{s} \sum_{j=1}^{s} I(\hat{y}_r^{(i)} = y_r^{(i)}), \tag{2.7}$$

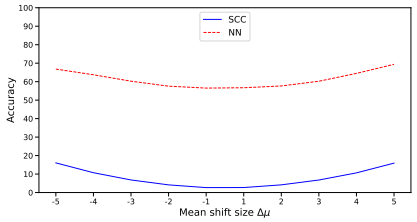
where $\hat{y}_r^{(i)}$ is the predicted target vector, N_{OC} the number of samples identified as OC and $I(\cdot)$ the indicator function

$$I(\hat{y}_r^{(i)} = y_r^{(i)}) = \begin{cases} 1 & \text{if } \hat{y}_r^{(i)} = y_r^{(i)} \\ & \qquad\qquad i = 1, 2, \ldots, N_{OC}, \quad r = 1, 2, \ldots, s. \\ 0 & \text{if } \hat{y}_r^{(i)} \neq y_r^{(i)} \end{cases} \tag{2.8}$$

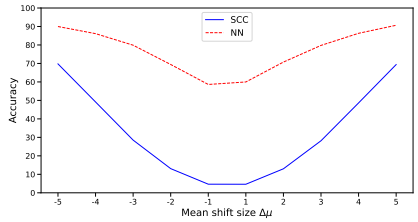
The performance is studied as a function of the number of streams, s, the sample size, n, and the standard deviation ratio, σ_A/σ_e. In the following section, the accuracy is defined as the average value of the Hamming score computed for each scenario in which different number $p = 1, 2, \ldots, s$ of streams are OC. The accuracy (averaged over p) of the SCC and the proposed NN approach is analyzed in Figure 2.2 for a MSP with $s = 4$, $\sigma_A = 4$, $\sigma_e = 1$ and for different sample sizes $n = 1, 3, 5$, considering all mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$ and the number of OC streams $p = 1, 2, \ldots, s$. As expected, the smaller shifts, i.e., the closer to the mean value $\mu = 0$, the lower the accuracy. As an example, the accuracy of the SCC at $n = 5$ and $\Delta\mu = \sigma_e$ is 4.63% which is much smaller than the 59.98% achieved by the proposed NN approach. Whereas, at $\Delta\mu = 5\sigma_e$, the accuracy of the SCC is 69.39%, and that of the proposed approach is 90.66%. The accuracy has a symmetric behavior along with the magnitude of the shifts. The proposed approach performs equally well for large and small shifts. Given $\Delta\mu$, the accuracy is an increasing function of the sample size n. The accuracy of the SCC is generally not large, except when $n = 5$ and $\Delta\mu = -5\sigma_e, -4\sigma_e, 4\sigma_e, 5\sigma_e$, where less than 50% of the simulated OC streams are correctly classified. Table 2.1 reports the



(a)



(b)



(c)

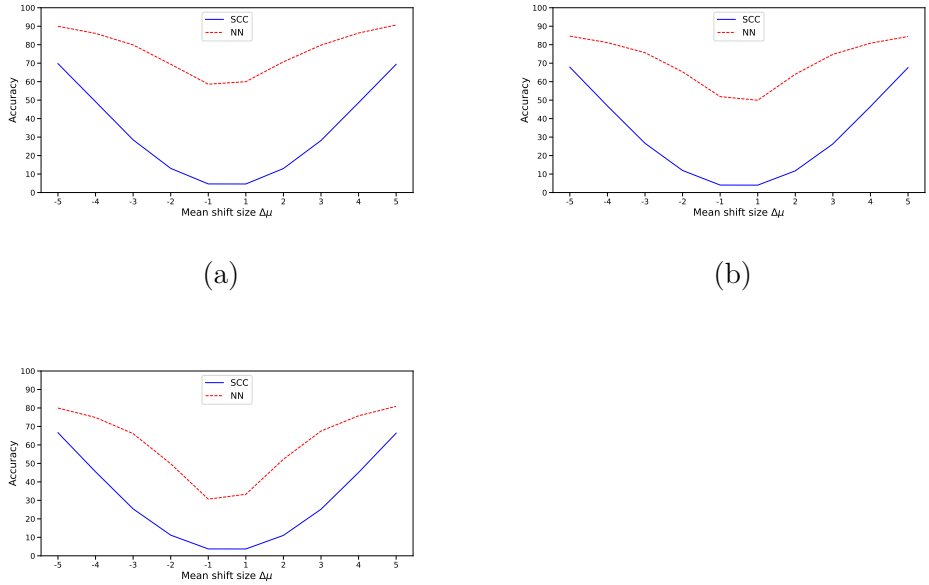
Figure 2.2: Accuracy (%) averaged over p of the SCC for each stream and the proposed NN approach for different sample sizes (a) $n = 1$, (b) $n = 3$, (c) $n = 5$ for a MSP with $s = 4$ streams and standard deviations $\sigma_A = 4$ and $\sigma_e = 1$, at different numbers $p = 1, 2, \ldots, s$ of OC streams with mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$.

Table 2.2: Accuracy (%) of the SCC and the proposed NN approach at different numbers of streams s for a MSP with $n = 5$, $\sigma_A = 4$, and $\sigma_e = 1$, averaged over different numbers $p = 1, 2, \ldots, s$ of OC streams with and mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. For each OC scenario, the highest accuracy value is marked in bold.

Number of streams s	**NN**	**SCC**
4	**77.15**	32.9
6	**71.27**	31.35
8	**61.11**	30.38

accuracy (averaged over p and $\Delta\mu$) of the two competing methods for several sample sizes. It can be seen that the maximum accuracy for both methods is achieved for $n = 5$ and that the proposed NN approach largely outperforms the SCC for each stream.

Figure 2.3 shows the accuracy of the proposed NN approach and the SCC in the OC stream identification for a different number of streams $s = 4, 6, 8$ averaged over all scenarios obtained by varying the number $p = 1, 2, \ldots, s$ of streams shift off-target and the mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. The sample size and the standard deviations are kept constant and set equal to $n = 5$, $\sigma_A = 4$, and $\sigma_e = 1$. The proposed NN approach



(a) (b)

(c)

Figure 2.3: Accuracy (%) averaged over p of the SCC for each stream and the proposed NN approach for a different number of streams (a) $s = 4$, (b) $s = 6$ and (c) $s = 8$ for a MSP with $n = 5$, $\sigma_A = 4$, and $\sigma_e = 1$, at different numbers $p = 1, 2, \ldots, s$ of OC streams with mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$.

Table 2.3: Accuracy (%) of the SCC and the proposed NN approach at different ratios σ_A/σ_e for MSP with $s = 6$, $n = 5$, $\sigma_e = 1$, averaged over different numbers $p = 1, 2, \ldots, s$ of OC streams and mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$. For each OC scenario, the highest accuracy value is marked in bold.

Process standard deviation σ_A	**NN**	**SCC**
2	**81.90**	60.96
4	**71.27**	26.77
6	**63.65**	17.02
8	**61.95**	11.11

outperforms the SCC in any simulated scenarios for any value of n and $\Delta\mu$. Table 2.2 reports the accuracy as a function of the number of streams s. The accuracy decreases with s, as the combinations of OC streams to be recognized increase.

Alternatively, in Figure 2.4 and Table 2.3, the accuracy is studied for different ratios of the common and within-stream standard deviation, $\sigma_A/\sigma_e = 2, 4, 6, 8$, averaged over the number $p = 1, 2, \ldots, s$ of OC streams and mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$, by keeping fixed the number of streams $s = 6$ and sample size $n = 5$. The accuracy of the

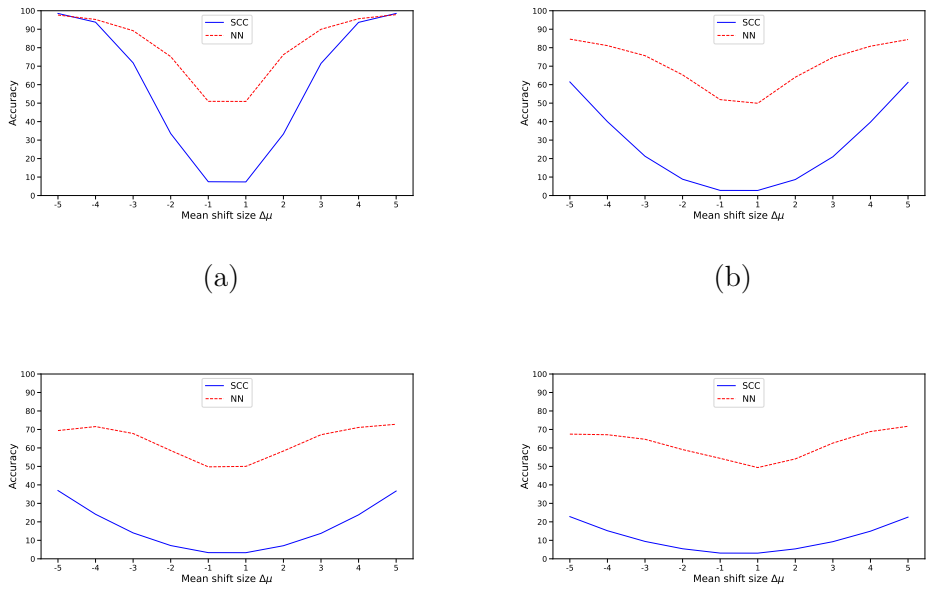


Figure 2.4: Accuracy (%) averaged over p of the SCC for each stream and the proposed NN approach for different common variations (a) $\sigma_A = 2$, (b) $\sigma_A = 4$, (c) $\sigma_A = 6$ and (d) $\sigma_A = 8$ for a MSP with $s = 6$, $n = 5$, $\sigma_e = 1$, at different numbers $p = 1, 2, \ldots, s$ of OC streams with mean shift size $\Delta\mu = -5\sigma_e, -4\sigma_e, \ldots, 5\sigma_e$.).

SCC decreases with σ_A/σ_e, i.e., when the common standard deviation gets larger than the within-stream one, because the width of control intervals is proportional to the overall process variation $\sigma_A^2 + \sigma_e^2$, making a control chart based on the quality variable Y_{tjk} quite insensitive to shifts in one or a few streams. As anticipated in the Introduction, like the SCC, Hotelling's T^2 control chart also fails in detecting this type of OC state as the variability of the plotted statistic still depends on σ_A^2 and is less powerful to detect a mean shift that does not affect all streams.

It should be noted that the correlation among streams slightly affects the performance of the proposed NN approach that achieves the largest accuracy in all simulated scenarios. For example, when $\sigma_A/\sigma_e = 8$ and $\Delta\mu = 3\sigma_e$, the accuracy achieved by the SCC is only 9.27%, whereas the proposed NN approach achieves 62.62%, i.e., a difference of about 53.35%.

To investigate how the accuracy is affected by the number of streams $p = 1, \ldots, s$ that shift off-target, we introduce the Exact Match ratio (EM) (Sorower 2010), defined as the percentage of samples that have all their labels correctly classified

$$\text{EM} = \frac{1}{N_{OC}} \sum_{i=1}^{N_{OC}} I(\mathbf{y}^{(i)} = \hat{\mathbf{y}}^{(i)}). \tag{2.9}$$

Table 2.4: EM (%) of the SCC and the proposed NN approach for a MSP with $s = 6$ streams, sample size $n = 5$, standard deviations $\sigma_A = 2$ and $\sigma_e = 1$, at different number $p = 1, 2, .., s$ of OC streams with a positive mean shift size $\Delta\mu = \sigma_e, 2\sigma_e, 3\sigma_e, 4\sigma_e, 5\sigma_e$. For each OC scenario, the largest EM value is marked in bold.

	SCC $\Delta\mu$					NN $\Delta\mu$				
	$1\sigma_e$	$2\sigma_e$	$3\sigma_e$	$4\sigma_e$	$5\sigma_e$	$1\sigma_e$	$2\sigma_e$	$3\sigma_e$	$4\sigma_e$	$5\sigma_e$
p										
1	5.86	31.52	69.62	**91.45**	**95.24**	**13.75**	**51.83**	**74.71**	86.44	92.16
2	2.67	22.41	61.42	89.36	95.63	**16.34**	**65.86**	**86.15**	**93.44**	**96.41**
3	1.77	18.5	56.98	87.83	96.08	**22.5**	**68.85**	**88.8**	**95.95**	**98.51**
4	1.48	16.44	53.98	86.78	96.62	**17.11**	**61.61**	**89**	**97.21**	**99.37**
5	1.49	15.16	52.05	86.12	97.32	**13.9**	**45.4**	**78.17**	**94.38**	**98.89**
6	1.78	14.7	50.91	**86.02**	**98.36**	**7.19**	**23.28**	**50.34**	75.22	89.94

EM is a more strict measure than accuracy as the latter overlooks the difference between completely and partially incorrect predictions, and measures the capability in the identification of *all* the OC streams. The accuracy is weighting positively also cases in which not all OC streams are correctly identified. If, for example, only half of the OC streams are identified the accuracy assigns 0.5 whereas EM assigns 0. This means that the accuracy increases on average with p and the probability of identifying at least one of the shifted p streams. Table 2.4 reports the EM values achieved by the proposed NN approach and SCC for different $p = 1, 2, \ldots, s$ and mean shift size $\Delta\mu = \sigma_e, 2\sigma_e, 3\sigma_e, 4\sigma_e, 5\sigma_e$, with $n = 5$, $\sigma_A = 2$ and $\sigma_e = 1$. The proposed NN approach is shown to outperform the SCC also in terms of EM in most cases, and, as expected, the EM turns out to be affected by the number of shifted streams out of s and the magnitude of the mean shift. For example, for $p = 1$ and $\Delta\mu = 2\sigma_e$, the EM of the SCC and the proposed NN approach are 31.52% and 51.83%, respectively, whereas, for $p = 6$, are equal to 14.7% and 23.28%, respectively.

2.3 A case study: identification of out-of-control streams from passenger train HVAC systems

Rail transportation in Europe has gained popularity as a potential alternative to other means of transport and has given rise to competition between operators to guarantee passenger satisfaction. In this regard, the thermal comfort of train passengers has become of paramount importance, especially for long trips, and is generally ensured through the regulation of the HVAC systems installed on each train coach. Over the past few years, European regulations (EN 2006) have been settled for operational standards in controlling air temperature and the comfort level of passenger train coaches. This stimulated railway companies to install onboard sensor systems and automatically gather massive amounts of observational data with the ultimate goal of monitoring the proper functioning of HVAC systems. The multiple data streams produced by the different HVAC systems installed on different coaches of the same train can be regarded as produced by a MSP (Lepore, Palumbo & Sposito 2022*a*). Whatever the scheme employed to monitor the MSP, the crucial step is the efficient and possibly automatic identification of abnormal streams. Given an OC alarm, the automatic and simultaneous identification of the faulty HVAC systems is crucial to optimize maintenance and reduce the relative costs to ensure passenger thermal comfort onboard. Based on the `HVAC` data mentioned in the Introduction, the main results of this case study are presented

Table 2.5: Operational variables measured for each of the $s = 6$ train coaches.

Variable Name	Symbol	Description
coach_j_InteriorTemp	T_{in}	Temperature inside coach
coach_j_OutsideTemp	T_{out}	Outside temperature
coach_j_SetpointTemp	T_{set}	Setpoint temperature set by European regulations (EN 2006)
coach_j_SupplyTemp	T_{supply}	Air temperature provided by the HVAC
coach_j_DeltaTemp	ΔT	Difference between the interior temperature and the outside temperature
Timestamp		Date
Vehicle		Train name

Note: coach_j denotes the jth train coach for $j = 1, 2, \ldots, 6$.

and discussed in Section, as well as the practical applicability and the superiority of the proposed NN approach with respect to the SCC, which remains the only competitor for the OC stream identification.

HVAC and variable description

The available variables used to describe the HVAC operating conditions for each of the six coaches are summarized in Table 2.5 as they are coded in the `HVAC` data set. HVAC systems are used to ensure interior air quality and thermal comfort through a combination of ventilation, heating, and cooling operations. Ventilation is the process of exchanging inside with outside air and involves the removal of smoke, dust and bacteria to provide better interior air quality. Heating and air conditioning instead generate heat and cooling for the train coaches, respectively. To guarantee the passenger thermal comfort throughout the range of the environmental and climatic conditions required by European regulations (EN 2006), a central unit is installed to regulate the heating and cooling mode of the HVAC systems by means of temperature sensors used to measure changes in the readings of the outside (T_{out}) and interior (T_{in}) temperatures for each coach. These regulations prescribe the setpoint temperature (T_{set}), which is the temperature at which each HVAC system aims, as a function of T_{out} at each time instant, and the difference $\Delta T = T_{\text{in}} - T_{\text{set}}$ to be no larger than 2°C. The latter quantity at each time instant describes any deviation of the interior temperature from the corresponding setpoint temperature and therefore is used as the quality variable for OC HVAC system identification purposes. When an HVAC system is no longer able to perform its designed function, T_{in} moves away from the setpoint temperature. Therefore, the ΔT sensor signals coming from each HVAC data acquisition system can be regarded as a MSP with $s = 6$ streams and $n = 5$. That is, with reference to Equation (2.1), Y_{tjk} denotes the kth ΔT measurement from the jth coach at each time instant t.

Results

`HVAC` data used in this study refer to operational data collected every two minutes from HVAC systems installed onboard two different passenger 6-coach trains, named train A and train B. Train names and the voyage year have been omitted for confidentiality reasons. To estimate the MSP model parameters, namely μ, σ_A^2 and σ_e^2, $m = 160$ IC reference samples of size $n = 1$ are collected from each coach of the train A. The common variability, σ_A^2, and the within-stream variability, σ_e^2, are estimated from the train A data using standard one-way analysis of variance techniques (Woodall & Thomas 1995) through $\hat{\sigma}_A^2$ and $\hat{\sigma}_e^2$ obtained as

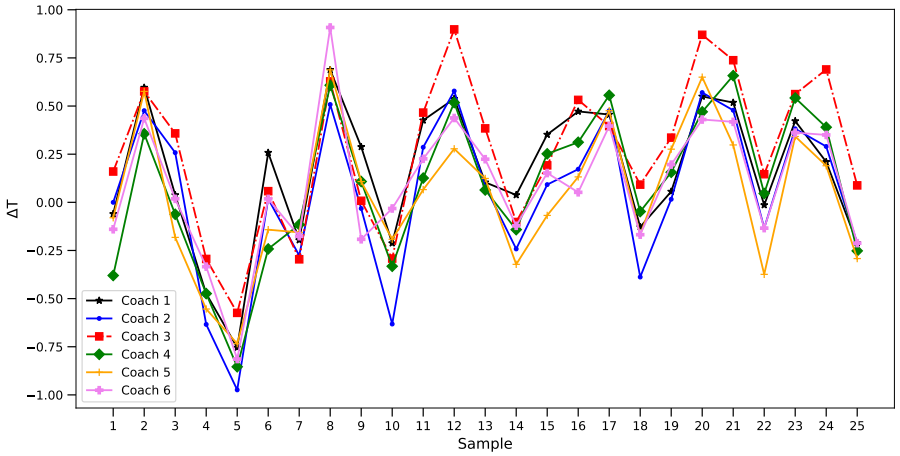


Figure 2.5: IC ΔT stream for each coach of train A over 25 samples collected on the 1st of July. The streams show similar behavior and seem to be IC. There is graphical evidence that the common variability, σ_A^2, is larger than the within-stream variability, σ_e^2.

follows

$$\hat{\sigma_e}^2 = \frac{\sum_{i=1}^{m}\sum_{j=1}^{s}(Y_{tj} - Y_t)^2}{m(s-1)} \tag{2.10}$$

and

$$\hat{\sigma_A}^2 = \frac{\sum_{i=1}^{m}(Y_t - Y)^2}{m-1} - \hat{\sigma_e}^2/s, \tag{2.11}$$

that yield $\hat{\sigma_A} = 0.85$ and $\hat{\sigma_e} = 0.33$. Note that $\hat{\sigma_A}$ is about two and a half times $\hat{\sigma_e}$. The proposed NN approach is demonstrated in the simulation study to outperform the competing SCC both when $\sigma_A/\sigma_e = 2$ and $\sigma_A/\sigma_e = 3$. As an example, 25 samples of size $n = 1$ of the ΔT signals coming from each coach of train A are plotted in Figure 2.5. The streams show similar behavior and seem to be IC. If all the HVAC systems can ensure thermal comfort in each passenger rail coach, the interior temperature T_{in} is expected to be equal to the setpoint temperature T_{set}, and the ΔT to be zero as appears for train A data. Furthermore, there is graphical evidence that the common variability is larger than the within-stream variability, as already highlighted by the computed values of σ_A^2 and σ_e^2. Although data are collected every 2 minutes, they are made available for analysis only every 10 minutes, thus the natural sample size is $n = 5$ as it corresponds to 10-minute-worth-of data. Therefore, to monitor the ΔT signals, the overall mean and range control charts (Mortell & Runger 1995) are designed with $n = 5$ and $\alpha = 0.05$. However, any other MSP control charts can be used The proposed NN approach is independent of the specific control chart employed to monitor the process as the NN is trained on true OC samples from a MSP. As illustrated in Section 2.2, a data set can be simulated from a MSP with $s = 6$, $n = 5$, $\sigma_A = 0.85$ and $\sigma_e = 0.33$ to effectively train and design the NN that will be used for the identification of the OC HVAC systems. The NN has 8 input neurons, corresponding to the 6 ΔT signals collected from each of the six coaches plus the two summary statistics, i.e., the overall mean and the range, and 6 output neurons that correspond to the 6 possible HVAC systems that can shift off-target.

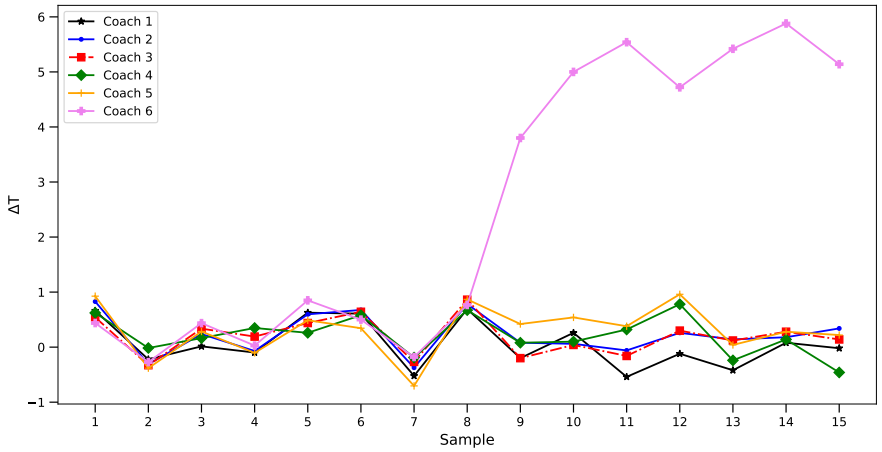


Figure 2.6: ΔT stream for each coach of train B over 15 samples, collected on the 8th of August only, from which we know that a fault occurred. The process is OC and an assignable cause is known to affect the output coach 6 from sample 8.

According to Section 2.2, a MLP with two hidden layers of 20 and 10 neurons, respectively, is trained with the gradient descent and backpropagation algorithms (Goodfellow et al. 2016) as in Equation (2.6). The chosen cost function is the cross entropy and the chosen activation function in the output layer is the sigmoid function. To demonstrate the capability of the proposed method of correctly identifying the HVAC systems that have shifted off-target, we use data from train B, collected on the 8th of August only, from which we know that a fault occurred. Figure 2.6 shows the ΔT signals from train B over 15 samples, where an assignable cause is known to affect the output of coach 6 from sample 8. Figures 2.7 show the range and overall mean control charts, respectively. The red horizontal line indicates the CLs at $\alpha = 0.05$. The range control chart starts signaling an OC state from sample 9. The value of the ΔT measurements of that sample together with the information about the range and the overall mean are processed by the trained NN. The predicted output is $\mathbf{o} = [0.012, 0.058, 0, 0.004, 0.019, 0.997]$. The use of the sigmoid activation function in the output layer allows the NN to predict, for any output neurons, a value between 0 and 1 that can be interpreted as the probability that a neuron and so the corresponding stream is responsible for the OC alarm. The streams that shifted off-target can be easily identified starting from $\mathbf{o}$ as those neurons whose output is larger than 0.5. The proposed NN approach is demonstrated to successfully recognize stream 6 as the stream responsible for the OC alarm.
Although, in this case study, it is possible to easily identify the OC stream by eye, it is worth stressing that the proposed NN approach is not limited to the data set hereby investigated and could go far beyond this specific application, as demonstrated by its superiority compared to the competitor in the more complex simulated scenarios.

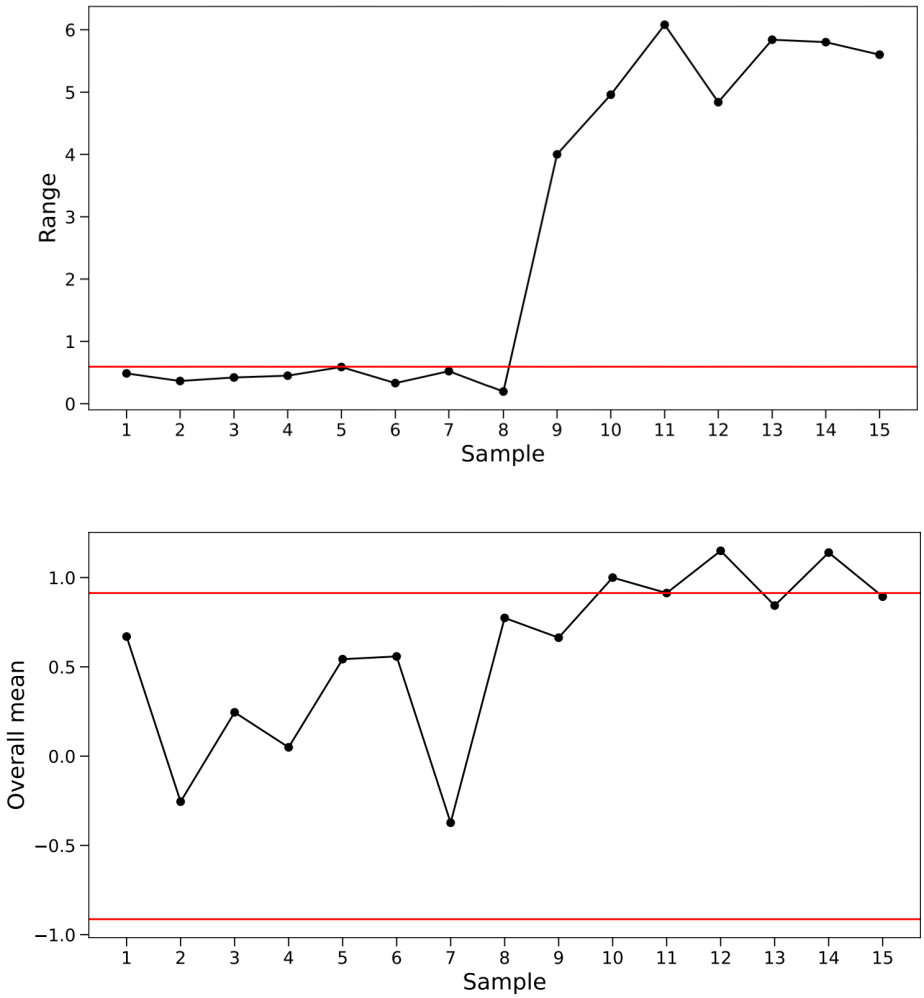


Figure 2.7: Range and overall mean control charts for the monitoring of the HVAC signals collected from each coach of train B. The red horizontal line indicates the CLs at $\alpha = 0.05$. The range control chart starts signaling from sample 9.

2.4 Conclusions

A MLP NN is employed for the first time to simultaneously and automatically identify how many and which streams have shifted from the IC state in a MSP, given an OC alarm issued by a control chart used for the monitoring of the MSP. This problem, referred to as OC stream identification, has been thoroughly studied for the first time in this chapter in terms of the correct classification of the shifted streams. By means of a simulation study, the proposed NN approach has proven to have better performance with respect to those achieved by using a SCC for each stream, which appeared in the literature as the only actual competitor when, as it often occurs in practice, more than one stream shifts off-target and the independence assumption between streams is unrealistic. In particular, the latter shows very poor performance when the common variability σ_A is large relative to the within-stream variation σ_e, while the accuracy of the NN has shown to be competitive at different sample sizes n, different number of streams s and different values of the ratio σ_A/σ_e. A discussion on the misuse of residuals, which are insensitive to the cross-correlation between streams, for the OC stream identification is given in the supplementary material. The practical applicability of the proposed approach has been illustrated by means of real operational data to address the issue of identifying faulty HVAC systems installed onboard railway passenger coaches. The validity of the proposal is obviously not limited to the case study here presented and can be extended to any MSP. Future research can further enhance the OC stream identification by leveraging information contained not only in the current OC sample but also in a given sample of past observations before the OC alarm.

2.5 Supplementary Material

The use of residuals for the identification of OC streams

From Equation (2.1), it is trivial to observe that when the common variability σ_A^2 is large with respect to the within-stream variability σ_e^2, a control chart based on the quality variable Y_{tjk} is quite insensitive to detect a shift in the mean of one or a few streams because the width of control intervals is proportional to the overall process variation $\sigma_A^2 + \sigma_e^2$. To overcome this issue, Mortell & Runger (1995) defined the residuals X_{tjk},

$$X_{tjk} = Y_{tjk} - Y_t \quad t = 1, 2, \ldots, T, \quad j = 1, 2, \ldots, s, \quad k = 1, 2, \ldots, n, \tag{2.12}$$

as the difference between the value of the kth observation from the jth stream at time t and the average value over all the s steams and the n observations at time t. From Equation (2.12), the standard deviation of the residuals X_{tjk} is $\sqrt{\frac{s-1}{s}\sigma_e^2}$ and the cross-correlation between the residuals X_{tjk} and $X_{tj'k}$ between the jth and j'th streams at a given subgroup point k and sample time t is $-\frac{1}{s-1}$ (Lepore, Palumbo & Sposito 2022*a*). However, such a cross-correlation is usually neglected for $s > 3$ streams (Epprecht et al. 2011). Mortell's idea is based on the observation that the standard deviation of X_{tjk} is independent of σ_A, and thus a control chart based on the residuals X_{tjk} in place of Y_{tjk} is more sensitive to assignable causes that may affect only a few streams. Along this line, Epprecht et al. (2011) proposed the *residual* group control chart to enhance the monitoring of shifts in only one stream. However, even though they claimed that the proposed scheme may be also used for the identification of the shifted stream, this is not true when a large mean shift occurs. As shown in Figure 2.8, when only one stream shifts off-target with a large mean shift size, say

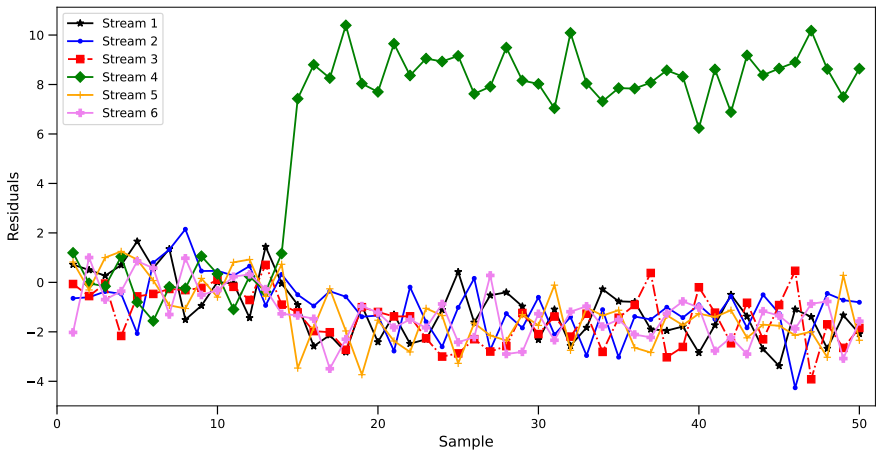


Figure 2.8: Time-series plot of the residuals from an OC MSP with $s = 6$, $\mu = 0$, $\sigma_e = 1$, $\sigma_A = 4$, $n = 1$ and an upward shift of magnitude $10\sigma_e$ in the 4th stream from sample 15.

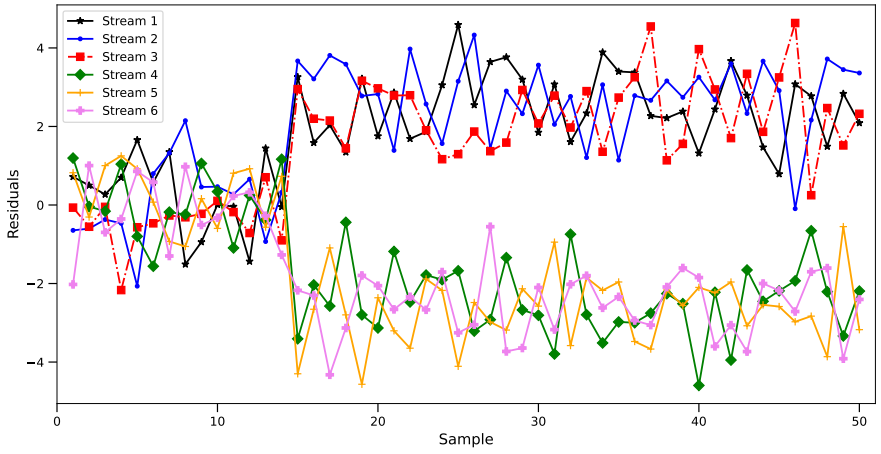


Figure 2.9: Time-series plot of the residuals from an OC MSP with $s = 6$, $\mu = 0$, $\sigma_e = 1$, $\sigma_A = 4$, $n = 1$ and an upward shift of magnitude $5\sigma_e$ in the first four streams from sample 15.

$\Delta\mu = 10\sigma_e$, the residuals of the OC stream tend to separate from the others so that the IC streams show extreme residual values. In such a case, the *residual* group control chart would plot the maximum value of the OC stream and the minimum value of one of the IC streams beyond the CLs, indicating both of them as responsible for the OC alarm. Furthermore, this problem is even more evident when more than one stream is OC, as shown in Figure 2.9, where an OC MSP is simulated from Equation (2.1) with $s = 6$, $\mu = 0$, $\sigma_e = 1$, $\sigma_A = 4$, $n = 1$ and a shift of magnitude $5\sigma_e$ applied to the mean of the first four streams starting from sample 15. The residuals tend to be divided into two groups. This is reasonable if one observes that X_{tjk} sum up to zero by construction for each t, i.e., $X_t = 0$. This should convince one that the idea of building a SCC for each stream based on X_{tjk} in place of Y_{tjk} is useless because, when more than half of the stream is OC, each control chart would raise an alarm and all streams would wrongly be identified as responsible for the OC alarm.

Chapter 3

Neural network for the statistical process control of HVAC systems in passenger rail vehicles

In the rail industry, coach temperature regulation has become a crucial task to improve passenger thermal comfort. Over the past few years, European standards have required rail operators to implement monitoring systems for the control of the HVAC of passenger rail vehicles. These systems, based on modern automated sensing technologies, have created new data-rich scenarios and call for new methods to deal with high-dimensional, high-correlated, and heterogeneous data. In this chapter, an AE, which is a particular type of NN developed to model unlabelled data and automatically extract significant features, is utilized to develop a nonparametric process monitoring approach. Two control charts based on statistics H^2 and SPE are built in the feature space and the residual space, respectively. Through operational HVAC data collected onboard passenger vehicles, the proposed approach is shown to be capable of simultaneously monitoring and detecting anomalies that may have occurred in the data streams acquired from each train coach, even though it is not limited to the application hereby investigated.

3.1 Introduction

In the rail industry, efficient temperature regulation is becoming a necessity in the face of strong competition and overcrowded carriages. One of the challenging aspects in this regard is the passenger thermal comfort of rail coaches, especially for long journeys. Over the past few years, new European standards have been developed, such as UNI EN 14750 (EN 2006), and prescribe requirements for controlling the air temperature, relative humidity and air speed within passenger compartments. In order to meet these standards, railway companies have been automatically collecting and storing data for the monitoring of HVAC systems installed onboard modern trains.

During the working life of a system, different kinds of faults may occur and compromise the originally designed functions. The timely identification of faults in a complex system is in fact a crucial task to ensure operation safety and quality as well as to reduce maintenance and operation costs. The challenge is to turn the collected high dimensional, correlated and heterogeneous data, also possibly affected by noise and environmental fluctuations,

into value. The ongoing digitalization creates in fact a new dimension in the diagnosis, maintenance and operation of these systems, in a new cost-effective and efficient manner. In this modern data-rich and computationally efficient environment, classical SPC (Montgomery 2019) and ML methods (James et al. 2013) interplay without sharp boundaries. Traditional methods, no matter if based on model-driven (Venkatasubramanian et al. 2003) or data-driven (Bersimis et al. 2007) approaches, may sometimes achieve poor performance, due to the increasing complexity of the data produced by modern industrial processes. As a result, alternative monitoring methods based on NNs, which in the past decades have been regarded as unaffordable because computationally expensive and large data demanding, have recently received great attention (Zhang et al. 2019, Lee et al. 2019, Lu et al. 2017) also in the field of SPC (Zorriassatine & Tannock 1998, Psarakis 2011).

In particular, the class of supervised ML methods (James et al. 2013) are naturally prone to automatically identify normal and OC system behaviors but require to be trained based on large labeled data sets with, ideally, the same number of data for each behavior. Unfortunately, this is not feasible in practice, because systems generally work under normal conditions for the most part of their life and faults are generally very rare, making real data sets unbalanced in this sense. On the other hand, unsupervised ML approaches do not require data with labels or additional information but have the drawback of showing lower performances for fault detection and classification. Hence, in this perspective, we explore the use of AEs (Goodfellow et al. 2016), which are a particular type of NN capable of extracting from unlabelled data significant features that are not necessarily linearly related to the original inputs. AE output will be exploited to be used in a nonparametric SPC charting scheme and mitigate the drawbacks of both supervised and unsupervised ML methods. In this way, we possibly avoid artificially changing normal system operating conditions to produce OC behaviors, and we can model the raw data signals directly, without the need for arbitrarily selecting and extracting problem-specific features based on past experience, if any.

We compare the proposed monitoring strategy with state-of-the-art feature extraction methods present in the literature by means of simulated data in terms of Fault Detection Rate (FDR) to demonstrate its effectiveness and applicability. The numerical results demonstrate that our proposed method outperforms the competitors.

The rest of the chapter is organized as follows. In Section 3.2, the proposed approach is introduced. In Section 3.3, it is applied to the operational data acquired for the monitoring of HVAC systems installed onboard passenger railway vehicles, and made available by the rail transport company Hitachi Rail STS. Finally, in Section 3.4, theoretical and practical conclusions are drawn.

3.2 The proposed approach

In the following subsections, the proposed SPC approach is introduced by briefly framing NNs and, in particular, AEs. We then present how to use the typical AE output for the perspective monitoring of new data points, usually referred to as Phase II in the SPC literature. It is then worth readily stating that, to start Phase II, we must preliminarily identify a clean set of observations that represents the process's normal operating conditions and will be hereinafter referred to as *reference set* or *training set*. The identification of the training set is called Phase I, and involves cyclically applying offline the following steps: (i) the training of the AE and hyperparameter optimisation on an initial, as large as possible, set of observations; (ii) the application of the SPC approach based on the AE output; (iii) the recognition of points that falls out the CLs; (iv) the possible elimination of outlying points marked as exceptional

by domain experts; (v) and the recalculation of the CLs (Montgomery 2019). For conciseness, and without loss of generality, we assume in the following subsections that a training set has been already identified and available. As a standard practice in NN literature, step (v) is performed without repeating step (i), and, in particular, hyperparameter optimization in step (i) is performed by means of a validation set or k-fold cross-validation approach (James et al. 2013) on the training set. The training set consists of N vectors denoted by $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, N$.

Artificial neural networks

A NN is a brain-inspired system, as the adjective *neural* suggests, with the goal of replicating the human learning process. NNs are usually introduced as advanced algorithms capable of extracting complex and nonlinear relationships among variables and are a popular tool used for learning and visualization problems, such as computer vision, speech recognition, natural language processing and function approximation. A NN consists of a collection of interconnected processing elements, called *neurons*, which loosely mimic the biological neuron in a human brain. Each neuron receives input from other neurons, then processes and transmits it to the neurons connected to it. Typically neurons are organized in *layers*, which are divided into three types: *input*, *hidden* and *output*. The input layer is used to bring the data to the network without performing any computations and passing the information to the hidden layer. The hidden layer, placed between input and output layers, is responsible for the relationship mapping among input variables in the model. More than one hidden layer can exist, depending on the complexity of the training set. The output layer, in turn, takes inputs from the hidden layer and returns the final output. Based on the type of connections among neurons, two successive layers are called *fully connected*, if each neuron of a layer is connected to each neuron in the successive layer, or *pooling*, if a group of neurons in one layer is connected to a single neuron in the successive layer. NNs with these types of connections are called *feedforward NN*, and the information moves only in one direction from the input layer to the output layer without forming any cycle. When feedforward NNs are extended to include connections between neurons in the same or previous layers, they are called *recurrent NNs*.

Autoencoder

An AE is a type of NN typically employed for dimensionality reduction and feature extraction to copy the input to the output (Goodfellow et al. 2016), possibly getting the significant aspects of the data to be copied. It usually consists of two parts: an encoder and a decoder. The former maps each input vector $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, N$, into $\mathbf{h}_i = f(\mathbf{x}_i) \in \mathbb{R}^{d'}$, called code, or latent, representation. Whereas, the latter maps $\mathbf{h}_i$ back to a reconstructed vector or output in the input space denoted by $\mathbf{x}'_i = g(\mathbf{h}_i) = g(f(\mathbf{x}_i)) \in \mathbb{R}^d$. AEs have one input layer, one output layer, and an appropriate number of hidden layers with an optimal number of neurons. In particular, the decoder and encoder layers are organized symmetrically and the output layer must have the same number of neurons as the input layer, since the purpose is to characterize the input itself. However, perfectly copying the input to the output may be useless, whereas the goal is to summarise, into the code layer only the input properties that are important for the problem at hand and discard the others. To achieve this, one may force $\mathbf{h}_i$ to have a dimension that is smaller than that of the input. An AE whose architecture is subject to this constraint is called *undercomplete*. Learning an undercomplete representation

forces the AE to separate important information from noise. If the hidden layer dimension is larger than or equal to that of the input, the AE is called *overcomplete* and can merely mimic the identity function. There exist various strategies for preventing this (Goodfellow et al. 2016). The proposed approach is based on an undercomplete, feedforward, and fully connected AE with only one hidden layer, for which $\mathbf{h}_i$ is computed for each input $\mathbf{x}_i$ as follows

$$\mathbf{h}_i = \sigma(\mathbf{W}\mathbf{x}_i + \mathbf{b}). \tag{3.1}$$

The function $\sigma(\cdot)$ is called the *activation function*. $\mathbf{W}$ and $\mathbf{b}$ are the $d' \times d$ *weight matrix* and the *bias vector*, respectively, and are referred to the neuron connections between the input layer and hidden layer. The most common activation functions are the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$, the tanh $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, the ReLU $\sigma(z) = \max(0, z)$ and the Exponential Linear Unit (ELU), which returns the input itself if it is positive, and $\gamma(e^z - 1)$ otherwise, $\gamma \in [0, 1]$. Then, $\mathbf{x}'_i$ can be accordingly computed as for each $i = 1, \ldots, N$

$$\mathbf{x}'_i = \sigma'(\mathbf{W}'\mathbf{h}_i + \mathbf{b}'), \tag{3.2}$$

where $\mathbf{W}'$ and $\sigma'(\cdot)$ are the $d \times d'$ weight matrix and the activation function of the decoder, respectively, and $\mathbf{b}'$ is the bias vector referred to the neuron connections between the input layer and hidden layer. AE training (step (i)) is performed by minimizing the MSE of all reconstruction errors $\mathbf{e}_i$ between input $\mathbf{x}_i$ and its reconstruction $\mathbf{x}'_i$. The objective function to be minimized is defined as follows

$$\mathrm{J}(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i, g(f(\mathbf{x}_i))) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{x}'_i), \tag{3.3}$$

where $\theta = \{\mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}'\}$ denotes the training parameter vector, $L(\cdot, \cdot)$ is referred to as the *loss function* and penalises $\mathbf{x}'_i$ for being dissimilar from the input vector $\mathbf{x}_i$, i.e., it is increasing with the reconstruction error $\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}'_i$. In this chapter, for each i $L(\mathbf{x}_i, \mathbf{x}'_i)$ is assumed to be the sum of squared errors and therefore, it can be explicitly written as the squared L_2-norm of $\mathbf{e}_i$

$$\begin{aligned} L(\mathbf{x}_i, \mathbf{x}'_i) &= \|\mathbf{x}_i - \mathbf{x}'_i\|^2 = \|\mathbf{e}_i\|^2 \\ &= \|\mathbf{x}_i - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x}_i + \mathbf{b})) + \mathbf{b}'\|^2. \end{aligned} \tag{3.4}$$

Coherently, we denote by $\theta_{\mathbf{o}} = \{\mathbf{W}_{\mathbf{o}}, \mathbf{W}'_{\mathbf{o}}, \mathbf{b}_{\mathbf{o}}, \mathbf{b}'_{\mathbf{o}}\}$ the optimal parameter vector that results from the minimization of the objective function obtained from Equation (3.3) and (3.4), as follows

$$\theta_{\mathbf{o}} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{x}_i - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x}_i + \mathbf{b})) + \mathbf{b}'\|^2. \tag{3.5}$$

This problem can be solved by means of the backpropagation algorithm (Rumelhart et al. 1986), where the gradient descent approach (Goodfellow et al. 2016) is used to update the parameter vector θ at each iteration t as follows

$$\theta^t = \theta^{t-1} - \eta(\vec{\boldsymbol{\nabla}}_\theta L(\theta^{t-1})), \tag{3.6}$$

where η is called the learning rate and $\vec{\boldsymbol{\nabla}}_\theta L(\theta^{t-1})$ is the gradient of the cost function for the weight matrices and the bias vectors at iteration $t-1$ for each layer. The weight matrices and the bias vectors are initialized through the Xavier initialization algorithm (Glorot &

Algorithm 1 AE training algorithm

Input: Reference set
Output: encoder and decoder NNs
Initialise the parameter vector θ by using Xavier initialisation
repeat
  Sample a minibatch from the reference set
  Compute the loss function $L(\mathbf{x}_i, g(f(\mathbf{x}_i)))$
  Update θ by gradient descent approach
until the early stopping criterion does not meet

Bengio 2010) and updated until an early stopping criterion (Goodfellow et al. 2016, Prechelt 2002) is met. The pseudocode of the AE training algorithm is given in Algorithm 1.

When the encoder and decoder have linear activation functions and the MSE is used as a loss function, an undercomplete AE performs the well-known PCA (Baldi & Hornik 1989). Conversely, when a nonlinear activation function is used, AEs may learn to span a different subspace from that identified by PCA (Jolliffe 2005). In this perspective, AEs generalize the PCA and are expected to achieve better performance in latent feature extraction, projection, and classification (Japkowicz et al. 2000).

It can be proved for example that a NN with one hidden layer can approximate any continuous function from the input patterns to the output patterns with an arbitrary degree of accuracy (Goodfellow et al. 2016), provided that there are enough neurons in the hidden layer. Anyway, the selection of a proper number of neurons in the hidden layer, the type of activation function, the learning rate, and the batch size is a delicate task belonging to the hyperparameter optimization of step (i).

Construction and control limit estimation of the monitoring statistics

The output of the AE from step (i) can be embedded in a control charting scheme and enhance a yet familiar SPC tool for practitioners. Implementation of process monitoring based on AEs is similar to those based on PCA (Jackson & Mudholkar 1979). Specifically, the monitoring statistics H^2 (Yan et al. 2016) and the SPE (MacGregor & Kourti 1995) are developed to improve the identification of possible faults that may have occurred in the process. For each input $\mathbf{x}_i$, the H^2 statistic (Yan et al. 2016) is defined in the feature space as the squared L_2-norm of the hidden representation $\mathbf{h}_i$ obtained from Equation (3.1), that is

$$H_i^2 = \mathbf{h}_i^\top \mathbf{h}_i = (\sigma(\mathbf{W}\mathbf{x}_i + \mathbf{b}))^\top \sigma(\mathbf{W}\mathbf{x}_i + \mathbf{b}), \quad i = 1, \ldots, N. \tag{3.7}$$

The H^2 statistic is the square distance of the projection of the input $\mathbf{x}_i$ from the origin of the feature space (MacGregor & Kourti 1995). Whereas, the SPE statistic (MacGregor & Kourti 1995) is defined in the residual space as the squared L_2-norm of the reconstruction error, explicitly written upon using Equation (3.2) as follows

$$\begin{aligned} SPE_i &= \mathbf{e}_i^\top \mathbf{e}_i = (\mathbf{x}_i - \mathbf{x}_i')^\top (\mathbf{x}_i - \mathbf{x}_i') \\ &= (\mathbf{x}_i - \sigma'(\mathbf{W}'\mathbf{h}_i + \mathbf{b}'))^\top (\mathbf{x}_i - \sigma'(\mathbf{W}'\mathbf{h}_i + \mathbf{b}')), \quad i = 1, \ldots, N. \end{aligned} \tag{3.8}$$

The SPE statistic measures how close the input $\mathbf{x}_i$ is to the residual space by the squared perpendicular distance of $\mathbf{x}_i$ from the feature space (MacGregor & Kourti 1995). These two monitoring statistics can be then plotted against a time index and form two different control

charts named H^2 and SPE control charts. In particular, the H^2 control chart monitors the variation inside the feature space spanned by the feature extracted by the AE, whereas changes along directions orthogonal to the latter space are monitored by the SPE control chart, which detects the occurrence of a fault that causes the input to move away from the feature space defined by the reference model. Therefore, the value of the H^2 statistic may be affected by a type of fault that does not increase the value of the SPE statistic, or viceversa. The UCLs of the H^2 and SPE control charts, denoted by H^2_{lim} and SPE_{lim} respectively, are estimated using the block bootstrap approach (Lahiri 1999, Bühlmann & Künsch 1999, Härdle et al. 2003, Phaladiganon et al. 2011). In particular, the CLs will be computed by taking an average of the $(1-\alpha')$-quantiles of the H^2 and SPE statistics computed on each of the 1000 bootstrap samples that are drawn from the reference set. The block bootstrap approach can be considered an alternate way of the traditional kernel density estimation (KDE) procedure when data are autocorrelated. KDE is derived under the assumption of independent and identically distributed data (Silverman 2018). These assumptions may be violated if non-negligible autocorrelation arises in H^2 and SPE statistics computed on the training/reference set. To overcome this issue, the block bootstrap method is instead considered as it is proven to allow estimation of a sampling distribution of a statistic even for strongly dependent data sequences (Lahiri 1999, Bühlmann & Künsch 1999, Härdle et al. 2003). Note that, to ensure a *family wise error rate* (FWER) (Lehmann et al. 2005) smaller than or equal to a desired threshold α, α' is chosen by using the Sidák correction (Lehmann et al. 2005) $\alpha' = 1-(1-\alpha)^{1/2}$. The control charts are then ready to be used in Phase II. Let $\mathbf{x}_{\text{new}}$ denote a new data point and $\mathbf{h}_{\text{new}} = \sigma(\mathbf{W}\mathbf{x}_{\text{new}} + \mathbf{b})$ its representation obtained through the trained encoder network. Then, the corresponding value of the two monitoring statistics to be reported in the relative control chart can be calculated by Equations (3.7) and (3.8), respectively, upon substituting $\mathbf{x}_i$ with $\mathbf{x}_{\text{new}}$ and $\mathbf{h}_i$ with $\mathbf{h}_{\text{new}}$. This allows practitioners to visualize any process drift (i.e, trends) and to issue an alarm if an observation of at least one monitoring statistic falls above H^2_{lim} and SPE_{lim}.

3.3 Real-case study

HVAC data collected onboard modern passenger vehicles, courtesy of Hitachi Rail STS, is employed to demonstrate the effectiveness of the proposed approach.

HVAC and data description

Real operational data are collected every two minutes for about two months in the summer season from modern HVAC systems installed onboard a passenger 6-coach rail vehicle. Table 3.1 summarises the 4 variables related to the HVAC operating conditions that are available for each of the 6 coaches and used for both training and testing purposes. That is, the dimension of the input observation is equal to 24. The training set contains 45776 samples drawn from the process under normal conditions. Whereas the test set contains 504 samples, and it is known that the process mean shifts starting from sample 176. Train names and voyage dates are intentionally omitted for confidentiality reasons.

A central unit is installed to control the HVAC system performance through temperature sensors that activate heating or cooling mode based on measurements of outside (T_{out}) and interior (T_{in}) temperatures. The aim of the central unit is to maintain comfortable

Table 3.1: Operational variables measured for each train's coach.

Variable	Description
T_{in}	Interior temperature
T_{out}	Outside temperature
T_{Set}	Setpoint temperature set by European regulations
T_{Supply}	Theoretical temperature provided by the HVAC

temperature and humidity levels throughout the range of environmental and climatic conditions required by the aforementioned European regulations UNI EN 14750-1 (EN 2006). In particular, the HVAC central unit implements (EN 2006) by dynamically setting, as a function of T_{in} and T_{out}, the setpoint temperature (T_{Set}) that is the desired temperature value at which the HVAC systems attempts to maintain the T_{in} value at each time instant. Actually, (EN 2006) allows $|T_{\text{in}} - T_{\text{Set}}|$ to be no larger than 2°C, otherwise the train cannot be operational.

Results

Table 3.2 reports the AE hyperparameters chosen for this real-case study. For comparison purposes with PCA, the number of hidden layer nodes is set equal to 2, i.e., equal to the optimal number of extracted features by PCA that explains at least a given percentage, say 80% of the total variance (Jolliffe 2005). Indeed, the first two principal components are able to account for the 82% of the total variance. Trivially, the number of input and output layer nodes are both set equal to the dimension of the input observation, that is 24. The remaining optimal AE hyperparameters are chosen by a numerical grid search as those achieving the smallest reconstruction error estimated through a 10-fold cross-validation applied to the training data with early stopping (Goodfellow et al. 2016), according to what is stated in Section 3.2. Within the grid search algorithm, different AEs are trained, each corresponding to a different combination of the selected hyperparameters, where the activation function is different for each layer. Each network is trained on a single core of an Intel Xeon Platinum 8160 node of the ENEA CRESCO6 system (2.10GHz, 192 GB RAM, no GPU) (Iannone et al. 2019). The proposed approach is numerically implemented by means of the open source software environment Python (Van Rossum & Drake Jr 1995) and Keras (Chollet 2015) with TensorFlow (Abadi et al. 2015) as the backend. Finally, a single hidden layer AE with two neurons is trained, with parameters $\mathbf{W_o}, \mathbf{W'_o}, \mathbf{b_o}, \mathbf{b'_o}$ learned by backpropagation and Adam's optimization (Goodfellow et al. 2016) algorithms. The CLs of the monitoring statistics are estimated through the block bootstrap with $\alpha = 0.05$.

For the sake of comparison, additionally to the proposed method, PCA, AE with more than one hidden layer (Deep AE), and Variational AutoEncoder (VAE) (Goodfellow et al. 2016) are also investigated. To make a fair comparison, AEs, PCA, and VAE have the same latent dimension, that is 2. As already stated in the previous sections, the hyperparameters of the deep AE and VAE are found as the ones that achieve the smallest reconstruction error through a 10-fold cross-validation on the training data. The overall network structures of the AE and VAE are $[24 - 8 - 2 - 8 - 24]$ and $[24 - 2 - 24]$, respectively. The H^2 and SPE statistics and the CLs for the competing methods are constructed in the same way as described in Section 3.2 with $\alpha = 0.05$. To evaluate the monitoring performance of the proposed method, four fault data sets are independently generated by drawing 10000 sample

Table 3.2: AE hyperparameter values. In bold, the hyperparameters, with their ranges, chosen through grid-search 10-fold cross-validation procedure

Hyperparameter	**Explored values**	**Chosen value**
Learning rate	{0.1,0.01,0.001,0.0001}	0.001
Number of input layer nodes		24
Number of hidden layer nodes		2
Number of output layer nodes		24
Activation function for hidden layer	{ReLU, ELU, tanh, sigmoid}	ELU
Activation function for output layer	{ReLU, ELU, tanh, sigmoid}	ELU
Batch size	{64,128,256,512}	128

Table 3.3: FDR (%) for a code layer dimension hyperparameter equal to 2 in the four scenarios described in Section 3.2. The best performance is highlighted in bold

Scenarios	**PCA**	**AE**	**DeepAE**	**VAE**
1	17.72	**18.29**	17.81	15.32
2	77.10	**78.40**	77.16	64.58
3	59.84	**60.28**	60.12	47.44
4	99.97	**99.98**	**99.98**	99.53

data from the training set and simulating failures as follows,

1. the mean of the T_{in} and T_{supply} of the first coach is shifted by two units of their standard deviation (Fault data set No. 1);

2. the mean of the T_{in} and T_{supply} of the first coach is shifted by three units of their standard deviation to simulate the second data set (Fault data set No. 2);

3. the mean of the T_{in} and T_{supply} of the first two coaches is shifted by two units of their standard deviation to simulate the third fault data set (Fault data set No. 3);

4. the mean of the T_{in} and T_{supply} of the first two coaches is shifted by three units of their standard deviation (Fault data set No. 4).

FDRs of the four competing methods are compared in Table 3.3 and are calculated as $FDR = TN/1000 \times 100\%$, where TN is the number of fault samples correctly identified. If the value of at least one of the monitoring statistics is greater than the corresponding UCL, the fault is successfully detected, otherwise, no alarm is signaled. From the numerical results in Table 3.3 it can be seen that all methods achieve high monitoring performance for the second and the fourth type of fault. In both cases, AE achieves the best result with the highest FDR. For fault data sets No. 1 and 3, although the results of the four competitors are unsatisfactory, the AE still has the better detection power. Additionally, it is clear from Table 3.3 that VAE's H^2 and SPE monitoring results are very bad, especially for fault data sets No. 2 and 3.

To demonstrate the applicability of the proposed approach, Figure 3.1 and Figure 3.2 report the H^2 and SPE control charts based on the single hidden layer AE for the HVAC data, respectively. The dashed vertical line indicates the instant at which a fault is known to occur, whereas the bold line indicates the UCLs, H^2_{lim} and SPE_{lim}, for each chart with $\alpha' = 0.025$. After the failure of the HVAC system, SPE_i statistics fall above SPE_{lim}. Whereas, H^2_i observations, even though showing an unusual trend, do not exceed H^2_{lim}. Hence, the fault is successfully identified as a change in the residual space, and not as a drift in the feature space.

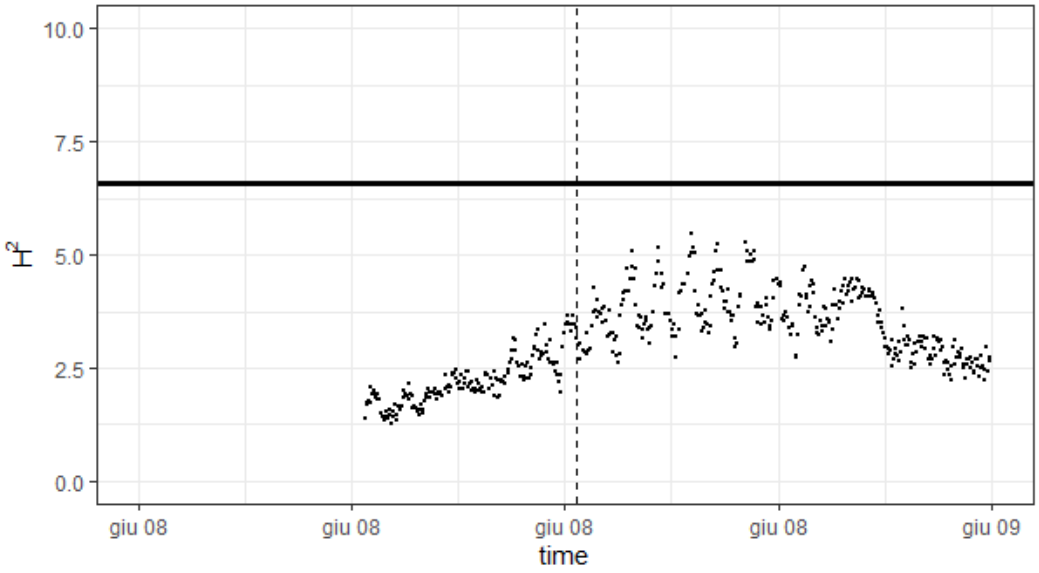


Figure 3.1: H^2 control chart used for the perspective monitoring of HVAC data. Each point indicates the monitoring statistic value at each point in time. The bold line indicates the UCL at $\alpha' = 0.025$, whereas the dashed line indicates the instant at which a fault is known to occur.

3.4 Conclusions

A nonparametric SPC approach is proposed through a properly designed AE NN, and the joint use of the H^2 and SPE control charts, which are built on the space spanned by the feature extracted by the AE and the corresponding reconstruction error, respectively. The performance of the proposed method has been investigated through four simulated fault data sets and has been compared with the VAE, a deeper AE, and PCA in terms of the FDR. To allow for a fair comparison all the comparing methods have the same latent dimension, chosen as equal to the optimal number of extracted features by PCA that explains at least the 80% of the total variance. The results showed that the proposed monitoring strategy performs better than the competitors for all types of faults. The approach is also shown to

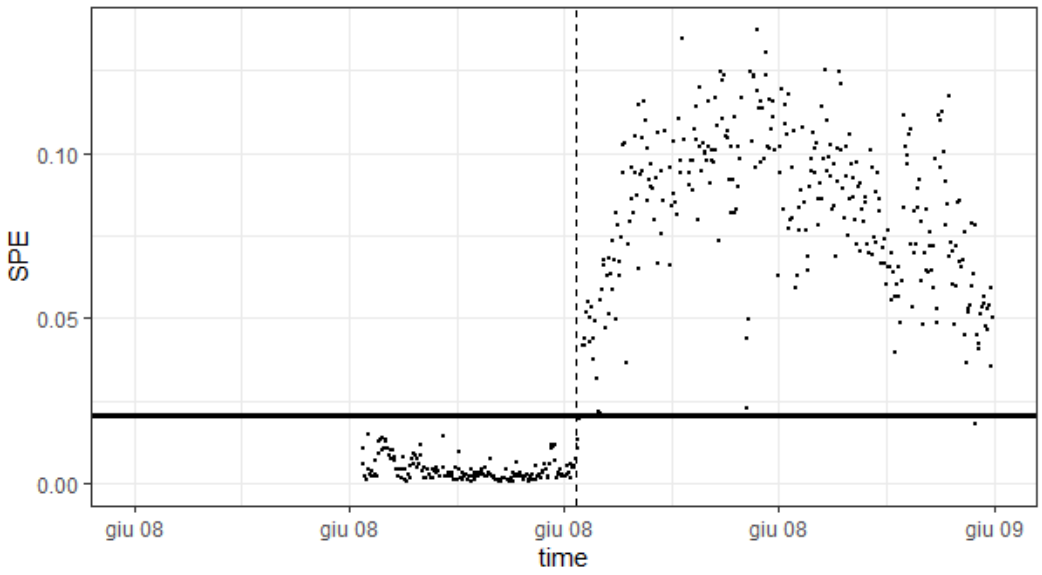


Figure 3.2: SPE control chart used for the perspective monitoring of HVAC data. Each point indicates the monitoring statistic value at each point in time. The bold line indicates the UCL at $\alpha' = 0.025$, whereas the dashed line indicates the instant at which a fault is known to occur.

be capable of exploiting the massive operational HVAC data collected by the rail transport company Hitachi Rail STS and promptly indicating if and when an anomaly occurs in the HVAC system performance, in a new automated and interpretable way.

In this direction, for future work, different and possibly larger real-data examples are expected to provide further evidence of this approach's benefits and applicability.

Chapter 4

Functional neural network control chart

In many Industry 4.0 data analytics applications, quality characteristic data acquired from manufacturing processes are better modeled as functions, often referred to as profiles. In practice, there are situations where a scalar quality characteristic, referred to also as *response*, is influenced by one or more variables in the form of functional data, referred to as *functional covariates*. To adjust the monitoring of the scalar response by the effect of this additional information, a new profile monitoring strategy is proposed on the residuals obtained from the FNN, which is able to learn a possibly nonlinear relationship between the scalar response and the functional covariates. An extensive Monte Carlo simulation study is performed to assess the performance of the proposed method with respect to other control charts that appeared in the literature before. Finally, a case study in the railway industry is presented with the aim of monitoring the HVAC installed onboard passenger trains.

4.1 Introduction

In many modern production processes, advanced data acquisition systems generate massive amounts of data in the form of curves or surfaces that vary over a continuum, such as time or space. Such data can be best modeled as functions that are more generally defined multidimensionally, and are usually referred to as *profiles* or *functional data* (Ramsay & Silverman 2005, Hsing & Eubank 2015, Menafoglio & Secchi 2017, Ferraty 2006). The statistical monitoring of a process best characterized by functional data is known as *profile monitoring*, which focuses on testing the stability over time of the functional quality characteristic of interest. As an application area within statistical process monitoring (SPM) (Montgomery 2019), profile monitoring aims to detect special causes of variation acting on the process, which, in such cases, is said to be out of control (OC). Otherwise, the process is said to be affected by only random causes of variation and hence in control (IC). Some examples of profile monitoring applications can be found in the works of Jin & Shi (1999), Woodall et al. (2004), Zou et al. (2007), Williams et al. (2007), Colosimo & Pacella (2010), Saghaei et al. (2013), Flores et al. (2021).

In practice, there are often situations where measurements of one or more concurrent variables, which may influence the quality characteristic, are also available in the form of functional data and referred to as *covariates*. That is, the quality characteristic, referred to also as *response* in this perspective, is monitored conditionally on the covariate levels. However, the existing approaches in the profile monitoring literature do not necessarily consider the relationship between the quality characteristic and covariates that may influence

it. A few exceptions are the recent works that extend the idea of the regression control chart (Mandel 1969) to functional data, where a scalar (Capezza et al. 2020) or functional (Centofanti et al. 2021) quality characteristic is monitored after being adjusted for the effect of one or more functional covariates. That is, their idea is to monitor the residuals of the regression of the quality characteristic on the covariates. In this way, their approach is able to consider the variance explained by the covariates and leverage their additional information with the aim of improving the effectiveness of the monitoring scheme. However, all the instances that can be reconducted to this framework, namely functional regression control chart (FRCC), are implemented through functional *linear* models, only.

In the meantime, NNs and Deep Learning (DL) techniques have been receiving increasing attention in time series, computer vision, speech recognition, and genetics. However, despite their success, their use for functional data is not completely investigated. Rossi et al. (2005) discussed how to incorporate functional pre-processing, e.g., FPCA, into a MLP, which is a particular type of NN. Rossi et al. (2002), Conan-Guez & Rossi (2002) introduced the Functional MultiLayer Perceptron (FMLP), extending the MLP to functional data through a proper reparameterization of the weight matrices. In particular, the former proposes an FMLP that simply requires a discretization of the functional covariates and, thus it cannot directly handle profiles with a different number of observations or measured at different time points. To overcome this issue, the latter uses the basis expansions (Wang et al. 2016) of the functional covariates. The theoretical properties of the FMLP are firstly studied by Rossi & Conan-Guez (2006), where the authors revealed that an FMLP is a universal approximator following the definition of Hornik et al. (1989). FMLP has also been further investigated by Wang, Zheng, Farahat, Serita & Gupta (2019), Wang, Zheng, Farahat, Serita, Saeki & Gupta (2019).

Another common approach to modeling functional data is through Convolutional Neural Network (CNN) (LeCun et al. 1998) and Recurrent Neural Network (RNN) (Rumelhart et al. 1986). In particular, long short-term memory variant (Hochreiter & Schmidhuber 1997) is the most popular, due to the ability to recognize patterns for a long duration of time. However, these may fail to learn the underlying smoothness of the functional data from raw noisy measurements, which are very common in real-data applications. Yao et al. (2021) proposed an adaptive functional NN (AdaFNN) that is a NN specifically designed for functional data through the definition of a new *basis* layer which implements a micro NN (Lin et al. 2013) to directly learn the most relevant base functions to represent the response value, thus avoiding a pre-specified choice. The authors demonstrate the superior performance of their proposal over the MLPs through an extensive simulation study, but they do not compare their proposal with any of the functional regression models already present in the functional data literature. Additionally, even though the authors claimed that the functional coefficients learned by AdaFNN are interpretable, it is not true in general as each micro NN learns a different functional coefficient and it is impossible to priorly know how to combine them and resemble the true functional coefficients. Recently, Thind et al. (2023) introduced a new DL architecture, known as FNN, which allows for deep architectures for scalar responses with multiple functional and scalar covariates through the definition of smooth weight functions. The *functional* weight can be visualized during the training, increasing the interpretability of the FNN while maintaining the nonlinear predictive power of traditional NNs. Furthermore, the number of parameters of the FNN can be lower than the amount needed in traditional MLPs, CNNs, and RNNs. Through simulated and benchmark data sets, Thind et al. (2023) demonstrated that FNN outperforms state-of-the-art DL architectures, functional linear models, and several other multivariate methods, e.g., least

square regression and random forest, in terms of prediction accuracy (Hastie et al. 2009). FNN is also proven to be a universal approximator (Cybenko 1989), that is it can be used to learn any continuous function to a desired degree of accuracy. In recent years, different DL-based have been proposed for the SPC of multivariate processes and functional data, and some relevant examples can be found in Stuart et al. (1996), Sergin & Yan (2021), Pacella & Semeraro (2011), Chen et al. (2020), Howard et al. (2018), Yeganeh et al. (2022), Yeganeh & Shadman (2021).

In this chapter, we propose a novel control chart, named FNNCC, that exploits the FNN architecture advantages for the monitoring of a scalar quality characteristic when functional and scalar covariates are available. The proposed FNNCC can be regarded as an implementation of the FRCC framework where the influence of functional or scalar covariates on a scalar response does not need to be necessarily linear as in Capezza et al. (2020, 2023*b*), which is based on a linear Scalar-On-Function (SOF) regression model (Reiss et al. 2017).

This chapter is structured as follows. Section 4.2 introduces some background on the SOF regression model and FNNs. Then, we describe the proposed FNNCC in detail. An extensive Monte Carlo simulation study is performed in Section 4.3 to quantify the FNNCC OC ARL (ARL_1) at a given IC ARL (ARL_0) (Qiu 2013) in identifying a mean shift in the scalar response in the presence or absence of drifts in the covariate mean and to compare it with other competing control charting schemes that have already appeared in the literature before. Additionally, we implement two other NN-based monitoring strategies and compare them to the proposed control chart. The practical applicability of the proposed method is illustrated in Section 4.4 through a case study in the monitoring of HVAC systems installed on passenger trains (Lepore, Palumbo & Sposito 2022*a*). The `HVAC` data set, courtesy of the rail transportation company *Hitachi Rail STS*, and the analysis code are available online at `https://github.com/unina-sfere/FNNCC`. The final section contains concluding remarks and outlines new directions for future research. All computations and plots have been obtained using the programming language R (R Core Team 2023)

4.2 Methodology

In this section, we briefly review the SOF linear regression model. Then, we introduce the FNN in Section and provide details of the monitoring strategy based on the FNN.

Functional linear regression

We begin by providing the necessary notation used in this chapter, followed by a summary of the linear model for SOF regression before describing the proposed methodology.
Let y_i and $\boldsymbol{X}_i = \left\{(X_{i1}, \ldots, X_{iP})^\top\right\}$, $i = 1, \ldots, n$, denote n observations of a scalar response covariates and a vector of P functional variables, respectively. $\boldsymbol{X}_i$ is a random element that takes values in the Hilbert space $L^2(\mathcal{T})^P$, i.e., $X_{i,1}, \ldots X_{i,P}$ belong to $L^2(\mathcal{T})$, the space of square-integrable functions defined on the compact interval $\mathcal{T}$. We also assume that $\mathbf{X}_i$ is fully observed (Kokoszka & Reimherr 2017) i.e., is densely observed on a set of discrete grid points. The aim is to learn the mapping $G : \mathcal{L}^2(\mathcal{T}) \times \ldots \times \mathcal{L}^2(\mathcal{T}) \to \mathcal{R}$ from the functional covariates $\mathbf{X}_i$ to the scalar response y_i

$$y_i = G\left(\mathbf{X}_i\right) + \eta_i \quad i = 1, \ldots, n, \tag{4.1}$$

where η_i is a scalar error term. The SOF regression has been extensively studied in the literature and, in this section, we give a succinct description of the SOF linear regression

model used by Capezza et al. (2020), defined as

$$y_i = \alpha + \sum_{p=1}^{P} \int_{\mathcal{T}} \beta_p(t) X_{ip}(t) dt + \eta_i \quad i = 1, \ldots, n, \tag{4.2}$$

where $\alpha \in \mathbb{R}$ is the scalar intercept, $\boldsymbol{\beta} = \left(\beta_1, \ldots, \beta_p, \ldots, \beta_P\right)^\top \in L^2(\mathcal{T})^P$ are the functional coefficients to be estimated and η_i are the error terms, which are assumed to be independent and identically distributed normal random variables with mean zero and variance σ^2. Comparing Equation (4.1) and Equation (4.2), it is trivial to note that in the linear SOF regression model $G\left(\mathbf{X}_i\right) = \alpha + \sum_{p=1}^{P} \int_{\mathcal{T}} \beta_p(t) X_{ip}(t) dt$. Without loss of generality, the functional covariates are assumed to be empirically standardized (Chou et al. 2014), that is each covariate observation is standardized by subtracting pointwise the corresponding sample mean and dividing the result by the relative standard deviation function. The coefficients α and $\boldsymbol{\beta}$ in Equation (4.2) can be estimated by solving the following minimization problem

$$\min_{\alpha \in \mathbb{R}, \boldsymbol{\beta} \in L^2(\mathcal{T})^P} \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{r=1}^{P} \int_{\mathcal{T}} \beta_p(t) X_{ip}(t) dt \right)^2. \tag{4.3}$$

Because of the infinite dimensionality of the functional data, the above minimization problem is not well-posed and the model cannot be estimated using the least squares approach James et al. (2013) directly. However, being square integrable, the functional covariates can be represented through the *Karhunen-Loéve expansion* as follows

$$\boldsymbol{X}_i(t) = \sum_{m=1}^{\infty} \xi_{im} \boldsymbol{\psi}_m(t), \quad t \in \mathcal{T}, \quad i = 1 \ldots, n, \tag{4.4}$$

where the $\boldsymbol{\psi}_m = (\psi_{m1}, \ldots, \psi_{mP})_{m \in \mathbb{N}} \in L^2(\mathcal{T})^P$ are the Multivariate Functional Principal Component (MFPC) defined as the eigenfunctions of the covariance function $\mathbf{c}(s,t)$ of the multivariate functional data i.e. they are the solutions to the equation

$$\int_{\mathcal{T}} \mathbf{c}(s,t) \boldsymbol{\psi}_m(s) ds = \lambda_m \boldsymbol{\psi}_m(t), \quad \forall t \in \mathcal{T}, \tag{4.5}$$

where λ_m are called the eigenvalues of $\mathbf{c}(s,t)$. The eigenvalues λ_m, and so the corresponding eigenfunctions $\boldsymbol{\psi}_m(t)$, are arranged in non-increasing order $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant 0$. The eigenfunctions $\boldsymbol{\psi}_m(t)$ are by construction such that

$$\sum_{p=1}^{P} \int_{\mathcal{T}} \psi_{m_1,p}(t) \psi_{m_2,p}(t) dt = \begin{cases} 1, & \text{if } m_1 = m_2 \\ 0, & \text{if } m_1 \neq m_2, \end{cases} \tag{4.6}$$

that is, they form an orthonormal basis of $L^2(\mathcal{T})^P$. The terms ξ_{im} in Equation (4.4) are called the MFPC scores, or simply scores, and defined as

$$\xi_{im} = \sum_{p=1}^{P} \int_{\mathcal{T}} X_{ip}(t) \psi_{mp}(t) dt \quad i = 1 \ldots, n. \tag{4.7}$$

It can be shown that $\mathrm{E}(\xi_m) = 0$, $\mathrm{E}(\xi_m^2) = \lambda_m$ and $\mathrm{E}(\xi_{m_1}, \xi_{m_2}) = 0$ when $m_1 \neq m_2$. The decomposition in Equation (4.4) is optimal in the sense that, for each finite $M \in N$,

$\boldsymbol{X}_i(t)$ is best approximated by $\hat{\mathbf{X}}_i = (\hat{X}_{i1}, \ldots, \hat{X}_{ip}, \ldots, \hat{X}_{iP})^T$, obtained by truncating the Karhunen-Loéve expansion, as follows

$$\hat{X}_{ip}(t) = \sum_{m=1}^{M} \xi_{im}\psi_{mp}(t), \quad t \in \mathcal{T}, \quad p = 1, \ldots, P, \quad i = 1 \ldots, n. \tag{4.8}$$

It can be proved that, upon using $\hat{\mathbf{X}}_i$ in place of $\mathbf{X}_i$, the functional coefficient $\boldsymbol{\beta}$ in Equation (4.2) can be replaced with $\hat{\boldsymbol{\beta}} = \left(\hat{\beta}_1, \ldots, \hat{\beta}_p, \ldots, \hat{\beta}_P\right)^\top \in L^2(\mathcal{T})^P$ by using the same truncated basis expansion, that is

$$\hat{\beta}_p(t) = \sum_{m=1}^{M} b_m\psi_{mp}(t), \quad t \in \mathcal{T}, \quad p = 1, \ldots, P, \tag{4.9}$$

where $b_1, \ldots, b_M$ are the basis coefficients. Upon using the approximation in Equations (4.8) and Equation (4.9), Equation (4.2) can be rewritten as

$$y_i = \alpha + \sum_{m=1}^{M} \xi_{im}b_m + \varepsilon_i^*, \quad i = 1, \ldots, n. \tag{4.10}$$

Hence, in this form, the model parameters, namely the intercept α and the coefficient b_m, $m = 1, \ldots, M$, can be estimated by using the least square approach, as $\hat{\alpha} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\hat{b}_m = \sum_{i=1}^{n} y_i\xi_{im} / \sum_{i=1}^{n} \xi_{im}^2$, respectively. Accordingly, the least square prediction of the scalar response y_i can be obtained as

$$\hat{y}_i = \hat{\beta}_0 + \sum_{p=1}^{P} \int_{\mathcal{T}} X_{ip}(t)\hat{\beta}_p(t)dt, \quad i = 1, \ldots, n. \tag{4.11}$$

As in the multivariate setting, the number M can be chosen such that the retained MFPCs $\boldsymbol{\psi}_m = (\psi_{m1}, \ldots, \psi_{mP})_{m=1,\ldots,M}$ explain at least a given percentage, say 80%, of the total variability. In this chapter, we use a different strategy based on the final model's predictive ability. That is, the M MFPCs to be retained are the first M that achieve a given reduction in the prediction sum of squares statistic, defined as $\sum_{i=1}^{n}\left(y_i - \hat{y}_{[i]}\right)^2$, where $\hat{y}_{[i]}$ is the i-th fitted value of the scalar response based on the SOF regression model with the i-th observation removed from the data set used to fit the linear model. In this way, this strategy ensures that the MFPCs with a small predictive ability are not retained in the SOF model. More details on this problem can be found in (Jolliffe & Cadima 2016, p. 173-177).

Functional neural network

NNs are computational models inspired by the structure and functioning of the human brain and consisting of interconnected artificial neurons, also known as nodes or units. The latter are organized into layers, typically an input layer, one or more hidden layers, and an output layer. The input layer receives the initial input data, and the output layer produces the final output or prediction. The hidden layers are intermediate layers between the input and output layers and play a crucial role in learning complex representations of the data. Each neuron takes multiple inputs from the previous layer, performs a weighted sum, and then applies a nonlinear function to produce an output. Let $n^{(j)}$ and $\mathbf{h}^{(j)}$ be the number of neurons in the

jth hidden layer and the output of the jth hidden layer, respectively. More formally, $\mathbf{h}^{(j)}$ is defined as $\mathbf{h}^{(j)} = g\left(\mathbf{W}^{(j)}\boldsymbol{h}^{(j-1)} + \boldsymbol{b}^{(j)}\right)$, where $\mathbf{h}^{(j-1)} \in R^{n^{(j-1)}}$ represents the output of the previous $(j-1)$th layer, $\mathbf{W}^{(j)}$ is a $n^{(j)} \times n^{(j-1)}$ weight matrix and $\boldsymbol{b}^{(j)} \in R^{n^{(j)}}$ is the intercept, often referred to as the bias in the ML field. The function $g : \mathbb{R}^{n^{(j)}} \to \mathbb{R}^{n^{(j)}}$ is called the activation function and introduces nonlinearity into the output of the neuron (Hastie et al. 2009), which may enable more complex pattern recognition and more accurate predictions. If g is the identity function, any NN is proved to specialize into a linear regression model. The choice of the activation function depends on the type of problem to be solved. Some common ones are (a) the sigmoid function (or logistic function) (Han & Moraga 1995) that maps the input to a range between 0 and 1, making it useful for binary classification problems; (b) the ReLU (Hahnloser et al. 2000) that returns the input if it is positive, and zero otherwise and is widely used due to its simplicity and computational efficiency; (c) the hyperbolic tangent (Rumelhart et al. 1986) that squashes the input values to the range between -1 and 1, making it useful for classification tasks; and (d) the softmax function (Ackley et al. 1985) that is commonly used in the output layer of a NN for multi-class classification problems, as it converts the outputs into a probability distribution.

Traditional NNs accept only finite-dimensional vectors as input and thus, they cannot easily handle profiles, whereas, the FNN introduced by Thind et al. (2023) is instead purposely designed for it.
Given $X_1(t), \ldots, X_P(t)$ and $z_1, \ldots, z_J$ functional and scalar covariates, respectively, Thind et al. (2023) introduce the functional weights $\boldsymbol{\gamma}(t) = \left(\gamma_1, \ldots, \gamma_p, \ldots, \gamma_P\right)^\top$ to effectively weigh the functional covariates at every point along their domain $\mathcal{T}$, and define the output $h_k^{(1)}$ of the kth neuron in the first hidden layer corresponding to the ith observation as follows

$$h_{ik}^{(1)} = g\left(\sum_{p=1}^{P}\int_{\mathcal{T}}\gamma_{kp}(t)X_{ip}(t)dt + \sum_{j=1}^{J} w_{kj}^{(1)} z_{ij} + b_k^{(1)}\right) \quad i = 1, \ldots, n \quad k = 1, \ldots, n^{(1)}, \tag{4.12}$$

where $n^{(1)}$ is the number of neurons in the first hidden layer and $g(\cdot)$ is the activation function. This layer is referred to as a *functional* hidden layer as it consists of neurons capable of handling infinite dimensional *functional* weights $\boldsymbol{\gamma}(t)$. Each neuron in the first layer produces a scalar value that is then fed into a regular NN. It is worth noting that only the output of the first hidden layer has this *functional* structure and this is the reason why $\gamma_{kp}(t)$ in Equation (4.12) does not need for any superscript. This implies that the rest of the hidden layers of the FNN can be of any of the usual forms (e.g., feedforward, recurrent, residual) (LeCun et al. 2015). Similarly to the results in Equation (4.9), the functional weights $\gamma_{kp}(t)$ in this first hidden layer can be approximated through a linear combination of basis functions

$$\hat{\gamma}_{kp}(t) = \sum_{m=1}^{M_p} c_{kpm}\zeta_{kpm}(t) = \boldsymbol{c}_{kp}^T\boldsymbol{\zeta}_{kp}(t) \quad p = 1, 2, \ldots, P \quad k = 1, 2, \ldots, n^{(1)}, \tag{4.13}$$

where $\boldsymbol{\zeta}_{kp}(t) = \left(\zeta_{kp1}(t), \ldots, \zeta_{kpM_p}(t)\right)^T$ is a vector of basis functions, $\boldsymbol{c}_{kp} = \left(c_{kp1}, \ldots, c_{kpM_p}\right)^T$ is the corresponding vector of basis coefficients to be estimated by the NN, and M_p denotes the number of basis functions for each of the P functional covariates. Using the basis approximation in Equation (4.13), the general form of the kth neuron in the first hidden

layer in Equation (4.12) can be rewritten as

$$\begin{aligned} h_k^{(1)} &= g\left(\sum_{p=1}^{P}\int_{\mathcal{T}}\sum_{m=1}^{M_P} c_{kpm}\zeta_{kpm}(t)X_p(t)dt + \sum_{j=1}^{J} w_{kj}^{(1)} z_j + b_k^{(1)}\right) \\ &= g\left(\sum_{p=1}^{P}\sum_{m=1}^{M_P} c_{kpm}\int_{\mathcal{T}}\zeta_{kpm}(t)X_p(t)dt + \sum_{j=1}^{J} w_{kj}^{(1)} z_j + b_k^{(1)}\right). \end{aligned} \tag{4.14}$$

In Equation (4.14), the integral can be approximated by using any numerical integration method, e.g., the Simpson's rule (Süli & Mayers 2003). To compute the integral, the functional covariate $X_p(t)$ can be replaced by $\hat{X}_p(t)$ as in Equation (4.8). The basis coefficients c_{kpm} are differently initialized for each functional weight $\gamma_{kp}(t)$ using the Xavier uniform distribution (Glorot & Bengio 2010), and then will be updated as the FNN learns together with the weights of the other non-functional layers of the FNN. However, any other weight initialization methods can be used, e.g., see Rastrigin (1963), Kim & Ra (1991), He et al. (2015). To train and assess the generalization performance of the FNN, the MSE $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ is used as the loss function, where $y_i, i = 1, \ldots, N$, is the true scalar response and $\hat{y}_i$ the FNN fitted output. The FNN can be trained with the usual backpropagation algorithm (Rumelhart et al. 1986) and the Adam optimizer (Kingma & Ba 2014). It is worth emphasizing that the FNN often requires fewer parameters compared to standard NNs (e.g., MLP, CNN, RNN) that directly process raw data. For instance, let us consider a single functional covariate, measured C times along its domain. The first hidden layer of a traditional NN has a number of parameters equal to $(C + 1) \times n^{(1)}$, whereas the FNN requires $(M_1 + 1) \times n^{(1)}$, where M_1 is typically lower than C to avoid functional weight to overfitting.

Before training a NN there are many parameters to be specified, sometimes referred to as hyperparameters. Tuning the values of the hyperparameters plays a critical role in the generalization of the NN model. Typical hyperparameters are the learning rate, which determines the step size in the optimization process and significantly affects the convergence speed and the possibility of getting stuck in local minima or overshooting global optima; the batch size, which determines the number of samples processed before updating the model's weights, affecting the trade-off between computational efficiency and generalization accuracy; the activation function, which introduces non-linearity feature and may affect the NN convergence and ability to handle vanishing or exploding gradients (Rumelhart et al. 1986); the NN architecture, which includes the number of layers and the number of neurons in each layer and significantly influence the ability of the NN to capture complex patterns in the data. The number and the type of the basis functions, which approximate each functional weight, can be considered hyperparameters as well. A manual hyperparameter tuning approach (Goodfellow et al. 2016) combined with a 5-fold cross-validation (Hastie et al. 2009) is used to optimize the generalization performance of the FNN. In this way, we ensure a comprehensive coverage of the hyperparameter space and identify the optimal parameter values that yield the smallest 5-fold cross-validated MSE, defined as $\sum_{b=1}^{5}\sum_{i\in S_b}(\hat{y}_i - y_i)^2/N$, where S_b is the bth set of observations in the held-out fold, and $\hat{y}_i$, is the predicted value for y_i by the FNN trained on the rest of the $B - 1$ folds. In the subsequent analysis, we tune all the FNN hyperparameters to some degree and then use the early stop strategy (Keskar & Socher 2017) to prevent overfitting and improve the generalization performance of the final model. Instead of training the model for a fixed number of epochs, the early stop strategy consists of monitoring the MSE on the *validation set*, which is a separate subset of unseen observations, and stops the training process when the latter increases.

The problem of interpretability in NNs has become a significant concern in the field of DL. While NNs have demonstrated remarkable performance across various tasks, their inherent black-box nature has hindered their wider adoption and acceptance, particularly in domains where interpretability and transparency are paramount. The black box refers to the opacity of NNs, where the internal decision-making processes are not easily understandable or explainable to humans. Researchers and practitioners are striving to develop techniques and methodologies that can shed light on the black box, enabling us to gain deeper insights into how NNs arrive at their predictions or classifications. The pursuit of interpretability, whose rigorous definition is still debated in ML literature (Molnar 2020, Lepore, Palumbo & Poggi 2022, Stevens & De Smedt 2023), is crucial for ensuring transparency and enabling domain experts to understand, validate, and trust the decisions made by NNs. To this aim, FNNs guarantee hidden semantic interpretability that refers to the human ability to understand hidden layers (Fan et al. 2021). Specifically, the functional weights introduced in Equation (4.14) differ from the traditional weights as they can be easily visualized over the continuum, helping the interpretation of the relationship between the functional covariates and the scalar response while preserving the autocorrelation structure associated with the data. These functional weights coincide with those estimated in the linear functional regression model reported in Equation (4.10). If the functional hidden layer has more than one neuron, the average of the estimated functional weights $\hat{\gamma}_p(t) = \sum_{k=1}^{n^{(1)}} \hat{\gamma}_{kp}(t)/n^{(1)}$ for each P functional covariate can be considered.

The functional neural network control chart

We propose a control charting procedure, referred to as FNNCC, to monitor a scalar quality characteristic adjusted by the effect, possibly nonlinear, of one or more functional covariates, which relies on the following main steps:

i The relationship between the scalar response and the functional covariates is modeled through an FNN.

ii The functional model is estimated through FNN hyperparameter tuning, objective function, and optimization algorithm definition (Goodfellow et al. 2016, p. 294-316).

iii The monitoring strategy of residuals $e_i = y_i - \hat{y}_i$, $i = 1, 2, \ldots, n$ obtained from the FNN is defined. Residuals act as the scalar quality characteristic of interest to indirectly monitor the stability of the functional relationship between the scalar response y_i and the multivariate functional covariates X_{ip}. For conciseness of notation, we will hereinafter denote by $\hat{y}_i$ the fitted value of y_i, even when an objective function different from MSE (as in Equation (4.11)) is used.

A current observation of the scalar response variable y_i, given the corresponding functional covariate vector $\mathbf{X}_i(t) = (X_{i1}, \ldots, X_{iP})$, is monitored by using the FRCC approach, that is the FNNCC results in a univariate control chart based on the FNN residual $e = y - \hat{y}$. This chapter focuses on the prospective monitoring of the residuals, referred to as Phase II. That is, a data set of observations representative of the IC process performance, referred to as *Phase I sample* or *reference data set*, is assumed to be available. It is worth noting that the retrospective monitoring, referred to as Phase I is crucial to check the stability of historical functional data and to obtain accurate estimates of the unknown parameters used for Phase II monitoring (Zhang et al. 2015). The reference data set is randomly split into

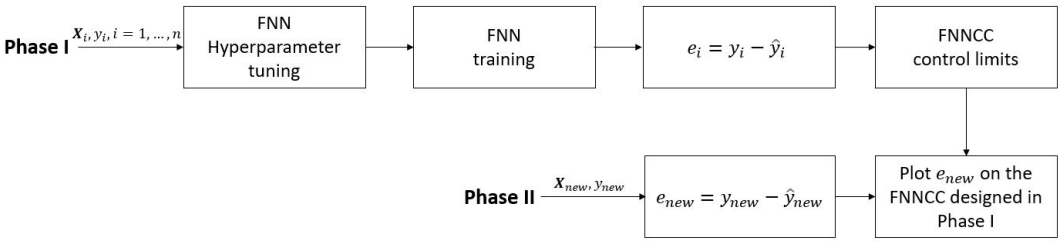


Figure 4.1: Outline of the FNNCC approach.

three non-overlapping sets, referred to as *training*, *validation*, and *tuning* sets. The first is used to design and train the FNN; the second to implement the early stopping strategy (Goodfellow et al. 2016), which allows stopping the training when the performance does not improve and thus, to prevent overfitting and improve the generalization of the NN; the third is used to estimate the upper and lower CLs, respectively, as $\alpha/2$ and $(1-\alpha/2)$ empirical quantiles of the sampling distribution of the FNN residuals estimated from the tuning set, being α the type-I error rate (Qiu 2013). In Phase II, the residual of a new observation $(\mathbf{X}_{new}, y_{new})$ is calculated as

$$e_{new} = y_{new} - \hat{y}_{new}, \tag{4.15}$$

where $\hat{y}_{new}$ is the value fitted by the FNN model identified in Phase I, according to the FNN model estimated with choices made in steps (i) and (ii). An alarm is issued if e_{new} is larger than UCL or lower than Lower Control Limit (LCL). The implementation of the FNNCC is outlined in Figure 4.1.

If the FNNCC issues an alarm, a change in the relationship between the response and the covariates may have occurred. This could be due to changes in the regression coefficients associated with one or more covariates, or potential causes outside the set of covariates included in the model may be investigated (Shu et al. 2004).

4.3 Simulation study

The overall performance of the proposed FNNCC is compared with FRCC and with a univariate SCC, referred to as SCC, which monitors the scalar response without considering any information on the functional covariates, in terms of ARL_1 at a given ARL_0 (Qiu 2013) through an extensive Monte Carlo simulation. Without loss of generality, the compact domain $\mathcal{T}$ is assumed as $[0, 1]$, and the number of covariates P is set equal to 1. To cover nonlinear process scenarios, as in Thind et al. (2023), the scalar response observations are

generated from an IC process as in the following scenarios

$$\text{Scenario A} \quad y^* = G(\mathbf{X}_i) + \eta^* = \alpha + \int_{\mathcal{T}} \beta(t)x(t)\mathrm{d}t + \eta^* \tag{4.16}$$

$$\text{Scenario B} \quad y^* = G(\mathbf{X}_i) + \eta^* = \exp\left(\alpha + \int_{\mathcal{T}} \beta(t)x(t)\mathrm{d}t\right) + \eta^* \tag{4.17}$$

$$\text{Scenario C} \quad y^* = G(\mathbf{X}_i) + \eta^* = \left|\alpha + \int_{\mathcal{T}} \beta(t)x(t)\mathrm{d}t\right| + \eta^* \tag{4.18}$$

$$\text{Scenario D} \quad y^* = G(\mathbf{X}_i) + \eta^* = \log\left(\left|\alpha + \int_{\mathcal{T}} \beta(t)x(t)\mathrm{d}t\right| + u\right) + \eta^* \tag{4.19}$$

$$\text{Scenario E} \quad y^* = G(\mathbf{X}_i) + \eta^* = \left(\alpha + \int_{\mathcal{T}} \beta(t)x(t)\mathrm{d}t\right)^2 + \eta^*, \tag{4.20}$$

$$\tag{4.21}$$

where the noise η^* is sampled from the Gaussian distribution $N(0, 0.1)$. where in each scenario, apart from Scenario A, $G(\cdot)$, defined in Equation (4.1), specializes to a nonlinear SOF mapping, and η_i^* is the error term. It is worth noting that in Scenario D, u is an arbitrary positive constant, which is set equal to 2 to avoid numerical problems due to small values. y_i and $\alpha + \int_{\mathcal{T}} \beta(t)x_i(t)\mathrm{d}t$ are generated using the functions `simulate_mfd()`, which is called by the wrapper function `sim_funcharts()` from the `funcharts` package (Capezza et al. 2023*a*). The mean and the variance function $\mu^X(t)$ and $v^X(t)$ of the functional covariates are generated according to the following model

$$f(z) = P(z) + r\sum_{i=1}^{I} h_i\left(z; m_i, s_i\right), \quad z \in (0, 1), \tag{4.22}$$

with

$$P(z) = az^2 + bz + c, \tag{4.23}$$

where a, b and c are real numbers, and the terms $h_i\left(z; m_i, s_i\right)$ are normal probability density functions with mean m_i and standard deviation s_i. The functional covariate $X(t)$ is characterized by the Bessel (Abramowitz et al. 1964) correlation function and is evaluated at 150 equally spaced discrete points of the functional domain $\mathcal{T} = [0, 1]$. As the simulated functional covariate data are observed at noisy discrete values, each functional observation is obtained by Equation (4.8) with $M = 30$ cubic B-splines estimated through the spline smoothing approach. Then, the scalar response is generated through a SOF linear model so that the determination coefficient R^2 defined by Yao et al. (2005), which measures the proportion of the variance in the response variable explained by the functional covariate in the model, is set equal to 0.97. The mean and the variance of y are set to $\mu^y = 0$ and $v^y = 1$, respectively.

The performance of the proposed control chart is studied under a shift in μ^y only, and both in the functional covariate mean $\mu^X(t)$ and μ^y. In the latter case, to study the unwanted effect of the shift in $\mu^X(t)$ on the FNNCC performance, a translation of the profile pattern is generated by using the model defined in Equation (4.22) with $P(z)$ defined as

$$P(z) = az^2 + bz + (c + \delta), \quad z \in (0, 1), \tag{4.24}$$

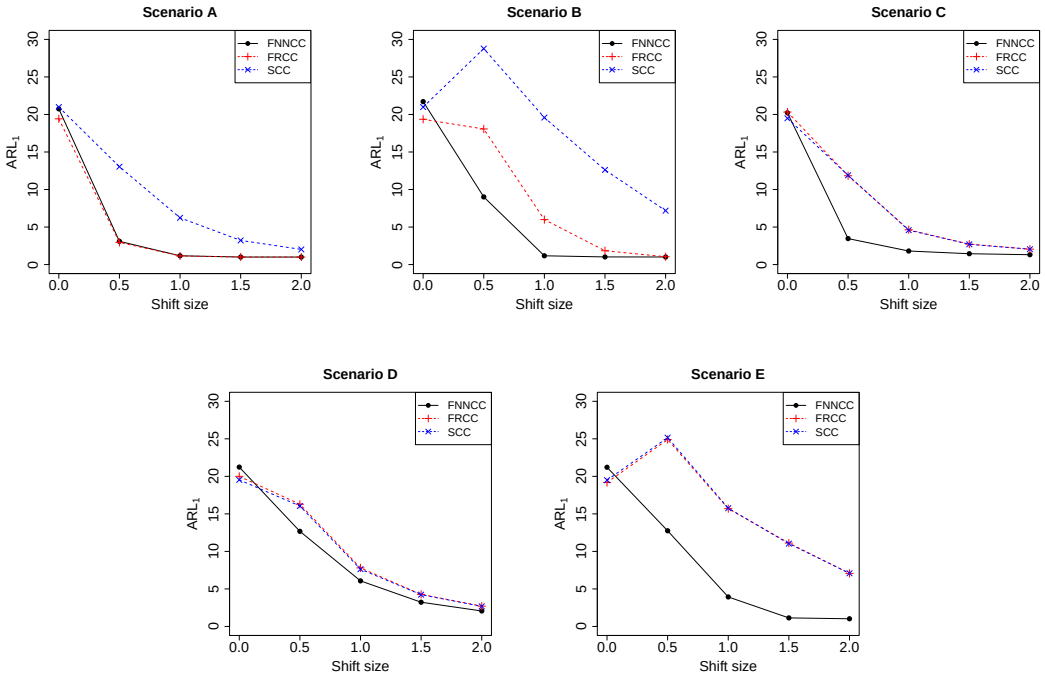


Figure 4.2: Estimated ARL_1 achieved by FNNCC, FRCC, and SCC for each simulated scenario, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$.

where δ is a real number defining the translation magnitude and is set to be equal to 0.5. In both scenarios, the mean shift in the scalar response is obtained by adding, to the simulated values, a fixed quantity that defines the mean shift size. That is, μ_y is shifted by as much as $\Delta\mu_y = \{0.5, 1, 1.5, 2\}s^{y^*}$, where $s^{y^*} = \sqrt{(v^{y^*})^2}$ is the standard deviation of the transformed scalar response.

For each simulated scenario, a set of 4000, 1000, and 10000 IC scalar responses and functional covariates are randomly generated to form the training, validation, and tuning sets, respectively, for the reason discussed in Section 4.2. To evaluate the ARL_1, an additional set of 20000 OC patterns is randomly generated. Although in practice the most commonly used for ARL_0 are larger than 200, in this simulation study, we set $ARL_0 = 20$, which corresponds to $\alpha = 0.05$. This choice however does not affect the OC performance comparison while reducing the simulation computational burden. In all scenarios, based on 5-fold cross-validation with a manual hyperparameter tuning approach, we use in step (ii) a two-layer FNN with 8 neurons and with ReLu and linear activation functions. Each functional coefficient is expressed as a linear combination of 5 cubic B-spline (see Equation (4.13)). Then, the FNN is trained in step (ii) using the backpropagation algorithm with the Adam optimizer to minimize the MSE chosen as the objective function. For the FRCC, regression coefficients are estimated using the training set, while the tuning set is used to compute the CLs as the $\alpha/2$ and $1 - \alpha/2$ of empirical quantiles of the scalar response.

Figures 4.2 graphically represent the ARL_1 performance achieved by FNNCC, FRCC,

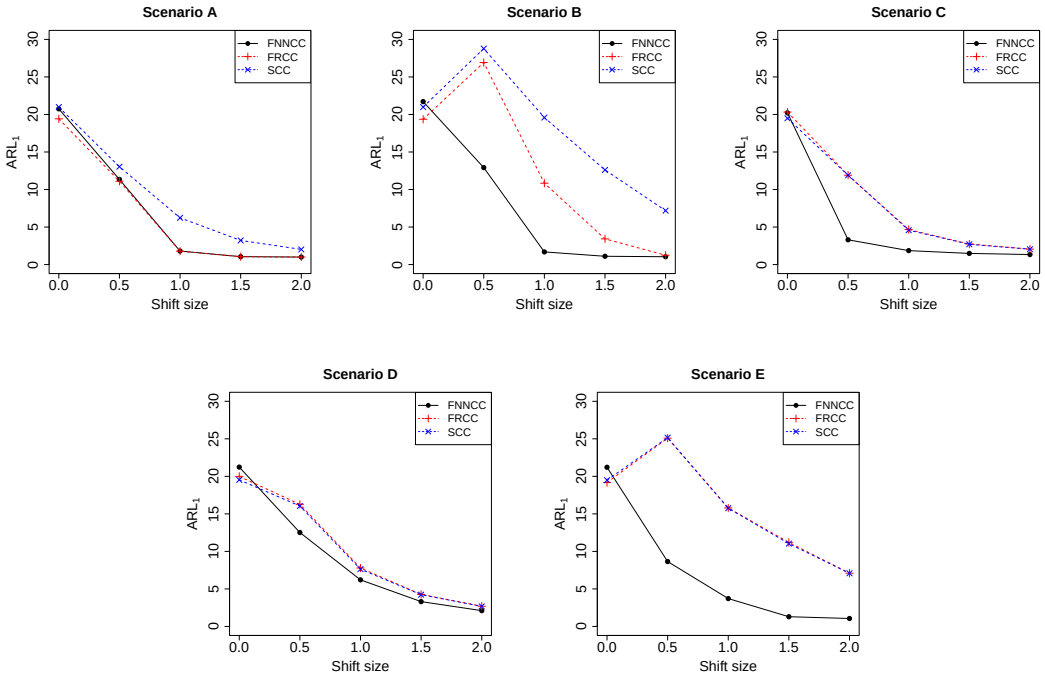


Figure 4.3: Estimated ARL_1 achieved by FNNCC, FRCC, and SCC for each simulated scenario, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y*}$ when the functional covariate is subject to a mean shift

and SCC for each simulated scenario, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y*}$, when no covariate mean shift is considered. As we would expect, the SCC shows the worst performance for all the considered scenarios and shifts as it is not capable of adjusting the monitoring of the scalar response by the effect of the additional information provided by the functional covariates. In Scenario A, the FRCC performs comparably to the FNNCC as the linear functional regression model can capture the true linear relationship between the scalar response and the functional covariate. In all other scenarios, the FNNCC outperforms the FRCC for all the considered shifts. This confirms the SOF regression model is not capable of modeling the true nonlinear relationship and, thus, the FRCC performance is the same as the SCC. For example, for Scenario C, the ARL_1 of the FNN, FRCC, and SCC are 3.46, 11.83, and 11.89, respectively, at $\Delta\mu_y = 0.5s^{y*}$. The gain in efficiency decreases as the shift size increases. As an example, when the relation between the response and the functional covariate is modeled by an exponential function (Scenario B), the ARL_1 of the FNNCC and the FRCC are 9.01 and 18.06 at a small shift size $\Delta\mu_y = 0.5s^{y*}$, respectively, whereas for a higher shift size, say $\Delta\mu_y = 1.5s^{y*}$, their performance decrease to 1.02 and 1.84, respectively. When the functional covariate is subject to a mean shift, Figure 4.3 displays the estimated ARL_1 of the three competing control charting schemes for all the simulated scenarios, as a function of the size of the response mean shift. This figure points out that shifts in the covariate mean function may impact the

ARL_1 of the FNNCC and FRCC. It is trivial to note that the SCC performance is unaffected by a change in $\mu^x(t)$. Simulation results, displayed in Figure 4.3, show that the FNNCC still results more sensitive than the FRCC in detecting OC condition of the scalar response and that the ARL_1 performance of the FNNC and FRCC generally increases (or at least remains the same) in the presence of a covariate mean shift. These results are consistent with Centofanti et al. (2021), Shu et al. (2004). It is worth noting that in scenarios B and E in Figures 4.2 and 4.3, the ARL_1 of the SCC exceeds the ARL_0. This is because the quantile-based control charting approach struggles to accurately handle the right-skewed distribution of the scalar response. The same phenomenon affects the performance of the FRCC (see scenarios B and E in Figure 4.3), where model mispecification leads to a skewed distribution of the residuals.

The simulation study clearly highlights the superiority of the proposed method in dealing with nonlinearity. As the true nature of the relation between the scalar response and the functional covariates is never known in real-world processes, FNNCC has proven to be a more flexible strategy to be recommended in situations where the influence of the functional covariates is not necessarily linear.

Moreover, by means of an extensive Monte Carlo simulation, the FNNCC is compared with other two NN-based monitoring strategies implemented by using an MLP in place of the FNN in step (i) of the monitoring strategy described in Section 4.2 to model the nonlinear relationships between the scalar response and the functional covariate. In particular, in the first control charting strategy, referred to as RawdataMLPCC, the raw discrete values of the functional covariate are directly fed into the MLP (Rossi et al. 2002), whereas the second competitor, namely BsplineMLPCC, involves a pre-processing step of the functional covariates into a vector of scores from its B-spline expansion (Rossi et al. 2005). The two MLPs, with the same set of the FNN hyperparameters, are trained on the training set using the backpropagation algorithm with the Adam optimizer, and the CLs of the two competing methods are estimated on the tuning set. Figures 4-7 show the estimated ARL_1 for the FNNCC, RawdataMLPCC, and BsplineMLPCC for the nonlinear scenarios from B to E, respectively, both in the presence and in the absence of a mean covariate shift. It is clear from these figures that the RawdataMLPCC shows worse performance than the *functional* counterparts as it is not able to account for the functional nature of the covariate. FNNCC and BsplineMLPCC achieve similar performance for all OC scenarios. However, the implementation of the FRCC is to be preferred thanks to its easier interpretability of the functional coefficients. The functional coefficients of the FNN enable visualizing the relationship between functional covariates and the scalar response. In contrast, the weights and intercepts of MLPs are challenging to interpret, and the existing literature on interpreting conventional NNs is primarily limited to computer vision applications.

4.4 Case study: monitoring of HVAC systems on modern passenger trains

The case study mentioned in the Introduction is presented to demonstrate the applicability of the proposed control chart in real situations. In recent years, European regulations have been established to set operational standards for the thermal comfort of passenger rail coaches. These standards, such as EN (2006), were developed to meet the different operating requirements of rail vehicles and to ensure a high-quality air environment for passengers. Given those standards, railway companies are increasingly installing sensing systems to

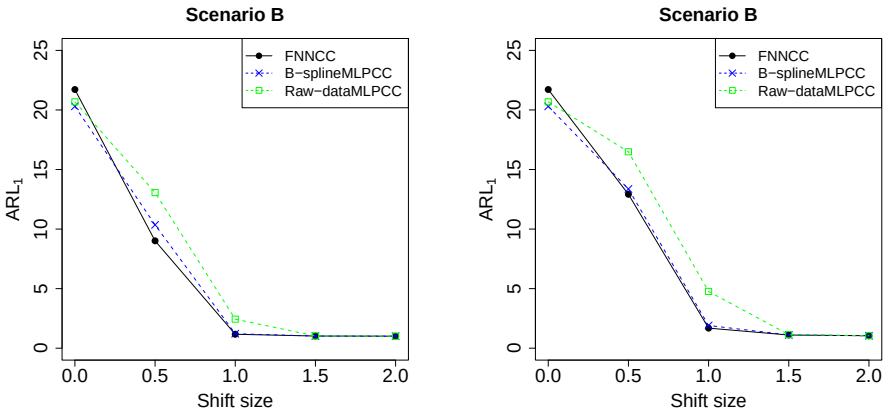


Figure 4.4: Estimated ARL_1 achieved by FNNCC, RawdataMLPCC, and BsplineMLPCC in Scenario B both in the presence (a) and absence (b) of a mean covariate shift, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$.

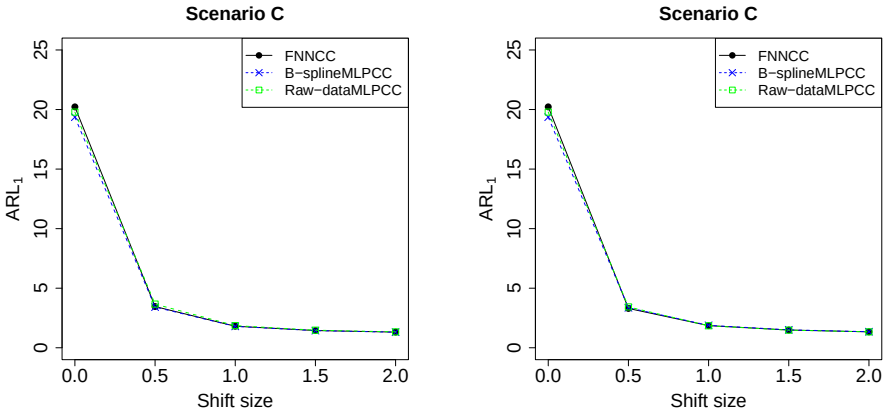


Figure 4.5: Estimated ARL_1 achieved by FNNCC, RawdataMLPCC, and BsplineMLPCC in Scenario C both in the presence (a) and absence (b) of a mean covariate shift, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$.

collect data from onboard HVAC systems.

HVAC systems regulate the interior temperature of each train's coach through a combination of ventilation, heating, and cooling operations. Ventilation is the process of replacing or exchanging interior air with outside air to remove harmful particles like dust, smoke, and bacteria. Heating and cooling, on the other hand, provide warmth or cold inside the coaches. An HVAC system has three main components: the compressor, the condenser, and the thermal expansion valve. The compressor moves the refrigerant gas to the condenser, where the gas changes into a liquid. Then, the liquid refrigerant moves through the evaporator section, where it evaporates into a cold gas, absorbing heat from the surrounding air and

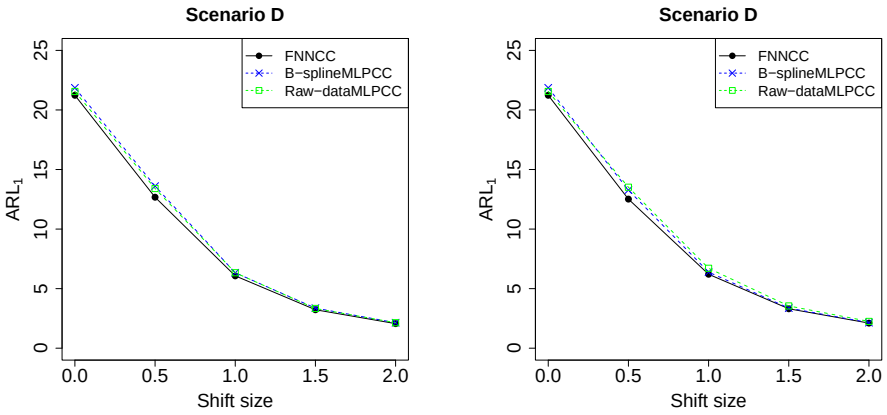


Figure 4.6: Estimated ARL_1 achieved by FNNCC, RawdataMLPCC, and BsplineMLPCC in Scenario D both in the presence (a) and absence (b) of a mean covariate shift, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$.

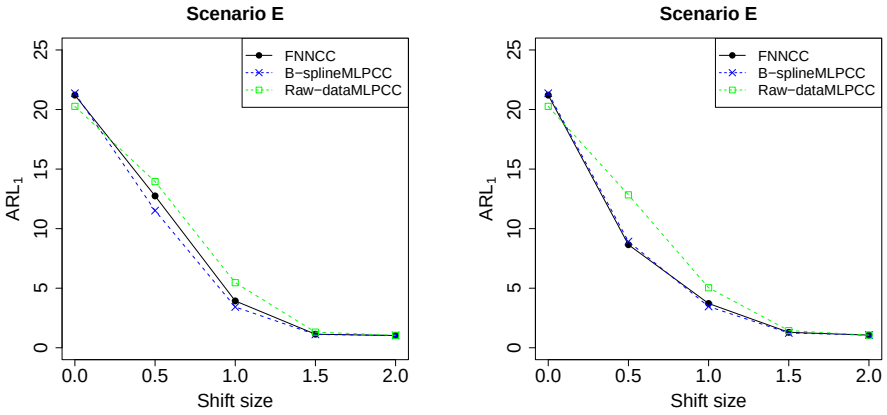


Figure 4.7: Estimated ARL_1 achieved by FNNCC, RawdataMLPCC, and BsplineMLPCC in Scenario E both in the presence (a) and absence (b) of a mean covariate shift, as a function of the mean shift size of the scalar response $\Delta\mu_y = \{0, 0.5, 1, 1.5, 2\}s^{y^*}$.

cooling down the coach interior. Eventually, the thermal expansion valve converts the cold gas back into a liquid, and the process repeats multiple times until the interior temperature reaches the desired level.

The five passenger trains object of this study have six coaches, which are equipped with a dedicated HVAC system, A central unit is installed to control the heating and cooling modes of each HVAC system based on temperature sensors that measure changes in the outside (T_{out}) and interior (T_{in}) temperature signals that are streamed to maintenance engineers for monitoring purposes and potentially improving reliability and maintenance programs. At each time instant, on each coach, the HVAC system activates until T_{in} matches the setpoint

temperature (T_{set}), which is automatically and independently set as a function of T_{out}, to comply with the current regulation on passengers' comfort. The raw measurements of these temperatures are contained in the `HVAC` data set where, for confidentiality reasons, the train names, acquisition year, and routes are omitted. Trains and coaches are then identified with a number from 1 to 5 and 1 to 6, respectively. This case study focuses on one specific route where temperature measurements are acquired at a regular grid of points equally spaced by 30 seconds. Railway engineers confirmed that the 30 HVAC systems can be assumed to be equal and working under the same operating conditions (e.g., outside temperature, and passenger load) on the same route. Thus, the coach effect is not considered hereinafter. Raw measurements are then grouped to form raw profiles referred to different voyages, which are identified by a unique Voyage Number (VN). Based on experts' opinion, exceptional voyages that do not represent normal operating conditions have been removed to define the Phase I sample that is thus formed by 1853 voyages and is randomly split into 740, 186, and 927 voyages to form training, validation, and tuning sets, respectively.

Temperature profiles are re-mapped as a function of the fraction of the total distance traveled by train at each voyage, and, to avoid the modeling of transitional regimes, due to the HVAC restart at the beginning of each voyage, the first 25% of the traveled distance of each profile is discarded. This operation can be regarded as a landmark registration (Ramsay & Silverman 2005, p. 129-132) of the functional data set from the function-specific temporal domain to the common domain $\mathcal{T}$, which, without loss of generality, is set as $[0, 1]$. To get smooth profiles, i.e., functional data observations, we choose a B-spline basis system with 70 basis functions and equally spaced knots estimated by solving a regularization problem with a roughness penalty on the integrated-squared second derivative and smoothing parameter chosen through a generalized cross-validation (Ramsay & Silverman 2005, p. 97- 99).

The scalar quality characteristic of interest is the root mean square of the difference between measurements $T_{\text{in}} and T_{\text{set}}$ acquired during each train voyage, referred to as `DevTemp`. That is, the scalar response, `DevTemp`, is computed once the voyage is ended as engineers cannot perform maintenance operations until the train has finished its voyage or reached the terminal station. `DevTemp` is assumed to be influenced by T_{out} and the T_{set} derivative (with respect to the fraction of the total distance travel), hereinafter denoted by $\dot{T}_{\text{set}}$. The former is included in the model to the extent of accounting for the thermal load under which the system works, whereas the latter, based on experts' opinion, is able to account for the thermal inertia of the process, which is the time the HVAC system needs to allow the T_{in} to match T_{set}. T_{set} may indeed vary as a function of T_{out}, which, in turn, may be subject to rapid changes, e.g., in under and over-ground route segments.

For illustrative purposes, Figure 4.8 displays a random slice of 100 observations from the training sample of the T_{out} and T_{set} derivative profiles. The FNN is trained using a backpropagation algorithm with Adam optimizer and early stopping, with the same set of hyperparameters discussed in the simulation study in Section 4.3. Specifically, we first use the tuning data set to build the FNNCC and estimate the CLs with a type-I error rate $\alpha = 0.05$. 41 additional profiles from `coach 4` of `train 5`, which are known to contain a fault due to a diagnosed failure in one of the two HVAC compressors, are used as Phase II observations. Figure 4.9 shows the Phase II monitoring of the aforementioned voyages. The x-axis label is the VN, while the FNN residuals are reported on the y-axis. Voyage 32 shows an overly large value of the monitoring statistics, thus correctly signaling the OC state of the corresponding HVAC system, which was promptly repaired by train maintenance service. Indeed, subsequent voyages plot inside the CLs, further demonstrating the proposed control chart to properly track both the IC and OC states in practice.

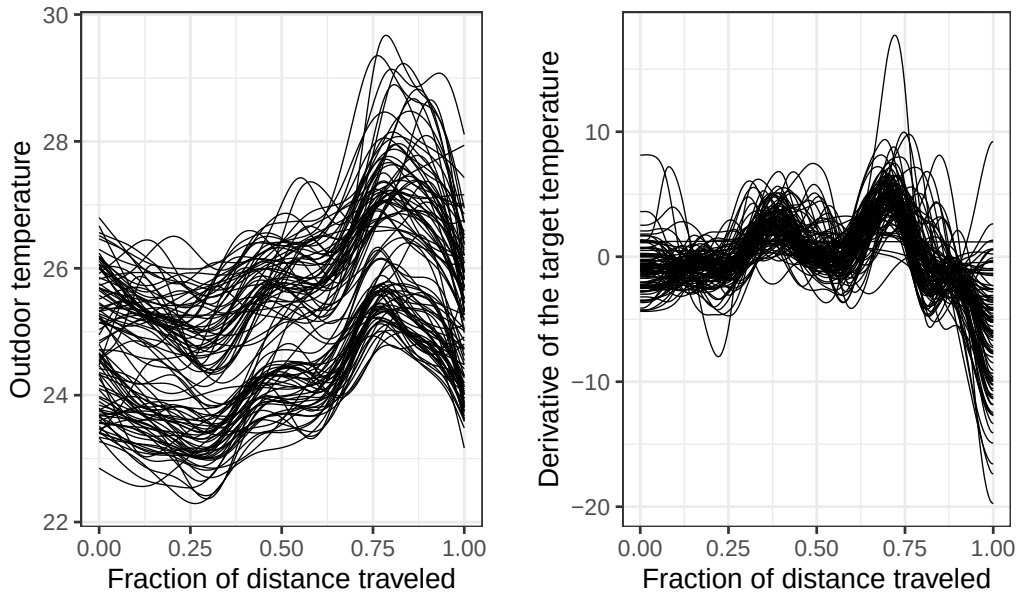


Figure 4.8: 100 random samples of the two functional profiles from the HVAC training set.

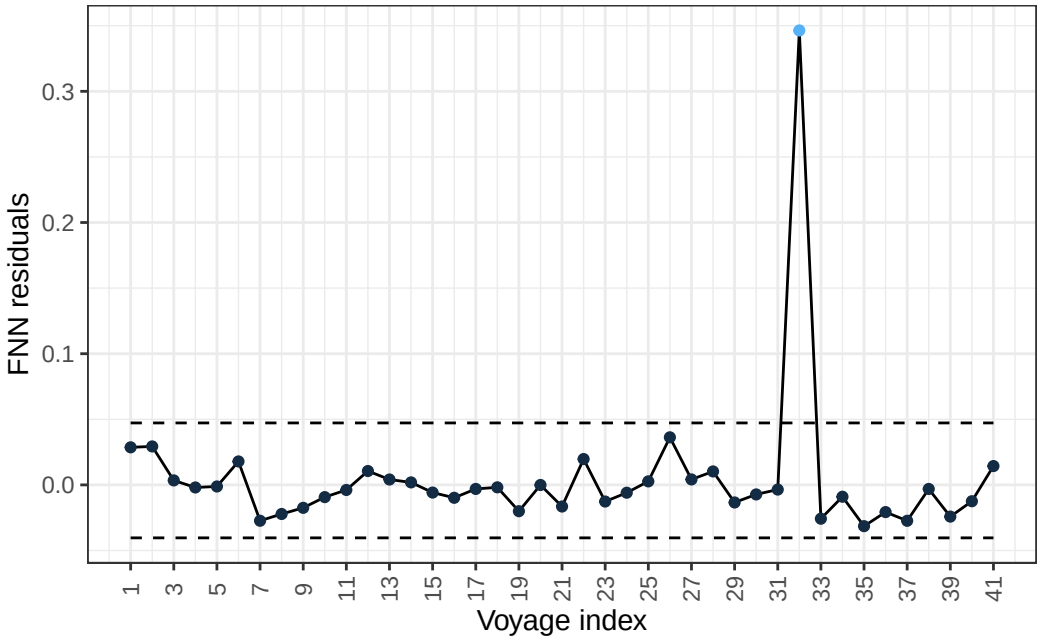


Figure 4.9: Phase II FNNCC. Each point corresponds to a voyage and the values of the corresponding residual are reported. Horizontal dashed lines are the CLs and the point above the CLs denotes the OC observation.

4.5 Conclusions

A novel profile monitoring charting scheme is proposed in this work and referred to as FNNCC. Based on a DL model, the FNNCC can adjust the monitoring of a scalar quality characteristic of interest for the nonlinear effect of the functional covariates when are available. Specifically, the FNNCC relies on FNN, which has recently appeared in the DL literature to allow a NN to learn possibly nonlinear relationships based on covariates in the form of functional data, aka profiles. The residuals obtained from the FNN are elaborated to build a FRCC, which also appeared in the statistical literature, although implemented only under the (functional) linearity assumption. The proposed FNNCC is thus the first DL-based profile monitoring scheme that can efficiently exploit additional information on functional covariates, in a possibly nonlinear fashion.

An extensive Monte Carlo simulation is carried out to assess the performance of the proposed FNNCC in identifying mean shifts in the quality characteristic of interest, which is in a scalar form, in the presence or absence of covariate mean shifts. Then, the FNNCC is compared with the FRCC and the SCC built on the scalar response. The results show that the FNNCC is far better than the two competitors when the relation between the scalar response and the functional covariate is nonlinear. Additionally, the FNNCC is compared with two NN-based control charting strategies and, even though they show similar performance, the FNNCC is preferred due to its interpretability of the functional coefficients. The practical applicability of the proposed control chart is finally illustrated through a case study in the monitoring of HVAC systems installed onboard six coach passenger trains, where the favorable performance of the proposed method in properly tracking the IC and OC states of the process is shown in practice.

The integration of NNs into functional data analysis remains an interesting topic as it efficiently enables nonlinear learning when the covariates are in the form of profiles. Future research can be addressed to extend the FNNCC to different and more sophisticated monitoring statistics.

Chapter 5

R and Python Packages

This chapter introduces the `Python` packages that provide the software for the methodologies proposed in Chapter 1 and 2. The packages are available on GitHub and have also been published on the Python Package Index (PyPI). Finally, the `R` code for the proposed FNNCC is provided to get the same results presented in the case study in Chapter 4.

Releasing an open package, whether it is software, research, or simple code, accelerates innovation as others can build upon your work, promotes transparency, and encourages the fruitful spread of novel and advanced methods among practitioners in the industry.

5.1 The `Python` package `NN4MSP`

The goal of `NN4MSP` is to provide a NN based control charting for MSPs. The methodology is described in Chapter 1. Additionally, a tutorial that shows how to apply the `NN4MSP` package to get the results discussed in the real-case study in Chapter 1 is available online at `https://github.com/unina-sfere/NN4MSP`.

Installation

You can install the development version from GitHub with

```
pip install git+https://github.com/unina-sfere/NN4MSP#egg=NN4MSP
```

or you can install the package using pip

```
pip install NN4MSP
```

Functions

`dataset_generator`() returns the simulated data set, as described in the reference chapter, to train and design the NN, and an array of zeros and ones, corresponding to the negative and positive MSP samples, respectively. The main arguments are:

- **s**: int, number of streams in the MSP model defined in Equation (1.3).
- **k**: int, sample size in the MSP model.
- **num_pos_samples**: int, number of positive samples.

- **num_neg_samples**: int, number of negative samples.
- **loc_res**: float, IC mean of the residuals defined in Equation (1.7)
- **scale_res**: float, IC standard deviation of the residuals.
- **set_seed**: int, random number generator.

`NN_model`() returns the MLP NN classifier. The main arguments are:

- **hidden_activation_function:**: a list containing the activation functions for each hidden layer.
- **num_hidden_layer**: an integer specifying the number of hidden layers in the NN model.
- **num_hidden_neuron**: a list containing the number of neurons in each hidden layer.

`ROC_AUC_plot`() plots the ROC curve and computes them AUC. The main arguments are:

- **classifier**: a *tensorflow* object that contains the trained MLP classifier model.
- **X_val**: array of positive and negative samples to be used to compute the ROC and AUC.
- **y_val**: array of zeros and ones that correspond to the true labels of X_val.
- **figure**: a *figure* object.

`set_cv_alpha`() returns the Type-I error corresponding to the given CV of the neuron in the output layer. The main arguments are:

- **scaler**: a *sklearn* object used to standardize the data by removing the mean and scaling to unit variance.
- **classifier**: *tensorflow* object of the trained MLP classifier model.
- **cv**: float, CV of the neuron in the output layer.

`control_chart`() plots the control chart based on the NN predicted probability. The main arguments are:

- **NN_pred**: a vector of the predicted probability, as returned by the MLP classifier.
- **fig_control_chart**: a *figure* object.
- **CV**: float, the CV of the neuron in the output layer

`load_HVAC_data`() returns the `HVAC` data set used in Chapter 1.

5.2 The `Python` package `NN4OCMSP`

`NN4OCMSP` provides a NN approach capable of identifying the stream or group of streams responsible for the OC alarm in a MSP. The methodology is described in Chapter 2. Additionally, a tutorial that shows how to apply the `NN4OCMSP` package to get the results discussed in the real-case study in Chapter 2 is available online at `https://github.com/unina-sfere/NN4OCMSP`.

Installation

You can install the development version from GitHub with

```
pip install git+https://github.com/unina-sfere/NN4OCMSP#egg=NN4OCMSP
```

or you can install the package using pip

```
pip install NN4OCMSP
```

Functions

`phaseI_estimation`() computes the mean, the common and within-stream variability using standard one-way analysis of variance techniques. The main argument is:

- **data**: IC data to estimate the parameters of the MSP model.

`dataset_generator`() returns the simulated data set, as described in the reference chapter, to train and design the NN for multi-label classification. The main arguments are:

- **s**: int, number of streams in the MSP model.
- **n**: int, sample size in the MSP model.
- **num_samples**: int, number of samples for each simulated OC scenario
- **mu**: int, IC process mean in the MSP model.
- **sigmaA_std**: standard deviation common to all the streams in the MSP model.
- **sigmaE_std**: within-stream standard deviation in the MSP model.

`NN_model`() returns the MLP NN classifier. The main arguments are:

- **hidden_activation_function:**: a list containing the activation functions for each hidden layer.
- **num_hidden_layer**: an integer specifying the number of hidden layers in the NN model.
- **num_hidden_neuron**: a list containing the number of neurons in each hidden layer.

`range_overall_mean`() computes the range and the overall mean for each sample. The main argument is:

- **data**: data to compute the range and the mean.

`prediction`() returns the label vectors predicted by the trained MLP classifier model. The main arguments are:

- **data**: data to be classified.
- **classifier**: a *tensorflow* object containing the trained MLP classifier model.
- **scaler**: a *sklearn* object used to standardize the data by removing the mean and scaling to unit variance.

- **overall_mean**: array, overall mean for each sample as returned by `range_overall_mean`.
- **sample_range**: array, range for each sample as returned by `range_overall_mean`.

`accuracy`() returns the Hamming score in a multi-label classification. The main arguments are:

- **y_true**: an array of the ground truth (correct) labels.
- **y_pred**: an array of the predicted labels, as returned by `prediction`.

`control_charts`() plots the range and the overall mean control charts. The main arguments are:

- **overall_mean**: an array of the overall mean for each sample returned by `range_overall_mean`.
- **sample_range**: an array of the range values for each sample returned by `range_overall_mean`.
- **fig_control_chart**: a *figure* object.
- **alpha_sim**: an integer specifying the Type-I error of the range and the overall mean control charts.

`load_HVAC_data`() returns the `HVAC` data set used in Chapter 2.

5.3 Functional neural network control chart

The goal of the proposed FNNCC is to adjust the monitoring of a scalar quality characteristic by the effect of functional covariates. The `R` code used to get the results presented in the case study in Chapter 4 is provided below and is also available online at `https://github.com/unina-sfere/FNNCC`. We are currently working on integrating the FNNCC control chart in the `R` package `funcharts` (Capezza et al. 2023*a*), already available on `R` software repository (CRAN), which provides functional control charts for SPC of multivariate functional data. It is worth noting that FNNCC can be applied to any industrial problem where a scalar variable depends on functional covariates.

An application to HVAC data

Import the required libraries and load the `HVAC` data.

```
# Library

library(funcharts)
library(fda)
library(tidyverse)
library(lubridate)
library(stringr)
library(FuncNN)
library("keras")
library("latex2exp")
library(ggpubr)

# Import HVA data set
load("HVAC_data_set.RData")
```

Let us start from the Phase I data. To get smooth profiles, we choose a B-spline basis system with 70 basis functions and equally spaced knots estimated by solving a regularization problem with a roughness penalty on the integrated-squared second derivative and smoothing parameter chosen through a generalized cross-validation. Then, the scalar quality characteristic is computed for each train voyage.

```
loglam <- seq(-4,0,0.25)
lambda_grid <- 10^loglam

mfdobj_phaseI <- HVAC_dataset %>%
filter(Phase == "Phase I") %>%
arrange(Train, VN, Time) %>%
filter(Percent_distance >= 0.25) %>% # the first 25% of the traveled
  distance of each profile is discarded
group_by(VN) %>%
mutate(Percent_distance = (Percent_distance - 0.25)/0.75) %>%
ungroup() %>%
mutate(VN = factor(VN)) %>%
get_mfd_df(domain = c(0, 1),
arg = "Percent_distance",
id = "VN",
lambda_grid = lambda_grid, #seq(0.05, 0.2, length=5)
variables = c("coach_OutdoorTemp",
"coach_SetPointTemp"),
n_basis = 70) # 70

# Derivative of the target temperature profile

dev_setpoint <-  deriv.fd(mfdobj_phaseI[,"coach_SetPointTemp"])

# scalar quality characteristic

scalar_train <- HVAC_dataset %>%
filter(Phase == "Phase I") %>%
filter(VN %in% unique(mfdobj_phaseI[["raw"]][["VN"]])) %>%
arrange(Train, VN, Time) %>%
filter(Percent_distance >= 0.25) %>%
select(DeltaTemp,VN,n) %>%
group_by(VN) %>%
mutate(dev_deltaTemp =sqrt(sum(DeltaTemp^2))/n) %>%
summarise(min = min(dev_deltaTemp)) %>% pull()
```

Plot of a random slice of 100 observations of Phase I temperature profiles

```
set.seed(3)
rows <- sample(1:dim(mfdobj_phaseI$coefs)[2], 100)

p1<- plot_mfd(mfdobj_phaseI[rows,"coach_OutdoorTemp"]) +
geom_line(mfdobj_phaseI = mfdobj_phaseI[rows], lty = 2, type_mfd = "raw",
  col = "darkgreen") +
xlab("Fraction of distance traveled") +
ylab("Outdoor temperature")
p1

dev_setpoint_plot <- get_mfd_fd(dev_setpoint)

set.seed(3) # 36
rows <- sample(1:dim(mfdobj_phaseI$coefs)[2], 100)
```

```
p2<- plot_mfd(dev_setpoint_plot[rows,]) +
geom_line(dev_setpoint_plot = dev_setpoint_plot[rows], lty = 2, type_mfd =
  "raw", col = "darkgreen") +
xlab("Fraction of distance traveled") +
ylab("Derivative of the target temperature")
p2

p<- ggarrange(p1,p2, ncol = 2, nrow = 1)
p
```

Phase I data are split into train/tuning set

```
voyage_id_PhaseI <- as.vector(unique( mfdobj_phaseI[["raw"]][["VN"]]))
set.seed(1)
voyage_id_train <- sample(voyage_id_PhaseI, size = length(voyage_id_PhaseI)
  /2,
replace = FALSE)

voyage_id_tun <- setdiff(voyage_id_PhaseI, voyage_id_train)
```

Let us define the Phase II observations.

```
mfdobj_test <- HVAC_dataset %>%
filter(Phase == "Phase II") %>%
filter(Percent_distance >= 0.25) %>% # the first 25% of the traveled
  distance of each profile is discarded
group_by(VN) %>%
mutate(Percent_distance = (Percent_distance - 0.25)/0.75) %>%
ungroup() %>%
mutate(VN = factor(VN)) %>%
get_mfd_df(domain = c(0, 1),
arg = "Percent_distance",
id = "VN",
lambda_grid = lambda_grid, #seq(0.05, 0.2, length=5)
variables = c("coach_OutdoorTemp",
"coach_SetPointTemp"),
n_basis = 70) # 70

dev_setpoint_test <-  deriv.fd(mfdobj_test[,"coach_SetPointTemp"])

scalar_test <- HVAC_dataset %>%
filter(Phase == "Phase II") %>%
filter(Percent_distance >= 0.25) %>%
group_by(VN) %>%
mutate(dev_deltaTemp =sqrt(sum(DeltaTemp^2))/n) %>%
summarise(min = min(dev_deltaTemp)) %>% pull()
```

The training of the FNN requires a pre-processing phase. In fact, the functional covariates need to be transformed in a tensor object to be processed by the FNN. In contrast, the scalar quality characteristic is simpler to handle and can be input as a vector. Once this step is done, the FNN is ready to be trained using the training set.

```
# Setting up tensor for fFNN
n_basis <- 70
data_fnnI <- array(dim = c(n_basis, length(voyage_id_train), 2))
```

```
data_fnnI[,,1] <- mfdobj_phaseI$coefs[,voyage_id_train,1]
data_fnnI[,,2] <- drop(dev_setpoint[["coefs"]])[,voyage_id_train]

data_fnnI_tun = array(dim = c(n_basis, length(voyage_id_tun), 2))
data_fnnI_tun[,,1] <- mfdobj_phaseI$coefs[,voyage_id_tun,1]
data_fnnI_tun[,,2] <- drop(dev_setpoint[["coefs"]])[,voyage_id_tun]

data_fnnII = array(dim = c(n_basis, dim(mfdobj_test$coefs)[2], 2))
data_fnnII[,,1] <-  mfdobj_test$coefs[,,1]
data_fnnII[,,2] <- drop(dev_setpoint_test[["coefs"]])

# FNN training

seed <- 1
set.seed(seed)
tensorflow::set_random_seed(seed)
fnn_sim <- fnn.fit(resp = scalar_train[voyage_id_PhaseI %in% voyage_id_
  train],
func_cov = data_fnnI,
scalar_cov = NULL,
basis_choice = c("bspline"),
num_basis = c(5),
hidden_layers = 2,
neurons_per_layer = c(8,8), # (16,16); (8,4)
activations_in_layers = c("relu", "linear"),
domain_range = list(c(0, 1)),
epochs = 250,
loss_choice = "mse",
metric_choice = list("mean_squared_error"),
val_split = 0.2,
patience_param = 25,
learn_rate = 0.001,
early_stop = T)
```

Finally, the control chart based on the residuals obtained by the FNN can be plotted to monitor the Phase II observations.

```
predictions <- fnn.predict(model = fnn_sim,
func_cov = data_fnnI_tun,
scalar_cov = NULL,
basis_choice = c("bspline"),
num_basis = c(5),
domain_range = list(c(0, 1))
)

# Control limit estimation

res_tun <- scalar_train[voyage_id_PhaseI %in% voyage_id_tun] - predictions

UCL_FNN_np <- quantile(res_tun, probs = 1 - 0.05/2)
UCL_FNN_np
LCL_FNN_np <- quantile(res_tun, probs = 0.05/2)
LCL_FNN_np


predictionsII <- fnn.predict(model = fnn_sim,
func_cov = data_fnnII,
scalar_cov = NULL,
basis_choice = c("bspline"),
```

```
num_basis = c(5),
domain_range = list(c(0, 1))
)

res <-  scalar_test - predictionsII # residuals

# Plot control chart

rows <- 50:90

df_test <- cbind(res[rows], seq(1, length(res[rows])),
res[rows] > UCL_FNN_np | res[rows] < LCL_FNN_np)
df_test  <- as.data.frame(df_test)
names(df_test) <- c("res", "id", "oc")


FNNCC_plot <- ggplot(df_test, aes(x = id, y = res)) +
geom_line() +
geom_point(aes(colour = oc)) +
geom_blank(aes(y = 0)) + geom_line(aes(y = UCL_FNN_np),
lty = 2) + geom_line(aes(y = LCL_FNN_np), lty = 2) +
theme_bw() + theme(axis.text.x = element_text(angle = 90,
vjust = 0.5)) + theme(legend.position = "none",
plot.title = element_text(hjust = 0.5)) +
xlab("Voyage index") + ylab(expression("FNN residuals"))
#ggtitle("FNNCC")

FNNCC_plot + scale_x_continuous(breaks = seq(1, nrow(df_test), by = 2))
```

Conclusion and future developments

In this dissertation, methods for SPC based on NNa have been investigated. The proposed methodologies have also been shown to be capable of promptly alerting railway maintenance engineers when an anomaly occurs in one or more HVAC systems installed on modern passenger trains. In particular, in the first part of this thesis, the simultaneous SPC of the sensor signals coming from each HVAC system installed on each train coach are regarded as produced by a MSP. In this context, Chapter 1 presents a new control charting framework based on NNa to improve the detection of changes in multiple streams. Whereas, Chapter 2 addresses the challenge of identifying OC streams after an OC state has been detected by a MSP control chart. Again stimulated by the railway case study, the last two chapters of this thesis present an application of a nonparametric SPC approach based on AEs and a new profile monitoring strategy based on a functional NN to monitor a scalar quality characteristic when functional covariates are available. Eventually, we provide open-source software packages that are not only applicable to the railway transportation industry but also generalizable to any industrial process monitoring scenario where (i) the process can be regarded as a MSP or (ii) a scalar quantity characteristic depends on a set of functional covariates.

The research presented in this thesis can be expanded upon in several directions, as outlined below.

A Compositional data approach for the statistical process control of multiple stream processes

The following future trajectory, which is part of an early stage research with Biagio Palumbo and Antonio Lepore from the University of Naples Federico II, aims at providing a novel monitoring strategy for MSP (Mortell & Runger 1995) through a compositional data (CoDa) approach (Aitchison 1994). This research is motivated by a case study from the railway industry in monitoring HVAC systems installed onboard passenger trains. In particular, we are interested in the number of times the *cooling mode status* C_{m} is equal to 1 in one hour's worth of data, where C_{m} is coded by 1 if the HVAC system is cooling at a given time instant, and 0 otherwise. Let $p_t = [p_{t1}, \ldots, p_{tj}, \ldots, p_{ts}]$ denote the vector of proportions of $C_{\mathrm{m}} = 1$ in one hour for each stream\coach $j = 1, 2, \ldots, s$ at a given time t. The idea is to monitor whether or not an HVAC system is cooling properly to ensure the passenger thermal comfort in compliance with the European regulations (EN 2006). That is, when an HVAC system j returns a p_{tj} that is significantly different from that of the other streams, then an alarm should be provided at time t.
To monitor this kind of process, Wludyka & Jacobs (2002*a*,*b*) proposed a Chi-square control chart and a new version of the group control chart (Boyd 1950) with both the one-sided

and two-sided runs test versions (Mortell & Runger 1995). They also assumed that p_{tj} are distributed as *independent binomial* random variable. However, the independence assumption is rarely met in modern complex processes, as well as in the HVAC data.
Stimulated by this challenge, we have started investigating an SPC approach for a MSP based on CoDa analysis techniques, which the quality characteristics, p_t, belong to. CoDa consists of vectors of non-negative components that are subject to a unit-sum constraint and play an important role in many disciplines, e.g., demography, economy, chemistry (Aitchison 1994). CoDa is often represented as vectors of proportions, percentages, concentrations, or frequencies. Typical examples of CoDa include the proportion of different chemical elements in a mineral, the concentration of different cell types in a patient's blood sample, the proportion of different nutrients in a beverage, the proportion of time spent on different tasks (Van den Boogaart & Tolosana-Delgado 2013). Traditional multivariate techniques (Johnson et al. 2002), which are designed for unconstrained data, cannot be used due to unity-sum constraint. Therefore, the statistics typically used in the context of the multivariate SPC (Lowry & Montgomery 1995, Bersimis et al. 2007) are thus developed and extended to the case of CoDa, by designing an appropriate compositional T^2 control chart in a suitable mathematical space, referred to as CoDa T^2 control chart (Vives-Mestres et al. 2014*a*,*b*, Vives-Mestres, Daunis-i Estadella & Martín-Fernández 2016). The CoDa T^2 control chart is based on the isometric log-ratio (ilr) transformation of the data (Egozcue et al. 2003) to which the standard T^2 control chart (Hotelling 1947) is applied (Vives-Mestres, Martín-Fernández & Kenett 2016). Furthermore, Tran et al. (2018) and Imran et al. (2022) extended the multivariate EWMA and CUSUM control chart to CoDa, respectively, based on the ilr transformation to eliminate the constant sum constraint.
With reference to the HVAC monitoring problem, the vector p_t can be monitored against time t through a CoDa T^2 control chart to detect any change in the vector composition, that is, in the relative values of the components.
The preliminary results are promising enough to encourage further effort in this direction.

Non-linear Function-on-Function regression control chart

Chapter 4 extends the work of Capezza et al. (2020, 2023*b*), proposing a control charting procedure able to monitor a *scalar* quality characteristic adjusted by the effect, possibly nonlinear, of one or more functional covariates through a functional NN. This methodology is motivated by the interest of the railway maintenance engineers in the cumulative deviation of the interior temperature from the setpoint temperature set by European regulations (EN 2006) at the end of each train voyage. However, future research can be addressed by extending the work of Centofanti et al. (2021) where the quality characteristic, which is in a functional form as well, is adjusted for the effects of the functional covariates through a Function-On-Function (FOF) linear regression model (Ramsay & Silverman 2005) and then monitored by using jointly the Hotelling's T^2 and the SPE control charts built on its FPCA decomposition (Pini et al. 2018, Grasso et al. 2017).
Recently, Rao & Reimherr (2023) introduces a modern non-linear acFOF regression using NNa, namely *Functional Direct NN* (FFDNN), for properly handling functional responses with multiple functional covariates. Therefore, a NN-based FRCC based on the residuals obtained from a FFDNN can be used to allow for a non-linear and more flexible regression of the quality characteristic profile on the functional covariates. Moreover, different residual monitoring strategies deserve further investigation. For instance, the residuals can be monitored by using the novel profile monitoring approach introduced by Sergin & Yan (2021)

that propose a new formulation of the SPE statistics built on the residual space learned by a VAE (Goodfellow et al. 2016), which can possibly learn a non-linear low-dimensional manifold.

Non-linear regression control chart for functional time series

In Chapter 4, the observations of the scalar response variable are assumed to be conditionally independent given the functional covariates. We believe that, conditionally on the covariate levels, the main temporal dependence structure is within individual voyages, and that this structure can be adequately represented by treating the data as profiles. However, this assumption is frequently violated in practice, and the presence of autocorrelation has a serious impact on the performance of control charts, causing a dramatic increase in the frequency of false alarms (Montgomery & Mastrangelo 1991).
Functional time series techniques (Kokoszka & Reimherr 2017, Horváth et al. 2014) can potentially capture the temporal relationships between voyages, thereby enhancing prediction and monitoring capabilities. Moreover, deep NNa have proved to be powerful and are achieving high accuracy in predicting time series (Ismail Fawaz et al. 2019, Torres et al. 2021), and a natural extension to functional data could be beneficial to better capture the autocorrelation structure.

Real-time Monitoring of Functional Data

Most of the profile monitoring methods aim at assessing the stability of functional quality characteristics that are completely observed (Centofanti et al. 2021, Capezza et al. 2020, 2023*b*). In the railway case study presented in Chapter 4, the main interest is in the cumulative deviation computed once the voyage is ended as railway engineers cannot perform maintenance operations until the train has finished its voyage or reached the terminal station. However, in some industrial processes, the interest relies on understanding if the process is working properly before its completion, that is, in detecting if assignable sources of variations (i.e., special causes) are present while the process is still running.
To this aim, Centofanti et al. (2022) propose a new method, namely functional real-time monitoring (FRTM), that is able to monitor a functional quality characteristic in real-time by taking into account both phase and amplitude variations through the following steps: (i) registration of the partially observed functional data to the appropriate reference curve; (ii) dimensionality reduction; (iii) monitoring of the partially observed functional quality characteristic in the reduced space. In the NN literature, there are methods based on CNN (Goodfellow et al. 2016) that may not require any alignments due to the translation equivariant property of the convolutional layers (Sergin & Yan 2021, Yu & Liu 2022), and, thus, can be used to solve the functional real-time monitoring problem. An effort can be made in this direction to improve the detection power of FRTM.

Acknowledgements

I am extremely grateful to the Operation Service and Maintenance Product Evolution Department of Hitachi Rail S.p.A. and, in particular, to engineers Giuseppe Giannini, Vincenzo Criscuolo and Guido Cesaro for their technological insights in the interpretation of results.

The computing resources and the related technical support used for this thesis work have been provided by CRESCO/ENEAGRID High-Performance Computing infrastructure and its staff (Iannone et al. 2019). CRESCO/ENEAGRID High-Performance Computing infrastructure is funded by ENEA, the Italian National Agency for New Technologies, Energy and Sustainable Economic Development, and by Italian and European research programmes, see `http://www.cresco.enea.it/english` for information.

Bibliography

Abadi, M., Agarwal, A. et al. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems', `http://tensorflow.org/`. Software available from tensorflow.org.

Abramowitz, M., Stegun, I. A. et al. (1964), *Handbook of Mathematical Functions*, Vol. 55, Dover New York.

Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. (1985), 'A learning algorithm for boltzmann machines', *Cognitive Science* **9**(1), 147–169.

Ahmadi-Javid, A. & Ebadi, M. (2020), 'A two-step method for monitoring normally distributed multi-stream processes in high dimensions', *Quality Engineering* **33**(1), 143–155.

Aitchison, J. (1994), 'Principles of compositional data analysis', *Lecture Notes-Monograph Series* pp. 73–81.

Allen Pugh, G. (1991), 'A comparison of neural networks to SPC charts', *Computers & Industrial Engineering* **21**, 253–255.

Ambrosino, F., Giannini, G., Lepore, A., Palumbo, B. & Sposito, G. (2021), Neural network for the statistical process control of hvac systems in passenger rail vehicles, *in* 'Convegno della Società Italiana di Statistica', Springer, pp. 393–406.

Amin, R. W., Wolff, H., Besenfelder, W. & Baxley Jr, R. (1999), 'EWMA control charts for the smallest and largest observations', *Journal of Quality Technology* **31**(2), 189–206.

Aparisi, F., Avendaño, G. & Sanz, J. (2006), 'Techniques to interpret T^2 control chart signals', *IIE Transactions* **38**(8), 647–657.

Bajaria, H. & Skog, F. (1994), 'Realistic multistream process charting', *Quality* **33**(12), 36.

Baldi, P. & Hornik, K. (1989), 'Neural networks and principal component analysis: Learning from examples without local minima', *Neural Networks* **2**(1), 53–58.

Barbeito, I., Zaragoza, S., Tarrío-Saavedra, J. & Naya, S. (2017), 'Assessing thermal comfort and energy efficiency in buildings by statistical quality control for autocorrelated data', *Applied energy* **190**, 1–17.

Bersimis, S., Psarakis, S. & Panaretos, J. (2007), 'Multivariate statistical process control charts: an overview', *Quality and Reliability Engineering International* **23**(5), 517–543.

Bersimis, S., Sgora, A. & Psarakis, S. (2017), 'Methods for interpreting the out-of-control signal of multivariate control charts: A comparison study', *Quality and Reliability Engineering International* **33**(8), 2295–2326.

Bersimis, S., Sgora, A. & Psarakis, S. (2022), 'A robust meta-method for interpreting the out-of-control signal of multivariate control charts using artificial neural networks', *Quality and Reliability Engineering International* **38**(1), 30–63.

Bishop, C. M. (1995), *Neural networks for pattern recognition*, Oxford University Press.

Boyd, D. (1950), 'Applying the group control chart for X and R', *Industrial Quality Control* **3**, 22–25.

Bühlmann, P. & Künsch, H. R. (1999), 'Block length selection in the bootstrap for time series', *Computational Statistics & Data Analysis* **31**(3), 295–310.

Capezza, C., Centofanti, F., Lepore, A., Menafoglio, A., Palumbo, B. & Vantini, S. (2023*a*), 'funcharts: control charts for multivariate functional data in R', *Journal of Quality Technology* pp. 1–18.

Capezza, C., Centofanti, F., Lepore, A., Menafoglio, A., Palumbo, B. & Vantini, S. (2023*b*), 'Functional regression control charts with an application to ship fuel consumption monitoring', *Wiley StatsRef: Statistics Reference Online* pp. 1–10.

Capezza, C., Lepore, A., Menafoglio, A., Palumbo, B. & Vantini, S. (2020), 'Control charts for monitoring ship operating conditions and CO2 emissions based on scalar-on-function regression', *Applied Stochastic Models in Business and Industry* **36**(3), 477–500.

Centofanti, F., Lepore, A., Kulahci, M. & Spooner, M. P. (2022), 'Real-time monitoring of functional data', *arXiv preprint arXiv:2205.06256* .

Centofanti, F., Lepore, A., Menafoglio, A., Palumbo, B. & Vantini, S. (2021), 'Functional regression control chart', *Technometrics* **63**(3), 281–294.

Chen, S., Yu, J. & Wang, S. (2020), 'Monitoring of complex profiles based on deep stacked denoising autoencoders', *Computers & Industrial Engineering* **143**, 106402.

Cheng, C.-S. & Cheng, H.-P. (2011), 'Using neural networks to detect the bivariate process variance shifts pattern', *Computers & Industrial Engineering* **60**(2), 269–278.

Chollet, F. (2015), 'Keras', `https://keras.io`.

Chou, S.-H., Chang, S. I. & Tsai, T.-R. (2014), 'On monitoring of multiple non-linear profiles', *International Journal of Production Research* **52**(11), 3209–3224.

Colosimo, B. M., del Castillo, E., Jones-Farmer, L. A. & Paynabar, K. (2021), 'Artificial intelligence and statistics for quality technology: An introduction to the special issue', *Journal of Quality Technology* **53**(5), 443–453.

Colosimo, B. M. & Pacella, M. (2010), 'A comparison study of control charts for statistical monitoring of functional data', *International Journal of Production Research* **48**(6), 1575–1601.

Company, W. E. (1958), *Statistical quality control handbook*, Western Electric.

Conan-Guez, B. & Rossi, F. (2002), Multi-layer perceptrons for functional data analysis: a projection based approach, *in* 'Artificial Neural Networks - International Conference Madrid, Spain, August 28–30, 2002 Proceedings', Springer, pp. 667–672.

Cybenko, G. (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals and Systems* **2**(4), 303–314.

Dehnad, K. (1987), 'Density estimation for statistics and data analysis'.

Dunn, O. J. (1958), 'Estimation of the means of dependent variables', *The Annals of Mathematical Statistics* pp. 1095–1111.

Dunson, D. B. (2018), 'Statistics in the big data era: Failures of the machine', *Statistics & Probability Letters* **136**, 4–9.

Egozcue, J. J., Pawlowsky-Glahn, V., Mateu-Figueras, G. & Barcelo-Vidal, C. (2003), 'Isometric logratio transformations for compositional data analysis', *Mathematical Geology* **35**(3), 279–300.

EN, B. (2006), 'UNI-EN 14750-1: Railway applications—air conditioning for urban and suburban rolling stock. part 1: Comfort parameters', *British standard. London: British Standards Institution* .

Epprecht, E. K. (2015), Statistical control of multiple-stream processes: A literature review, *in* 'Frontiers in Statistical Quality Control', Springer, pp. 49–64.

Epprecht, E. K., Barbosa, L. F. M. & Simões, B. F. T. (2011), 'SPC of multiple stream processes: a chart for enhanced detection of shifts in one stream', *Production* **21**(2), 242–253.

Fan, F.-L., Xiong, J., Li, M. & Wang, G. (2021), 'On interpretability of artificial neural networks: A survey', *IEEE Transactions on Radiation and Plasma Medical Sciences* **5**(6), 741–760.

Fawcett, T. (2006), 'An introduction to ROC analysis', *Pattern recognition letters* **27**(8), 861–874.

Ferraty, F. (2006), *Nonparametric functional data analysis*, Springer.

Ferrer, A. (2014), 'Latent structures-based multivariate statistical process control: A paradigm shift', *Quality Engineering* **26**(1), 72–91.

Flores, M., Fernández-Casal, R., Naya, S. & Tarrío-Saavedra, J. (2021), 'Statistical quality control with the qcr package', *R Journal* **13**(1), 194–217.

Funahashi, K.-I. (1989), 'On the approximate realization of continuous mappings by neural networks', *Neural Networks* **2**(3), 183–192.

Glorot, X. & Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* 'Proceedings of the thirteenth International Conference on Artificial Intelligence and Statistics', JMLR Workshop and Conference Proceedings, pp. 249–256.

Godbole, S. & Sarawagi, S. (2004), Discriminative methods for multi-labeled classification, *in* 'Pacific-Asia Conference on Knowledge Discovery and Data Mining', Springer, pp. 22–30.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. (2016), *Deep learning*, Vol. 1, MIT Press Cambridge.

Grasso, M., Colosimo, B. M. & Tsung, F. (2017), 'A phase I multi-modelling approach for profile monitoring of signal data', *International Journal of Production Research* **55**(15), 4354–4377.

Grimshaw, S. D., Rex Bryce, G. & Meade, D. J. (1999), 'Control limits for group charts', *Quality Engineering* **12**(2), 177–184.

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J. & Seung, H. S. (2000), 'Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit', *Nature* **405**(6789), 947–951.

Han, J. & Moraga, C. (1995), The influence of the sigmoid function parameters on the speed of backpropagation learning, *in* 'From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks Malaga-Torremolinos, Spain, June 7–9, 1995', Springer, pp. 195–201.

Härdle, W., Horowitz, J. & Kreiss, J.-P. (2003), 'Bootstrap methods for time series', *International Statistical Review* **71**(2), 435–459.

Hastie, T., Tibshirani, R., Friedman, J. H. & Friedman, J. H. (2009), *The elements of statistical learning: data mining, inference, and prediction*, Vol. 2, Springer.

He, K., Zhang, X., Ren, S. & Sun, J. (2015), Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *in* 'Proceedings of the IEEE International Conference on Computer Vision', pp. 1026–1034.

He, Q. P. & Wang, J. (2018), 'Statistical process monitoring as a big data analytics tool for smart manufacturing', *Journal of Process Control* **67**, 35–43.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural Computation* **9**(8), 1735–1780.

Hornik, K., Stinchcombe, M. & White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* **2**(5), 359–366.

Horváth, L., Kokoszka, P. & Rice, G. (2014), 'Testing stationarity of functional time series', *Journal of Econometrics* **179**(1), 66–82.

Hotelling, H. (1947), 'Multivariate quality control-illustrated by the air testing of sample bombsights', *Techniques of Statistical Analysis* pp. 11–184.

Howard, P., Apley, D. W. & Runger, G. (2018), 'Identifying nonlinear variation patterns with deep autoencoders', *IISE Transactions* **50**(12), 1089–1103.

Hsing, T. & Eubank, R. (2015), *Theoretical foundations of functional data analysis, with an introduction to linear operators*, Vol. 997, John Wiley & Sons.

Hwarng, H. B. (2014), *Neural Networks in Statistical Process Control*, Wiley Online Library.

Iannone, F., Ambrosino, F. et al. (2019), Cresco enea hpc clusters: A working example of a multifabric gpfs spectrum scale layout, *in* '2019 International Conference on High Performance Computing & Simulation (HPCS)', IEEE, pp. 1051–1052.

Imran, M., Sun, J., Zaidi, F. S., Abbas, Z. & Nazir, H. Z. (2022), 'Multivariate cumulative sum control chart for compositional data with known and estimated process parameters', *Quality and Reliability Engineering International* **38**(5), 2691–2714.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. & Muller, P.-A. (2019), 'Deep learning for time series classification: a review', *Data Mining and Knowledge Discovery* **33**(4), 917–963.

Jackson, J. E. & Mudholkar, G. S. (1979), 'Control procedures for residuals associated with principal component analysis', *Technometrics* **21**(3), 341–349.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013), *An introduction to statistical learning*, Vol. 112, Springer.

Japkowicz, N., Hanson, S. J. & Gluck, M. A. (2000), 'Nonlinear autoassociation is not equivalent to pca', *Neural Computation* **12**(3), 531–545.

Jin, J. & Shi, J. (1999), 'Feature-preserving data compression of stamping tonnage information using wavelets', *Technometrics* **41**(4), 327–339.

Jirasettapong, P. & Rojanarowan, N. (2011), 'A guideline to select control charts for multiple stream processes control', *Engineering Journal* **15**(3), 1–14.

Johnson, R. A., Wichern, D. W. et al. (2002), *Applied multivariate statistical analysis*, Vol. 5, Prentice Hall Upper Saddle River, NJ.

Jolliffe, I. (2005), 'Principal component analysis', *Encyclopedia of Statistics in Behavioral Science* .

Jolliffe, I. T. & Cadima, J. (2016), 'Principal component analysis: a review and recent developments', *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences* **374**(2065), 20150202.

Keskar, N. S. & Socher, R. (2017), 'Improving generalization performance by switching from adam to sgd', *arXiv preprint arXiv:1712.07628* .

Kim, Y. & Ra, J. (1991), Weight value initialization for improving training speed in the backpropagation network, *in* 'Proceedings 1991 IEEE International Joint Conference on Neural Networks', IEEE, pp. 2396–2401.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Kokoszka, P. & Reimherr, M. (2017), *Introduction to functional data analysis*, CRC Press.

Kresta, J. V., Macgregor, J. F. & Marlin, T. E. (1991), 'Multivariate statistical monitoring of process operating performance', *The Canadian Journal of Chemical Engineering* **69**(1), 35–47.

Kulahci, M., Lepore, A., Palumbo, B. & Sposito, G. (2023), 'Functional neural network control chart', *arXiv preprint arXiv:2311.11050* .

Lahiri, S. N. (1999), 'Theoretical comparisons of block bootstrap methods', *Annals of Statistics* pp. 386–404.

Lanning, J. W., Montgomery, D. C. & Runger, G. C. (2002), 'Monitoring a multiple stream filling operation using fractional samples', *Quality Engineering* **15**(2), 183–195.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Lee, S., Kwak, M., Tsui, K.-L. & Kim, S. B. (2019), 'Process monitoring using variational autoencoder for high-dimensional nonlinear processes', *Engineering Applications of Artificial Intelligence* **83**, 13–27.

Lehmann, E. L., Romano, J. P. & Casella, G. (2005), *Testing statistical hypotheses*, Vol. 3, Springer.

Lepore, A., Palumbo, B. & Poggi, J.-M. (2022), *Interpretability for Industry 4.0: Statistical and Machine Learning Approaches*, Springer Nature.

Lepore, A., Palumbo, B. & Sposito, G. (2021), 'Github repository for the paper neural network based control charting for multiple stream processes with an application to HVAC systems in passenger railway vehicle', `https://github.com/unina-sfere/NN4MSP`.

Lepore, A., Palumbo, B. & Sposito, G. (2022*a*), 'Neural network based control charting for multiple stream processes with an application to HVAC systems in passenger railway vehicles', *Applied Stochastic Models in Business and Industry* **38**(5), 862–883.

Lepore, A., Palumbo, B. & Sposito, G. (2022*b*), 'NN4MSP: Neural network for multiple stream processes. python package version 2.28', `https://pypi.org/project/NN4MSP/`.

Lepore, A., Palumbo, B. & Sposito, G. (2022*c*), 'NN4OCMSP: Neural network for out-of-control multiple stream process. python package version 1.1.1', `https://pypi.org/project/NN4OCMSP/`.

Lin, M., Chen, Q. & Yan, S. (2013), 'Network in network', *arXiv preprint arXiv:1312.4400* .

Liu, X., MacKay, R. J. & Steiner, S. H. (2008), 'Monitoring multiple stream processes', *Quality Engineering* **20**(3), 296–308.

Lowry, C. A. & Montgomery, D. C. (1995), 'A review of multivariate control charts', *IIE Transactions* **27**(6), 800–810.

Lu, C., Wang, Z.-Y., Qin, W.-L. & Ma, J. (2017), 'Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification', *Signal Processing* **130**, 377–388.

MacGregor, J. F. & Kourti, T. (1995), 'Statistical process control of multivariate processes', *Control Engineering Practice* **3**(3), 403–414.

Mahmoud, M. A., Henderson, G. R., Epprecht, E. K. & Woodall, W. H. (2010), 'Estimating the standard deviation in quality-control applications', *Journal of Quality Technology* **42**(4), 348–357.

Mandel, B. (1969), 'The regression control chart', *Journal of Quality Technology* **1**(1), 1–9.

Mason, R. L., Tracy, N. D. & Young, J. C. (1995), 'Decomposition of T2 for multivariate control chart interpretation', *Journal of Quality Technology* **27**(2), 99–108.

Mason, R. L., Tracy, N. D. & Young, J. C. (1997), 'A practical approach for interpreting multivariate T2 control chart signals', *Journal of Quality Technology* **29**(4), 396–406.

Megahed, F. M. & Jones-Farmer, L. A. (2015), 'Statistical perspectives on "big data"', *Frontiers in Statistical Quality Control* pp. 29–47.

Megahed, F. M., Woodall, W. H. & Camelio, J. A. (2011), 'A review and perspective on control charting with image data', *Journal of Quality Technology* **43**(2), 83–98.

Menafoglio, A. & Secchi, P. (2017), 'Statistical analysis of complex and spatially dependent data: a review of object oriented spatial statistics', *European Journal of Operational Research* **258**(2), 401–410.

Meneces, N. S., Olivera, S. A., Saccone, C. D. & Tessore, J. (2008), 'Statistical control of multiple-stream processes: a shewhart control chart for each stream', *Quality Engineering* **20**(2), 185–194.

Molnar, C. (2020), *Interpretable machine learning*, Lulu.com.

Montgomery, D. C. (2017), *Design and analysis of experiments*, John wiley & sons.

Montgomery, D. C. (2019), *Introduction to statistical quality control*, John Wiley & Sons.

Montgomery, D. C., Jennings, C. L. & Kulahci, M. (2015), *Introduction to time series analysis and forecasting*, John Wiley & Sons.

Montgomery, D. C. & Mastrangelo, C. M. (1991), 'Some statistical process control methods for autocorrelated data', *Journal of Quality Technology* **23**(3), 179–193.

Montgomery, D. C., Peck, E. A. & Vining, G. G. (2021), *Introduction to linear regression analysis*, John Wiley & Sons.

Morris, J. S. (2015), 'Functional regression', *Annual Review of Statistics and Its Application* **2**, 321–359.

Mortell, R. R. & Runger, G. C. (1995), 'Statistical process control of multiple stream processes', *Journal of Quality Technology* **27**(1), 1–12.

Murphy, K. P. (2012), *Machine learning: a probabilistic perspective*, MIT Press.

Nelson, L. S. (1986), 'Control chart for multiple stream processes', *Journal of Quality Technology* **18**(4), 255–256.

Niaki, S. A. & Abbasi, B. (2008), 'Detection and classification mean-shifts in multi-attribute processes by artificial neural networks', *International Journal of Production Research* **46**(11), 2945–2963.

Nielsen, M. A. (2015), *Neural networks and deep learning*, Vol. 2018, Determination Press San Francisco, CA.

Ott, E. R. & Snee, R. D. (1973), 'Identifying useful differences in a multiple-head machine', *Journal of Quality Technology* **5**(2), 47–57.

Pacella, M. & Semeraro, Q. (2011), 'Monitoring roundness profiles based on an unsupervised neural network algorithm', *Computers & Industrial Engineering* **60**(4), 677–689.

Phaladiganon, P., Kim, S. B., Chen, V. C., Baek, J.-G. & Park, S.-K. (2011), 'Bootstrap-based T^2 multivariate control charts', *Communications in Statistics—Simulation and Computation* **40**(5), 645–662.

Pini, A., Vantini, S., Colosimo, B. M. & Grasso, M. (2018), 'Domain-selective functional analysis of variance for supervised statistical profile monitoring of signal data', *Journal of the Royal Statistical Society Series C: Applied Statistics* **67**(1), 55–81.

Prechelt, L. (2002), Early stopping-but when?, *in* 'Neural Networks: Tricks of the trade', Springer, pp. 55–69.

Psarakis, S. (2011), 'The use of neural networks in statistical process control charts', *Quality and Reliability Engineering International* **27**(5), 641–650.

Pugh, G. A. (1989), 'Synthetic neural networks for process control', *Computers & Industrial Engineering* **17**(1-4), 24–26.

Qiu, P. (2013), *Introduction to Statistical Process Control*, Chapman & Hall.

R Core Team (2023), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *https://www.R-project.org/*

Ramsay, J. O. & Silverman, B. W. (2005), *Functional Data Analysis*, Wiley Online Library.

Rao, A. R. & Reimherr, M. (2023), 'Modern non-linear function-on-function regression', *Statistics and Computing* **33**(6), 1–12.

Rastrigin, L. (1963), 'The convergence of the random search method in the extremal control of a many parameter system', *Automaton & Remote Control* **24**, 1337–1342.

Reis, M. S. & Gins, G. (2017), 'Industrial process monitoring in the big data/industry 4.0 era: From detection, to diagnosis, to prognosis', *Processes* **5**(3), 35.

Reiss, P. T., Goldsmith, J., Shang, H. L. & Ogden, R. T. (2017), 'Methods for scalar-on-function regression', *International Statistical Review* **85**(2), 228–249.

Rossi, F. & Conan-Guez, B. (2006), 'Theoretical properties of projection based multilayer perceptrons with functional inputs', *Neural Processing Letters* **23**(1), 55–70.

Rossi, F., Conan-Guez, B. & Fleuret, F. (2002), Functional data analysis with multi layer perceptrons, *in* 'Proceedings of the 2002 International Joint Conference on Neural Networks', Vol. 3, IEEE, pp. 2843–2848.

Rossi, F., Delannay, N., Conan-Guez, B. & Verleysen, M. (2005), 'Representation of functional data in neural networks', *Neurocomputing* **64**, 183–210.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning representations by back-propagating errors', *Nature* **323**(6088), 533–536.

Runger, G., Alt, F. & Montgomery, D. (1996), 'Controlling multiple stream processes with principal components', *International Journal of Production Research* **34**(11), 2991–2999.

Russell, S. J. & Norvig, P. (2010), *Artificial intelligence a modern approach*, London.

Saghaei, A., Noorossana, R. & Amiri, A. (2013), *Statistical analysis of profile monitoring*, Wiley Online Library.

Secchi, P. (2018), 'On the role of statistics in the era of big data: A call for a debate', *Statistics & Probability Letters* **136**, 10–14.

Sergin, N. D. & Yan, H. (2021), 'Toward a better monitoring statistic for profile monitoring via variational autoencoders', *Journal of Quality Technology* **53**(5), 454–473.

Shang, H. L. (2014), 'A survey of functional principal component analysis', *AStA Advances in Statistical Analysis* **98**, 121–142.

Shu, L., Tsung, F. & Tsui, K.-L. (2004), 'Run-length performance of regression control charts with estimated parameters', *Journal of Quality Technology* **36**(3), 280–292.

Šidák, Z. (1967), 'Rectangular confidence regions for the means of multivariate normal distributions', *Journal of the American Statistical Association* **62**(318), 626–633.

Silverman, B. W. (2018), *Density estimation for statistics and data analysis*, Routledge.

Sorower, M. S. (2010), 'A literature survey on algorithms for multi-label learning', *Oregon State University, Corvallis* **18**, 1–25.

Stevens, A. & De Smedt, J. (2023), 'Explainability in process outcome prediction: Guidelines to obtain interpretable and faithful models', *European Journal of Operational Research* .

Stuart, M., Mullins, E. & Drew, E. (1996), 'Statistical quality control and improvement', *European journal of operational research* **88**(2), 203–214.

Süli, E. & Mayers, D. F. (2003), *An introduction to numerical analysis*, Cambridge University Press.

Thind, B., Multani, K. & Cao, J. (2023), 'Deep learning with functional inputs', *Journal of Computational and Graphical Statistics* **32**(1), 171–180.

Thomson, A. (1988), 'Real-time artificial intelligence for process monitoring and control', *IFAC Proceedings Volumes* **21**(13), 67–72.

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F. & Troncoso, A. (2021), 'Deep learning for time series forecasting: a survey', *Big Data* **9**(1), 3–21.

Tran, K. P., Castagliola, P., Celano, G. & Khoo, M. B. (2018), 'Monitoring compositional data using multivariate exponentially weighted moving average scheme', *Quality and Reliability Engineering International* **34**(3), 391–402.

Van den Boogaart, K. G. & Tolosana-Delgado, R. (2013), *Analyzing compositional data with R*, Vol. 122, Springer.

Van Rossum, G. & Drake Jr, F. L. (1995), *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam.

Venkatasubramanian, V., Rengaswamy, R., Yin, K. & Kavuri, S. N. (2003), 'A review of process fault detection and diagnosis: Part I: Quantitative model-based methods', *Computers & Chemical Engineering* **27**(3), 293–311.

Vicario, G. & Coleman, S. (2020), 'A review of data science in business and industry and a future view', *Applied Stochastic Models in Business and Industry* **36**(1), 6–18.

Vives-Mestres, M., Daunis-I-Estadella, J. & Martin-Fernandez, J.-A. (2014*a*), 'Individual T2 control chart for compositional data', *Journal of Quality Technology* **46**(2), 127–139.

Vives-Mestres, M., Daunis-i Estadella, J. & Martín-Fernández, J.-A. (2014*b*), 'Out-of-control signals in three-part compositional T2 control chart', *Quality and Reliability Engineering International* **30**(3), 337–346.

Vives-Mestres, M., Daunis-i Estadella, J. & Martín-Fernández, J.-A. (2016), 'Signal interpretation in Hotelling's T2 control chart for compositional data', *IIE Transactions* **48**(7), 661–672.

Vives-Mestres, M., Martín-Fernández, J.-A. & Kenett, R. S. (2016), 'Compositional data methods in customer survey analysis', *Quality and Reliability Engineering International* **32**(6), 2115–2125.

Wang, J.-L., Chiou, J.-M. & Müller, H.-G. (2016), 'Functional data analysis', *Annual Review of Statistics and Its Application* **3**, 257–295.

Wang, Q., Zheng, S., Farahat, A., Serita, S. & Gupta, C. (2019), Remaining useful life estimation using functional data analysis, *in* '2019 IEEE International Conference on Prognostics and Health Management', IEEE, pp. 1–8.

Wang, Q., Zheng, S., Farahat, A., Serita, S., Saeki, T. & Gupta, C. (2019), Multilayer perceptron for sparse functional data, *in* '2019 International Joint Conference on Neural Networks', IEEE, pp. 1–10.

Weese, M., Martinez, W., Megahed, F. M. & Jones-Farmer, L. A. (2016), 'Statistical learning methods applied to process monitoring: An overview and perspective', *Journal of Quality Technology* **48**(1), 4–24.

Williams, J. D., Woodall, W. H. & Birch, J. B. (2007), 'Statistical monitoring of nonlinear product and process quality profiles', *Quality and Reliability Engineering International* **23**(8), 925–941.

Wise, B., Veltkamp, D., Davis, B., Ricker, N. & Kowalski, B. (1988), 'Principal components analysis for monitoring the west valley liquid fed ceramic melter', *Waste Management* **88**, 811–818.

Wludyka, P. S. & Jacobs, S. L. (2002*a*), 'Controlling homogeneous multistream binomial processes with a chi-squared control chart', *Proceedings of the 33rd Annual Meeting of the Decision Sciences Institute* pp. 2254–2263.

Wludyka, P. S. & Jacobs, S. L. (2002*b*), 'Runs rules and p-charts for multistream binomial processes', *Communications in Statistics-Simulation and Computation* **31**, 97–142.

Woodall, W. H., Spitzner, D. J., Montgomery, D. C. & Gupta, S. (2004), 'Using control charts to monitor process and product quality profiles', *Journal of Quality Technology* **36**(3), 309–320.

Woodall, W. H. & Thomas, E. V. (1995), 'Statistical process control with several components of common cause variability', *IIE Transactions* **27**(6), 757–764.

Xu, L. D., Xu, E. L. & Li, L. (2018), 'Industry 4.0: state of the art and future trends', *International Journal of Production Research* **56**(8), 2941–2962.

Yan, W., Guo, P. et al. (2016), 'Nonlinear and robust statistical process monitoring based on variant autoencoders', *Chemometrics and Intelligent Laboratory Systems* **158**, 31–40.

Yao, F., Müller, H.-G. & Wang, J.-L. (2005), 'Functional linear regression analysis for longitudinal data'.

Yao, J., Mueller, J. & Wang, J.-L. (2021), Deep learning for functional data analysis with adaptive basis layers, *in* 'International Conference on Machine Learning', PMLR, pp. 11898–11908.

Yeganeh, A., Abbasi, S. A., Pourpanah, F., Shadman, A., Johannssen, A. & Chukhrova, N. (2022), 'An ensemble neural network framework for improving the detection ability of a base control chart in non-parametric profile monitoring', *Expert Systems with Applications* **204**, 117572.

Yeganeh, A. & Shadman, A. (2021), 'Monitoring linear profiles using artificial neural networks with run rules', *Expert Systems with Applications* **168**, 114237.

Yu, J. & Liu, X. (2022), 'One-dimensional residual convolutional auto-encoder for fault detection in complex industrial processes', *International Journal of Production Research* **60**(18), 5655–5674.

Zhang, J., Ren, H., Yao, R., Zou, C. & Wang, Z. (2015), 'Phase I analysis of multivariate profiles based on regression adjustment', *Computers & Industrial Engineering* **85**, 132–144.

Zhang, Z., Jiang, T., Li, S. & Yang, Y. (2018), 'Automated feature learning for nonlinear process monitoring–an approach using stacked denoising autoencoder and k-nearest neighbor rule', *Journal of Process Control* **64**, 49–61.

Zhang, Z., Jiang, T., Zhan, C. & Yang, Y. (2019), 'Gaussian feature learning based on variational autoencoder for improving nonlinear process monitoring', *Journal of Process Control* **75**, 136–155.

Zorriassatine, F. & Tannock, J. (1998), 'A review of neural networks for statistical process control', *Journal of Intelligent Manufacturing* **9**(3), 209–224.

Zou, C., Tsung, F. & Wang, Z. (2007), 'Monitoring general linear profiles using multivariate exponentially weighted moving average schemes', *Technometrics* **49**(4), 395–408.